ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ
ΑΝΑΠΤΥΞΗ ΛΟΓΙΣΜΙΚΟΥ ΓΙΑ ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ

Αναγνώριση Ταυτότητας / Αποσαφήνιση Μεγάλου Όγκου Δεδομένων (Big Data Entity Resolution)

Μιχάλης Βουρτζόυμης
Μερόπη Φανού

2020-2021

ΠΕΡΙΕΧΟΜΕΝΑ

1. ΕΙΣΑΓΩΓΗ

Η εφαρμογή που πρόκειται να παρουσιαστεί στη συνέχεια υλοποιήθηκε στα πλαίσια του μαθήματος Ανάπτυξη Λογισμικού για Πληροφοριακά Συστήματα με θέμα τη διαχείρηση big data και την αναγνώριση ταυτότητας σε συνδυασμό με τεχνικές μηχανικής μάθησης.

Τα δεδομένα που χρησιμοποιούνται ως είσοδος της εφαρμογής αναπαριστούν ένα σύνολο από προϊόντα, τα οποία έχουν εξαχθεί από διαφορετικά web sites. Κάθε προϊόν έχει ορισμένα χαρακτηριστικά που ορίζουν την προδιαγραφή του. Βάσει αυτών των χαρακτηριστικών καλείται η εν λόγω εφαρμογή να είναι σε θέση να αναγνωρίζει με επιτυχία τόσο τα όμοια προϊόντα όσο και αυτά που διαφέρουν μεταξύ τους.

Στα πλαίσια αυτής της αναφοράς θα γίνει παρουσίαση τόσο των δομών δεδομένων που επιλέχθηκαν για την ευκολότερη αποθήκευση και διαχείρηση των δεδομένων αλλά και των αλγορίθμων μηχανικής μάθησης που δοκιμάστηκαν για την αποτελεσματικότερη ταυτοποίηση των προϊόντων. Παράλληλα με κάθε σχεδιαστική ή προγραμματιστική επιλογή που θα παρουσιάζεται θα δίνονται και τα αντίστοιχα διαγράμματα με πειραματικές τιμές από ενδεικτικές παραμέτρους όπως χρόνος εκτέλεσης, κατανάλωση μνήμης κ.ά. .

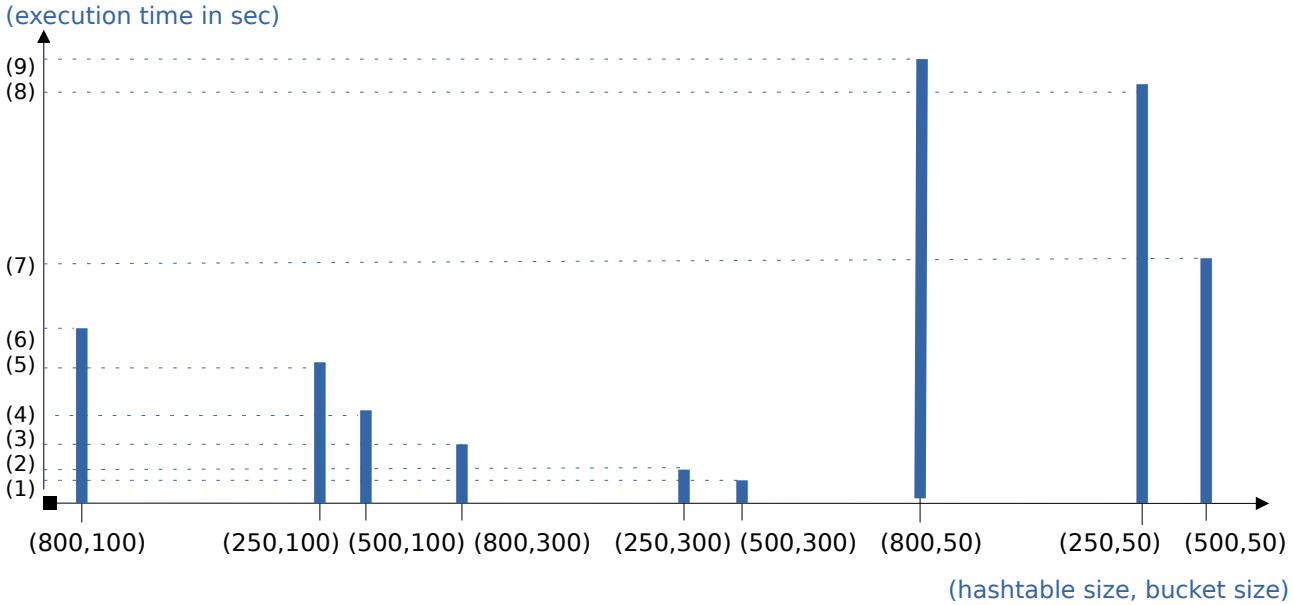
Τέλος θα γίνει σύνοψη των επιλογών των δομών και των τεχνικών που απέφεραν συγκριτικά τα καλύτερα αποτελέσματα, παραθέτοντας και τις συνθήκες υπό τις οποίες πραγματοποιήθηκαν οι δοκιμές.

2. ΑΠΟΘΗΚΕΥΣΗ ΔΕΔΟΜΕΝΩΝ ΕΙΣΟΔΟΥ

Η εφαρμογή λαμβάνει ως δεδομένα ένα σύνολο από προϊόντα (dataset X) και ένα σύνολο με τις σχέσεις ανά ζεύγη για κάποια από αυτά (dataset W). Κάθε προϊόν έχει ένα σύνολο από ιδιότητες που το χαρακτηρίζουν. Οι ιδιότητες αυτές έχουν την μορφή < key: values>. Έτσι, για ένα προϊόν αποθηκεύεται ένας πίνακας με κάθε στοιχείο να αποτελείται από το key και μια λίστα από τα values της αντίστοιχης ιδιότητας.

