Πανεπιστήμιο
Ιωαννίνων

ΣΧΟΛΗ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Ταξινόμηση Κειμένου με Τεχνικές Μηχανικής Μάθησης

Σπύρος Στραβοράβδης
th1151537 7@uoi.gr

Επιβλέπων: Σταύρος Π. Αδάμ
Επίκουρος Καθηγητής

Σεπτέμβριος 2020

Text Classification Using Machine Learning Techniques

Εγκρίθηκε από τριμελή εξεταστική επιτροπή

Ελλάδα, 29/09/2020

ΕΠΙΤΡΟΠΗ ΑΞΙΟΛΟΓΗΣΗΣ

1. Επιβλέπων καθηγητής
   Αδάμ Σταύρος,
   Επίκουρος Καθηγητής

2. Μέλος Επιτροπής
   Αντωνιάδης Νικόλαος,
   Καθηγητής

3. Μέλος Επιτροπής
   Καρβέλης Πέτρος,
   Πανεπιστημιακός Υπότροφος

Δήλωση μη λογοκλοπής

Δηλώνω υπεύθυνα και γνωρίζοντας τις κυρώσεις του Ν. 2121/1993 περί Πνευματικής Ιδιοκτησίας, ότι η παρούσα πτυχιακή εργασία είναι εξ ολοκλήρου αποτέλεσμα δικής μου ερευνητικής εργασίας, δεν αποτελεί προϊόν αντιγραφής ούτε προέρχεται από ανάθεση σε τρίτους. Όλες οι πηγές που χρησιμοποιήθηκαν (κάθε είδους, μορφής και προέλευσης) για τη συγγραφή της περιλαμβάνονται στη βιβλιογραφία.

Στραβοράβδης Σπυρίδων

Υπογραφή

Θέλω να ευχαριστήσω τον επιβλέποντα μου, Σταύρο Αδάμ, για τη βοήθεια και απέραντη υπομονή του κατά την εκπόνηση αυτής της πτυχιακής εργασίας.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Κατάλογος Παραδειγμάτων Κώδικα

Περίληψη

Η ταξινόμηση κειμένου ή εγγράφων αποτελεί ένα πρόβλημα στην επιστήμη των υπολογιστών για ανάθεση δεδομένων σε μία ή περισσότερες κλάσεις ή κατηγορίες. Ένας τρόπος επίλυσης του είναι με μεθόδους επεξεργασίας φυσικής γλώσσας. Ο κλάδος αυτός ασχολείται με τις αλληλεπιδράσεις μεταξύ υπολογιστών και ανθρώπινης γλώσσας, συγκεκριμένα τον προγραμματισμό μηχανών για επεξεργασία και ανάλυση μεγάλου όγκου δεδομένων φυσικής γλώσσας. Για την επίτευξη του στόχου αυτού χρησιμοποιούνται διάφορες μέθοδοι, οι οποίες συνεχώς εξελίσσονται. Μία από αυτές είναι η ενσωμάτωση λέξεων (ή διάνυσμα λέξεων), η οποία συνήθως κάνει χρήση νευρωνικών δικτύων.

Σκοπός της συγκεκριμένης εργασίας είναι να γίνει μία λεπτομερής αναφορά στην ταξινόμηση κειμένου και την επεξεργασία φυσικής γλώσσας, την ιστορική εξέλιξη τους και τις μεθόδους ή τεχνικές που χρησιμοποιούνται ευρέως στους συγκεκριμένους τομείς. Έπειτα γίνεται μνεία στην ενσωμάτωση λέξεων και παλιότερες μεθόδους που εφαρμόστηκαν για επίλυση των ίδιων προβλημάτων. Κατόπιν αναλύεται διεξοδικά η τεχνική Word2Vec, μία από τις πιο δημοφιλείς ενσωματώσεις λέξεων, τα μοντέλα που περιλαμβάνει, ο τρόπος λειτουργίας της και διάφορες μέθοδοι βελτιστοποίησης. Τέλος, γίνεται προγραμματιστική υλοποίηση του Word2Vec με τμηματοποίηση και παρουσίαση των βημάτων που απαιτούνται για τη χρήση της.

Abstract

Text or document classification is a problem in computer science for data assignment to one or more classes or categories. One way to solve it is with language processing methods. This field deals with the interactions between computers and human language, in particular the programming of machines for processing and analyzing large amounts of natural language data. Various methods are used to achieve this goal, which are constantly evolving. One of these is word embedding (or word vector), which usually uses neural networks to function.

The purpose of this paper is to make a detailed and clear reference to text classification and natural language processing, their historical development and the methods or techniques that are widely used in these specific fields. After that, the subject of word embedding is described and older methods used to solve the same problems before it. Then the Word2Vec technique, one of the most popular word embeddings, is examined along with the models it includes, its modus operandi and various optimization methods. Finally, a Word2Vec implementation is provided with a step-by-step presentation of the algorithm's internals.

1 Νευρωνικά δίκτυα

Στην ενότητα αυτή γίνεται μία περίληψη των εννοιών που σχετίζονται με τα νευρωνικά δίκτυα, καθώς υπάρχουν διάσπαρτες αναφορές σε αυτά στο υπόλοιπο κείμενο.

1.1 Εισαγωγή

Ένα νευρωνικό δίκτυο είναι ένα δίκτυο ή κύκλωμα νευρώνων, ή με μια σύγχρονη έννοια, ένα τεχνητό νευρωνικό δίκτυο αποτελείται από τεχνητούς νευρώνες ή κόμβους. [1]

Τα φυσικά νευρωνικά δίκτυα είναι ασφαλώς οι εγκέφαλοι των βιολογικών οργανισμών. Τα τεχνητά νευρωνικά δίκτυα *(ΤΝΔ)* είναι προσπάθεια αντιγραφής των φυσικών, ώστε να δημιουργηθούν υπολογιστικές μηχανές με χαρακτηριστικά που δεν είναι εγγενή σε συστήματα von Neumann, [2] όπως:

- παράλληλη επεξεργασία
- κατανεμημένη απεικόνιση και παραλληλισμό
- δυνατότητα μάθησης
- δυνατότητα γενίκευσης
- προσαρμοστικότητα
- εγγενή επεξεργασία πληροφοριών με βάση τα συμφραζόμενα
- ανοχή σε σφάλματα
- χαμηλή κατανάλωση ενέργειας

Τα ΤΝΔ είναι κομβικά για τον τομέα της μηχανικής μάθησης. Ως μηχανική μάθηση ορίζεται η μελέτη των υπολογιστικών αλγορίθμων που βελτιώνονται αυτόματα μέσω εμπειρίας. [3] Θεωρείται υποσύνολο της τεχνητής νοημοσύνης.

1.2 Δομή τεχνητού νευρώνα

Η έρευνα στα τεχνητά νευρωνικά δίκτυα ξεκίνησε τη δεκαετία του 1940, από τους McCulloch και Pitts, οι οποίοι περιέγραψαν ένα απλό μοντέλο νευρώνα, όπου οι είσοδοι εξαρτώνται από συναπτικά βάρη (synaptic weights) και χρησιμοποιείται η βηματική συνάρτηση (step function), ένα απλός τύπος συνάρτησης ενεργοποίησης. [4]

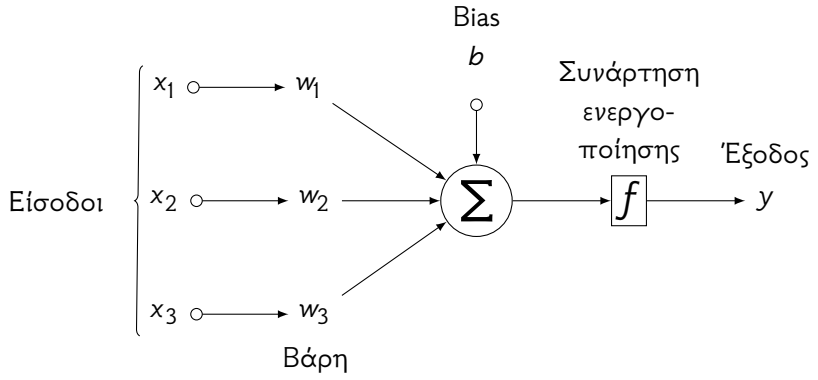
$$u = \sum_{i=1}^{n} w_i x_i$$

Άθροισμα u του φορτίου που δέχεται ο νευρώνας

$$f(u) = \begin{cases} 0 & \text{if } u \leq 0 \\ 1 & \text{if } u > 0 \end{cases}$$

Βηματική συνάρτηση

Η συνάρτηση ενεργοποίησης μπορεί να διαφέρει. Μερικά γνωστά παραδείγματα αποτελούν η γραμμική, η σιγμοειδής, η υπερβολική εφαπτομένη, η συνάρτηση κατωφλίου και η συνάρτηση ράμπας. [2, 4]



Σχήμα 1.1: Μοντέλο νευρώνα των McCulloch-Pitts.

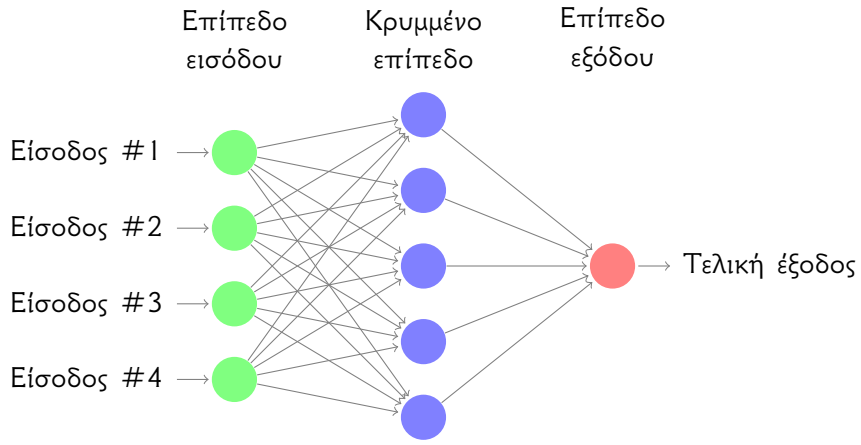
1.3 Αρχιτεκτονικές δικτύων

Τα ΤΝΔ μπορούν να θεωρηθούν σαν σταθμισμένοι κατευθυνόμενοι γράφοι στους οποίους οι τεχνητοί νευρώνες είναι κόμβοι και οι κατευθυνόμενες ακμές (με βάρη) είναι συνδέσεις μεταξύ εισόδων και εξόδων νευρώνων.

Με βάση την αρχιτεκτονική σύνδεσης, τα ΤΝΔ μπορούν να χωριστούν σε δύο κύριες κατηγορίες:

1. δίκτυα εμπρόσθιας τροφοδότησης *(feed-forward network)*, όπου οι συνδέσεις μεταξύ κόμβων δεν σχηματίζουν κύκλο

2. επαναλαμβανόμενα δίκτυα *(recurrent networks)*, όπου σχηματίζονται βρόχοι εξαιτίας συνδέσεων ανατροφοδότησης



Σχήμα 1.2: Ένα απλό νευρωνικό δίκτυο εμπρόσθιας τροφοδότησης.

1.4 Εκμάθηση

Η δυνατότητα εκμάθησης είναι θεμελιώδες χαρακτηριστικό νοημοσύνης. Παρότι ένας ακριβής ορισμός της μάθησης είναι δύσκολο να διατυπωθεί, μία μαθησιακή διαδικασία στο πλαίσιο των ΤΝΔ μπορεί να οριστεί ως το πρόβλημα της ενημέρωσης της αρχιτεκτονικής του δικτύου και των βαρών σύνδεσης έτσι ώστε ένα δίκτυο να μπορεί να εκτελεί αποτελεσματικά μία συγκεκριμένη εργασία. Το δίκτυο συνήθως μαθαίνει τα βάρη σύνδεσης από διαθέσιμα πρότυπα εκπαίδευσης. Η απόδοση βελτιώνεται με την πάροδο του χρόνου ενημερώνοντας επαναληπτικά τα βάρη στο δίκτυο. Η ικανότητα των ΤΝΔ να μαθαίνουν αυτόματα από παραδείγματα τα καθιστούν ελκυστικά και συναρπαστικά. Αντί να ακολουθούν ένα σύνολο κανόνων που καθορίζονται από ειδικούς, τα ΤΝΔ φαίνεται να μαθαίνουν τους βασικούς κανόνες (όπως οι σχέσεις εισόδου-εξόδου) από τη συλλογή αντιπροσωπευτικών παραδειγμάτων που δίνεται. Αυτό είναι ένα από τα σημαντικότερα πλεονεκτήματα των νευρωνικών δικτύων σε σχέση με τα παραδοσιακά έμπειρα συστήματα *(expert systems)*. [2]

Υπάρχουν δύο βασικά είδη παραδειγμάτων μάθησης:

1. Επιβλεπόμενη μάθηση *(Supervised learning)*: κατηγορία μηχανικής μάθησης στην οποία ένα υπολογιστικό σύστημα μαθαίνει να αντιστοιχεί μία είσοδο *(input)* σε μία έξοδο *(output)* με βάση ήδη υπάρχοντα ζευγάρια input-output.

2. Μη-επιβλεπόμενη μάθηση *(Unsupervised learning)*: τεχνική μάθησης που αναζητά πρότυπα σε κάποιο data-set, χωρίς υπάρχουσες ετικέτες και με ελάχιστη ανθρώπινη επίβλεψη.

1.4.1 Επίπεδα

Τα περισσότερα νευρωνικά δίκτυα αποτελούνται από πολλαπλά επίπεδα *(σχήμα 1.2)*. Τα πιο συχνά είδη αυτών είναι:

- το επίπεδο είσοδου, το οποίο λαμβάνει τις αρχικές πληροφορίες
- τα κρυμμένα επίπεδα, τα οποία δεν έχουν άμεση επαφή με το περιβάλλον (εξού και η ονομασία τους). Μπορεί να υπάρχουν 0, 1 ή περισσότερα κρυμμένα επίπεδα σε ένα δίκτυο, τα οποία συνήθως περιέχουν τον ίδιο αριθμό νευρώνων. Οι κόμβοι σε αυτά εκτελούν υπολογισμούς και μεταφέρουν πληροφορίες από το επίπεδο εισόδου στο επίπεδο εξόδου
- το επίπεδο εξόδου, με κόμβους που συνδέονται διαδοχικά (πλήρως ή τοπικά), χωρίς συνδέσεις μεταξύ των κόμβων στο ίδιο επίπεδο και χωρίς συνδέσεις ανατροφοδότησης μεταξύ επιπέδων.

1.4.2 Διόρθωση λαθών και οπίσθια τροφοδότηση

Ο αλγόριθμος Back-propagation προτάθηκε από τον Paul Werbos στη δεκαετία του 1970 στα πλαίσια της ανάλυσης μοντέλων οικονομικής και πολιτικής πρόβλεψης. [5] Τη δεκαετία του 1980, έγινε αντιληπτό ότι η μέθοδος μπορούσε να μεταφερθεί αυτούσια στην εκπαίδευση νευρωνικών δικτύων πολλών στρωμάτων, και έκτοτε έγινε η πιο γνωστή και η πιο διαδεδομένη μέθοδος για το σκοπό αυτό. [6, 4] Βασικό χαρακτηριστικό της μεθόδου αυτής είναι η ύπαρξη στόχων, επομένως το μοντέλο ανήκει στην κατηγορία των δικτύων που εκπαιδεύονται με επίβλεψη.

Βασίζεται στην αρχή διόρθωσης λαθών. Επομένως, αν λάβουμε μία συνάρτηση απώλειας *(loss function)*, όπως η συνάρτηση μέσου τετραγωνισμένου κόστους *(Mean Squared Error, MSE)*:

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

όπου Y είναι το διάνυσμα των παρατηρούμενων τιμών και $\hat{Y}$ το διάνυσμα των προβλεπόμενων τιμών, τότε θα επιστραφεί ένα νούμερο που θα συμβολίζει την απώλεια του νευρωνικού δικτύου. Σε μία ιδανική περίπτωση, το νούμερο αυτό θα ήταν μηδέν και ο στόχος είναι να πλησιάσει σταδιακά σε αυτό όσο γίνεται. Η οπίσθια τροφοδότηση είναι η μείωση της συνάρτησης απώλειας με μεταβολή των βαρών και κλίσεων του νευρωνικού δικτύου.

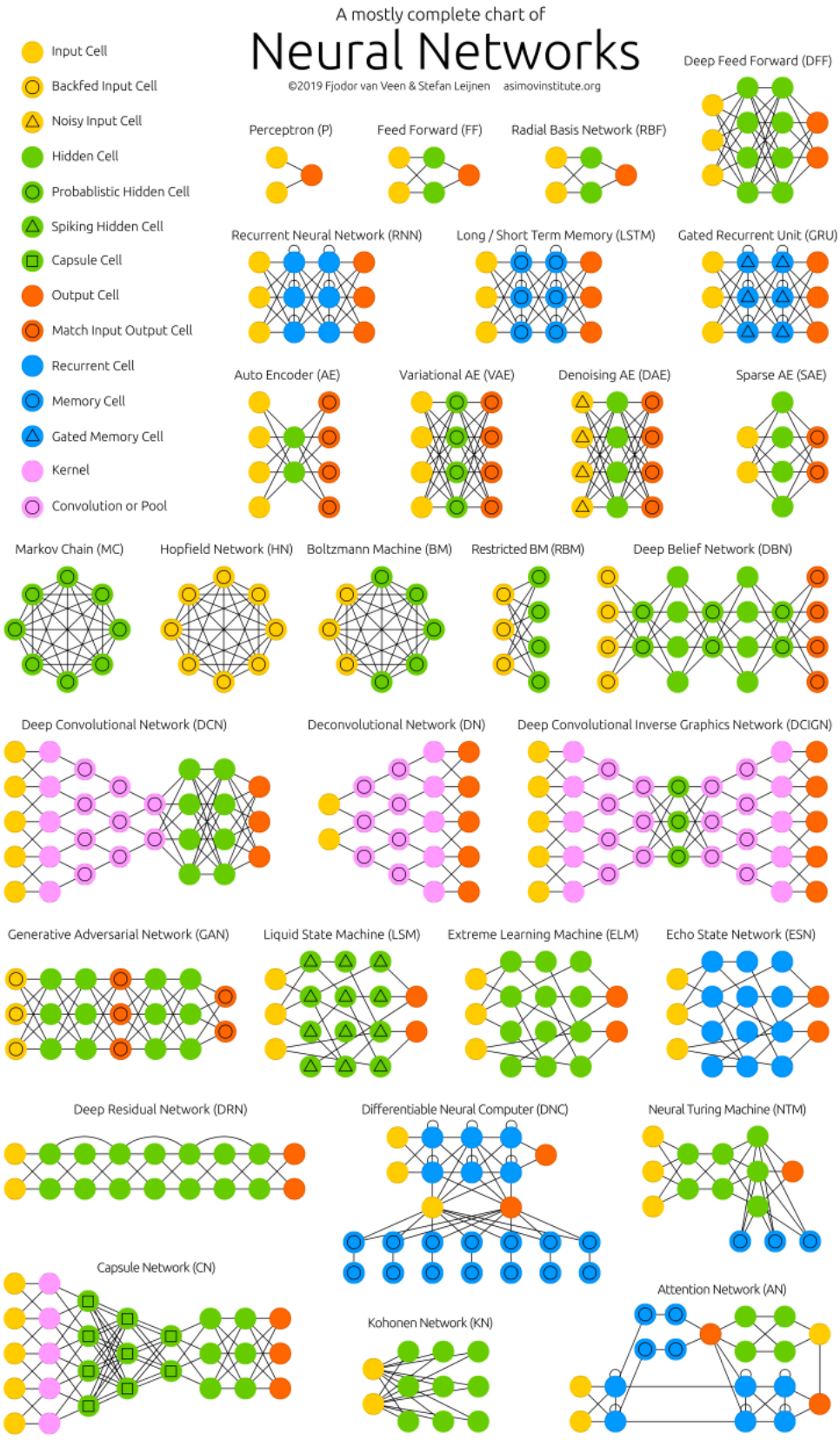
Ο αλγόριθμος οπίσθιας τροφοδότησης λειτουργεί υπολογίζοντας την κλίση της συνάρτησης απώλειας σε σχέση με κάθε βάρος με βάση τον κανόνα της αλυσίδας, υπολογίζοντας την κλίση κάθε φορά, επαναλαμβάνοντας προς τα πίσω από το τελευταίο επίπεδο για να αποφευχθούν περιττοί υπολογισμοί των ενδιάμεσων όρων στον κανόνα της αλυσίδας. Η διαδικασία επαναλαμβάνεται πολλαπλές φορές, για καλύτερα αποτελέσματα. [7]

Συνάρτηση απώλειας *(loss function)*: Οι συναρτήσεις απώλειας χρησιμοποιούνται για τον προσδιορισμό του σφάλματος (ή απώλειας) μεταξύ της εξόδου των αλγορίθμων και της δεδομένης στοχευμένης τιμής.

Κανόνας αλυσίδας *(chain rule)*: τύπος για τον υπολογισμό της παραγώγου μιας σύνθετης συνάρτησης.

1.5 Τύποι νευρωνικών δικτύων

Υπάρχουν διάφοροι τύποι νευρωνικών δικτύων· μία παρουσίαση των σημαντικότερων μπορεί να βρεθεί στο *σχήμα 1.3*.



Σχήμα 1.3: Λίστα τύπων νευρωνικών δικτύων. [8]

2 Ταξινόμηση κειμένου

Ως ταξινόμηση κειμένου (ή ταξινόμηση εγγράφων) ορίζεται η διαδικασία κατά την οποία ανατίθενται ετικέτες ή κατηγορίες σε κείμενο, ανάλογα με το περιεχόμενο του. Η διαδικασία αυτή μπορεί να γίνει χειροκίνητα ή αυτοματοποιημένα με τη χρήση αλγορίθμων.

2.1 Κατηγορίες συστημάτων για ταξινόμηση κειμένου

2.1.1 Συστήματα βασισμένα σε κανόνες (Rule-based systems)

Τα συστήματα που λειτουργούν με βάση τους κανόνες ταξινομούν το κείμενο σε οργανωμένες ομάδες χρησιμοποιώντας ένα σύνολο χειροποίητων γλωσσικών κανόνων. Αυτοί οι κανόνες δίνουν εντολή στο σύστημα να χρησιμοποιεί σημασιολογικά σχετικά στοιχεία ενός κειμένου για να προσδιορίσει σχετικές κατηγορίες με βάση το περιεχόμενό του. Κάθε κανόνας αποτελείται από ένα προηγούμενο ή μοτίβο και μια προβλεπόμενη κατηγορία.

Τα συγκεκριμένα συστήματα είναι κατανοητά από τον άνθρωπο και μπορούν να βελτιωθούν με την πάροδο του χρόνου. Αλλά αυτή η προσέγγιση έχει μερικά μειονεκτήματα. Για αρχάριους, αυτά τα συστήματα απαιτούν βαθιά γνώση του τομέα. Είναι επίσης χρονοβόρα, καθώς η δημιουργία κανόνων για ένα πολύπλοκο σύστημα μπορεί να είναι αρκετά δύσκολη και συνήθως απαιτεί πολλή ανάλυση και δοκιμή. Τα συστήματα που βασίζονται σε κανόνες είναι επίσης δύσκολο να διατηρηθούν και δεν κλιμακώνονται καλά, δεδομένου ότι η προσθήκη νέων κανόνων μπορεί να επηρεάσει τα αποτελέσματα των προϋπάρχοντων κανόνων.

2.1.2 Συστήματα βασισμένα σε μηχανική μάθηση (Machine learning based systems)

Σε αντίθεση με τα συστήματα βασισμένα σε κανόνες, οι τεχνικές μηχανικής μάθησης κάνουν ταξινόμηση με βάση προηγούμενες παρατηρήσεις των δεδομένων. Χρησιμοποιώντας προ-επισημασμένα παραδείγματα ως δεδομένα εκπαίδευσης, ένας αλγόριθμος μηχανικής μάθησης μπορεί να μάθει τις εγγενείς συσχετίσεις μεταξύ των κειμένων και των ετικετών τους. Έτσι, οι μέθοδοι που βασίζονται στη μηχανική μάθηση μπορούν να ανιχνεύσουν κρυμμένα μοτίβα στα δεδομένα, είναι πιο επεκτάσιμες και μπορούν να εφαρμοστούν σε ποικίλες εργασίες, σε αντίθεση με τις μεθόδους που βασίζονται σε κανόνες, οι οποίες χρειάζονται διαφορετικά σύνολα κανόνων για διαφορετικές εργασίες.

Τα μοντέλα μηχανικής μάθησης έχουν τραβήξει το ενδιαφέρον των ερευνητών τα τελευταία χρόνια. [9] Τα περισσότερα μοντέλα κλασικής μηχανικής μάθη-

σης ακολουθούν τη διαδικασία δύο βημάτων. Στο 1ο βήμα μερικά *χειροποίητα (hand-crafted)* χαρακτηριστικά εξάγονται από τα έγγραφα (ή οποιαδήποτε άλλη μονάδα κειμένου). Στο 2ο βήμα αυτά τα χαρακτηριστικά εισάγονται σε έναν ταξινομητή για να κάνει μια πρόβλεψη. Το *bag-of-words* (αναπαράσταση του κειμένου σαν πολυσύνολο των λέξεων που περιέχονται, χωρίς να δίνεται σημασία στη γραμματική και τη σειρά των λέξεων, παρουσιάζεται στην *ενότητα 5*) και οι επεκτάσεις του αποτελούν δημοφιλή hand-crafted χαρακτηριστικά. Δημοφιλείς επιλογές αλγορίθμων ταξινόμησης είναι οι Naive Bayes, Support Vector Machines *(SVM)*, κρυφά μοντέλα Markov *(HMM)*, gradient boosting trees και random forests.

Οι προσεγγίσεις δύο βημάτων ωστόσο έχουν διάφορους περιορισμούς. Για παράδειγμα, η εξάρτηση από τα χειροποίητα χαρακτηριστικά απαιτεί κοπιαστική ανάλυση και οργάνωση των χαρακτηριστικών, που χρειάζονται για τροφοδότηση των μοντέλων, ώστε να επιτευχθεί καλή απόδοση. Επιπλέον, η ισχυρή εξάρτηση από τη γνώση του τομέα για το σχεδιασμό χαρακτηριστικών καθιστά τη μέθοδο δύσκολη τη γενίκευση σε νέες εργασίες. Τέλος, αυτά τα μοντέλα δεν μπορούν να επωφεληθούν πλήρως από μεγάλες ποσότητες δεδομένων εκπαίδευσης επειδή τα χαρακτηριστικά (ή τα πρότυπα λειτουργιών) είναι προκαθορισμένα. [10]

2.1.3 Υβριδικά συστήματα

Τα υβριδικά συστήματα συνδυάζουν έναν βασικό ταξινομητή εκπαιδευμένο με μηχανική μάθηση και ένα σύστημα βασισμένο σε κανόνες, το οποίο χρησιμοποιείται για την περαιτέρω βελτίωση των αποτελεσμάτων.

2.2 Αλγόριθμοι ταξινόμησης κειμένου

2.2.1 Naive Bayes

Οι ταξινομητές Naive Bayes είναι μία κατηγορία απλών αλλά αποδοτικών γραμμικών ταξινομητών που βασίζονται στο θεώρημα του Bayes.

Ακολουθεί το θεώρημα του Bayes:

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

όπου:

- A και B είναι ενδεχόμενα και $P(B) \neq 0$
- $P(A \mid B)$: η πιθανότητα να συμβεί το ενδεχόμενο A δεδομένου ότι το B είναι αληθές

- $P(B \mid A)$: η πιθανότητα να συμβεί το ενδεχόμενο B δεδομένου ότι το A είναι αληθές
- $P(A)$ και $P(B)$ είναι οι πιθανότητες των A και B αντίστοιχα

Το επίθετο *"naïve" (αδαείς - αφελείς)* που περιγράφει τους συγκεκριμένους ταξινομητές προέρχεται από την εικασία ότι τα χαρακτηριστικά στο dataset είναι αμοιβαία ανεξάρτητα. Στην πράξη, η εικασία ανεξαρτησίας παραβιάζεται, ωστόσο οι ταξινομητές Naive Bayes ανταγωνίζονται με επιτυχία πιο σύνθετους ταξινομητές. [11]

Οι συγκεκριμένοι ταξινομητές μπορούν και έχουν εφαρμοστεί σε χρήσεις ταξινόμησης κειμένου. Για παράδειγμα, ας υποτεθεί ότι υπάρχει μία συλλογή 500 εγγράφων από τα οποία τα 100 είναι spam μηνύματα. Αν προστεθεί ένα καινούργιο μήνυμα που περιέχει το κείμενο "Hello world", πρέπει να υπολογιστεί η υπό συνθήκη πιθανότητα να ανήκουν σε μία ομάδα αν θεωρηθεί ότι είναι spam. Το πρότυπο αποτελείται από δύο χαρακτηριστικά, "hello" και "world" και η υπό συνθήκη πιθανότητα να ανήκουν σε μία ομάδα είναι το γινόμενο της "πιθανότητας να εμφανιστεί η λέξη "hello" δεδομένου ότι το μήνυμα είναι spam" και "της πιθανότητας να εμφανιστεί η λέξη "world" δεδομένου ότι το μήνυμα είναι spam". [12]

$$P(\mathbf{x} = [\text{hello, world}] \mid \omega = \text{spam}) = P(\text{hello} \mid \text{spam}) \cdot P(\text{world} \mid \text{spam})$$

Με βάση το υπάρχον dataset 500 εγγράφων μπορεί να γίνει χρήση της εκτίμησης μέγιστης πιθανοφάνειας για να υπολογιστούν αυτές οι πιθανότητες, κοινώς πόσο συχνά αυτές οι λέξεις εμφανίζονται στο σώμα κειμένων *(corpus)* των spam μηνυμάτων: [12]

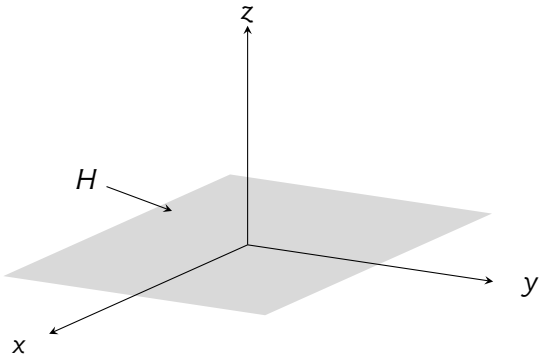
$$\hat{P}(\mathbf{x} = [\text{hello, world}] \mid \omega = \text{spam}) = \frac{20}{100} \cdot \frac{2}{100} = 0.004$$

2.2.2 Μηχανές Διανυσμάτων Υποστήριξης

Οι μηχανές διανυσμάτων υποστήριξης *(Support Vector Machines, SVM)* είναι μία μέθοδος ταξινόμησης, η οποία βασίζεται στη στατιστική θεωρία μάθησης. Για να λειτουργήσει αναζητά ένα υπερεπίπεδο σε χώρο N διαστάσεων (όπου N = ο αριθμός των χαρακτηριστικών), το οποίο διαχωρίζει ξεκάθαρα τα σημεία των δεδομένων.

> Υπερεπίπεδο: Γεωμετρικά, είναι ένας υποχώρος του οποίου η διάσταση είναι μία λιγότερη από αυτή του περιβάλλοντα χώρου.

- Σε δισδιάστατο χώρο ένα υπερεπίπεδο είναι μία γραμμή.
- Σε τρισδιάστατο χώρο ένα υπερεπίπεδο είναι ένα επίπεδο.
- Σε χώρους μεγαλύτερων διαστάσεων η οπτικοποίηση είναι δύσκολη.



Σχήμα 2.1: Ένα επίπεδο H δύο διαστάσεων μέσα σε έναν τρισδιάστατο χώρο.

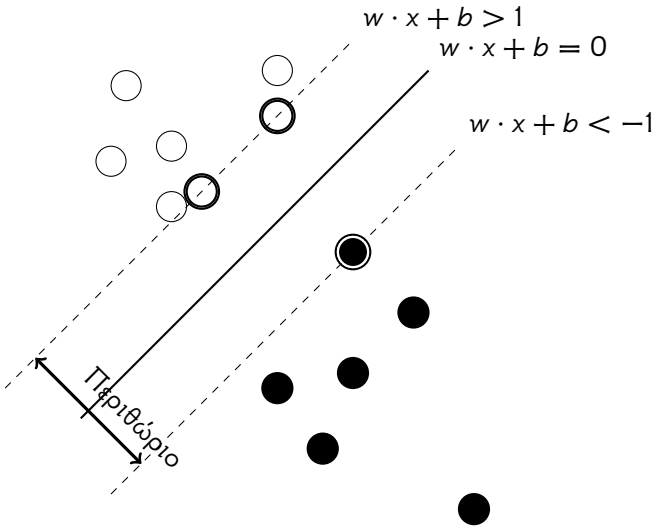
Η απόσταση ανάμεσα στο υπερεπίπεδο και το κοντινότερο σημείο είναι γνωστή ως περιθώριο. Ο στόχος είναι να βρεθεί ένα υπερεπίπεδο με όσο το δυνατό μεγαλύτερο περιθώριο με κάθε σημείο στο σύνολο εκπαίδευσης, προκειμένου να υπάρχει μεγαλύτερη πιθανότητα σωστής ταξινόμησης νέων δεδομένων. [13]

Το περιθώριο έχει δύο πιθανές καταστάσεις: [14]

- Σκληρό περιθώριο *(Hard-margin)*: Η ιδανική περίπτωση, τα δεδομένα είναι γραμμικά διαχωρίσιμα και ο διαχωρισμός μπορεί να γίνει με δύο παράλληλες γραμμές, οι οποίες συμβολίζουν τα όρια του περιθωρίου.
- Μαλακό περιθώριο *(Soft-margin)*: Τα δεδομένα δεν είναι γραμμικά διαχωρίσιμα, ωστόσο με ένα μικρό βαθμό ανεκτικότητας μπορεί να βρεθεί μία γραμμή που να διαχωρίζει το μεγαλύτερο ποσοστό των δεδομένων. Ο βαθμός ανεκτικότητας εξαρτάται από μία συνάρτηση απώλειας που ονομάζεται hinge loss.

2.2.3 Deep learning

Το 2012, το AlexNet, ένα μοντέλο βαθιάς μάθησης κέρδισε τον διαγωνισμό ImageNet με διαφορά. Αυτό οδήγησε σε ραγδαία αύξηση έρευνας στον συγκεκριμένο τομέα. Από τότε, τα μοντέλα βαθιάς εκμάθησης έχουν εφαρμοστεί επιτυχώς



Σχήμα 2.2: Συσταδοποίηση σε δύο διακριτές ομάδες με χρήση SVM. [15]

σε πληθώρα εργασιών, όπως μηχανική όραση ή επεξεργασία φυσικής γλώσσας. Πέρα από την ανακάλυψη κρυμμένων προτύπων σε δεδομένα, τα μοντέλα αυτά είναι πιο εύκολο να μεταφερθούν από μία εφαρμογή σε μία άλλη.

Σύμφωνα με μία άποψη [10], τα μοντέλα βαθιάς εκμάθησης μπορούν να ταξινομηθούν στις ακόλουθες κατηγορίες:

- Μοντέλα βασισμένα σε δίκτυα πρόσθιας τροφοδότησης *(feed-forward)*, τα οποία διακρίνουν το κείμενο σαν ένα bag-of-words
- Μοντέλα βασισμένα σε επαναλαμβανόμενα νευρωνικά δίκτυα *(recurrent neural networks, RNNs)*, τα οποία διακρίνουν το κείμενο σαν σειρά λέξεων
- Μοντέλα βασισμένα σε δίκτυα συνέλιξης *(convolutional neural networks, CNNs)*, τα οποία εκπαιδεύονται για αναγνώριση προτύπων σε κείμενο, ώστε να αποτυπώσουν εξαρτήσεις λέξεων και δομές κειμένου
- Νευρωνικά Δίκτυα με Κάψουλες *(capsule networks)*, τα οποία διορθώνουν την απώλεια πληροφορίας που παρατηρείται στα CNNs και πρόσφατα εφαρμόστηκαν σε ταξινόμηση κειμένου
- Attention mechanism, που χρησιμεύει στην αναγνώριση σχετικών λέξεων σε κείμενο

- Memory-augmented δίκτυα, που συνδυάζουν νευρωνικά δίκτυα με μορφές εξωτερικής μνήμης, από την οποία τα μοντέλα διαβάζουν ή γράφουν σε αυτή
- Transformers, που επιτρέπουν περισσότερη παραλληλοποίηση από τα RNNs, κάνοντας πιο εύκολη την εκπαίδευση μεγάλων γλωσσικών μοντέλων με χρήση GPU clusters
- Νευρωνικά δίκτυα σε γράφους, που έχουν σχεδιαστεί για αποτύπωση εσωτερικών δομών γράφων φυσικής γλώσσας, όπως τα συντακτικά και σημασιολογικά δέντρα
- Σιαμέζικα νευρωνικά δίκτυα *(Siamese neural networks)*, σχεδιασμένα για αντιστοίχιση κειμένου, ειδική περίπτωση ταξινόμησης κειμένου
- Υβριδικά μοντέλα, που συνδυάζουν attention, RNNs, CNNs για αποτύπωση τοπικών και καθολικών χαρακτηριστικών προτάσεων και λέξεων
- Επιπλέον τεχνολογίες μη-επιβλεπόμενης μάθησης, όπως autoencoders ή reinforcement learning.

3 Επεξεργασία φυσικής γλώσσας

Η επεξεργασία φυσικής γλώσσας *(Natural language processing, NLP)* σχετίζεται με τη δημιουργία συστημάτων που "καταλαβαίνουν" μία γλώσσα προκειμένου να εκτελέσουν συγκεκριμένες εργασίες. Αυτές οι εργασίες μπορεί να περιλαμβάνουν: [41]

- Απάντηση ερωτήσεων (απαντάται σε εικονικούς βοηθούς όπως Siri, Alexa, Cortana)
- Ανάλυση συναισθημάτων (αν μία πρόταση έχει θετικό ή αρνητικό συνειρμό)
- Απεικόνιση εικόνας σε κείμενο (πχ. δημιουργία μίας λεζάντας για μία εισαγόμενη εικόνα)
- Μηχανική μετάφραση (αυτόματη μετάφραση κειμένου σε άλλη γλώσσα)
- Αναγνώριση ομιλίας (μετατροπή περιεχομένου ομιλίας σε γραπτή μορφή)
- Ετικετοποίηση ομιλίας (*speech tagging*, πχ. διαχωρισμός ρημάτων, ουσιαστικών, επιθέτων κ.ο.κ.)
- Αναγνώριση ονομασίων οντοτήτων (*name entity recognition*, πχ. ονοματεπώνυμα, τοποθεσίες κτλ.)

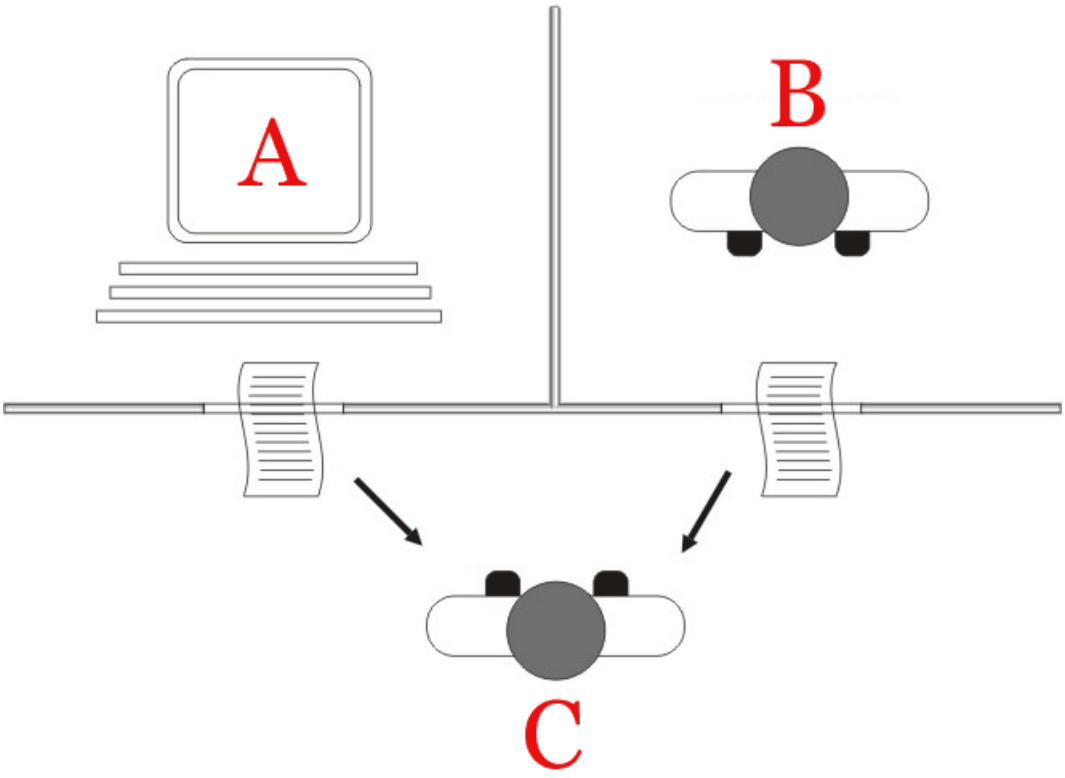
3.1 Ιστορικό

Η ιστορία της επεξεργασίας φυσικής γλώσσας ξεκινάει τη δεκαετία του 1950. Το 1950, ο Alan Turing[1] δημοσίευσε το *Computing Machinery and Intelligence* [16]. Στο συγκεκριμένο paper εξηγεί την έννοια που μετέπειτα ονομάστηκε Turing test. Το Turing test θέτει το ερώτημα αν κάποια μηχανή μπορεί να σκεφτεί. Επειδή η προηγούμενη πρόταση είναι διφορούμενη και μπορεί να παρερμηνευτεί, ο Turing θέτει ως παράδειγμα το "παιχνίδι της μίμησης".

Στο συγκεκριμένο νοητικό παιχνίδι υπάρχουν 3 παίκτες, οι Α, Β και C. Ο παίκτης Α είναι άντρας, ο παίκτης Β είναι γυναίκα και ο παίκτης C, ο ανακριτής (το φύλο του οποίου δεν έχει σημασία) προσπαθεί να εντοπίσει το φύλο τους με μία σειρά ερωτήσεων μέσω γραπτών μηνυμάτων. Ο παίκτης Α προσπαθεί να ξεγελάσει τον C, ενώ ο Β να τον βοηθήσει. Σε αυτό το σημείο ο Turing αναρωτιέται αν στην περίπτωση που μία μηχανή είναι στο ρόλο του παίκτη Α, ο ανακριτής θα έχει τα ίδια ποσοστά επιτυχίας που είχε στην προηγούμενη περίπτωση, όταν

[1] γνωστός και από τη δημιουργία της μηχανής Turing καθώς και τη συμμετοχή του στην αποκρυπτογράφηση μηνυμάτων της μηχανής Enigma κατά τον Β' Παγκόσμιο Πόλεμο

στο παιχνίδι συμμετείχαν ένας άντρας και μία γυναίκα. Στο ίδιο paper ο Turing αναφέρει και μία ελαφρώς αλλαγμένη έκδοση του συλλογισμού, στην οποία ο παίκτης Α (μηχανή) και ο παίκτης Β (άντρας) προσπαθούν να ξεγελάσουν τον ανακριτή.



Σχήμα 3.1: Η συνηθισμένη μορφή του Turing test. [17]

Ο Turing αναλύει επίσης, στην ίδια εργασία, εννιά συνηθισμένες απόψεις που συναντώνται και διαφωνούν με την ιδέα της ύπαρξης τεχνητής νοημοσύνης. [16]

Το 1954 εκτελείται μία από τις πρώτες σημαντικές παρουσιάσεις μηχανικής μετάφρασης, από το πανεπιστημίο του Georgetown και την IBM. Η δοκιμή μετάφρασης 60 απλών προτάσεων από τα ρώσικα στα αγγλικά στέφθηκε με επιτυχία. Ωστόσο, οι προγνώσεις της εποχής που ανέφεραν ότι το πρόβλημα της μηχανικής μετάφρασης θα λυθεί σε 3 με 5 χρόνια δεν επαληθεύτηκαν.