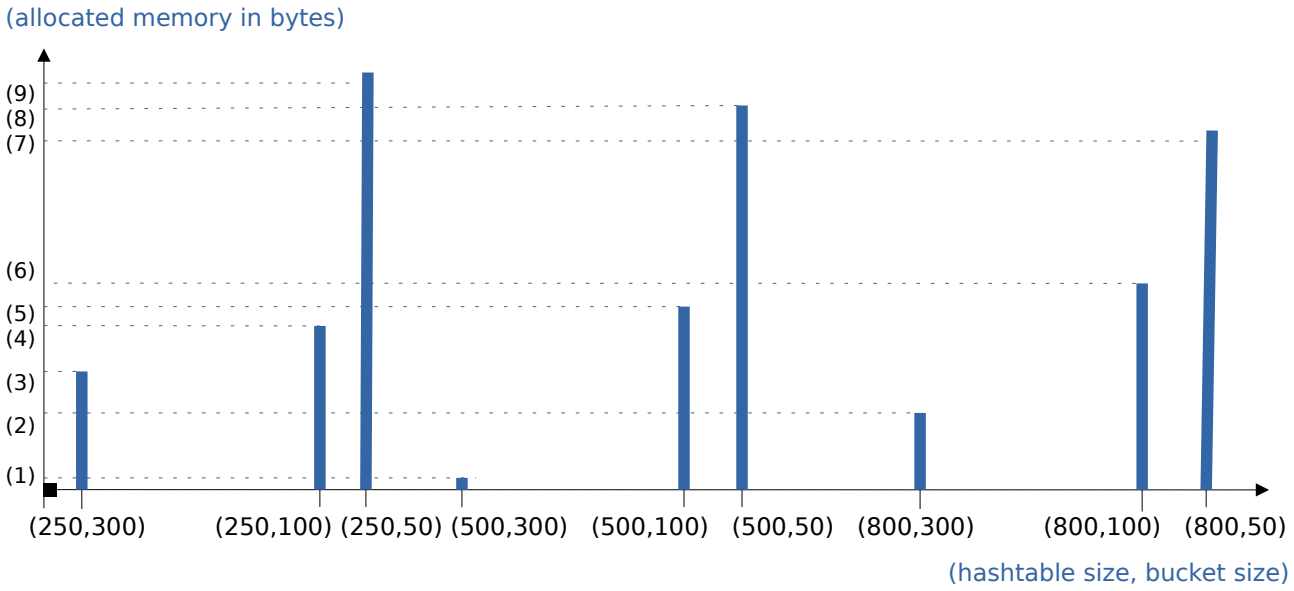
Επίσης, κάθε προϊόν έχει ένα δικό του μοναδικό id. Για τον εντοπισμό ενός προϊόντος που γνωρίζουμε το id του, χρησιμοποιείται η δομή ενός πίνακα κατακερματισμού (hashtable). Η επιλογή της δομής αυτής έγινε κυρίως λόγω του πλεονεκτήματος που διαθέτει σε σχέση με άλλες δομές ως προς τον χρόνο ανάκτησης των δεδομένων που χαρακτηρίζονται με κάποιο μοναδικό τρόπο (στη δική μας περίπτωση το id του προϊόντος). Συγκεκριμένα ο πίνακας αυτός επιτρέπει την ανάκτηση ενός προϊόντος σε O(1) χρόνο. Κάτι τέτοιο θεωρήθηκε ιδιαίτερα χρήσιμο στην υλοποίηση της συγκεκριμένης εφαρμογής, καθώς απαιτείται συχνή ανάκτηση ορισμένων προϊόντων από ένα σημαντικά μεγάλο σύνολο δεδομένων, χωρίς να υπάρχει κάποια προφανής δυνατότητα ταξινόμησής τους.

Ορίζουμε ένα πίνακα κατακερματισμού μεγέθους 500 κάδων (buckets), με κάθε κάδο να έχει μέγεθος 100bytes, ώστε να έχουμε μια καλή ισορροπία ανάμεσα σε χρόνο εκτέλεσης και κατανάλωση μνήμης. Στα παρακάτω διαγράμματα φαίνεται αντίστοιχα η μεταβολή των δύο αυτών παραμέτρων για τα διάφορα μεγέθη του hashtable και των buckets του



όπου

(1): 3m 52.234s [(500,300), trainSize:683,499]
(2): 4m 2.959s [(250,300), trainSize:683,357]
(3): 4m 37.287s [(800,300), trainSize:679,628]
(4): 5m 6.877s [(500,100), trainSize:679,015]
(5): 5m 57.772s [(250,100), trainSize:682,279]
(6): 6m30.425s [(800,100), trainSize:680,540]
(7): 7m 44.201s [(500,50), trainSize:681,246]
(8): 11m 0.097s [(250,50), trainSize:681,273]
(9): 11m 11.361s [(800,50), trainSize:681,520]



όπου

(1): 80,795,852,783 b [trainSize:683,499]
(2): 80,795,964,247 b [trainSize:679,628]
(3): 80,796,038,071 b [trainSize:683,357]
(4): 80,796,129,391 b [trainSize:682,279]
(5): 80,796,177,031 b [trainSize:679,015]
(6): 80,796,185,039 b [trainSize:681,273]
(7): 80,796,466,478 b [trainSize:681,520]
(8): 80,796,532,415 b [trainSize:681,246]
(9): 80,796,550,519 b [trainSize:681,273]