Την δεκαετία του 1960 υπήρξε πρόοδος σε προγράμματα κατανόησης φυσικής γλώσσας. Η ELIZA (1964-1966, Joseph Weizenbaum) με απλή εύρεση

ορισμών και αντικαταστάσεις φράσεων ή λέξεων προσομοίωνε συνομιλίες με χρήστες. Στην πραγματικότητα δεν υπήρχε κάποιο νοηματικό πλαίσιο, αλλά έδινε την ψευδαίσθηση κατανόησης και υπήρξε ιδιαίτερα δημοφιλής και επιδραστική. Το SHRDLU (1968-1970, Terry Winograd) επέτρεπε σε χρήστες να δίνουν εντολές σε έναν υπολογιστή που λειτουργούσε σε έναν απλοϊκό "κόσμο κύβων". Το πείραμα έτρεψε τις προσδοκίες για περαιτέρω έρευνα με βάση αυτό, ωστόσο μόλις επιχειρήθηκε να χρησιμοποιηθεί σε ρεαλιστικά περιβάλλοντα φάνηκαν οι περιορισμοί του.

> Οντολογία: τρόπος παρουσίασης των ιδιοτήτων μίας θεματικής περιοχής και πως αυτές σχετίζονται, καθορίζοντας ένα σύνολο εννοιών και κατηγοριών που αντιπροσωπεύουν το θέμα.

Λίγα χρόνια αργότερα χρησιμοποιήθηκαν *εννοιολογικές οντολογίες*, προκειμένου να μετατραπούν πληροφορίες σε μορφή εύκολα κατανοητή από τον υπολογιστή. Εκείνη την περίοδο έκαναν την εμφάνιση τους τα πρώτα chat bots.

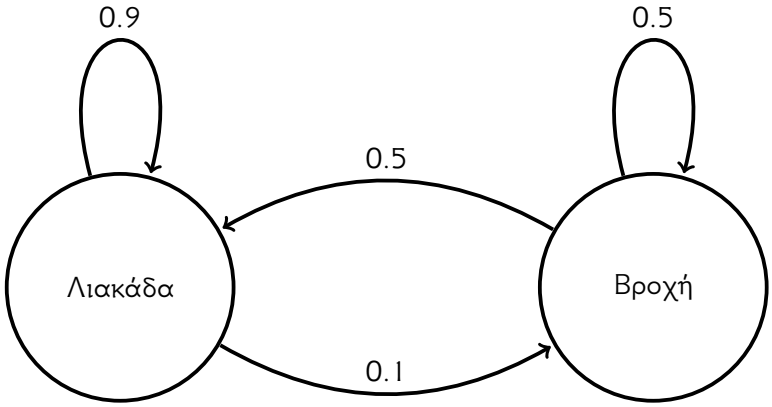
Από το 1980, υπήρξε ραγδαία πρόοδος στον τομέα της επεξεργασίας φυσικής γλώσσας χάρη στην εισαγωγή των πρώτων αλγορίθμων μηχανικής μάθησης. Αυτό έγινε δυνατό τόσο χάρη στην αύξηση της υπολογιστικής ισχύος, όσο και στην αλλαγή νοοτροπίας που οδήγησε σε μείωση της ισχύος των θεωριών του Chomsky.

3.2 Αλυσίδες Markov (Markov chains)

Ένα από τα παλαιότερα μοντέλα που μπορούν να χρησιμοποιηθούν στην επεξεργασία φυσικής γλώσσας είναι οι αλυσίδες Markov. Στη συνέχεια, πριν δοθεί κάποιος αυστηρός ορισμός, ακολουθεί ένα απλό παράδειγμα της χρήσης τους:

Ας θεωρήσουμε ως μία χρήση αλυσίδας Markov την πρόβλεψη του καιρού, δηλαδή ότι ο τωρινός καιρός σχετίζεται με τον καιρό της επόμενης ημέρας. Αν μαζέψουμε δεδομένα μπορεί να διακρίνουμε ότι η πιθανότητα μία ηλιόλουστη ημέρα να ακολουθήσει μία συννεφιασμένη ημέρα είναι $P(sunny)$, επομένως η πιθανότητα μίας συνεφιασμένης ημέρας μετά από μία επίσης συννεφιασμένης ημέρας είναι $P(rainy) = 1 - P(sunny)$ (για λόγους απλότητας υποθέτουμε ότι τα δεδομένα μας δεν έχουν άλλους παράγοντες). Αυτή η σχετικά απλή λογική μπορεί να χρησιμοποιηθεί για την πρόβλεψη του καιρού των επόμενων ημερών, πάντα με βάση την προηγούμενη κατάσταση.

Ουσιαστικά, μία αλυσίδα Markov είναι ένα σύνολο μεταβάσεων, που αποφασίζονται από μία πιθανοτική κατανομή και ικανοποιούν την ιδιότητα Markov



Σχήμα 3.2: Μία απλή αλυσίδα Markov.

(Markov property), δηλαδή εξαρτώνται μόνο από τη τωρινή κατάσταση και όχι την ακολουθία γεγονότων που προηγήθηκε. Αυτή η συλλογιστική μπορεί να εφαρμοστεί με επιτυχία και σε ακολουθίες λέξεων. Αν X_1, X_2, X_3 είναι τυχαίες μεταβλητές που ικανοποιούν την ιδιότητα Markov, τότε συμβολίζεται μαθηματικά ως:

$$Pr(X_{n+1} = x \mid X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n) = \Pr(X_{n+1} = x \mid X_n = x_n)$$

αρκεί οι δεσμευμένες πιθανότητες να είναι σαφώς καθορισμένες:

$$Pr(X_1 = x_1, \ldots, X_n = x_n) > 0$$

Οι αλυσίδες Markov όταν εφαρμόζονται σε ένα κείμενο μπορούν να παρέχουν ποσοστά αλληλοεπιδράσεων μίας λέξης με κάποια άλλη και στη συνέχεια να προβλέψουν με αξιοπρεπές ποσοστό επιτυχίας μετά την λέξη x ποια άλλη λέξη μπορεί να ακολουθήσει.

Ωστόσο, αναφέρθηκε παραπάνω ότι το χαρακτηριστικό που διαφοροποιεί τις αλυσίδες Markov είναι ότι δεν έχουν μνήμη των προηγούμενων καταστάσεων. Αυτό τις εμποδίζει από την παραγωγή νοηματικού περιεχομένου το οποίο βασίζεται σε συμφραζόμενα και λοιπές συνθήκες. [54, 55]

Σχήμα 3.3: Παράδειγμα γεννήτριας κειμένου με χρήση αλυσίδας Markov. Πηγή δεδομένων: https://www.soliantconsulting.com/blog/title-generator-using-markov-chains/

3.3 Λανθάνουσα Σημασιολογική Ανάλυση (LSA)

Η Λανθάνουσα Σημασιολογική Ανάλυση *(Latent Semantic Analysis, LSA)* είναι μία τεχνική στην επεξεργασία φυσικής γλώσσας για εξαγωγή και αναπαράσταση των συγκείμενων εννοιών *(contextual meanings)* των λέξεων με στατιστικούς υπολογισμούς που εφαρμόζονται σε σώματα κειμένων *(corpora, εν. corpus)* μεγάλου μεγέθους. Η βασική ιδέα είναι ότι λέξεις με "γειτονικό" νόημα θα εμφανίζονται σε παρόμοια δείγματα κειμένου ή αντιστοίχως η απουσία τους από κείμενα μπορεί να υποδηλώνει ομοιότητα. [21]

3.3.1 Λειτουργία

Η LSA ξεκινάει με πίνακα όρων-εγγράφων *(term document matrix)* που αποτελείται από γραμμές οι οποίες χρησιμοποιούνται για μοναδικές λέξεις και στήλες οι οποίες αντιστοιχούν σε ένα έγγραφο. Επομένως η καταχώριση είναι η συχνότητα με την οποία η λέξη της γραμμής εμφανίζεται στο έγγραφο της στήλης.

Αντί για απλή συχνότητα όρων μπορεί να χρησιμοποιηθεί ένα μετασχηματισμός tf-idf *(term frequency–inverse document frequency)* που αντιστοιχεί στη συχνότητα ενός όρου μέσα σε ένα έγγραφο, διαιρούμενο από τη συχνότητα στο συνολικό σώμα κειμένων.

Συχνότητα όρων: $tf(t,d) = f_{t,d}$ όπου $f_{t,d}$ είναι ο μετρητής του όρου στο κείμενο.

Υπάρχουν και παραλλαγές, όπως:

$$tf(t,d) = 0.5 + 0.5 \cdot \frac{f_{t,d}}{\max\{f_{t',d} : t' \in d'\}}$$

όπου διαιρούμε τη συχνότητα του όρου με τη συχνότητα του πιο κοινού όρου του εγγράφου (για να αποφευχθεί η πόλωση προς μεγαλύτερα σε έκταση κείμενα).

Αντίστροφη συχνότητα όρων:

$$idf(t,D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

όπου N ο συνολικός αριθμός των εγγράφων στο σώμα κειμένων $N = |D|$ και $|\{d \in D : t \in d\}|$ ο αριθμός των εγγράφων στα οποία εμφανίζεται ο όρος t.

Επομένως η συχνότητα όρων - αντίστροφη συχνότητα εγγράφων (tf-idf) ισούται με:

$$tfidf(t, d, D) = tf(t, d) \cdot idf(t, D)$$

Επομένως οι όροι που είναι συχνοί (πχ. συνηθισμένες λέξεις όπως "the", "a", "it" στα αγγλικά ή "ο", "και", "οι" στα ελληνικά και παρόμοιοι αντίστοιχοι) συνεισφέρουν λιγότερο στο έγγραφο από τους όρους που είναι σχετικοί / ειδικευμένοι με το θέμα του εγγράφου.

Ανάλυση πίνακα σε ιδιάζουσες τιμές *(Singular Value Decomposition, SVD)*: Ας υποθέσουμε ότι έχουμε έναν πίνακα A μεγέθους m × n. Η SVD είναι μία μέθοδος που μας επιτρέπει να παραγοντοποιήσουμε τον A σε:

- ορθογώνιο πίνακα U m × n
- διαγώνιο πίνακα S n × n
- ορθογώνιο πίνακα V n × n

Η συγκεκριμένη διαδικασία διευκολύνει σημαντικά την εξέταση πολύπλοκων προβλημάτων, καθώς μέσω της διάσπασης σε διαφορετικές μήτρες μπορούν να εξαληφθούν ιδιάζουσες τιμές οι οποίες είναι οι λιγότερο σημαντικές και να απλοποιηθούν τα δεδομένα. Οι χρήσεις της SVD βρίσκονται διάσπαρτες σε επιστημονικούς τομείς, από τη συμπίεση δεδομένων μέχρι την επίλυση γραμμικών εξισώσεων και είναι η βάση για την ανάλυση σε βασικές συνιστώσες *(principal component analysis, PCA)*. [22]

Αν παραγοντοποιήσουμε τον πίνακα με SVD λαμβάνουμε 3 πίνακες: U, D, V.

$$A = UDV^T$$

Όπως αναφέραμε, οι στήλες του U και του V είναι ορθοκανονικές και ο πίνακας D είναι διαγώνιος με θετικές εγγραφές, οι αποκαλούμενες ιδιάζουσες τιμές *(singular values)*. Οι τιμές αξιολογούν τις διαστάσεις στον U, επιτρέποντας έτσι να ανακαλύψουμε τις διαστάσεις στις οποίες αποτυπώνεται η κυριότερη διακύμανση στο χώρο των χαρακτηριστικών από το tf-idf. Αυτό σημαίνει ότι τα διανύσματα των λέξεων που επιδεικνύουν παρόμοια μεταβολή σε έγγραφα καταλήγουν να είναι κοντά το ένα με το άλλο στον καινούργιο αυτό χώρο διανυσμάτων, ο οποίος ονομάζεται χώρος θεμάτων *(topic space)*. [61]

Σε τεχνικές όπως η LSA χρησιμοποιείται συχνά το μοντέλο Bag-of-Words (BoW), το οποίο αναπαριστά ένα έγγραφο απλά μετρώντας πόσες φορές μία λέξη από το λεξιλόγιο εμφανίζεται σε αυτό. Στην *ενότητα 5* γίνεται λεπτομερής αναφορά του μοντέλου. Μειονέκτημα των μεθόδων που βασίζονται στο Bag-of-Words είναι ότι η συνεμφάνιση λέξεων σε έγγραφα παράγει ένα μάλλον "ρηχό" είδος θεματικής ομοιότητας. Για παράδειγμα, σε μία αξιολόγηση λέξεων σε σώμα κειμένων προερχόμενο από την αγγλική Wikipedia *(Πίνακας 3.1)*, το BoW στη λέξη "florida" αντιστοιχεί τοποθεσίες της συγκεκριμένης πολιτείας, στη λέξη "hogwarts" λέξεις από το σύμπαν του Harry Potter και στη λέξη "turing" επίθετα που περιγράφουν τα έργα του. [23] Είναι φανερό ότι με τον τρόπο αυτό λαμβάνουμε αποτελέσματα που αντικατοπτρίζουν τον τομέα της λέξης, αντί για την πιο αμυδρή σημασιολογία της. Ο Turney αναφέρει τη διαφοροποίηση αυτή σαν *θεματική ομοιότητα (domain similarity)* έναντι της *συναρτησιακής ομοιότητας (functional similarity)* [24] Εναλλακτικά αποτελέσματα θα ήταν ονομασίες άλλων πολιτειών, σχολείων και γνωστών ατόμων της πληροφορικής, αντίστοιχα για τις 3 αναφερόμενες λέξεις.

Target word	BoW	DEPS
hogwarts	dumbledore hallows half-blood malfoy snape	sunnydale collinwood calarts greendale millfield
turing	nondeterministic non-deterministic computability deterministic finite-state	pauling hotelling heting lessing hamming
florida	gainesville fla jacksonville tampa lauderdale	texas louisiana georgia california carolina

Πίνακας 3.1: Σύγκριση BoW με DEPS *(dependency-based syntactic contexts)*

4 Ενσωμάτωση λέξεων (word embedding)

Ως ενσωμάτωση λέξεων *(word embedding)* ορίζεται ένα σύνολο τεχνικών και μεθόδων για επεξεργασία φυσικής γλώσσας που αντιστοιχούν λέξεις ή φράσεις από το λεξιλόγιο σε διανύσματα πραγματικών αριθμών ώστε αυτά: [26]

1. να εμπεριέχουν το νόημα τους
2. να επιτρέπουν τον υπολογισμό βαθμολογίας ομοιότητας για ένα ζευγάρι λέξεων. Η βαθμολογία ομοιότητας είναι δεκαδική τιμή ανάμεσα στο -1.0 και 1.0, με υψηλότερες τιμές να υποδηλώνουν μεγαλύτερη ομοιότητα.

Οι ενσωματώσεις λέξεων σχετίζονται εννοιολογικά με τον μαθηματικό όρο της ενσωμάτωσης από ένα χώρο με πολλές διαστάσεις ανά λέξη σε ένα συνεχή χώρο διανυσμάτων με σημαντικά μικρότερη διάσταση. Λέγονται και διανύσματα λέξεων *(word vectors)*.

4.1 Ιστορικό

Οι ενσωματώσεις λέξεων βασίζονται στην ιδέα ότι η συγκείμενη πληροφορία είναι αρκετή για να γίνει εφικτή μία αναπαράσταση των γλωσσικών αντικειμένων, σε αντίθεση με τη λογική της τυπικής γλωσσολογίας και την παράδοση του Chomsky. Οι πρώτες προσπάθειες χρήσης αναπαράστασης χαρακτηριστικών για σημασιολογική ομοιότητα ήταν με *χειροποίητα* χαρακτηριστικά, χωρίς να γίνεται χρήση κάποιου αλγόριθμου. Ένα παράδειγμα είναι η σημασιολογική διαφοροποίηση *(semantic differential)* του Charles E. Osgood. Παρόμοιες αναπαραστάσεις χρησιμοποιήθηκαν και σε πρόωρες μελέτες στην τεχνητή νοημοσύνη τη δεκαετία του 1980. [19]

> Τυπική γλωσσολογία *(Formal linguistics)*: Το τμήμα της γλωσσολογίας στο οποίο χρησιμοποιούνται πρακτικές μαθηματικές μέθοδοι για την ανάλυση των φυσικών γλωσσών. Η ιδέα έχει θεωρητικές ρίζες σε μελέτες της δεκαετίας του 1950.

Το 1990 άρχισαν να εμφανίζονται μέθοδοι για χρήση αυτόματα δημιουργούμενων χαρακτηριστικών με βάση τα συμφραζόμενα. Μία από τις επιδραστικές μεθόδους ήταν η LSA,[2] η οποία αποτελεί τον πρόγονο των σημερινών μοντέλων θεμάτων *(topic models)*. Την ίδια περίπου περίοδο υπήρχαν αρκετά διαφορετικά μοντέλα που αναπτύχθηκαν κατά την έρευνα για τεχνητά νευρωνικά δίκτυα, και τα οποία χρησιμοποιούσαν αναπαραστάσεις με βάση τα συμφραζόμενα. Τα πιο

[2] *Ενότητα 3.3*

γνωστά από αυτά είναι οι Αυτο-οργανούμενοι Χάρτες *(Self Organizing Maps, SOM)* και τα Απλά Επαναλαμβανόμενα Δίκτυα *(Simple Recurrent Networks, SRN)*.

Οι μετέπειτα εξελίξεις στον κλάδο αποτελούνται κυρίως από βελτιώσεις στα ήδη αναφερθέντα μοντέλα. Τα μοντέλα θεμάτων είναι βελτιώσεις του LSA με μεθόδους όπως probabilistic LSA (PLSA) και Latent Dirichlet Allocation (LDA). Λειτουργούν διαισθητικά, με βάση την ιδέα ότι τα έγγραφα που αναλύονται περιέχουν περισσότερα από ένα θέματα και οι λέξεις του εγγράφου σχετίζονται με κάποιο από αυτά. [25] Από την άλλη τα νευρωνικά γλωσσικά μοντέλα βασίζονται στις ίδιες εφαρμογές που στόχευαν τα SRN και περιέχουν αρχιτεκτονικές όπως τα Συνελικτικά Νευρωνικά Δίκτυα *(Convolutional Neural Networks, CNN)* και οι Autoencoders.

Δύο συνηθισμένες παρανοήσεις για τις ενσωματώσεις λέξεων είναι: [19]

1. Δεν απαιτούνται βαθιά νευρωνικά δίκτυα για κατασκευή αποτελεσματικών ενσωματώσεων λέξεων. Αντίθετα τα Continuous-Bag-of-Words και Skip-gram του Word2Vec, δύο από τα πιο επιτυχημένα μοντέλα των τελευταίων ετών, είναι ρηχά νευρωνικά δίκτυα.

2. Δεν υπάρχει κάποια ουσιαστική διαφορά ανάμεσα στα σημερινά μοντέλα προγνωστικών νευρωνικών δίκτυων *(predictive neural network)* και count-based κατανεμημένων σημασιολογικών μοντέλων *(distributional semantics models)*. Πρακτικά αποτελούν δύο διαφορετικά μονοπάτια από τα οποία κάποιος καταλήγει στον ίδιο τύπο σημασιολογικού μοντέλου.

> Κατανεμημένη σημασιολογία: βασίζεται στην Κατανεμημένη Υπόθεση *(Distributional Hypothesis)*, που διατυπώνει ότι η ομοιότητα στο νόημα οδηγεί στην ομοιότητα της γλωσσικής κατανομής: λέξεις που συνδέονται σημασιολογικά χρησιμοποιούνται σε παρόμοια γλωσσικά συμφραζόμενα. Η κατανεμημένη σημασιολογία επάγει σημασιολογικές αναπαραστάσεις από τα συμφραζόμενα που παρατηρούνται σε μεγάλα δείγματα γλωσσικών δεδομένων. [20, 30]

4.2 Διανύσματα χαρακτηριστικών

Τα διανύσματα λέξεων αναπαριστούν τις λέξεις με τρόπο ο οποίος κωδικοποιεί το νόημα τους. Ένα διάνυσμα είναι απλά πίνακας από κλάσματα. Κάθε διάνυσμα λέξης περιέχει αρκετές δεκαδικές τιμές οι οποίες μεμονωμένες δεν αντιστοιχούν

σε πληροφορία που μπορεί να ερμηνευτεί διαισθητικά. Αντίθετα, τα διανύσματα λέξεων πρέπει να γίνονται αντιληπτά ως σημεία σε χώρο πολλών διαστάσεων.

Σε συντεταγμένες δισδιάστατου ή τρισδιάστατου χώρου είναι εύκολη η οπτικοποίηση τους, ώστε να γίνει αντιληπτή η απόσταση δύο σημείων, αν είναι κοντά ή μακριά. Επειδή όμως τα διανύσματα αποτελούνται από συντεταγμένες εξαιρετικά πολλών διαστάσεων[3] δεν είναι δυνατόν να οπτικοποιηθούν. Ωστόσο η έννοια της απόστασης ανάμεσα στα σημεία συνεχίζει να ισχύει. [26]

4.3 Βαθμός ομοιότητας

Η ευκλείδεια απόσταση ανάμεσα σε δύο σημεία μπορεί να επεκταθεί για κάθε αριθμό διαστάσεων. Ακολουθεί ο μαθηματικός τύπος, όπου το n ισούται με τον αριθμό των διαστάσεων στη συγκεκριμένη περίπτωση.

$$dist(a, b) = \sqrt{\sum_{i=0}^{n} (a_i - b_i)^2}$$

Ωστόσο τα αποτελέσματα είναι καλύτερα αν χρησιμοποιηθεί συνημιτονοειδής ομοιότητα *(cosine similarity)*. Για δύο διανύσματα A και B, η συνημιτονοειδής ομοιότητα βρίσκεται υπολογίζοντας το εσωτερικό γινόμενο τους και διαιρώντας το με το Ευκλείδιο μήκος τους. [26]

$$similarity_{cos} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$

Το Ευκλείδιο μήκος (ή Ευκλείδια νόρμα) ενός διανύσματος x συμβολίζεται ως $\|x\|$ και υπολογίζεται ως:

$$\|x\| := \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$$

[3] Ενδεικτικά, σε προεκπαιδευμένο μοντέλο του Google News υπάρχουν 300 τιμές για κάθε διάνυσμα λέξης.

5 Word2Vec

King - Man + Woman = Queen

Το Word2vec είναι μία τεχνολογία μοντέλων για ενσωμάτωση λέξεων. Τα μοντέλα αυτά αποτελούνται από νευρωνικά δίκτυα δύο επιπέδων τα οποία έχουν εκπαιδευτεί για να ανακατασκευάσουν τη γλωσσική σύνδεση με συγκείμενες λέξεις *(context words)*.

Δημιουργήθηκε το 2013 από μία ομάδα στη Google καθοδηγούμενη από τον Tomas Mikolov. Το συγκεκριμένο toolkit για ενσωμάτωση λέξεων μπορεί να εκπαιδεύσει μοντέλα χώρου διανυσμάτων πιο γρήγορα από προηγούμενες προσπάθειες και προσεγγίσεις.

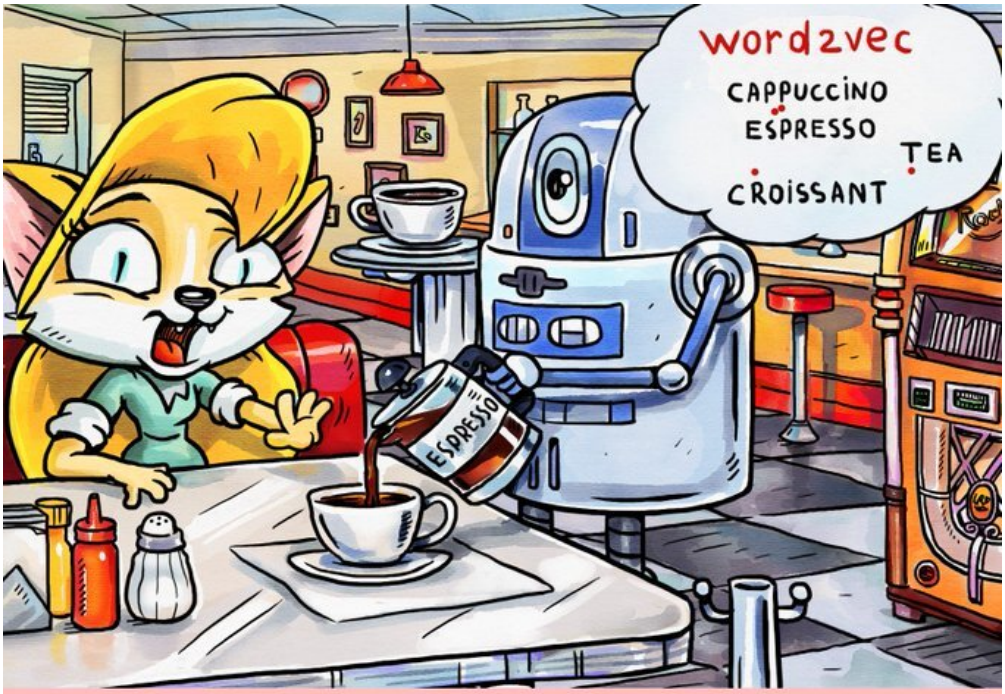
Το Word2vec δέχεται ένα μεγάλο (σε έκταση) σώμα κειμένων και παράγει ένα χώρο διανυσμάτων, συνήθως αρκετών εκατοντάδων διαστάσεων, με κάθε μοναδική λέξη στο σώμα κειμένων να αντιστοιχίζεται με ένα ανάλογο διάνυσμα στο χώρο αυτό. Τα διανύσματα είναι τοποθετημένα έτσι ώστε οι λέξεις που μοιράζονται κάποιο λεκτικό συγκειμενικό πλαίσιο να βρίσκονται κοντά το ένα στο άλλο στο χώρο.

Το Word2Vec δεν είναι ένας μόνο αλγόριθμος αλλά ένας συνδυασμός δύο τεχνικών που μπορούν να χρησιμοποιηθούν για να παράγουν μία κατανεμημένη αναπαράσταση των λέξεων: ένα συνεχές bag of words *(continuous bag of words, CBOW)* και ένα συνεχές skip-gram. Και οι δύο αυτές τεχνικές είναι ρηχά νευρωνικά δίκτυα *(shallow neural networks)* που αντιστοιχίζουν λέξεις με στοχευμένες μεταβλητές που είναι επίσης λέξεις. Χρησιμοποιούν βάρη που δρουν σαν αναπαραστάσεις των διανυσμάτων λέξεων. [27, 28]

5.1 Μοντέλα

5.1.1 Ιστορικό

Το Bag-of-words είναι ένα σχετικά κλασσικό μοντέλο. Οι περισσότερες σύγχρονες πηγές αποδίδουν την πρώτη αναφορά στον Harris, που περιέγραψε τη βασική λογική το 1954, [30] αν και η ιδέα είχε ήδη προταθεί το 1953 από τον Wittgenstein, στο μετά θάνατον δημοσιευμένο *Philosophical Investigations*. [31, 32] Σταδιακά επικράτησε ως η κυρίαρχη μέθοδος για επεξεργασία φυσικής γλώσσας. Τα χαρακτηριστικά που εξάγονται από το BoW τροφοφοτούν στη συνέχεια ταξινομητές, όπως ο Naive Bayes (ο οποίος είναι αρκετά συνηθισμένος, όπως αναφέρθηκε ήδη). Μία μικρή βελτίωση που αναφέρθηκε σε προηγούμενο

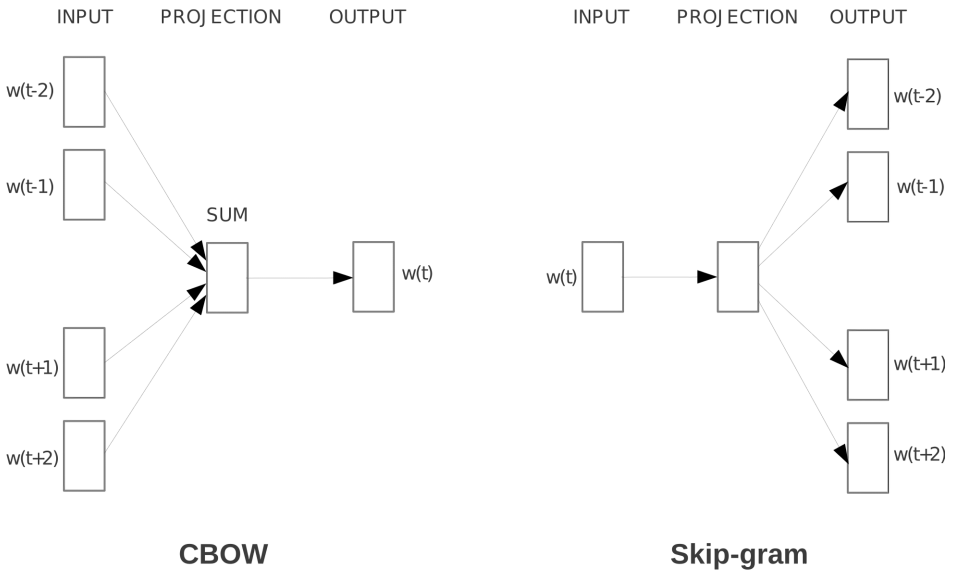


Σχήμα 5.1: Υποθετική χρήση του Word2Vec. [29]

κεφάλαιο ήταν το tf-idf *(term frequency–inverse document frequency)*, το οποίο λαμβάνει υπόψη και τη γενικότερη σπανιότητα ή μη μίας λέξης.

Το Skip-gram αποτελεί γενίκευση των ν-γραμμάτων *(n-gram)* τα οποία είναι μία συνεχόμενη ακολουθία ν τεμαχίων από μία συγκεκριμένη ακολουθία κειμένου,[33] με το skip-gram να εισάγει την ιδέα ότι οι λέξεις στο κείμενο δεν χρειάζεται να είναι συνεχόμενες, αλλά μπορούν να υπάρχουν κενά που παραλείπονται.[34] Σε αντίθεση με το Bag-of-words, άρχισε να χρησιμοποιείται πιο πρόσφατα, με πρώτη περιγραφή της βασικής λογικής το 1993[4] και των ανώτερων αποτελεσμάτων που παρείχε σε σύγκριση με προηγούμενες μεθόδους. 15 χρόνια μετά η χρήση του γενικεύτηκε. [35, 36]

[4] αναφέρεται ως *"long distance bigram"* αντί της τωρινής ονομασίας



Σχήμα 5.2: Τα μοντέλα του Word2Vec. Το CBoW προβλέπει την τωρινή λέξη με βάση τα συμφραζόμενα, ενώ το Skip-gram προβλέπει τις κοντινές λέξεις με βάση την τωρινή. [38]

5.1.2 Που χρησιμεύει το καθένα

Σύμφωνα με τον Mikolov, το καθένα από τα δύο μοντέλα ενδείκνυται για συγκε-κριμένες χρήσεις. Το Skip-gram εμφανίζει αρκετά καλά αποτελέσματα με μικρό όγκο δεδομένων για εκπαίδευση και αναπαριστά με ακρίβεια ακόμα και σπάνιες λέξεις ή φράσεις. Το CBOW είναι αρκετές φορές γρηγορότερο από το skip-gram και παρέχει ελαφρώς καλύτερη ακρίβεια για συχνά εμφανιζόμενες λέξεις. Η σύγκριση γίνεται πιο περίπλοκη αν λάβουμε υπόψη τους διαφορετικούς τρόπους εκπαίδευσης των μοντέλων: με κανονικοποιημένη soft-max ή μη κανονικοποιημένη αρνητική δειγματοληψία, καθώς οι δύο μέθοδοι λειτουργούν με διαφορετικό τρόπο. [37]

5.2 Skip-gram

5.2.1 Διαδικασία για Skip-gram

Στο Word2Vec, για το Skip-gram εκπαιδεύουμε ένα απλό νευρωνικό δίκτυο. Ωστόσο δεν θα χρησιμοποιήσουμε το συγκεκριμένο νευρωνικό δίκτυο για την εργασία στην οποία εκπαιδεύτηκε. Αντίθετα, σκοπός είναι να μάθουμε απλά τα

	CBoW	Skip-gram
Μικρά datasets		√
Ταχύτητα εκπαίδευσης	√	
Σπάνιες λέξεις		√
Συνήθεις λέξεις	√ (ελάχιστα καλύτερο)	
Συντακτική ομοιότητα	√ (ελάχιστα καλύτερο)	

Πίνακας 5.1: Σύγκριση CBoW με Skip-gram, που αποδίδει καλύτερα το κάθε μοντέλο

βάρη του κρυμμένου επιπέδου. Στην πορεία διαπιστώνουμε ότι αυτά τα βάρη είναι τα διανύσματα λέξεων που προσπαθούμε να μάθουμε.

Αν το νευρωνικό δίκτυο λάβει μία λέξη X_1 (input) η οποία βρίσκεται στη μέση κάποιας πρότασης, θα ελέγξει τις υπόλοιπες κοντινές λέξεις και θα επιλέξει τυχαία μία X_2. Ύστερα, θα βρει την πιθανότητα η κάθε λέξη του λεξιλογίου να είναι η X_2 που επιλέξαμε νωρίτερα.

Οι κοντινές λέξεις που θα ελεγχθούν εξαρτώνται από μία παράμετρο που ονομάζεται *μέγεθος παραθύρου (window size)*. Αν το μέγεθος παραθύρου είναι 3, αυτό σημαίνει ότι θα ελεγχθούν οι τρεις λέξεις πριν από την επιλεγμένη και οι τρεις μετά από αυτή.

> Στην αυθεντική υλοποίηση του Word2Vec επιλέγεται ένα τυχαίο μέγεθος παραθύρου, ανάμεσα στο μικρότερο δυνατό μέγεθος *(1)* και το μέγιστο δυνατό *(μεταβλητή `window_size`)*. Λόγω αυτής της αλλαγής διαφέρει το βάρος των συγκείμενων λέξεων ανάλογα με την απόσταση τους από τη μεσαία λέξη. Η τεχνική αυτή αναλύεται λεπτομερώς στην *ενότητα 5.3.6*.

Στο τέλος του κεφαλαίου *(σχήμα 5.10)* παρουσιάζεται ένα παράδειγμα στο οποίο γίνεται φανερό πως λειτουργεί το μέγεθος παραθύρου.

Τα ποσοστά που θα εξαχθούν θα είναι συνδεδεμένα με το πόσο πιθανό είναι να βρεθεί κάθε λέξη του λεξιλογίου κοντά στο input. Για παράδειγμα, αν το input αυτό ήταν η λέξη "Βυζαντινή" τότε υπάρχει μεγαλύτερη πιθανότητα να

συνδέεται με τη λέξη "Αυτοκρατορία", παρά με κάτι άσχετο, όπως "πατάτα" ή "λεοπάρδαλη".

Ακολουθεί ένα απλό παράδειγμα:

Ας υποθέσουμε ότι σαν input υπάρχει ένα λεξιλόγιο 10000 λέξεων. Τα αρχικά δεδομένα θα είναι σε ένα one-hot διάνυσμα 10000 θέσεων.

> One-hot διάνυσμα: Διάνυσμα στο οποίο ένα μοναδικό στοιχείο αναπαρίσταται με 1 (το input) και όλα τα υπόλοιπα με 0.

Αν επιλέξουμε τη λέξη "ants" θα τοποθετήσουμε 1 στη θέση που αυτή βρίσκεται και 0 σε όλες τις υπόλοιπες.

Το αποτέλεσμα του δικτύου στο τελικό επίπεδο θα είναι κι αυτό ένα διάνυσμα με 10000 χαρακτηριστικά και θα περιέχει, για κάθε λέξη του λεξιλογίου, την πιθανότητα μία τυχαία επιλεγμένη κοντινή λέξη να είναι αυτή που αναζητείται.

Το *σχήμα 5.3* παρουσιάζει οπτικά τη συγκεκριμένη διαδικασία.

5.2.2 Κρυμμένο επίπεδο

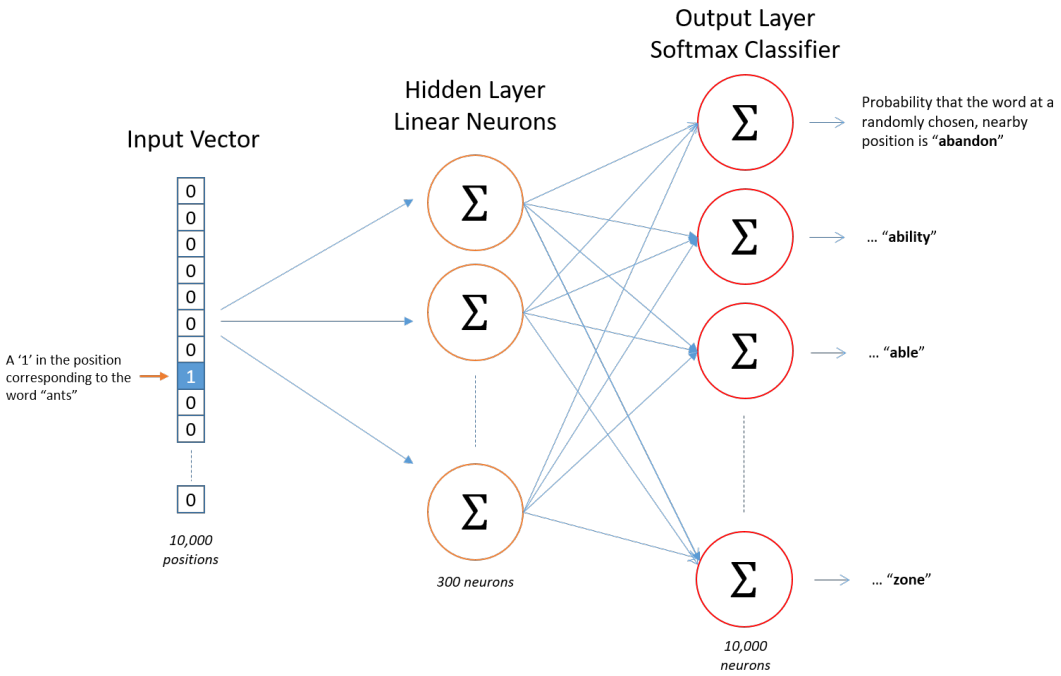
Στο παράδειγμα υποθέτουμε ότι τα διανύσματα λέξεων έχουν 300 χαρακτηριστικά. Επομένως το κρυμμένο επίπεδο θα αποτελείται από έναν πίνακα βαρών με 10.000 γραμμές (μία για κάθε λέξη) και 300 στήλες (μία για κάθε νευρώνα).

Αυτός ακριβώς ο πίνακας είναι που θα χρειαστεί από την διαδικασία, καθώς το τελικό επίπεδο θα απορριφθεί στη συνέχεια.

> 300 νευρώνες χρησιμοποιούνται και σε ένα προεκπαιδευμένο με Word2Vec dataset της Google, με δεδομένα προερχόμενα από το Google News *(https://code.google.com/archive/p/word2vec/)*

Σε αυτό το σημείο πρέπει να θυμήσουμε ότι σχεδόν όλο το αρχικό διάνυσμα είναι γεμάτο με 0. Ως αποτέλεσμα, ο πολλαπλασιασμός [1x10000] με [10000x300] θα έχει ως αποτέλεσμα να επιλεγεί η αντίστοιχη γραμμή του πίνακα που αντιστοιχεί στο 1 *(σχήμα 5.4)*.

Το κρυμμένο επίπεδο του μοντέλου επομένως λειτουργεί σαν πίνακας αναζήτησης, αφού η έξοδος που δίνει είναι απλά το διάνυσμα λέξης για την επιλεγμένη λέξη.



Σχήμα 5.3: Αναπαράσταση του μοντέλου Skip-gram στο Word2Vec, με παρουσίαση του one-hot διανύσματος με τα δεδομένα στο αρχικό επίπεδο και της παλινδρόμησης softmax σαν συνάρτηση ενεργοποίησης στο τελικό. [42]

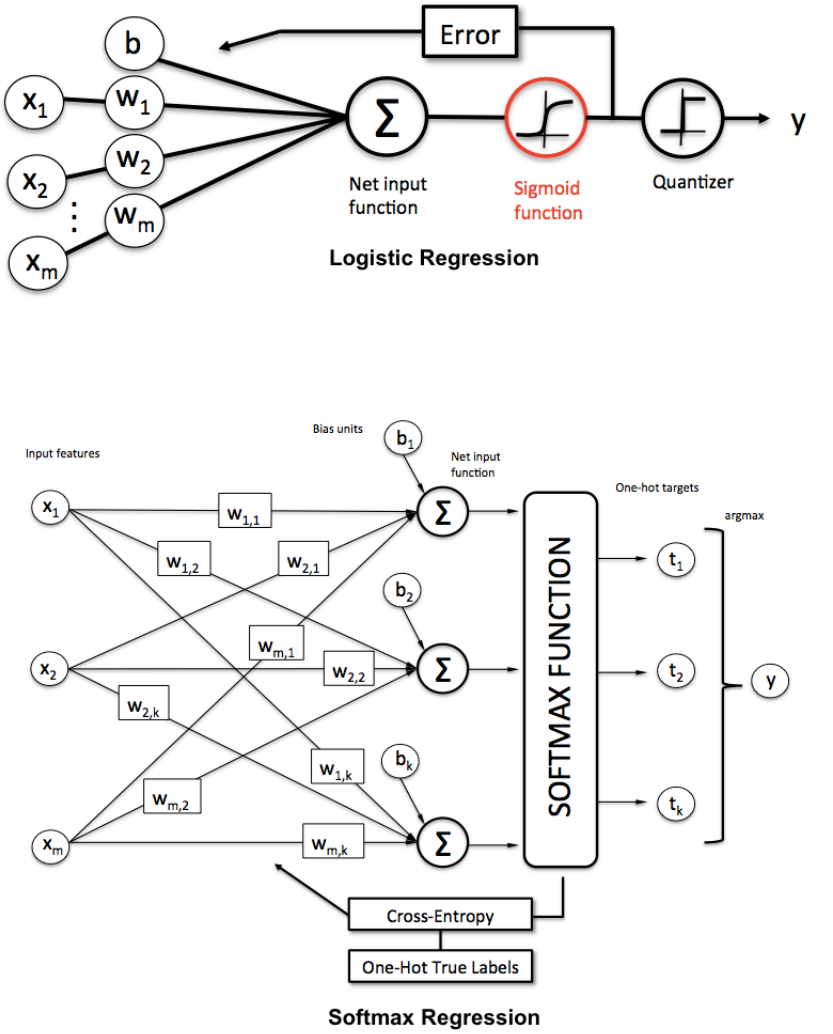
Πίνακας αναζήτησης *(Lookup table)*: Ένας πίνακας που αντικαθιστά μία ακριβή υπολογιστική πράξη με κάποια τιμή η οποία αντιστοιχεί σε αυτή. Απλό παράδειγμα: στις τριγωνομετρικές πράξεις παρέχονται κάποια συνηθισμένα αποτελέσματα για να μην χρειάζεται να υπολογίζονται συνέχεια από το άτομο που επιλύει κάποιον τύπο.

5.2.3 Λειτουργία παλινδρόμησης softmax

Η παλινδρόμηση softmax *(softmax regression)* αποτελεί ένα είδος λογιστικής παλινδρόμησης. Χρησιμοποιείται συχνά σαν συνάρτηση ενεργοποίησης στο επίπεδο εξόδου των νευρωνικών δικτύων. Κανονικοποιεί μία εισηγμένη τιμή σε ένα διάνυσμα τιμών που ακολουθούν μία πιθανοτική κατανομή της οποίας το άθροισμα του συνόλου ισούται με 1. Οι τιμές αυτές έχουν εύρος τιμών [0-1]. Στο παράδειγμα μας, οι νευρώνες εξόδου (ένας ανά λέξη) είναι οι τιμές του διανύσματος τιμών.

$$[0 \quad 0 \quad 0 \quad 1 \quad 0] \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ 10 & 12 & 19 \\ 11 & 18 & 25 \end{bmatrix} = [10 \quad 12 \quad 19]$$

Σχήμα 5.4: Πολλαπλασιασμός one-hot διανύσματος με κρυμμένο επίπεδο λέξεων και χαρακτηριστικών τους. [42]



Σχήμα 5.5: Σύγκριση λογιστικής παλινδρόμησης και παλινδρόμησης softmax. [45]

Λογιστική παλινδρόμηση *(Logistic regression)*: Είναι ένα στατιστικό μοντέλο που στη βασική του μορφή χρησιμοποιεί τη λογιστική συνάρτηση (γνωστή σιγμοειδής καμπύλη) για να μοντελοποιήσει μία δυαδική εξαρτημένη μεταβλητή. Έχει δύο πιθανές τιμές, πχ. κερδίζω / χάνω, περνάω / κόβομαι, ζωντανός / νεκρός.