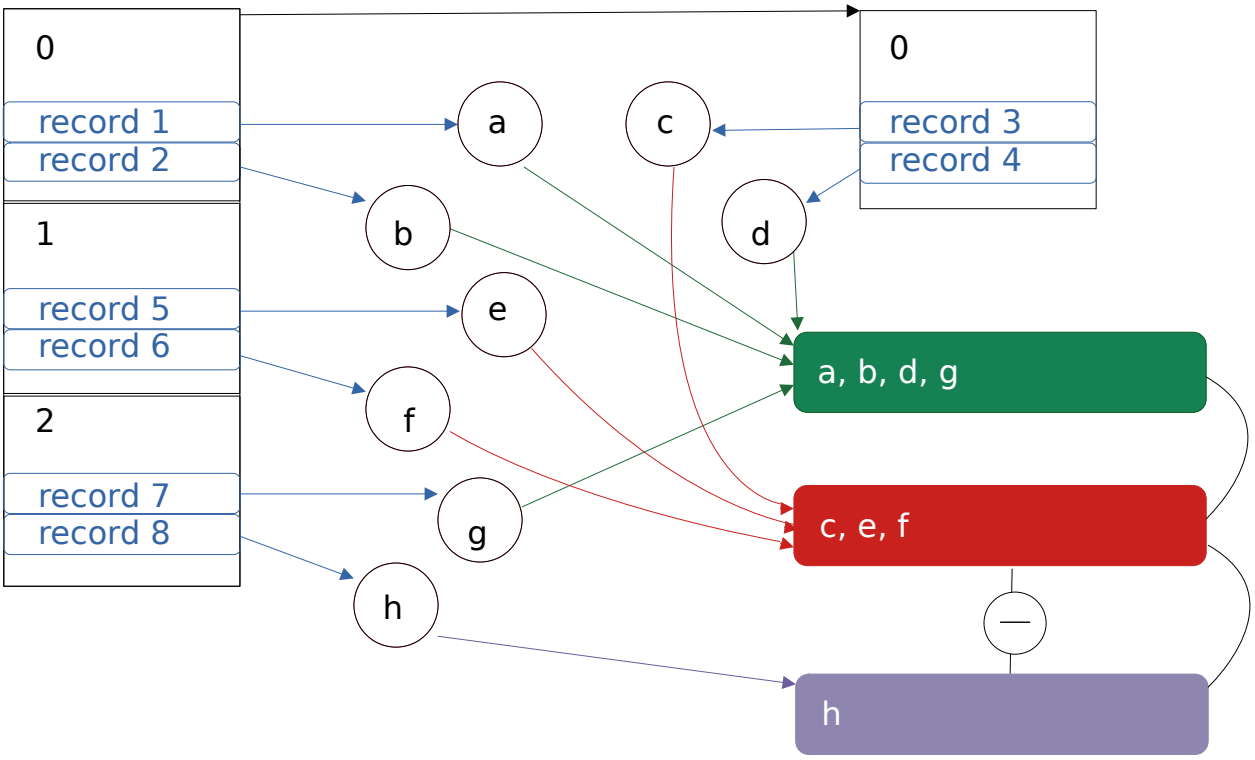
Πέρα από την αποθήκευση και την ανάκτηση των προϊόντων αυτών καθεαυτών, η άλλη πληροφορία που δίνεται στην εφαρμογή είναι η ύπαρξη ομοιότητας ή διαφοράς ανα ζεύγη για κάποια από τα προϊόντα. Για την αναπαράσταση αυτής της πληροφορίας χρησιμοποιούνται οι λεγόμενες κλίκες. Οι κλίκες είναι σύνολα από προϊόντα που έχουν όλα την ίδια σχέση μεταξύ τους (ομοιότητας ή διαφοράς). Ειδικότερα, αρκεί να υλοποιηθούν πλήρως όλες οι διακριτές κλίκες ομοιότητας που προκύπτουν, με κάθε προϊόν να είναι άμεσα συνδεδεμένο με την κλίκα στην οποία ανήκει (O(1) χρόνος). Στη συνέχεια μπορούμε να συμπεράνουμε ότι αν ένα προϊόν A διαφέρει από ένα προϊόν B, τότε το A θα διαφέρει κι από όλα τα προϊόντα που ανήκουν στην ίδια κλίκα με το B. Επομένως οι γνωστές διαφορές μεταξύ προϊόντων αρκεί να αναπαρασταθούν ως διαφορές μεταξύ κλικών. Τέλος, είναι φυσικό να υπάρχουν ζεύγη προϊόντων και κατ' επέκταση κλικών για τα οποία δεν έχουμε κάποια άμεση ή έμμεση πληροφορία ως προς το αν ταιριάζουν ή όχι.

Για παράδειγμα έστω ότι έχουμε τα προϊόντα με id a,b,c,d,e,f,g,h και έστω ότι ισχύουν τα εξής:

hash(a) = hash(b) = hash(c) = hash(d) = 0
hash(e) = hash(f) = 1
hash(g) = hash(h) = 2 και

a,b	όμοια	c,e	όμοια
b,g	όμοια	c,f	όμοια
a,d	όμοια	e,h	διαφορετικά

Τότε, σχηματικά, οι δομές στη μνήμη θα έχουν την παρακάτω μορφή



3. ΔΥΑΔΙΚΟΣ ΤΑΞΙΝΟΜΗΤΗΣ

Για την αναγνώριση ταυτότητας των δεδομένων εισόδου της εφαρμογής, δηλαδή για να αποφασίζεται με επιτυχία αν δύο άγνωστα προϊόντα είναι όμοια ή διαφορετικά, κατασκευάζεται και χρησιμοποιείται ένας δυαδικός ταξινομητής. Ένας δυαδικός ταξινομητής είναι μια συνάρτηση που ταξινομεί στιγμιότυπα σε δύο κλάσεις π.χ. 0 (διαφορετικά προϊόντα) και 1 (όμοια προϊόντα). Για να μπορεί ένας ταξινομητής να αποφασίζει αποτελεσματικά σε ποιά κλάση θα αντιστοιχίσει κάποιο άγνωστο στιγμιότυπο, θα πρέπει πρώτα να εκπαιδευτεί κατάλληλα πάνω στα ήδη γνωστά δεδομένα. Για το λόγο αυτό, χωρίζουμε τα δεδομένα μας (την πληροφορία για τις σχέσεις μεταξύ ζευγών – dataset W) σε 3 μέρη:

- το training set, ένα υποσύνολο των αρχικών προϊόντων πάνω στο οποίο θα εκπαιδευτεί το μοντέλο (ο ταξινομητής) λαμβάνοντας κάθε φορά ένα ζεύγος από τα προϊόντα και την πληροφορία για το αν είναι όμοια ή όχι. Τα προϊόντα που θα αποτελέσουν το σύνολο του training set αντιστοιχούν σε όλα τα προϊόντα του dataset X για τα οποία παίρνουμε το 60% των πληροφοριών του dataset W. Προφανώς το μοντέλο τροφοδοτείται μόνο με ζευγάρια προϊόντων για τα οποία γνωρίζουμε τη σχέση τους (όμοια ή διαφορετικά).

- το testing set, δηλαδή το ύποσύνολο των προϊόντων στο οποίο θα εξασκηθεί το μοντέλο στο να προβλέπει αν δύο προϊόντα ταιριάζουν ή όχι βάσει της εκμάθησής του από το προηγούμενο στάδιο. Από την ακρίβεια (accuracy) που παίρνουμε από το testing set μπορούμε να κάνουμε εκτιμήσεις για την απόδοση του μοντέλου. Τα προϊόντα που θα αποτελέσουν το σύνολο του testing set αντιστοιχούν σε όλα τα προϊόντα του dataset X για τα οποία παίρνουμε το 20% των πληροφοριών στο dataset W. Προφανώς το μοντέλο τροφοδοτείται μόνο με ζευγάρια προϊόντων για τα οποία γνωρίζουμε τη σχέση τους (όμοια ή διαφορετικά).

- το validation set, δηλαδή το σύνολο των προΐόντων του dataset X στα οποία αντιστοιχεί το υπόλοιπο 20% των σχέσεων του dataset W. Το validation set χρησιμοποίται για την τελική εκτίμηση της απόδοσης του μοντέλου που κατασκευάστηκε. Ωστόσο στην εν λόγω εφαρμογή, μετά κι από τη διαδικασία του testing, το μοντέλο καλείται να κάνει προβλέψεις για κάθε δυνατό ζευγάρι που μπορεί να παραχθεί από όλο το dataset X. Η διαδικασία αυτή (all with all· θα αναλυθεί στη συνέχεια του paper) υπερκαλύπτει τη διαδικασία εκτίμησης του μοντέλου που θα γινόταν μόνο βάσει του validation set.

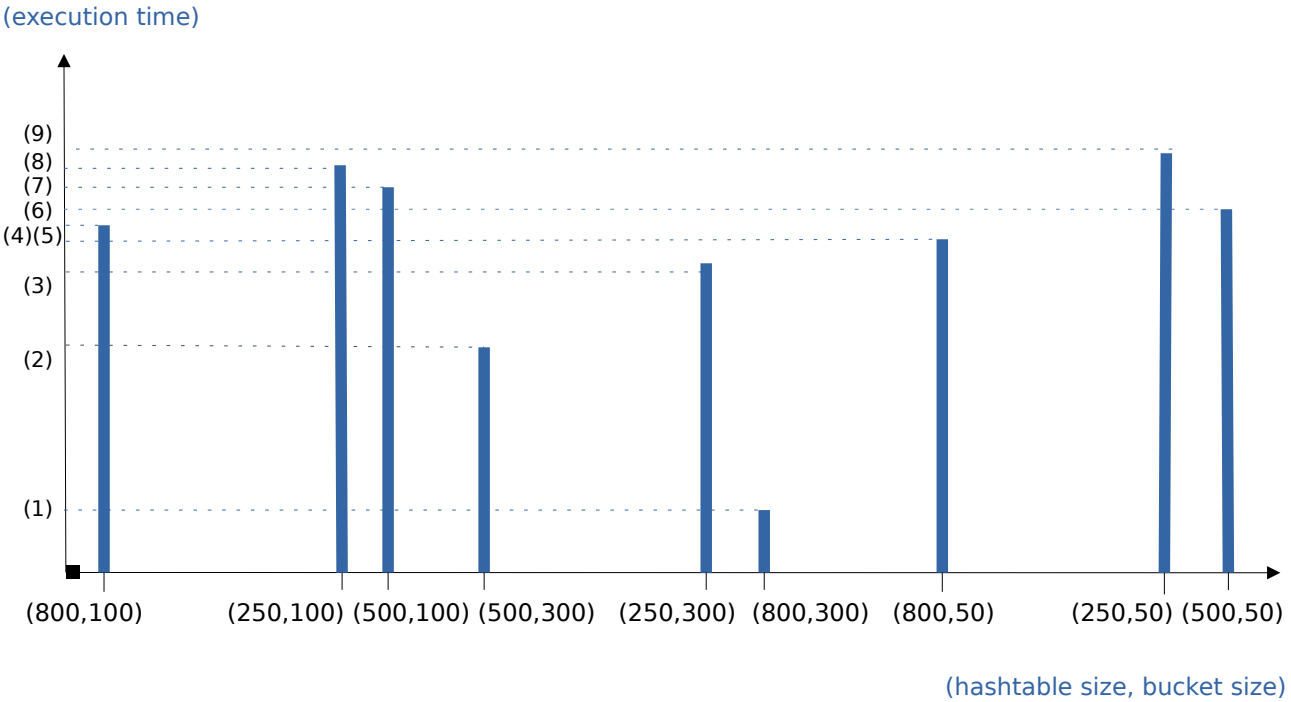
Για να δεχτεί το μοντέλο τα διάφορα ζεύγη προϊόντων έτσι ώστε πρώτα να εκπαιδευτεί με τις γνωστά πληροφορίες και στη συνέχεια να χρησιμοποιηθεί ως ένας αξιόπιστος ταξινομητής των άγνωστων δεδομένων, θα χρειαστεί πρώτανα γίνει η κατάλληλη προεπεξεργασία των δεδομένων. Η διαδικασία αυτή περιγράφεται αναλυτικότερα στην επόμενη ενότητα.