Συνάρτηση ενεργοποίησης *(activation function)*: Χρησιμοποιείται σε κόμβους νευρωνικών δικτύων και ορίζει την έξοδο αυτών δοθέντος 1 ή παραπάνω εισόδων. Μπορεί να παρομοιαστεί με έναν διακόπτη σε ηλεκτρικό κύκλωμα, που αναλόγως την περίπτωση βρίσκεται σε κατάσταση ON ή OFF.

Ακολουθεί ο τύπος της παλινδρόμησης softmax:

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

όπου [46]:

- $\vec{z}$: Το διάνυσμα εισόδου στη συνάρτηση softmax, αποτελείται από ($z0, ...zK$)
- z_i: Όλες οι τιμές z_i είναι τα στοιχεία του διανύσματος εισόδου στη συνάρτηση softmax και μπορούν να πάρουν οποιαδήποτε πραγματική τιμή, θετική, μηδενική ή αρνητική. Για παράδειγμα, ένα νευρωνικό δίκτυο θα μπορούσε να έχει σαν έξοδο ένα διάνυσμα όπως (-0.62, 8.12, 2.53), το οποίο δεν είναι έγκυρη κατανομή πιθανότητας, συνεπώς το softmax θα ήταν απαραίτητο για κανονικοποίηση των δεδομένων.
- e^{z_i}: Η τυπική εκθετική συνάρτηση εφαρμόζεται σε κάθε στοιχείο του διανύσματος εισόδου. Δίνει θετική τιμή πάνω από 0, η οποία θα είναι πολύ μικρή αν η είσοδος ήταν αρνητική και πολύ μεγάλη αν η είσοδος ήταν μεγάλη. Ωστόσο, εξακολουθεί να μην είναι στο εύρος (0, 1) που απαιτείται στις πιθανότητες.
- $\sum_{j=1}^{K} e^{z_j}$: Ο όρος στο κάτω μέρος του τύπου είναι ο όρος κανονικοποίησης. Διασφαλίζει ότι όλες οι τιμές εξόδου της συνάρτησης θα αθροιστούν στο 1 και κάθε μία θα βρίσκεται στο εύρος (0, 1), αποτελώντας έτσι μια έγκυρη κατανομή πιθανότητας.
- K: Ο αριθμός τάξεων στον ταξινομητή πολλαπλών τάξεων.

Συνεπώς, κάθε νευρώνας εξόδου έχει ένα διάνυσμα βάρους που πολλαπλασιάζεται με το διάνυσμα λέξεων (του κρυμμένου επιπέδου) και έπειτα εφαρμόζεται

μία εκθετική συνάρτηση στο αποτέλεσμα. Τέλος, διαιρούμε το αποτέλεσμα με το άθροισμα των αποτελεσμάτων από τους 10000 κόμβους εξόδου.

Αν δύο διαφορετικές λέξεις έχουν κοντινά γλωσσικά συμφραζόμενα, δηλαδή πολλές γειτονικές λέξεις τους είναι κοινές, τότε το μοντέλο μας πρέπει να εξάγει πολύ κοντινά αποτελέσματα γι' αυτές. Ένας τρόπος για το νευρωνικό δίκτυο να εξάγει παρόμοιες προβλέψεις για τις λέξεις είναι ελέγχοντας αν τα διανύσματα λέξεων έχουν ομοιότητες. Οπότε αν δύο λέξεις έχουν παρόμοια συμφραζόμενα, τότε το δίκτυο έχει "κίνητρο" να αναζητήσει περισσότερα διανύσματα λέξεων γι' αυτές τις δύο λέξεις.

Με βάση τα παραπάνω, αν συγκριθούν δύο λέξεις μπορούμε να καταλάβουμε αν υπάρχει γλωσσική σύνδεση μεταξύ τους. Πχ. οι λέξεις "έξυπνος" και "ευφυής", είναι συνώνυμα με πολύ κοντινά γλωσσικά συμφραζόμενα, άρα το δίκτυο που αναφέρθηκε παραπάνω θα έχει παρόμοια διανύσματα λέξεων. [42]

5.2.4 Υπολογισμός απώλειας

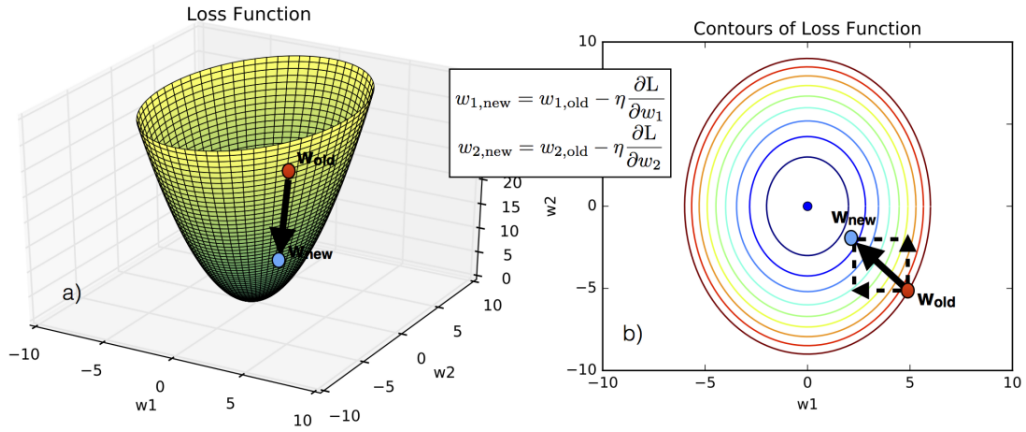
Χάρη στη softmax που εφαρμόστηκε στο προηγούμενο βήμα μπορεί τώρα να υπολογιστεί η απώλεια για ένα συγκεκριμένο σύνολο στοχευμένων και συγκείμενων λέξεων. Η απώλεια είναι ποσοτική τιμή που εκφράζει τη διαφορά μεταξύ της προβλεπόμενης εξόδου και της πραγματικής εξόδου. Η απώλεια είναι αντιστρόφως ανάλογη με την ορθότητα του μοντέλου, επομένως όσο μεγαλύτερη είναι η απώλεια, τόσο λιγότερο ακριβές είναι το μοντέλο.

5.2.5 Οπίσθια τροφοδότηση (Back-propagation)

Σε αυτό το σημείο έχει εκτελεστεί η παραπάνω διαδικασία μία φορά. Ωστόσο για καλύτερα αποτελέσματα και μείωση της απώλειας που βρέθηκε παραπάνω, πρέπει να εκτελεστεί κάποιες φορές ακόμα με σταδιακή ενημέρωση των βαρών και εύρεση πιο ακριβών αποτελεσμάτων. Για να γίνει αυτό θα εκτελεστεί το βήμα της οπίσθιας τροφοδότησης. Ένας συνηθισμένος τρόπος ελαχιστοποίησης του τετραγωνικού σφάλματος της συνάρτησης απώλειας είναι με τη χρήση του αλγορίθμου gradient descent. Έπειτα η ενημέρωση των βαρών προκύπτει από την παραγώγιση του σφάλματος ως προς κάθε συνιστώσα του διανύσματος των βαρών.

Για προβλήματα με δύο μόνο μεταβλητές, είναι εύκολη η οπτικοποίηση της λειτουργίας του gradient descent. Στο *σχήμα 5.6* η πρώτη εικόνα δείχνει μια τρισδιάστατη γραφική παράσταση της λειτουργίας απώλειας ως συνάρτηση των βαρών w_1 και w_2. Αρχικά, δεν γνωρίζουμε ποιες είναι οι βέλτιστες τιμές τους,

δηλαδή δεν γνωρίζουμε ποιες τιμές w_1 και w_2 ελαχιστοποιούν τη λειτουργία απώλειας. Ας υποθέσουμε ότι ξεκινάμε από το κόκκινο σημείο. Εάν γνωρίζουμε πώς διαφέρει η συνάρτηση απώλειας καθώς αλλάζουμε την τιμή των βαρών, δηλαδή αν γνωρίζουμε τις μερικές παραγώγους $\frac{\partial L}{\partial w_1}$ και $\frac{\partial L}{\partial w_2}$, τότε μπορούμε να κινηθούμε από το κόκκινο σημείο σε ένα σημείο πιο κοντά στο ελάχιστο της συνάρτησης απώλειας, η οποία αντιπροσωπεύεται από ένα μπλε σημείο στην εικόνα. Το πόσο προχωρούμε από το σημείο εκκίνησης υπαγορεύεται από μια παράμετρο η που συνήθως ονομάζεται παράμετρος εκμάθησης. [43]



Σχήμα 5.6: Οπτική εξήγηση του αλγορίθμου gradient descent. [43]

Η πρώτη ενημέρωση των βαρών δεν θα είναι και η τελευταία. Αντίθετα, πρέπει να εκτελεστεί η διαδικασία αυτή (εμπρόσθια τροφοδότηση, παλινδρόμηση, υπολογισμός απώλειας και ενημέρωση βαρών) μέχρις ότου η απώλεια να μειωθεί σε επίπεδα που δεν επηρεάζουν σημαντικά τις προβλέψεις.

5.2.6 Υπαρκτά προβλήματα με την παραπάνω διαδικασία

- Για κάθε δείγμα δεν είναι απαραίτητο να ενημερωθούν όλα τα βάρη, μόνο ένα ποσοστό από αυτά. Με την πλήρη ενημέρωση γίνεται σπατάλη υπολογιστικών πόρων χωρίς λόγο.

- Ο υπολογισμός των τελικών πιθανοτήτων με softmax είναι μία υπολογιστικά ακριβή διαδικασία, καθώς περιλαμβάνει άθροισμα των αποτελεσμάτων όλων των λέξεων (πιθανόν αρκετών χιλιάδων / εκατομμυρίων) για κανονικοποίηση.

  Για κάθε δείγμα εκπαίδευσης πραγματοποιούμε μία πράξη για υπολογισμό πιθανοτήτων σε λέξεις των οποίων το βάρος μπορεί να μην ενημερωθεί

καθόλου ή να ενημερωθεί ελάχιστα. Θέλουμε να αποφύγουμε αυτούς τους περιττούς υπολογισμούς.

5.3 Βελτιστοποίηση

Στο προηγούμενο παράδειγμα, είχαμε διανύσματα λέξεων με 300 συνιστώσες και λεξιλόγιο 10000 λέξεων. Χωρίς κάποια βελτιστοποίηση, κατά την εκπαίδευση τα δεδομένα θα έπρεπε να πολλαπλασιαστούν μεταξύ τους. Το αποτέλεσμα θα ήταν το κρυμμένο επίπεδο και το τελικό επίπεδο να έχουν πίνακες βαρών με 3 εκατομμύρια βάρη. [47]

Για προφανείς λόγους, όσο αυξάνονται τα δεδομένα, η εκπαίδευση του νευρωνικού δικτύου γίνεται όλο και πιο δύσκολη. Για να αντιμετωπιστεί αυτό, η ομάδα ερευνητών που δημιούργησε το Word2Vec ακολούθησαν με μία δεύτερη δημοσιευμένη εργασία, στην οποία προτείνουν τρόπους να αντιμετωπιστούν τα αναφερθέντα προβλήματα. [39]

- Με δειγματοληψία να γίνεται επιλογή των πιο συνηθισμένων λέξεων, ώστε να μειωθεί ο αριθμός των δεδομένων προς εκπαίδευση.
- Χρήση της τεχνικής αρνητικής δειγματοληψίας (negative sampling), προκειμένου να ενημερώνεται μόνο ένα μικρό ποσοστό από τα βάρη του μοντέλου και όχι το σύνολο.

5.3.1 Υπο-δειγματοληψία συνηθισμένων λέξεων

Συνηθισμένες λέξεις *(stopwords)* όπως το "το" δημιουργούν προβλήματα:

- ένα ζευγάρι λέξεων όπως "το μήλο" δεν περιέχει πολλές πληροφορίες για τη λέξη "μήλο", καθώς τα άρθρα εμφανίζονται στις περισσότερες λέξεις. Αυτό φυσικά ισχύει και για τα υπόλοιπα ζευγάρια λέξεων τύπου "το, ...".
- Θα έχουμε πολύ περισσότερα δείγματα για τη λέξη "το" από όσα χρειαζόμαστε για να λάβουμε ικανοποιητικά αποτελέσματα για το τι εκφράζει.

Η δειγματοληψία του Word2Vec αντιμετωπίζει το συγκεκριμένο πρόβλημα. Κάθε λέξη που συναντούμε έχει πιθανότητα να διαγραφεί από το κείμενο. Η πιθανότητα αυτή καθορίζεται από τη συχνότητα εμφάνισης της.

Αν έχουμε ένα παράθυρο με μέγεθος 10 (για κάθε λέξη γίνονται συνδυασμοί με άλλες 9 συγκείμενες λέξεις) και αφαιρέσουμε ένα συγκεκριμένο "το" από το κείμενο:

- Εκπαιδεύοντας με τις υπόλοιπες λέξεις, αυτό δεν θα εμφανίζεται
- Θα έχουμε 10 δείγματα λιγότερα σε σχέση με πριν

Επομένως, τα αναφερόμενα προβλήματα θα έχουν αντιμετωπιστεί αποτελεσματικά.

5.3.2 Υπολογισμός πιθανοτήτων για υπο-δειγματοληψία

Για να εκτελεστεί η υπο-δειγματοληψία, χρειάζεται να οριστεί ο τρόπος υπολογισμού της πιθανότητας να παραμείνει μία λέξη στο λεξιλόγιο που θα εκπαιδεύσουμε. Έστω ότι η λέξη απεικονίζεται με w_i, $z(w_i)$ είναι το ποσοστό των συνολικών λέξεων του σώματος κειμένων που ισούνται με τη λέξη αυτή και η πιθανότητα να παραμείνει η λέξη είναι $P(w_i)$. [47] Στον κώδικα της αρχικής υλοποίησης του Word2Vec χρησιμοποιείται η ακόλουθη εξίσωση: [48]

$$P(w_i) = (\sqrt{\frac{z(w_i)}{0.001}} + 1) \cdot \frac{0.001}{z(w_i)}$$

Στην πράξη, η διαδικασία αυτή θα αφαιρέσει έναν μικρό αριθμό λέξεων, οι οποίες όμως εμφανίζονται εξαιρετικά συχνά στα κείμενα, βοηθώντας πολύ στη μείωση του όγκου δεδομένων.

5.3.3 Αρνητική δειγματοληψία (negative sampling)

Αναφέραμε παραπάνω ότι κατά τον κλασσικό τρόπο εκπαίδευσης ενός νευρωνικού δικτύου τα βάρη μεταβάλλονται ώστε να μπορούν να προβλέψουν το δοθέν δείγμα με περισσότερη ακρίβεια. Επομένως για κάθε δείγμα με βάση το οποίο εκπαιδεύεται γίνεται μεταβολή όλων των βαρών.

Λόγω του μεγέθους του λεξιλογίου αυτό σημαίνει ότι το skip-gram νευρωνικό δίκτυο θα έχει τεράστιο αριθμό βαρών που θα πρέπει να ενημερώνονται για κάθε δείγμα που δοκιμάζεται, προφανώς δημιουργώντας πρακτικά υπολογιστικά προβλήματα.

Η αρνητική δειγματοληψία επιλύει το πρόβλημα, μεταβάλλοντας ένα μικρό ποσοστό των βαρών για κάθε δείγμα, αντί για όλα.

Παράδειγμα: το ζευγάρι λέξεων "μήλο, νόστιμο". Έχουμε ένα one-hot διάνυσμα, στο οποίο το σωστό αποτέλεσμα αναπαρίσταται με 1 και τα υπόλοιπα με 0. Με την αρνητική δειγματοληψία, θα διαλέξουμε ένα μικρό αριθμό αρνητικών λέξεων *(negative words)* για τις οποίες θα ενημερώσουμε τα βάρη. Στη συγκεκριμένη

περίπτωση αρνητική λέξη είναι αυτή για την οποία θέλουμε το δίκτυο να παράγει 0. Θα ενημερώσουμε επίσης τα βάρη για τη θετική *(positive)* λέξη ("νόστιμο").

> Σύμφωνα με το άρθρο των Mikolov et al προτείνεται να χρησιμοποιήσουμε 5-20 λέξεις για μικρά datasets και 2-5 λέξεις για μεγαλύτερα. [39]

Το τελικό επίπεδο του μοντέλου έχει έναν πίνακα μεγέθους 300 x 10000. Θα ενημερώσουμε τα βάρη για τη θετική λέξη ("νόστιμο") και για τα βάρη των άλλων 5 λέξεων που θέλουμε να εξάγουν 0. Ως αποτέλεσμα, θα υπάρχουν 6 νευρώνες εξόδου και 1800 τιμές βαρών συνολικά, σημαντικά μικρότερο νούμερο (μόλις το 0.06%) από τα αρχικά 3 εκατομμύρια βάρη.

Στο κρυμμένο επίπεδο μόνο τα βάρη για το input ενημερώνονται, αν και αυτό είναι πάντα αληθές, χωρίς να επηρεάζεται από την αρνητική δειγματοληψία.

Για κάποιο λόγο, η τηλεοθόνη στο λίβινγκ ρουμ ήταν τοποθετημένη ασυνήθιστα. Αντί να είναι, όπως ήταν το κανονικό, στον ακρινό τοίχο, απ' όπου θα μπορούσε να επιθεωρεί όλο το δωμάτιο, βρισκόταν στο μακρύτερο τοίχο, απέναντι απ' το παράθυρο. Στη μια πλευρά του υπήρχε μια μικρή εσοχή όπου καθόταν τώρα ο Γουίνστον, η οποία, όταν χτιζόταν το διαμέρισμα, φαίνεται ότι προοριζόταν για ράφια βιβλιοθήκης. Με το να κάθεται σ' αυτή την εσοχή, και προς τα πίσω, ο Γουίνστον κατάφερνε να βρίσκεται έξω από το οπτικό πεδίο της τηλεοθόνης. Μπορούσαν βέβαια να τον ακούν, αλλά όσο βρισκόταν σ' αυτή τη θέση δεν μπορούσαν να τον δουν. Λίγο αυτή η ασυνήθιστη γεωγραφία του δωματίου, λίγο τούτο το τετράδιο που μόλις είχε βγάλει από το συρτάρι, του είχαν υποβάλει την ιδέα να κάνει ό,τι ετοιμαζόταν να κάνει τώρα.

- στοχευμένη λέξη
- συγκείμενες "θετικές" λέξεις (μέγεθος παραθύρου = 3)
- τυχαία επιλεγμένες "αρνητικές" λέξεις

Σχήμα 5.7: Παράδειγμα αρνητικής δειγματοληψίας

5.3.4 Επιλογή αρνητικών δειγμάτων

Παραμένει το ερώτημα πως μπορούν να επιλεγούν αποτελεσματικά τα αρνητικά δείγματα (στο παράδειγμα οι 5 λέξεις που παράγουν 0). Χρησιμοποιείται μία μονογραμματική κατανομή *(unigram distribution)*, όπου οι συχνά εμφανιζόμενες λέξεις είναι πιο πιθανό να επιλεγούν γι' αυτό το σκοπό [39, 47, 49]. Για παράδειγμα, στην περίπτωση που υπάρχει ολόκληρο το σώμα κειμένων ως λίστα

λέξεων και επιλεχθούν τα 5 αρνητικά δείγματα τυχαία από τη λίστα, η πιθανότητα επιλογής της λέξης "τηλεόραση" θα είναι ίση με τον αριθμό των φορών που εμφανίζεται η "τηλεόραση" στο σώμα, διαιρούμενη με τον συνολικό αριθμό των λέξεων στο σώμα κειμένων. Αυτό εκφράζεται με την ακόλουθη εξίσωση: [47]

$$P(w_i) = \frac{f(w_i)}{\sum_{j=0}^{n} \left(f(w_j) \right)}$$

Προτείνεται για καλύτερα αποτελέσματα ο μετρητής των λέξεων να υψωθεί στη δύναμη 3/4: σε σύγκριση με την απλούστερη επιλογή λέξεων η αλλαγή αυτή έχει την τάση να αυξάνει την πιθανότητα επιλογής λιγότερο συχνών λέξεων αντί για τις πιο συχνές, επομένως αποφεύγεται η πιθανότητα να επιλεγούν άρθρα ή λέξεις χωρίς σημασιολογική σημασία. Έτσι ο προηγούμενος τύπος παίρνει την ακόλουθη μορφή: [47]

$$P(w_i) = \frac{f(w_i)^{3/4}}{\sum_{j=0}^{n} \left(f(w_j)^{3/4} \right)}.$$

5.3.5 Ιεραρχική softmax

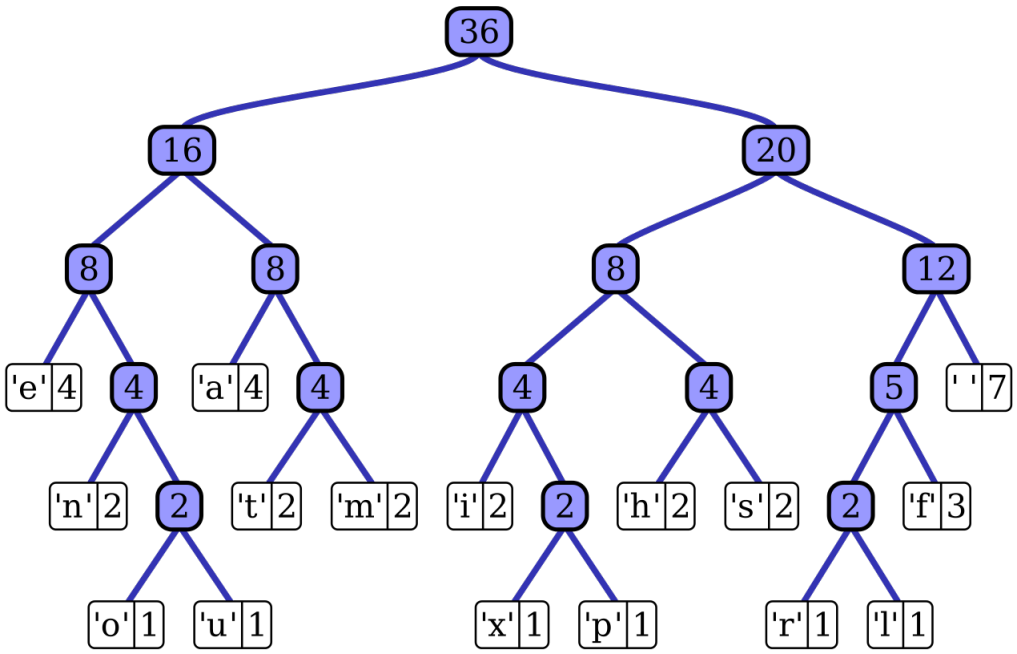
Η ιεραρχική softmax *(hierarchical softmax)* είναι μία υπολογιστικά αποδοτική προσέγγιση της πλήρους softmax. [39] Στο πλαίσιο των γλωσσικών μοντέλων νευρωνικών δικτύων παρουσιάστηκε πρώτη φορά το 2005, από τους Morin και Bengio. [51] Πρακτικά, στο Word2Vec χρησιμοποιείται ως τεχνική εναλλακτική της αρνητικής δειγματοληψίας. [38, 39] Και οι δύο είναι μέθοδοι για μείωση του κόστους υπολογισμού με επιλογή μόνο ένος μικρού αριθμού λέξεων στο τελικό επίπεδο.