4. ΠΡΟΕΠΕΞΕΡΓΑΣΙΑ ΔΕΔΟΜΕΝΩΝ

Το μοντέλο μηχανικής μάθησης που χρησιμοποείται από την εφαρμογή δέχεται δεδομένα σε μορφή διανυσμάτων σταθερού μήκους. Για την μετατροπή των προϊόντων, τα οποία βρίσκονται σε μορφή κειμένων (αποθηκευμένα με τον τρόπο που περιγράφηκε στη $2^{η}$ ενότητα), στη ζητούμενη μορφή χρησιμοποιείται το μοντέλο Bag Of Words ή BOW. Η αναπαράσταση αυτή μετατρέπει κείμενο σε διανύσματα σταθερού μήκους μετρώντας πόσες φορές εμφανίζεται κάθε λέξη στο κείμενο.

Ειδικότερα, μας ενδιαφέρει ο αριθμός εμφανίσεων οποιασδήποτε λέξης του λεξιλογίου μας (dataset X) σε κάθε κείμενο (προϊόν) ξεχωριστά. Για να διατηρήσουμε την πληροφορία αυτή χρησιμοποιείται ένας πίνακας κατακερματισμού, με αλυσίδες υπερχείλισης και κάθε εγγραφή του πίνακα αντιστοιχεί σε μια λέξη του λεκιλογίου. Σε αυτή την περίπτωση η ίδια η λέξη χρησιμοποιείται ως το μοναδικό χαρακτηριστικό για την αποθήκευση και την ανάκτησή της από τον πίνακα σε O(1) χρόνο. Όπως και με τα ίδια τα προϊόντα έτσι με το λεξιλόγιο, ο μεγάλος όγκος δεδομένων και η ανάγκη για τη συχνή ανάκτησή τους οδήγησαν στην επιλογή της συγκεκριμένης δομής. Επιπλέον, για κάθε έγγραφη (δηλαδή λέξη) μέσα στον πίνακα διατηρείται και η πληροφορία για το σε ποιά προϊόντα (κείμενα) αυτή εμφανίζεται αλλά και πόσες φορές.

Ορίζουμε ένα πίνακα κατακερματισμού μεγέθους 500 κάδων (buckets), με κάθε κάδο να έχει μέγεθος 100bytes, ώστε να έχουμε μια καλή ισορροπία ανάμεσα σε χρόνο εκτέλεσης και κατανάλωση μνήμης. Στα παρακάτω διαγράμματα φαίνεται αντίστοιχα η μεταβολή των δύο αυτών παραμέτρων για τα διάφορα μεγέθη του hashtable και των buckets του.



όπου

(1): 4m 0.294s [(800,300), trainSize:682,335]

(2): 4m 9.556s [(500,300),trainSize:682,364]

(3): 4m 47.798s [(250,300),trainSize:682,370]

(4): 5m 6.518s [(800,50), trainSize:682,179]

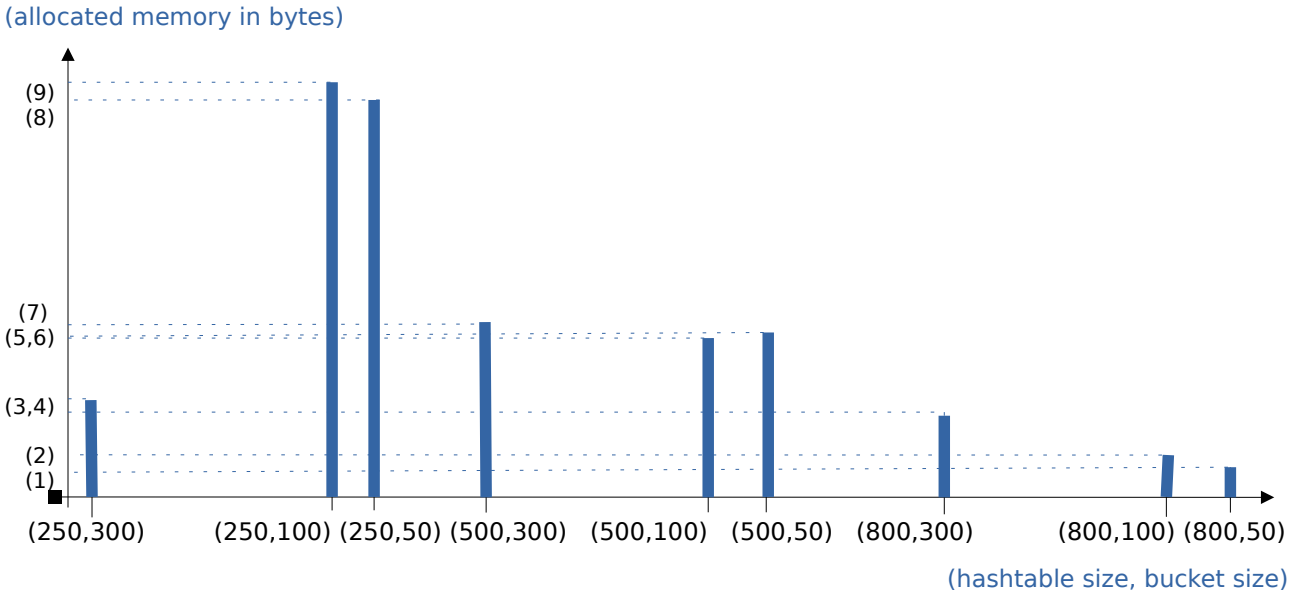
(5): 5m 9.274s [(800,100), trainSize:682,179]

(6): 5m 16.208s [(500,50), trainSize:682,366]

(7): 5m 32.557s [(500,100), trainSize:682,361]

(8): 5m 44.526s [(250,100),trainSize:682,365]

(9): 5m 49.588s [(250,50), trainSize:682,375]



όπου

(1): 80,746,814,281 b [(800,100)]

(2): 80,746,949,797 b [(800,50)]

(3): 80,768,518,824 b [(800,300)]

(4): 80,768,987,668 b [(250,300)]

(5): 80,796,177,031 b [(500,100)]

(6): 80,797,258,813 b [(500,50)]

(7): 80,797,297,542 b [(500,300)]

(8): 80,892,023,169 b [(250,100)]

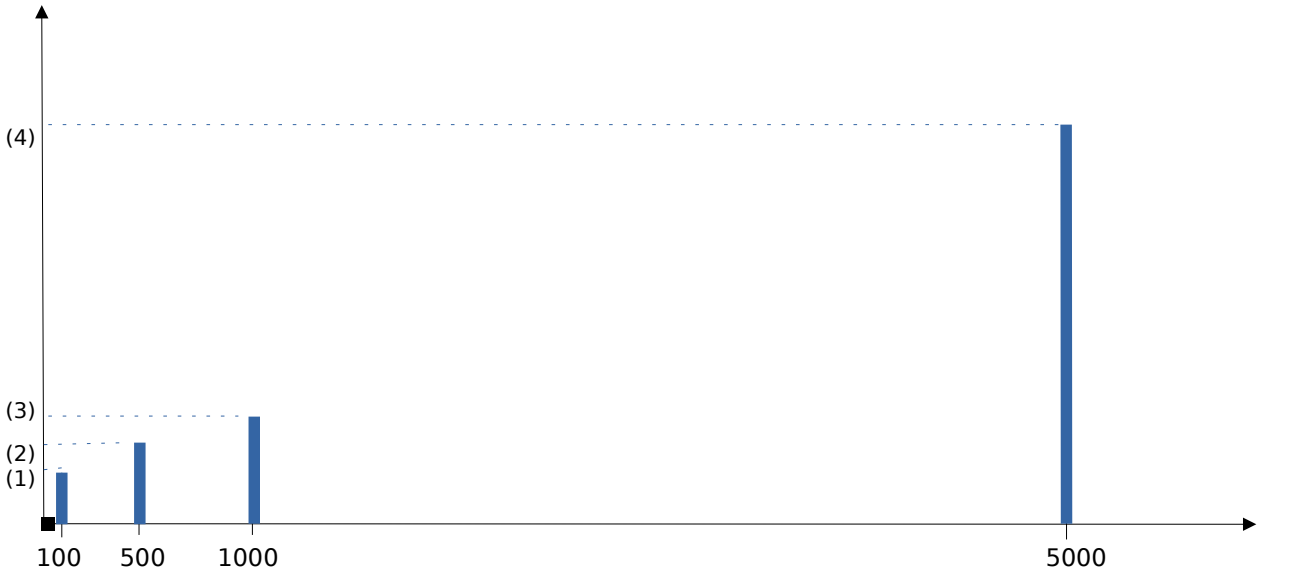
(9): 80,891,998,193 b [(250,50)]

Να σημειωθεί ότι για την καλύτερη αποτύπωση της πληροφορίας ενός κειμένου σε μορφή διανύσματος αποφεύγεται να ληφθούν υπόψη κοινές λέξεις όπως άρθρα ή αντωνυμίες καθώς και σημεία στήξης. Για τον ίδιο σκοπό, αποφεύγεται και η δημιουργία διπλότυπων λέξεων, αφού όλες οι λέξεις πρώτου χρησιμοποιηθούν μετατρέπονται σε πεζούς χαρακτήρες. Παρόλο που τα παραπάνω μέτρα είναι ιδιαίτερα χρήσιμα, για να δοθεί ακόμα μεγαλύτερη έμφαση στις σημαντικότερες λέξεις του λεξιλογίου (οι οποίες θα επιλεχθούν αργότερα και ως ενδεικτικές για κάθε κείμενο) χρησιμοποιείται επιπλέον η τεχνική TF-IDF (Text Frequency – Inverse Document Frequency). Για κάθε λέξη στον πίνακα κατακερματισμού και για κάθε κείμενο στο οποίο αυτή εμφανίζεται ξεχωριστά, υπολογίζεται η τιμή tf-idf η οποία αυξάνει ανάλογα με τον αριθμό των εμφανίσεων της λέξης στο συγκεκριμένο κείμενο και μειώνεται με τον συνολικό αριθμό των κειμένων στα οποία εμφανίζεται. Έτσι για κάθε λέξη έχουμε και από μια tf-idf τιμή της για κάποιο κείμενο στο οποίο εμφανίζεται.