Η αρνητική δειγματοληψία αποφασίζει έναν δεδομένο αριθμό αρνητικών λέξεων οι οποίες θα χρησιμοποιηθούν για κάθε είσοδο προς εκπαίδευση. Αυτές διαλέγονται τυχαία από το λεξιλόγιο, αλλά με πιθανότητα που σχετίζεται με τη συχνότητα της λέξης.

Η ιεραρχική softmax διαφέρει σε δύο σημεία:

1. Ο αριθμός των αρνητικών δειγμάτων ποικίλει.
2. Οι λέξεις που χρησιμοποιούνται σαν αρνητικές ορίζονται από πριν για κάθε λέξη του λεξιλογίου.

Το Word2Vec οργανώνει τις λέξεις του λεξιλογίου σε ένα δυαδικό δέντρο με χρήση κωδικοποίησης Huffman. [38, 39] Με τον τρόπο αυτό, το δέντρο που



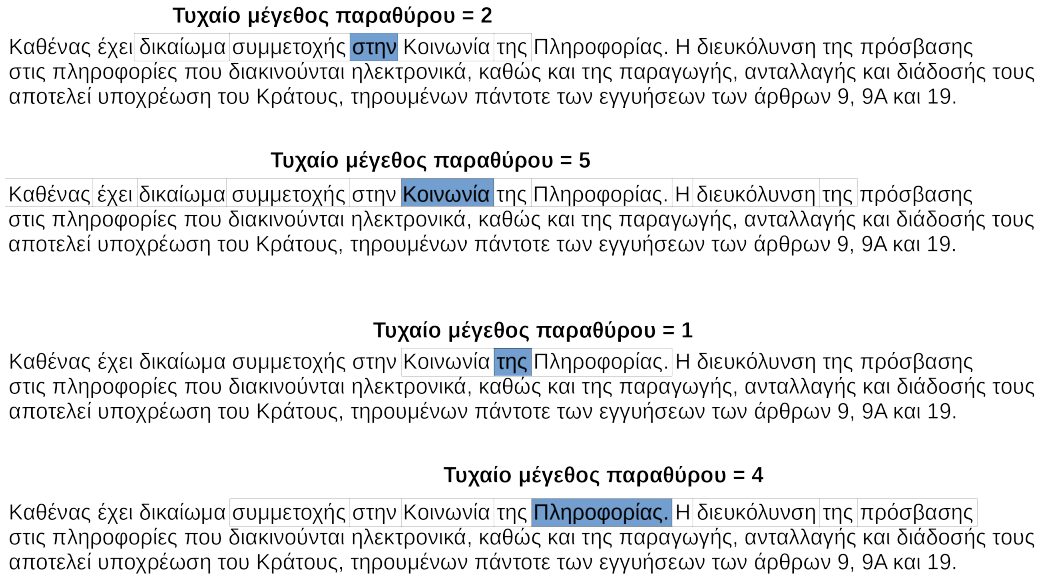
Σχήμα 5.8: Ένα απλό δέντρο Huffman για τη φράση "this is an example of a huffman tree". Οι χαρακτήρες που εμφανίζονται συχνότερα είναι πιο κοντά στη ρίζα. [52]

προκύπτει περιέχει σύντομα μονοπάτια για τις συνήθεις λέξεις και μακρύτερα για τις σπανιότερες. Ως αποτέλεσμα, οι συνήθεις λέξεις έχουν λιγότερα αρνητικά δείγματα, μειώνοντας το κόστος εκπαίδευσης.

5.3.6 Στάθμιση με βάση τη θέση των συμφραζομένων

Όπως αναφέρθηκε ήδη, η αυθεντική υλοποίηση του Word2Vec σταθμίζει αποτελεσματικά τις συγκείμενες λέξεις ανάλογα με τη θέση τους μέσα στο παράθυρο περιβάλλοντος. Οι συγκείμενες λέξεις που βρίσκονται πιο κοντά στην κεντρική λέξη έχουν περισσότερο βάρος από τις λέξεις που βρίσκονται πιο μακριά. Η στάθμιση αυτή επιτυγχάνεται μειώνοντας τυχαία το μέγεθος του παραθύρου συμφραζομένων.

Στον κώδικα της αρχικής υλοποίησης του Word2Vec (στη γλώσσα προγραμματισμού C), κάθε φορά που το παράθυρο συμφραζομένων μετακινείται στην επόμενη λέξη σε μια πρόταση, επιλέγεται ένα τυχαίο μέγεθος παραθύρου στο εύρος τιμών [1, window_size]. Για παράδειγμα, αν το word2vec εκπαιδευτεί με μέγεθος παραθύρου 5, τότε το πραγματικό μέγεθος παραθύρου θα ποικίλλει ομοιόμορφα μεταξύ 1 και 5 *(σχήμα 5.9)*.



Σχήμα 5.9: Παράδειγμα παραθύρων τυχαίου μεγέθους.

Δεδομένου ότι όλα τα μεγέθη παραθύρων στο εύρος αυτό είναι εξίσου πιθανά, κάθε θέση περιβάλλοντος έχει αναλογική πιθανότητα να συμπεριληφθεί. Ωστόσο οι λέξεις που είναι αμέσως πριν και μετά από τη στοχευμένη περιλαμβάνονται πάντα, σε αντίθεση με άλλες που βρίσκονται πιο μακριά.

5.3.7 Ζευγάρια λέξεων - φράσεις

Μία ακόμα άξια αναφοράς βελτιστοποίηση στο Word2Vec είναι η συνένωση φράσεων από μεμονωμένες λέξεις. Για παράδειγμα, το "New York Times" έχει διαφορετική σημασία από τις τρεις λέξεις ξεχωριστά. Επόμενως όταν εμφανίζεται στο κείμενο αντιμετωπίζεται σαν μία ενιαία λέξη με δική της αναπαράσταση σε διάνυσμα.

Για την εύρεση φράσεων χρησιμοποιείται ο ακόλουθος μαθηματικός τύπος:

$$score(w_i, w_j) = \frac{count(w_i \cdot w_j) - \delta}{count(w_i) \times count(w_j)}$$

Το δ είναι σταθερά που εμποδίζει τον σχηματισμό πολλών φράσεων από ασυνήθιστες λέξεις. Τα διγράμματα *(bigrams)* με βαθμολογία μεγαλύτερη του

κατωφλίου χρησιμοποιούνται σαν φράσεις. Κατά κανόνα, η εκτέλεση 2-4 περασμάτων στα δεδομένα εκπαίδευσης με σταδιακά μικρότερη τιμή κατωφλίου επιτρέπει το σχηματισμό μεγαλύτερων φράσεων που αποτελούνται από αρκετές λέξεις (όπως οι "New York Times" που προαναφέρθηκαν).

5.4 Συνεχές Bag-of-words

Το μοντέλο bag-of-words είναι μία απλοποιημένη αναπαράσταση που χρησιμοποιείται στην επεξεργασία φυσικής γλώσσας αλλά και στην ανάκτηση πληροφοριών.

> Πολυσύνολο *(multiset)*: Είναι παραλλαγή της έννοιας του συνόλου *(set)*, με βασική διαφορά ότι επιτρέπει πολλαπλές αναφορές κάθε στοιχείου μέσα σε αυτό.
>
> Multiplicity: μετρητής λέξεων, πόσες φορές εμφανίζονται στο κείμενο

Στο μοντέλο αυτό το κείμενο (για παράδειγμα μία πρόταση ή ένα έγγραφο) αναπαρίσταται σαν ένα πολυσύνολο των λέξεων του, χωρίς να δίνεται σημασία στη γραμματική ή τη σειρά των λέξεων αλλά κρατώντας έναν μετρητή λέξεων.

5.4.1 Διαδικασία για CBOW

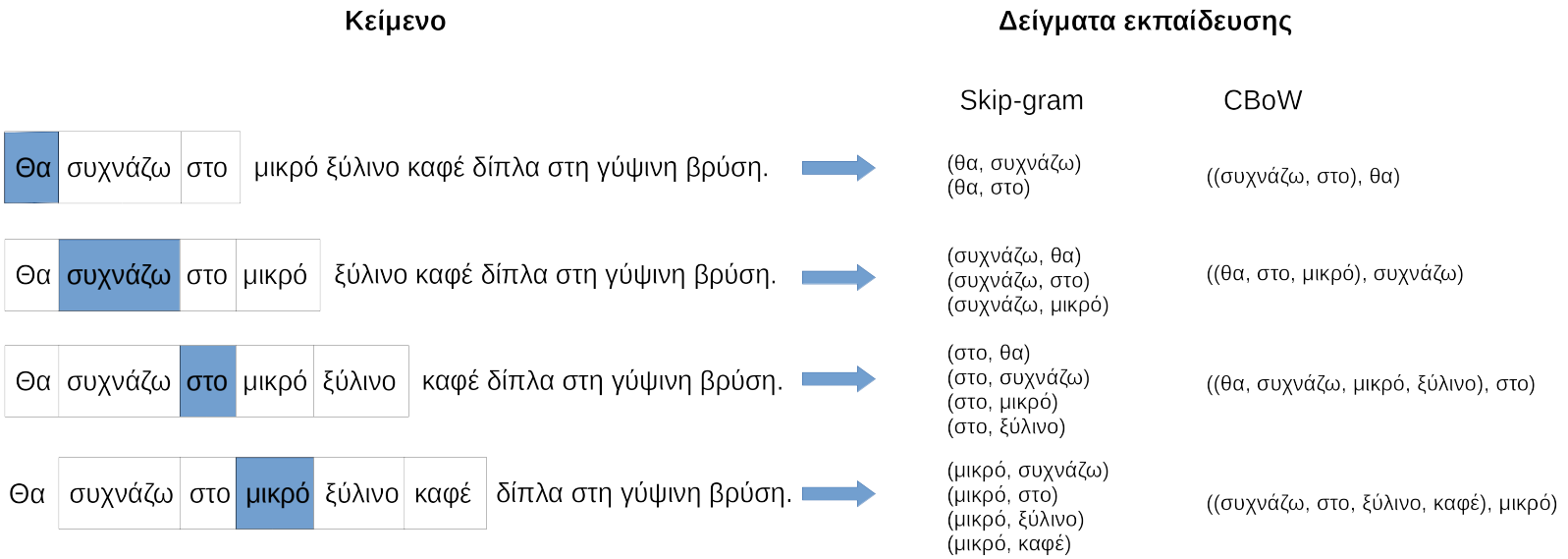
Στο CBOW, όπως και στο Skip-gram, παίρνουμε ζευγάρια λέξεων και εκπαιδεύουμε το μοντέλο αν εμφανίζονται συχνά μαζί. Η διαφορά είναι ότι η λειτουργία του είναι πρακτικά η αντίθετη του Skip-gram *(Σχήμα 5.2, σελ. 38)*, δηλαδή αντί σαν είσοδο να υπάρχει μία στοχευμένη λέξη *(target word)* και να αναζητούνται οι συγκείμενες της, ξεκινάει από συμφραζόμενα λέξεων και προσπαθεί να προβλέψει μία στοχευμένη λέξη σχετική με αυτές.

Οι διαστάσεις του κρυμμένου επιπέδου και του τελικού επιπέδου παραμένουν ίδιες. Θα αλλάξει μόνο η διάσταση του αρχικού επιπέδου και ο υπολογισμός του κρυμμένου επιπέδου. Έστω ότι V είναι ο αριθμός των λέξεων του λεξιλογίου και E το μέγεθος του διανύσματος λέξης. Αν έχουμε 4 συγκείμενες λέξεις για μία στοχευμένη λέξη τότε θα υπάρχουν 4 διανύσματα εισόδου διαστάσεων $1 \times V$. Αυτά θα πολλαπλασιαστούν με το κρυμμένο επίπεδο $V \times E$ και θα επιστραφούν διανύσματα διαστάσεων $1 \times E$. Από τα 4 διανύσματα διαστάσεων $1 \times E$ θα υπολογιστεί μία μέση τιμή για κάθε στοιχείο ώστε να αποκτηθεί η τελική ενεργοποίηση για το επίπεδο της softmax. [27]

Στο *Σχήμα 5.3 (σελ. 41)* παρουσιάστηκε η λειτουργία του Word2Vec με χρήση Skip-gram. Η μόνη διαφορά στο Word2Vec είναι πως το αρχικό one-hot

διάνυσμα (με μοναδική τιμή 1 και 0 στις υπόλοιπες θέσεις), θα αντικατασταθεί με ένα διάνυσμα bag-of-words, στο οποίο πολλαπλές θέσεις περιέχουν την τιμή 1 (όσες αντιστοιχούν σε συγκείμενες λέξεις).

Στο *Σχήμα 5.10 (σελ. 54)* εμφανίζονται οι διαφορές ανάμεσα στις δύο αρχιτεκτονικές κατά τον υπολογισμό των δειγμάτων εκπαίδευσης. Με παράθυρο μεγέθους δύο θέσεων, το Skip-gram μπορεί να παράξει μέχρι και τέσσερα δείγματα ανά λέξη, ενώ το CBoW δημιουργεί ένα.



Σχήμα 5.10: Δείγμα εκπαίδευσης με Skip-gram και CBoW. Με παράθυρο μεγέθους 2 θέσεων, το Skip-gram μπορεί να παράξει μέχρι και 4 δείγματα ανά λέξη, ενώ το CBoW δημιουργεί ένα.

6 Εφαρμογή του Word2Vec μέσω Wikipedia2Vec

Το Wikipedia2Vec είναι μία εφαρμογή του Word2Vec πάνω σε σώμα κειμένων προερχόμενο από τη Wikipedia. [58] Εκπαιδεύοντας σε κάποιο επιλεγμένο dataset γίνεται φανερή η χρησιμότητα του Word2Vec στην εύρεση παρεμφερών λέξεων.

Εκπαιδεύοντας πάνω στο σύνολο της ελληνικής (`elwiki-20200501-pages-articles.xml.bz2`) και της ιταλικής Wikipedia (`itwiki-20200501-pages-articles.xml.bz2`) είναι δυνατόν στη συνέχεια να δοκιμαστούν συγκεκριμένοι ορισμοί ώστε να εμφανιστούν οι συγκείμενες έννοιες.

Όπως γίνεται φανερό, το διάνυσμα λέξεων για μία επιλεγμένη λέξη δεν φαίνεται να δίνει προφανείς πληροφορίες:

```
print(wiki2vec.get_word_vector('άνθρωπος'))
```

```
[-4.37613070e-01 -7.93129385e-01  1.32850826e-01 -6.05473459e-01
  4.87971544e-01  3.36074620e-01 -1.99092910e-01 -2.26280496e-01
  2.60736374e-03 -1.71225488e-01  3.83156776e-01  5.75389452e-02
  4.22033280e-01  2.85893708e-01 -3.88226569e-01 -2.64715999e-01
  3.89433801e-01  4.81684774e-01 -2.66082913e-01  2.50984401e-01
 -7.64248520e-03 -4.14747357e-01 -5.01581669e-01 -1.99527919e-01
 -4.83685434e-01  1.24407746e-01  1.73290566e-01  1.68370366e-01
 -1.92520872e-01  8.39709878e-01 -1.14041425e-01  5.92677832e-01
 -4.62448955e-01  9.96714175e-01 -2.89678425e-02 -3.08448523e-01
 -1.36048704e-01  3.46713930e-01 -3.03338498e-01 -9.25918866e-04
 -3.75726223e-01 -2.45534763e-01  2.62487888e-01 -2.33224273e-01
  4.59548980e-01  1.59411699e-01 -1.05576515e-01 -3.38971585e-01
  2.55468227e-02 -7.70277381e-01  2.82264590e-01 -8.50640312e-02
  2.17919737e-01 -6.26160026e-01 -1.68440387e-01 -1.64014444e-01
 -3.61233205e-01  2.25079671e-01 -3.46617494e-03 -5.39516568e-01
 -4.79966909e-01  2.67220974e-01 -2.40704641e-01  1.75751507e-01
 -1.01509072e-01 -6.81954861e-01  1.89982325e-01  7.34624267e-02
  9.80500802e-02  1.31639987e-02  3.05175573e-01 -3.76395613e-01
 -1.96448669e-01  8.82621631e-02  4.80996013e-01  1.99847624e-01
  2.13445634e-01 -3.07612538e-01 -1.29667521e-01  2.51591176e-01
  4.28921282e-02  2.11317390e-01 -4.92740154e-01  6.79963827e-02
  1.83990046e-01  3.97859693e-01  5.46263039e-01  1.96152538e-01
 -1.53201759e-01 -1.19557500e-01  1.36978388e-01 -3.26302826e-01
```

```
-6.25194788e-01 -2.68188059e-01  1.52750745e-01  4.43351477e-01
-5.20339906e-01 -3.02035175e-02  2.34596714e-01 -3.82158160e-01]
```

Ωστόσο αν ζητηθούν οι πιο κοντινές λέξεις σε μία συγκεκριμένη η χρησιμότητα αυξάνεται:

```
print(wiki2vec.most_similar(wiki2vec.get_entity('Λάμπρος Κωνσταντάρας'), 5))
print(wiki2vec.most_similar(wiki2vec.get_entity('Θανάσης Βέγγος'), 5))
```

Τα αποτελέσματα του Πίνακα 6.1 δείχνουν ότι 2 ηθοποιοί με παρόμοιο στυλ κωμωδίας (Βέγγος - Χατζηχρήστος) έχουν περισσότερα κοινά στοιχεία από άλλα ζευγάρια ηθοποιών.

Δοσμένη λέξη (βαθμολογία)	**Κοντινές λέξεις (βαθμολογία)**
<Entity Λάμπρος Κωνσταντάρας> (1.0000001)	<Entity Κώστας Χατζηχρήστος> (0.79891473) <Entity Σταύρος Παράβας> (0.7983009) <Entity Γιώργος Φούντας> (0.79027617) <Entity Δημήτρης Χορν> (0.7889478)
<Entity Θανάσης Βέγγος> (1.0000001)	<Entity Κώστας Χατζηχρήστος> (0.8145666) <Entity Ναπολέων Ελευθερίου> (0.7924234) <Entity Νίκος Σταυρίδης> (0.7915481) <Entity Χρήστος Γιαννακόπουλος> (0.7912624)

Πίνακας 6.1: Εκπαίδευση του Wikipedia2Vec πάνω στα δεδομένα της ελληνικής Wikipedia, εύρεση κοντινών λέξεων

Παρομοίως, αν επαναληφθεί το πείραμα σε δεδομένα της ιταλικής Wikipedia (Πίνακας 6.2), εμφανίζεται ότι ο ζητούμενος ηθοποιός έχει σαν κοντινά στοιχεία συν-ηθοποιούς από ταινίες στις οποίες είχαν κοινούς ρόλους.

Μάλιστα στη συγκεκριμένη περίπτωση γίνεται εμφανής η αναγνώριση ονομάτων οντοτήτων που έχει υλοποιηθεί στο Wikipedia2Vec, καθώς υπάρχει διαφορά ανάμεσα σε απλές λέξεις και πραγματικά φυσικά πρόσωπα.

Δοσμένη λέξη (βαθμολογία)	**Κοντινές λέξεις (βαθμολογία)**
<Entity Toni Servillo> (0.99999994)	<Word servillo> (0.7994142) <Entity Donatella Finocchiaro> (0.75151604) <Entity Mario Martone> (0.74370784) <Entity Antonio Neiwiller> (0.7381443)
<Word uomo> (1.0)	<Word assassino> (0.7671828) <Word individuo> (0.7649235) <Word indagatore> (0.72888863) <Word ragazzo> (0.72473836)

Πίνακας 6.2: Εκπαίδευση του Wikipedia2Vec πάνω στα δεδομένα της ιταλικής Wikipedia, εύρεση κοντινών λέξεων

7 Εναλλακτικοί αλγόριθμοι ενσωμάτωσης λέξεων

Στη συνέχεια παρουσιάζονται κάποια ακόμα μοντέλα που παρέχουν βελτιώσεις στη διαδικασία που εισήγαγε το Word2Vec.

7.1 GloVe

Το GloVe (από Global Vectors) είναι μοντέλο για αποκεντρωμένη απεικόνιση λέξεων. Προσπαθεί να επιτύχει δύο στόχους:

1. Τη δημιουργία διανυσμάτων λέξεων που αποτυπώνουν νόημα σε χώρο διανυσμάτων

2. Την εκμετάλλευση καθολικών στατιστικών στοιχείων εκτός από τοπικές πληροφορίες

Στο Word2Vec τα Skipgram μοντέλα προσπαθούν να αποτυπώσουν τη συνεμφάνιση λέξεων με τη χρήση "παραθύρων", λαμβάνοντας υπόψη μόνο τις πιο κοντινές λέξεις. Αντίθετα, το GloVe δημιουργεί έναν πίνακα συνεμφάνισης για μέτρηση εμφάνισης μίας λέξης σε συμφραζόμενα. Για να μειωθεί το μέγεθος του πίνακα παραγοντοποιείται ώστε να αποτυπωθεί σε λιγότερες διαστάσεις.

Πιθανότητα & αναλογία	k = *solid*	k = *gas*	k = *water*	k = *fashion*
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

Πίνακας 7.1: Πιθανότητες συνεμφάνισης για τις λέξεις *ice* και *steam* με κάποιες επιλεγμένες λέξεις από ένα σώμα κειμένων 6 δισεκατομμυρίων στοιχείων. [60]

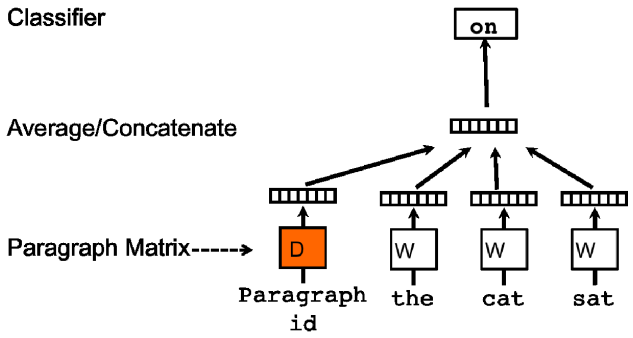
Στον Πίνακα 7.1 εμφανίζονται οι πιθανότητες συνεμφάνισης για τις λέξεις *ice* και *steam* με κάποιες επιλεγμένες λέξεις από ένα σώμα κειμένων 6 δισεκατομμυρίων στοιχείων. Η αναλογία (τελευταία γραμμή) φιλτράρει το θόρυβο από μη περιγραφικές λέξεις όπως το *water* (σχετικό και με τις δύο στοχευμένες λέξεις) ή το *fashion* (άσχετο με τις δύο στοχευμένες λέξεις). Οι υψηλές τιμές, κατά πολύ μεγαλύτερες του 1, δηλώνουν συσχέτιση με τη λέξη ice, ενώ οι χαμηλές τιμές, κατά πολύ μικρότερες του 1, δηλώνουν συσχέτιση με τη λέξη steam. [60] Η αναλογία συνεμφάνισης μεταξύ λέξεων είναι ιδιαίτερα αποτελεσματική, γι' αυτό τον λόγο χρησιμοποιείται από το GloVe. Προστίθεται επίσης μία συνάρτηση βάρους για βέλτιστα αποτελέσματα.

Πρακτικά, η κυριότερη διαφορά των δύο ενσωματώσεων εξαρτάται από τα δεδομένα στα οποία εφαρμόζονται. Κάποια από αυτά αποδίδουν καλύτερα με χρήση Word2Vec, ενώ άλλα πετυχαίνουν βέλτιση απόδοση μέσω του GloVe. Το GloVe συνήθως απαιτεί παραπάνω μνήμη λόγω του πίνακα συνεμφάνισης.

7.2 Doc2Vec

Το Doc2Vec είναι μία διαφορετική ενσωμάτωση λέξεων και βελτίωση του Word2Vec. Σκοπός του είναι να δημιουργήσει μία δυαδική αναπαράσταση ενός εγγράφου, ανεξαρτήτως του μεγέθους του. Αντίθετα με τις λέξεις, τα έγγραφα δεν έχουν λογική δομή όπως αυτές και χρειάζεται να χρησιμοποιηθούν άλλες μέθοδοι.

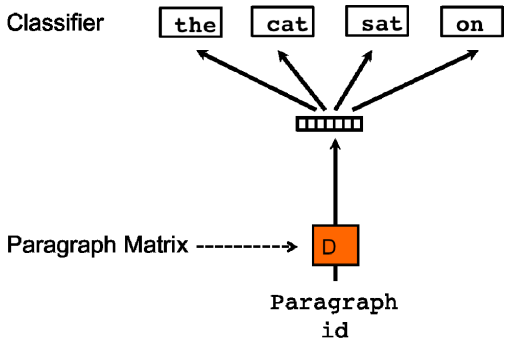
Η ιδέα των Le και Mikolov [64] είναι απλή: πάνω στη λογική του Word2Vec προστέθηκε άλλο ένα διάνυσμα, το Paragraph id του Σχήματος 7.1. Ουσιαστικά, πρόκειται για προσθήκη διανύσματος χαρακτηριστικών στο CBoW. Κατά την εκπαίδευση των διανυσμάτων λέξεων W, γίνεται εκπαίδευση και του διανύσματος εγγράφου D και στο τέλος της εκπαίδευσης περιέχει μία αριθμητική αναπαράσταση του εγγράφου. Το μοντέλο αυτό ονομάζεται PV-DM *(Distributed Memory version of Paragraph Vector)* και λειτουργεί σαν μνήμη όσων λείπουν από τα τωρινά συμφραζόμενα, όπως το θέμα της παραγράφου.



Σχήμα 7.1: Το μοντέλο PV-DM του Doc2Vec. [64]

Όπως και στο Word2Vec υπάρχει το Skip-gram, ένας αντίστοιχος αλγόριθμος είναι το PV-DBOW *(Distributed Bag of Words version of Paragraph Vector)*. Σε αντίθεση με το Word2Vec, εδώ είναι πιο γρήγορος και καταναλώνει λιγότερη μνήμη, καθώς δεν χρειάζεται να αποθηκεύουμε διανύσματα λέξεων. [65]

Στην υλοποίηση του Doc2Vec χρησιμοποιήθηκε ένας συνδυασμός των δύο μεθόδων, αν και οι συγγραφείς αναφέρουν ότι το PV-DM παρουσιάζει καλύτερα αποτελέσματα και μπορεί να χρησιμοποιηθεί μόνο του. [64, 65]



Σχήμα 7.2: Το μοντέλο PV-DBOW του Doc2Vec. [64]

7.3 FastText

Το FastText, παρότι αποτελεί επέκταση του Word2Vec, διαφέρει από αυτό, καθώς κάθε λέξη αντί να αποτελεί ξεχωριστή οντότητα, αντιμετωπίζεται σαν σύνθεση πολλαπλών n-γραμμάτων αποτελούμενα από χαρακτήρες (μοντέλο υπολέξεων). Επομένως το διάνυσμα για μία λέξη είναι το άθροισμα των συγκεκριμένων n-γραμμάτων με τους χαρακτήρες που την απαρτίζουν. Για παράδειγμα, η λέξη `where` είναι άθροισμα των διανυσμάτων από τα n-γράμματα χαρακτήρων: ‹`wh, whe, her, ere, re`›, όταν $n = 3$. Τα σύμβολα "‹" και "›" συμβολίζουν την αρχή και το τέλος της λέξης, προσφέροντας τη δυνατότητα διαφοροποίησης προθημάτων και επιθημάτων από άλλες ακολουθίες χαρακτήρων. Επομένως, η ακολουθία ‹`her`› για τη λέξη `her` είναι διαφορετική από το τρίγραμμα `her` της λέξης `where`. [66]

Η λογική του Word2Vec προσφέρει ορισμένα πλεονεκτήματα: [66, 67]

- Ανάλυση μίας λέξης σε έγκυρα μορφήματα (η ελάχιστη μονάδα σημασίας σε μία λέξη), εύρεση επιθημάτων, εύρεση κλίσεων λέξεων
- Δημιουργία διανυσμάτων λέξεων για λέξεις που δεν υπάρχουν στο σώμα εκπαίδευσης
- Απόδοση που ανταγωνίζεται δίκτυα βαθιάς μάθησης (χρειάζεται να ρυθμιστούν κατάλληλα οι υπερ-παράμετροι του αλγορίθμου)

8 Επίδειξη τρόπου λειτουργίας του αλγορίθμου Word2Vec

8.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα παρουσιαστεί μία προγραμματιστική υλοποίηση του Word2Vec σε γλώσσα Python, προκειμένου να γίνουν πιο ξεκάθαρα οι μέθοδοι και τεχνικές που περιγράφτηκαν προηγουμένως. Η υλοποίηση χρησιμοποιεί τον αλγόριθμο Skip-gram για να λειτουργήσει, έχει ένα κρυμμένο επίπεδο και είναι παρόμοια με το *σχήμα 5.3*, στη *σελίδα 41*.

Για την υλοποίηση χρησιμοποιήθηκαν βιβλιοθήκες της Python, με κυριότερες τη Numpy, η οποία παρέχει έτοιμες δομές πινάκων και διανυσμάτων πολλών διαστάσεων και το Natural Language Toolkit (NLTK) που βοήθησε στην αφαίρεση των πιο συνηθισμένων λέξεων της αγγλικής γλώσσας.

8.2 Υλοποίηση

Αρχικά γίνονται import όσες βιβλιοθήκες θα χρειαστούν *(Κώδικας 1)*:

```
1 import numpy as np
2 from collections import defaultdict
3 import re
4 import time
5 import nltk
6 from nltk.corpus import stopwords
7 from nltk.tokenize import word_tokenize
8 from nltk.tokenize.treebank import TreebankWordDetokenizer
```

Listing 1: Εισαγωγή απαραίτητων βιβλιοθηκών που θα χρησιμοποιηθούν

Διαβάζονται τα δεδομένα από ένα αρχείο. Στο συγκεκριμένο παράδειγμα χρησιμοποιείται η ιστορία μικρού μήκους "The Most Dangerous Game".[5] Αφού διαβαστούν, αφαιρούνται από τα δεδομένα διάφοροι περιττοί χαρακτήρες που μπορεί να υπάρχουν. Στη συνέχεια εκτελείται αφαίρεση των stopwords όπως περιγράφεται στις *ενότητες 5.3.1 & 5.3.2* και αποθηκεύονται σε αλφαριθμητική μεταβλητή. [70] *(Κώδικας 2)*

[5]Κείμενο σε καθεστώς public domain, διαθέσιμο ελεύθερα στο Internet Archive: https://ia800304.us.archive.org/15/items/TheMostDangerousGame_129/danger.txt.
Το αρχείο μετατράπηκε πρώτα σε UTF-8 για να μπορεί να προσπελαστεί από τις βιβλιοθήκες της Python χωρίς προβλήματα κωδικοποίησης.

Κατά την αφαίρεση των stopwords εξαλείφθηκαν μόλις 58 λέξεις, αλλά αντιπροσώπευαν το 43% των συνολικών λέξεων του dataset.

```
#Read file data
def read_file():
    with open('danger.txt') as f:
        expression = f.read().replace('\n', ' ')
        remove_special_characters = re.sub('[^A-Za-z ]+', ' ',
                                           expression)
        nltk.download('stopwords')
        stop_words = set(stopwords.words('english'))
        tokens = word_tokenize(remove_special_characters)
        tokenized = [i for i in tokens if not i in stop_words]
        detokenized = TreebankWordDetokenizer().detokenize(tokenized)
    return detokenized
```

Listing 2: Ανάγνωση δεδομένων από αρχείο

	Πριν την αφαίρεση των stopwords	**Μετά την αφαίρεση των stopwords**
Μοναδικές λέξεις (μέγεθος λεξιλογίου)	1927	1869
Συνολικές λέξεις	8138	4616

Πίνακας 8.1: Διαφορά στον αριθμό των λέξεων χάρη στην αφαίρεση των stopwords.

Ορίζεται μία κλάση μέσα στην οποία θα υπάρχει όλος ο κώδικας του αλγόριθμου. Έπειτα, δηλώνονται οι μεταβλητές που πρέπει να δοθούν για την εκπαίδευση του αλγόριθμου,[6] όπως πόσες φορές πρέπει να επαναληφθεί η εκτέλεση, το μέγεθος παραθύρου, την παράμετρο εκμάθησης και τις διαστάσεις των ενσωματώσεων λέξεων, οι οποίες επηρεάζουν το μέγεθος του κρυφού επιπέδου. *(Κώδικας 3)*

Από το σώμα κειμένων αντλούνται όλες τις λέξεις που υπάρχουν ώστε να αποθηκευτούν σε συγκεκριμένες μεταβλητές και να δημιουργηθεί το λεξιλόγιο του σώματος κειμένων. Θα υπάρχει μία εγγραφή για κάθε λέξη, ακόμα κι αν αυτή εμφανίζεται πολλαπλές φορές μέσα στο αρχικό κείμενο. Επίσης δημιουργούνται

[6] `__init__()`, αντίστοιχο των constructors σε Python

```python
# The code for Word2Vec
class Word2Vec():

    def __init__(self):
        self.dimensions = default_values['dimensions']
        self.learning_param = default_values['learning_param']
        self.iterations = default_values['iterations']
        self.window_size = default_values['window_size']
```

Listing 3: Ορισμός κλάσης, δήλωση τιμών που χρησιμοποιούνται πολλαπλές φορές

2 dictionaries που αριθμούν τις λέξεις (με το ένα από αυτά να δείχνει από το id που δόθηκε ενώ το άλλο το ακριβώς αντίθετο). *(Κώδικας 4)*

```python
    def create_training_data(self, settings, corpus):
        # Unique words
        vocabulary = defaultdict(int)

        for row in corpus:
            for word in row:
                # word = word.lower()
                vocabulary[word] += 1
        print(vocabulary)

        # Number of unique words
        self.vocabulary_size = len(vocabulary.keys())
        # All the unique words
        self.vocabulary_list = list(vocabulary.keys())

        # Dictionary (the -> 0, most -> 1, dangerous -> 2)
        self.word_to_id = dict((word, i) for i,
                    word in enumerate(self.vocabulary_list))

        # Dictionary (0 -> the, 1 -> most, 2-> dangerous)
        self.id_to_word = dict((i, word) for i,
                    word in enumerate(self.vocabulary_list))
```

Listing 4: Αποθήκευση λέξεων σε μεταβλητές και δομή λεξιλογίων της Python

Έπειτα το σώμα κειμένων ξαναδιαπερνάται ώστε να αντληθούν οι συσχετίσεις της κάθε στοχευμένης λέξης (target word) με τις γειτονικές της συγκείμενες (context words). Καλείται η συνάρτηση one_hot_vectors για δημιουργία της μεταβλητής word_target και του πίνακα word_context και έπειτα προστίθενται στον πίνακα training_data[7] που αρχικά είναι κενός και με κάθε επανάληψη αποθηκεύει τις στοχευμένες και συγκείμενες λέξεις. *(Κώδικας 5)*

```
        training_data = []

        # Pass through through dataset
        for dataset in corpus:
            dataset_length = len(dataset)
            print(dataset)
            print(dataset_length)

            # Pass through each word in dataset
            for i, word in enumerate(dataset):
                word_target = self.one_hot_vectors(dataset[i])

                word_context = []

                for j in range(i - self.window_size,
                               i + self.window_size + 1):
                    if j != i and j <= dataset_length -1 \
                            and j >= 0:
                        word_context.append\
                            (self.one_hot_vectors(dataset[j]))
                training_data.append([word_target, word_context])

        return np.array(training_data,dtype=object)
```

Listing 5: Επανάληψη στα δεδομένα για δημιουργία διανυσμάτων με στοχευμένες λέξεις και τις συγκείμενες τους λέξεις. Καλείται η συνάρτηση one_hot_vectors που δηλώνεται παρακάτω.

Στη συνάρτηση one_hot_vectors δημιουργείται ένα one-hot διάνυσμα για κάθε στοχευμένη λέξη. Αρχικά όλες οι θέσεις γεμίζουν με 0 και στη συνέχεια βρίσκεται το id της στοχευμένης λέξης και αντικαθίσταται με 1. *(Κώδικας 6)*

[7] δομή numpy.array

```
    def one_hot_vectors(self, word):
        word_vec = [0 for i in range(0, self.vocabulary_size)]

        word_to_id = self.word_to_id[word]

        word_vec[word_to_id] = 1

        return word_vec
```

Listing 6: Δημιουργία αρχικού one-hot διανύσματος, με όλες τις τιμές πλην μίας να είναι 0

Η forward_propagation ορίζει τον μηχανισμό εμπρόσθιας τροφοδότησης. Δημιουργείται το κρυμμένο επίπεδο πολλαπλασιάζοντας το διάνυσμα κάθε στοχευμένης λέξης με το πρώτο σύνολο βαρών[8] και έπειτα γίνεται το ίδιο για το τελικό επίπεδο, αυτή τη φορά με το δεύτερο σύνολο βαρών. Τέλος, το τελικό επίπεδο στέλνεται στη συνάρτηση softmax για παλινδρόμηση. *(Κώδικας 7)*

```
    def forward_propagation(self, target_word_vector):
        # Compute dot product
        hidden_layer = np.dot(target_word_vector, self.weight_1)
        # Compute dot product
        output_layer = np.dot(hidden_layer, self.weight_2)
        y_c = self.softmax(output_layer)
        return y_c, hidden_layer, output_layer
```

Listing 7: Εμπρόσθια τροφοδότηση

Η softmax δέχεται το τελικό επίπεδο και εκτελεί εκθετική συνάρτηση σε κάθε τελικό επίπεδο κόμβου, διαιρούμενο με το άθροισμα όλων των εκθετικών αποτελεσμάτων. *(Κώδικας 8)*

Έπειτα υπολογίζεται το σφάλμα ανάμεσα στο αποτέλεσμα της softmax και τις συγκείμενες λέξεις. *(Κώδικας 9)*

Τα λάθη του προηγούμενου βήματος χρησιμοποιούνται στην οπίσθια τροφοδότηση, η οποία υπολογίζει την αλλαγή που πρέπει να γίνει στα βάρη χρησιμο-

[8] Υπολογισμός εσωτερικού γινομένου σε Numpy

```python
    def softmax(self, x):
        # Compute softmax values
        e_x = np.exp(x - np.max(x))
        return e_x / e_x.sum(axis=0)
```

Listing 8: Υλοποίηση softmax συνάρτησης

```python
    def error_calculation(self, y_pred, context_word_vector):
        # Difference of softmax - context words
        error = np.sum([np.subtract(y_pred, word)
                        for word in context_word_vector], axis=0)
        return error
```

Listing 9: Υπολογισμός λαθών

ποιώντας το αποτέλεσμα της προηγούμενης συνάρτησης. Έπειτα το εφαρμόζει, αφαιρώντας από την υπάρχουσα τιμή τους το νούμερο, πολλαπλασιασμένο με την παράμετρο εκμάθησης. *(Κώδικας 10)*

```python
    def back_propagation(self, error_calculation, hidden_layer, x):
        # Compute outer product
        dl_dw2 = np.outer(hidden_layer, error_calculation)
        dl_dw1 = np.outer(x, np.dot(self.weight_2, error_calculation.T))

        # Update weights
        self.weight_1 = self.weight_1 - (self.learning_param * dl_dw1)
        self.weight_2 = self.weight_2 - (self.learning_param * dl_dw2)
```

Listing 10: Οπίσθια τροφοδότηση, ενημέρωση βαρών με χρήση του υπολογισμού λαθών

Η loss_calculation υπολογίζει την απώλεια που υπάρχει σε κάθε επανάληψη. Αποτελείται από 2 μέρη, το πρώτο είναι το αρνητικό άθροισμα όλων των στοιχείων του τελικού επιπέδου, και το δεύτερο πολλαπλασιάζει τον αριθμό των συγκείμενων λέξεων με τον λογάριθμο του αθροίσματος για όλα τα στοιχεία (τα οποία περνάνε πρώτα από εκθετική συνάρτηση) στο τελικό επίπεδο. *(Κώδικας 11)*

```
1     def loss_calculation(self, output_layer, context_word_vector):
2         self.loss = 0
3         self.loss += -np.sum([output_layer[word.index(1)]
4                               for word in context_word_vector]) \
5                      + len(context_word_vector) \
6                      * np.log(np.sum(np.exp(output_layer)))
7         return self.loss
```

Listing 11: Συνάρτηση υπολογισμού απώλειας

Στη συνάρτηση train *(Κώδικας 12)* καλούνται όλα τα προηγούμενα βήματα (εμπρόσθια τροφοδότηση, softmax, υπολογισμός λαθών, οπίσθια τροφοδότηση) και ορίζονται τα βάρη που χρησιμοποιούνται πριν το κρυμμένο επίπεδο και μετά από αυτό. Αυτή η διαδικασία εκτελείται πολλαπλές φορές, ανάλογα με τις επαναλήψεις που έχουμε δηλώσει στη μεταβλητή `iterations`, με σταδιακή μείωση της απώλειας που εμφανίζεται.

Τα βάρη στη συγκεκριμένη περίπτωση δεν έχουν σταθερές τιμές, αλλά τυχαίες εναλλασσόμενες. Αυτό έχει ως αποτέλεσμα, αφού μεταβάλλονται αυτά, να μεταβάλλονται και τα εξαγώμενα δεδομένα μετά από κάθε εκπαίδευση. Επομένως αν θέλουμε να γίνει σύγκριση 2 εκτελέσεων του κώδικα και των αποτελεσμάτων που εμφανίζουν θα πρέπει να γίνει αλλαγή και να αντικατασταθούν με σταθερές τιμές ή seed της επιλογής μας.

Η συνάρτηση get_vector_of_word *(Κώδικας 13)* καλείται μετά την εκπαίδευση και επιστρέφει το διάνυσμα που αντιστοιχεί σε μία επιλεγμένη στοχευμένη λέξη.

Η συνημιτονοειδής ομοιότητα *(Κώδικας 14)* είναι απαραίτητη για να εμφανιστούν οι κοντινές λέξεις μίας επιλεγμένης από το dataset. Οι γραμμές 9-11 υλοποιούν τη θεωρία που περιγράφεται στην *υποενότητα 4.3*.

```
    def train(self, training_data):
        # Use random values for weights:
        # different results at each training
        self.weight_1 = np.random.uniform\
            (-1, 1, (self.vocabulary_size, self.dimensions))
        self.weight_2 = np.random.uniform\
            (-1, 1, (self.dimensions, self.vocabulary_size))

        for i in range(self.iterations):
            for target_word_vector, context_word_vector \
                    in training_data:
                y_pred, hidden_layer, output_layer = \
                        self.forward_propagation(target_word_vector)
                total_error = self.error_calculation\
                        (y_pred,context_word_vector)
                self.back_propagation(total_error,
                        hidden_layer, target_word_vector)
                total_loss = self.loss_calculation\
                        (output_layer,context_word_vector)

            print('Iteration:', i, "Loss:", total_loss)
```

Listing 12: Κλήση όλων των παραπάνω συναρτήσεων για εκπαίδευση. Χρησιμοποιείται επανάληψη για να εκτελεστούν όσες φορές δηλώνουμε στη μεταβλητή iterations.

```
    # Get vector of word
    def get_vector_of_word(self, word):
        w_id = self.word_to_id[word]
        v_w = self.weight_1[w_id]
        return v_w
```

Listing 13: Συνάρτηση για εμφάνιση διανύσματος στοχευμένης λέξης

```
    # Input vector, returns nearest word(s)
    def cosine_similarity(self, word, number_of_nearest):
        v_w1 = self.get_vector_of_word(word)
        word_similarity = {}

        for i in range(self.vocabulary_size):
            # Find the similarity score for each word in vocabulary
            v_w2 = self.weight_1[i]
            total_dot = np.dot(v_w1, v_w2)
            total_norm = np.linalg.norm(v_w1) * np.linalg.norm(v_w2)
            theta = total_dot / total_norm

            word = self.id_to_word[i]
            word_similarity[word] = theta

        nearest_words = sorted(word_similarity.items(),
                               key=lambda kv: kv[1], reverse=True)

        for word, nearest in nearest_words[:number_of_nearest]:
            print(word, nearest)
```

Listing 14: Συνάρτηση που υπολογίζει συνημιτονοειδή ομοιότητα και βρίσκει τις πιο κοντινές λέξεις μίας επιλεγμένης στοχευμένης λέξης

Τέλος, μετά τις υλοποιήσεις των συναρτήσεων υπάρχει το κύριο μέρος του προγράμματος *(Κώδικας 15)*. Αρχικοποιούνται οι τιμές που αναφέρθηκαν στον *κώδικα 3*, έπειτα καλούνται όλες οι υπόλοιπες που απαιτούνται για την εκπαίδευση και στο τέλος γίνεται χρήση των get_vector_of_word και cosine_similarity για να ληφθούν αποτελέσματα.