Το διάνυσμα που εξάγεται από τη δομή BOW για ένα κείμενο αποτελείται από στήλες με όλες τις λέξεις του λεξιλογίου και τιμές τις αντίστοιχες tf-idf τιμές (0 αν κάποια λέξη δεν εμφανίζεται στο συγκεκριμένο κείμενο). Επειδή το λεξιλόγιο αποτελείται από υπερβολικά μεγάλο αριθμό λέξεων και τα διανύσματα που θα παίρναμε για κάθε κείμενο θα ήταν αντίστοιχα μεγάλων διαστάσεων, επιλέγουμε τις 1000 πιο σημαντικές λέξεις του λεξιλογίου για τη διαδικασία της διανυσματοποιήσης. Για την αξιολόγηση των λέξεων σε λιγότερο ή περισσότερο σημαντικές εξετάζουμε τη μέση tf-idf τιμή μιας λέξης από όλα τα κείμενα στα οποία αυτή εμφανίζεται. Για την επιλογή των 1000 πιο σημαντικών λέξεων χρησιμοποιείται δομή min-binary-heap 1000 κόμβων. Η δομή αυτή κρίθηκε ως η πλέον κατάλληλη για την επιλογή των σημαντικότερων λέξεων, διότι επιτρέπει την γρήγορη εύρεση του κόμβου με τη 1000-οστή μικρότερη μέση τιμη tf-idf (σε O(1) χρόνο), αλλά και την αρκετά γρήγορη εξαγωγή του (O(logn)) εάν βρεθεί κάποια “σημαντικότερή” λέξη.

Ο αριθμός των 1000 σημαντικότερων λέξεων επιλέχθηκε ως μια λύση με σχετικά μικρή επιβάρυνση στο συνολικό χρόνο εκτέλεσης της εφαρμογής και σχετικά καλή αναπαράσταση του λεξιλογίου από τα διανύσματα που παράγονται και κατ’ επέκταση σχετικά καλή απόδοση του ταξινομητή. Στα παρακάτω διαγράμματα φαίνονται η μεταβολές τόσο στο χρόνο εκτέλεσης όσο και στην απόδοση του μοντέλου για τους διάφορους αριθμούς σημαντικών λέξεων που μπορεί να επιλεχθούν.

(execution time)



(number of significant words)

όπου

(1): 4m 32.722s [100 significant words & training size 682,370]

(2): 4m 58.693s [500 significant words & training size 682,361]

(3): 5m 29.787 [1000 significant words & training size 683,262]

(4): 9m 42.241s [5000 significant words & training size 678,318]

(accuracy of testing step)

(number of significant words)

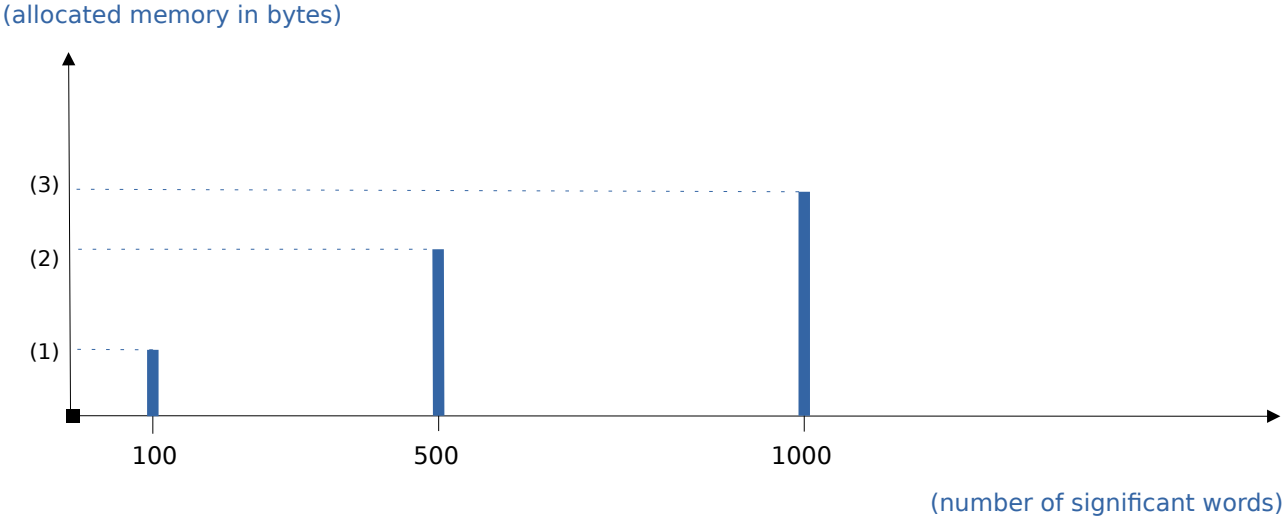
όπου

(1): 0.8014 [100 significant words & training size 682,370]

(2): 0.8165 [500 significant words & training size 682,361]

(3): 0.8247 [1000 significant words & training size 683,262]

(4): 0.8373 [5000 significant words & training size 678,318]