Ενδεικτικά, τα αποτελέσματα που εμφανίζονται για την λέξη "dangerous" μετά από μία εκτέλεση του προγράμματος είναι:

```
dangerous [ 0.44217787  1.10197754 -0.63665796 -1.25105528  1.56235813
  1.09652871  0.11017171  0.77586346  1.55657706  0.72740048]

dangerous 1.0
surprise 0.9237047879597611
precisely 0.9106447864338071
```

```
default_values = {
    'window_size': 10,
    'dimensions': 10,
    'iterations': 50,
    'learning_param': 0.01
}

text = read_file()
print(text)
corpus = [[word.lower() for word in text.split()]]

# Assign object
w2V = Word2Vec()
training_data = w2V.create_training_data(default_values, corpus)
# Training
w2V.train(training_data)
# Get vector for word
word = "dangerous"
vec = w2V.get_vector_of_word(word)
print(word, vec)
# Find similar words
w2V.cosine_similarity("hunter", 3)
w2V.cosine_similarity("dangerous", 3)

toc = time.perf_counter()
print('Ran in ' + str(toc - tic) + ' seconds.')
```

Listing 15: Αρχικοποίηση τιμών, κλήση των παραπάνω συναρτήσεων

8.3 Βελτιώσεις

Ασφαλώς όσο μεγαλώνει το μέγεθος του dataset τόσο πιο χρονοβόρα γίνεται η διαδικασία εκπαίδευσης και αυξάνεται η κατανάλωση πόρων. Στο συγκεκριμένο dataset η εκτέλεση σε ένα laptop του 2014 παίρνει περίπου 40 λεπτά για 50 επαναλήψεις και καταναλώνει πάνω από 1.5GB RAM (η χρήση μνήμης είναι άμεσα συνδεδεμένη με το μέγεθος παραθύρου και το μέγεθος του dataset). Η λύση για καλύτερες επιδόσεις είναι η εφαρμογή μίας από τις τεχνικές που περιγράφονται στην *ενότητα 5.3*:

- αρνητική δειγματοληψία για μείωση των δειγμάτων προς εκπαίδευση

  ή

- ιεραρχική softmax

Μία ακόμα τεχνική με πιθανώς θετικά αποτελέσματα είναι το παράθυρο τυχαίου μεγέθους, ώστε να μειώνονται τα σύνολα λέξεων που πρέπει να ελεγχθούν.

9 Συμπεράσματα

Η επεξεργασία φυσικής γλώσσας κρίνεται απαραίτητη για αποτελεσματική μελέτη των γραπτών κειμένων που συναντώνται στη σημερινή εποχή. Λόγω του όγκου δεδομένων και της ποικιλίας τους η ανθρώπινη επεξεργασία είναι χρονοβόρα και μη βέλτιστη.

Πολλές λέξεις ή φράσεις παρουσιάζουν εννοιολογικές ομοιότητες με άλλες λέξεις ή φράσεις. Χρησιμοποιώντας συστήματα μηχανικής μάθησης είναι δυνατή η ταξινόμηση αυτών με αυτοματοποιημένο τρόπο. Η εμπειρία ενός συστήματος είναι σημαντικός παράγοντας, καθώς επηρεάζει τον βαθμό ακρίβειας που επιτυγχάνει ο αλγόριθμος.

Η ενσωμάτωση λέξεων είναι ένας τρόπος για επίλυση του συγκεκριμένου προβλήματος. Αντιστοιχώντας τις λέξεις από μία συλλογή γραπτών κειμένων σε ένα διάνυσμα πραγματικών αριθμών, είναι δυνατή η εύρεση συγκείμενων εννοιών με υπολογισμό της απόστασης κάθε αριθμού (λέξης) από έναν άλλο, με τους πιο κοντινούς να παρουσιάζουν ομοιότητες.

Το Word2Vec αποτελεί έναν αλγόριθμο που υλοποιεί την ιδέα των ενσωματώσεων λέξεων με χρήση ενός νευρωνικού δικτύου. Παρά την φαινομενική απλότητα του, παράγει ικανοποιητικά και σχετικά ακριβή αποτελέσματα. Χάρη στις δύο τεχνικές που χρησιμοποιεί, το Skip-gram και το Bag-of-Words, είναι δυνατή η επιλογή της πιο αποδοτικής κρίνοντας από το είδος του σώματος κειμένων προς εκπαίδευση. Το γεγονός ότι σε ένα μεγάλο σώμα κειμένων, όπως μία έκδοση της Wikipedia, η εκπαίδευση με αυτό καταφέρνει να ξεχωρίσει συναφείς λέξεις ή έννοιες (πχ. ηθοποιούς) είναι δείγμα της αποτελεσματικότητας του αλγορίθμου.

Η υλοποίηση μπορεί να βελτιωθεί με πλήθος βελτιστοποιήσεων, ώστε να λειτουργεί αποδοτικά κατά την εκπαίδευση με μεγάλα σύνολα εκπαίδευσης. Αρκετές από αυτές λειτουργούν με τη λογική να απορρίψουν λέξεις που εμφανίζονται υπερβολικά συχνά και στερούνται εννοιών από μόνες τους. Άλλες δίνουν περισσότερο βάρος στην τοποθέτηση των λέξεων μέσα στις προτάσεις. Τέλος, η τυχαία επιλογή κάποιων λέξεων προς εκπαίδευση με μία άλλη επιλεγμένη (αντί να γίνει εκπαίδευση με το σύνολο των λέξεων) έχει αποδειχθεί πειραματικά ότι είναι χρήσιμη για ανάλυση μεγάλου όγκου δεδομένων, αφού η πληροφορία που πιθανόν χάνεται αντισταθμίζεται από τα ήδη ικανοποιητικά αποτελέσματα και την ελαχιστοποίηση των απαιτούμενων υπολογιστικών πόρων που είναι απαραίτητοι για να ολοκληρωθεί η εκπαίδευση.

Μάλιστα, όπως φαίνεται στην ενότητα 7, το Word2Vec μπορεί να βελτιωθεί

με κάποιες αλλαγές που κατέληξαν σε εναλλακτικές υλοποιήσεις, όπως το Glove ή το FastText. Αυτές κάνουν αλλαγές στη λειτουργία του αλγορίθμου ή τον χρησιμοποιούν σαν βάση και πειραματίζονται με επιπλέον καινοτομίες και ιδέες που αποφέρουν θετικά αποτελέσματα σε αρκετές περιπτώσεις.

Εν κατακλείδι, οι παραπάνω μέθοδοι μπορούν να εφαρμοστούν για πολλαπλές χρήσεις στην ανάλυση γραπτών κειμένων, καθιστώντας τες πολύ χρήσιμα εργαλεία στη διάθεση των ερευνητών του συγκεκριμένου κλάδου της μηχανικής μάθησης.

Περαιτέρω μελέτη μπορεί να γίνει σε άλλες εναλλακτικές ενσωματώσεις λέξεων, τόσο σε αυτές που βασίζονται στο Word2Vec και τις ιδέες που παρουσίασε, όσο και λοιπές που ακολουθούν διαφορετική πορεία και ενδέχεται να παρουσιάσουν αποτελέσματα ακόμα πιο κοντά στην ανθρώπινη λογική. Όπως είναι φανερό από την γρήγορη ανάπτυξη του κλάδου της επεξεργασίας φυσικής γλώσσας, οι καινοτομίες διαδέχονται η μία την άλλη, με αποτέλεσμα τεχνολογίες ή ιδέες που παλιότερα παρουσίαζαν την καλύτερη επίδοση για διάφορες εργασίες που τους αναθέτονταν, πλέον να χρησιμοποιούνται σε λιγότερες εφαρμογές, έχοντας αντικατασταθεί από διάδοχους που παρέχουν καλύτερα αποτελέσματα.

10 Βιβλιογραφία

[1] Hopfield, John J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences, 79(8), 2554-2558. (https://www.pnas.org/content/pnas/79/8/2554.full.pdf)*

[2] Jain, Anil K., Jianchang Mao, and K. Moidin Mohiuddin (1996). Artificial neural networks: A tutorial. *Computer 29.3, 31-44. (http://metalab.uniten.edu.my/~abdrahim/mitm613/Jain1996_ANN%20-%20A%20Tutorial.pdf)*

[3] Mitchell, Tom (1996). Machine Learning. *McCraw Hill, United States.*

[4] Διαμαντάρας Κωνσταντίνος (2007). Τεχνητά Νευρωνικά Δίκτυα. *Κλειδάριθμος. Αθήνα, Ελλάδα.*

[5] Werbos Paul (1974). Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences *Ph.D thesis, Harvard University.*

[6] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1985). Learning internal representations by error propagation. *No. ICS-8506. California Univ San Diego La Jolla Inst for Cognitive Science.*

[7] Thomas Wood (Ανάκτηση 28/09/2020). Backpropagation *(https://deepai.org/machine-learning-glossary-and-terms/backpropagation)*

[8] Fjodor van Veen (2016-2019). (Ανάκτηση 21/09/2020). The Neural Network Zoo *(https://www.asimovinstitute.org/neural-network-zoo/)*

[9] Kowsari, Kamran, et al. (2019). Text Classification Algorithms: A Survey. *arXiv preprint arXiv:1904.08067 (https://arxiv.org/abs/1904.08067)*

[10] Minaee, Shervin, et al. (2020). Deep learning based text classification: A comprehensive review. *arXiv preprint arXiv:2004.03705 (https://arxiv.org/abs/2004.03705)*

[11] Rish, Irina (2001). An empirical study of the naive Bayes classifier. *IJCAI 2001 workshop on empirical methods in artificial intelligence. Vol. 3. No. 22. (https://www.cc.gatech.edu/~isbell/reading/papers/Rish.pdf)*

[12] Raschka, Sebastian (2014). Naive bayes and text classification I - Introduction and Theory. *arXiv preprint arXiv:1410.5329 (https://arxiv.org/abs/1410.5329)*

[13] Noel Bambrick (2016). (Ανάκτηση 23/09/2020). Support Vector Machines for dummies; A Simple Explanation *(https://blog.aylien.com/support-vector-machines-for-dummies-a-simple-explanation/)*

[14] Lujing Chen (2019). (Ανάκτηση 23/09/2020). Support Vector Machine — Simply Explained *(https://towardsdatascience.com/support-vector-machine-simply-explained-fee28eba5496)*

[15] Alejandro Alcalde (2017). Creating trees, Dependency graphs and Support Vector Machines in LaTeX with Tikz *(https://elbauldelprogramador.com/en/creating-trees-dependency-graphs-svms-in-tikz/)*

[16] Turing, Alan M. (1950). Computing machinery and intelligence. *Parsing the Turing Test. Springer, Dordrecht, 2009. 23-65. (http://cogprints.org/499/1/turing.html)*

[17] Wikimedia Commons - Juan Alberto Sánchez Margallo (2017). Diagram of the Turing test *(https://commons.wikimedia.org/wiki/File:Turing_test_diagram.png)*

[18] Manning, Christopher D., Christopher D. Manning, and Hinrich Schütze. (1999). Foundations of statistical natural language processing. *MIT press.*

[19] Magnus Sahlgren (2015). (Ανάκτηση 09/05/2020). A brief history of word embeddings (and some clarifications) *(https://www.linkedin.com/pulse/brief-history-word-embeddings-some-clarifications-magnus-sahlgren/)*

[20] Boleda, Gemma (2020). Distributional semantics and linguistic theory. *Annual Review of Linguistics. (https://www.annualreviews.org/doi/full/10.1146/annurev-linguistics-011619-030303)*

[21] Landauer, Thomas K., Peter W. Foltz, and Darrell Laham. (1998). An introduction to latent semantic analysis. *Discourse processes 25.2-3 (1998): 259-284. (https://mainline.brynmawr.edu/Courses/cs380/fall2006/intro_to_LSA.pdf)*

[22] Andrew Gibiansky. (2013). (Ανάκτηση 14/09/2020). Cool Linear Algebra: Singular Value Decomposition *(https://andrew.gibiansky.com/blog/mathematics/cool-linear-algebra-singular-value-decomposition/)*

[23] Levy, Omer, and Yoav Goldberg. (2014). Dependency-based word embeddings. *Proceedings of the 52nd Annual Meeting of the Association for*

Computational Linguistics (Volume 2: Short Papers). (https://www.aclweb.org/anthology/P14-2050.pdf)

[24] Turney, P. D. (2012). Domain and Function: A Dual-Space Model of Semantic Relations and Compositions. *Journal of Artificial Intelligence Research. (https://arxiv.org/abs/1309.4035)*

[25] Blei, David M. (2012). Probabilistic topic models. *Communications of the ACM 55.4 (2012): 77-84. (https://dl.acm.org/doi/pdf/10.1145/2133806.2133826)*

[26] Chris McCormick (2019). The Inner Workings of word2vec. *Self-published.*

[27] RIA KULSHRESTHA (2019). (Ανάκτηση 09/05/2020). NLP 101: Word2Vec — Skip-gram and CBOW *(https://towardsdatascience.com/nlp-101-word2vec-skip-gram-and-cbow-93512ee24314)*

[28] Neeraj Singh Sarwan (2017). (Ανάκτηση 09/05/2020). An Intuitive Understanding of Word Embeddings: From Count Vectors to Word2Vec *(https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/)*

[29] Gregory Piatetsky, KDnuggets (2017). (Ανάκτηση 14/07/2020). Cartoon: the distance between Espresso and Cappuccino *(https://www.kdnuggets.com/2017/04/cartoon-word2vec-espresso-cappuccino.html)*

[30] Harris, Zellig S. (1954). Distributional structure. *Word 10.2-3: 146-162. (https://www.tandfonline.com/doi/pdf/10.1080/00437956.1954.11659520)*

[31] Wittgenstein, Ludwig (1954, 2009). Philosophical investigations. *John Wiley & Sons.*

[32] Skansi, Sandro (2018). Introduction to Deep Learning: from logical calculus to artificial intelligence. *Springer.*

[33] Γεώργιος Μικρός (2015). Υπολογιστική υφολογία [ηλεκτρ. βιβλ.] *(https://repository.kallipos.gr/handle/11419/4860)*

[34] Guthrie, D., Allison, B., Liu, W., Guthrie, L., & Wilks, Y. (2006, May). A closer look at skip-gram modelling. *LREC (Vol. 6, pp. 1222-1225). (https://www.aclweb.org/anthology/L06-1210/)*

[35] Huang, Xuedong, et al. (1993). The SPHINX-II speech recognition system: an overview. *Computer Speech & Language 7.2: 137-148. (https://pdfs.semanticscholar.org/9269/fbc18214e47da663ea04ed23c1cd396e0c38.pdf)*

[36] Guthrie, David, et al. (2006). A closer look at skip-gram modelling. *LREC. Vol. 6. (http://www.lrec-conf.org/proceedings/lrec2006/pdf/357_pdf.pdf)*

[37] Tomas Mikolov *(https://groups.google.com/forum/#!searchin/word2vec-toolkit/c-bow/word2vec-toolkit/NLvYXU99cAM/E51d8LcDxlAJ)*

[38] Mikolov, Tomas, et al. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781 (https://arxiv.org/abs/1301.3781)*

[39] Mikolov, Tomas, et al. (2013). Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems. (http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and)*

[40] Jay Alammar (2019). (Ανάκτηση 09/05/2020). The Illustrated Word2vec *(https://jalammar.github.io/illustrated-word2vec/)*

[41] Adit Deshpande (2017). (Ανάκτηση 09/05/2020). Deep Learning Research Review Week 3: Natural Language Processing *(https://adeshpande3.github.io/Deep-Learning-Research-Review-Week-3-Natural-Language-Processing)*

[42] Chris McCormick (2016). (Ανάκτηση 09/05/2020). Word2Vec Tutorial - The Skip-Gram Model *(http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/)*

[43] Claudio Bellei (2018). (Ανάκτηση 15/09/2020). The backpropagation algorithm for Word2Vec *(http://www.claudiobellei.com/2018/01/06/backprop-word2vec/)*

[44] Computer Science Department, Stanford University (2013). (Ανάκτηση 09/05/2020). Softmax Regression - Unsupervised Feature Learning and Deep Learning Tutorial *(http://ufldl.stanford.edu/tutorial/supervised/SoftmaxRegression/)*

[45] Sebastian Raschka (2016). (Ανάκτηση 22/08/2020). What is Softmax regression and how is it related to Logistic regression? *(https://sebastianraschka.com/faq/docs/softmax_regression.html)*

[46] Thomas Wood (Ανάκτηση 28/09/2020). Softmax Function *(https://deepai.org/machine-learning-glossary-and-terms/softmax-layer)*

[47] Chris McCormick (2017). (Ανάκτηση 09/05/2020). Word2Vec Tutorial Part 2 - Negative Sampling *(http://mccormickml.com/2017/01/11/word2vec-tutorial-part-2-negative-sampling/)*

[48] Word2Vec contributors, McCormick Chris (2013-2019). (Ανάκτηση 28/09/2020). Word2Vec - Commented code [Subsampling of Frequent Words] *(https://github.com/chrisjmccormick/word2vec_commented/blob/master/word2vec.c#L869)*

[49] Caselles-Dupré, Hugo, Florian Lesaint, and Jimena Royo-Letelier. (2018). Word2Vec applied to Recommendation: Hyperparameters Matter. *arXiv preprint arXiv:1804.04212 (https://arxiv.org/abs/1804.04212)*

[50] Edward Ma (2019). (Ανάκτηση 09/05/2020). How Negative Sampling work on word2vec? *(https://medium.com/@makcedward/how-negative-sampling-work-on-word2vec-7bf8d545b116)*

[51] Morin, F., and Bengio, Y. (2005, January). Hierarchical probabilistic neural network language model. *Aistats (Vol. 5, pp. 246-252). (http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.221.8829&rep=rep1&type=pdf#page=255)*

[52] Wikimedia Commons - Meteficha (2007). Huffman tree *(https://commons.wikimedia.org/wiki/File:Huffman_tree_2.svg)*

[53] RIA KULSHRESTHA (2019). (Ανάκτηση 09/05/2020). NLP 101: Negative Sampling and GloVe *(https://towardsdatascience.com/nlp-101-negative-sampling-and-glove-936c88f3bc68)*

[54] Devin Soni (2018). (Ανάκτηση 09/05/2020). Introduction to Markov Chains *(https://towardsdatascience.com/introduction-to-markov-chains-50da3645a50d)*

[55] Joseph Rocca (2019). (Ανάκτηση 09/05/2020). Introduction to Markov Chains *(https://towardsdatascience.com/brief-introduction-to-markov-chains-2c8cab9c98ab)*

[56] Rohan Varma (2017). (Ανάκτηση 09/05/2020). Language Models, Word2Vec, and Efficient Softmax Approximations *(https://rohanvarma.me/Word2Vec/)*

[57] Levy, Omer, and Yoav Goldberg. (2014). Neural word embedding as implicit matrix factorization. *Advances in neural information processing systems. (https://pdfs.semanticscholar.org/e0d7/69c5698275c2f745c1aac2aebc323f51aa24.pdf)*

[58] Yamada, Ikuya, et al. (2018). Wikipedia2Vec: an optimized tool for learning embeddings of words and entities from Wikipedia. *arXiv preprint arXiv:1812.06280 (https://arxiv.org/abs/1812.06280)*

[59] Yamada, Ikuya, et al. (2016). Joint learning of the embedding of words and entities for named entity disambiguation. *arXiv preprint arXiv:1601.01343 (https://arxiv.org/abs/1601.01343)*

[60] Pennington, Jeffrey, Richard Socher, and Christopher D. Manning. (2014). Glove: Global vectors for word representation. *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). (https://www.aclweb.org/anthology/D14-1162.pdf)*

[61] Jan Bussieck (2017). (Ανάκτηση 09/05/2020). Demystifying Word2Vec *(https://www.deeplearningweekly.com/blog/demystifying-word2vec/)*

[62] Collobert, Ronan, et al. (2011) Natural language processing (almost) from scratch. *Journal of machine learning research 12.ARTICLE: 2493-2537. (http://www.jmlr.org/papers/volume12/collobert11a/collobert11a.pdf)*

[63] Radim Řehůřek (2014.) (Ανάκτηση 26/06/2020). Making sense of word2vec *(https://rare-technologies.com/making-sense-of-word2vec/)*

[64] Le, Quoc, and Tomas Mikolov (2014). Distributed Representations of Sentences and Documents. *arXiv preprint arXiv:1405.4053 (https://arxiv.org/abs/1405.4053)*

[65] Gidi Shperber (2017). (Ανάκτηση 06/09/2020). A gentle introduction to Doc2Vec *(https://medium.com/wisio/a-gentle-introduction-to-doc2vec-db3e8c0cce5e)*

[66] Bojanowski, Piotr, et al. (2016). Enriching Word Vectors with Subword Information. *arXiv preprint arXiv:1607.04606 (https://arxiv.org/abs/1607.04606)*

[67] Joulin, Armand, et al. (2016). Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759 (https://arxiv.org/abs/1607.01759)*

[68] Rong, Xin (2014). word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738 (https://arxiv.org/abs/1411.2738)*

[69] Weng, Lilian (2017). (Ανάκτηση 12/09/2020). Learning Word Embedding *(https://lilianweng.github.io/lil-log/2017/10/15/learning-word-embedding.html)*

[70] Davydova Olga (2018). (Ανάκτηση 09/09/2020). Text Preprocessing in Python: Steps, Tools, and Examples *(https://medium.com/@datamonsters/text-preprocessing-in-python-steps-tools-and-examples-bf025f872908)*