όπου

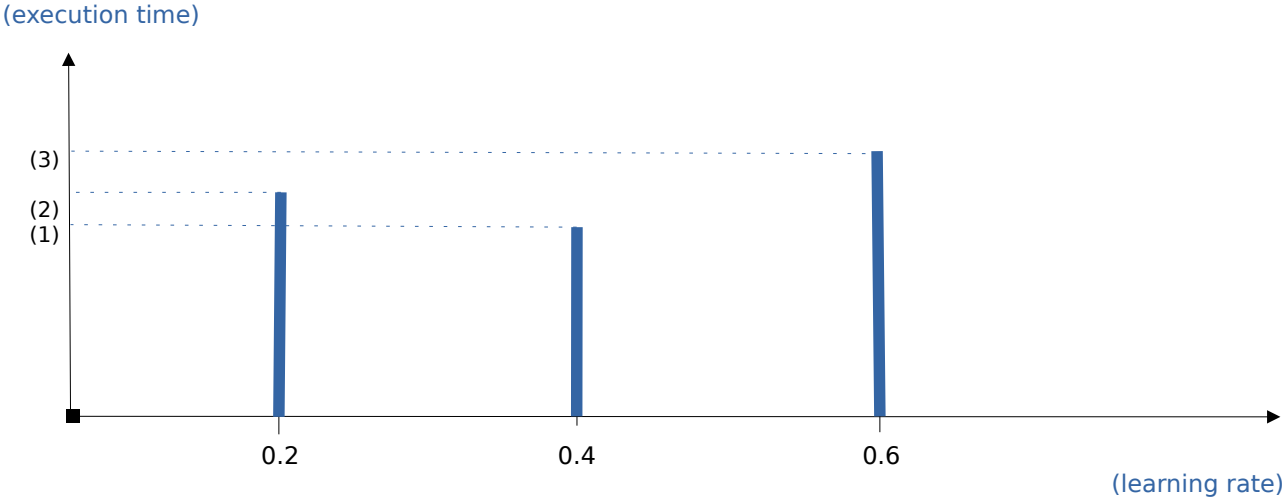
(1): 80,749,234,987 b [100 significant words & training size 682,370]

(2): 80,780,024,501 b [500 significant words & training size 682,361]

(3): 80,798,161,339 b [1000 significant words & training size 683,262]

5. ΜΟΝΤΕΛΟ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ - ΕΚΠΑΙΔΕΥΣΗ

Η υλοποίηση του μοντέλου της εφαρμογής βασίζεται στον αλγόριθμο ταξινόμησης μηχανικής μάθησης, δυαδική λογιστική παλινδρόμηση (binary logistic regression). Η λογιστική παλινδρόμηση είναι ένας γραμμικός ταξινομητής, όπου οι συντελεστές της αντιστοιχούν στις τιμές των διανυσμάτων που δημιουργούνται για κάθε προϊόν και οι εκτιμητές των συντελεστών ονομάζονται βάρη πρόγνωσης. Κατά την εκπαίδευση του μοντέλου στόχος είναι η εύρεση των βέλτιστων βαρών πρόγνωσης χρησιμοποιώντας τις διαθέσιμες πληροφορίες, έτσι ώστε οι προβλέψεις του μοντέλου για τα ζεύγη προϊόντων να τείνουν κατά το δυνατόν περισσότερο προς τις πραγματικές σχέσεις (ομοιότητα ή διαφορά). Για τον υπολογισμό των βέλτιστων βαρών πρόβλεψης υλοποιείται η ελαχιστοποίηση της συνάρτησης σφάλματος στη λογιστική παλινδρόμηση. Η διαδικασία αυτή μπορεί να επαναλαμβάνεται σε πολλά βήματα παράγοντας όλο και καλύτερες εκτιμήσεις και επηρεάζεται από ορισμένους παράγοντες. Συγκεκριμένα, η διαφοροποίηση κάθε βάρους πρόγνωσης επηρεάζεται από τον βαθμό εκμάθησης του μοντέλου (learning rate). Μετά από διάφορες δοκιμές η τιμή του βαθμού εκμάθησης ορίστηκε να είναι ίση με 0.6, ως η πλέον αποτελεσματική όπως φαίνεται και στο παρακάτω διάγραμμα.



όπου

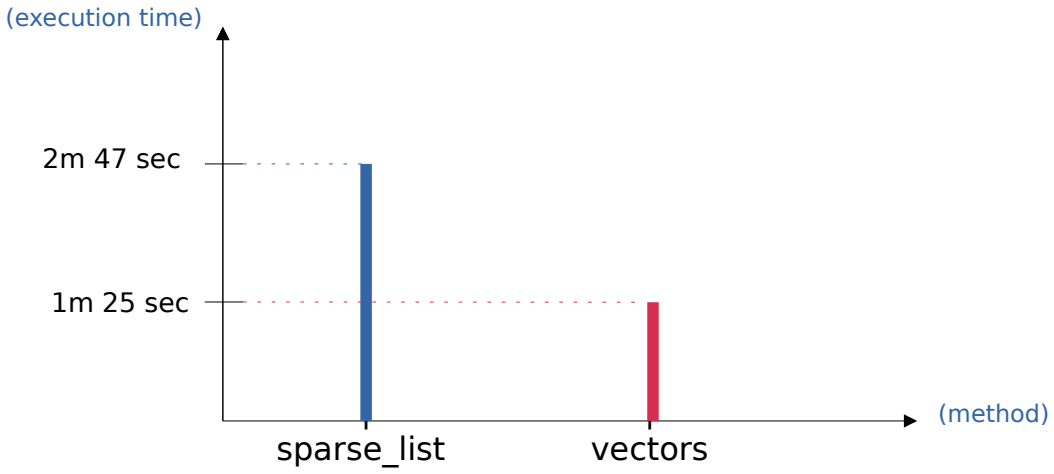
(1): 4m 2.6s [training size: 678,551]

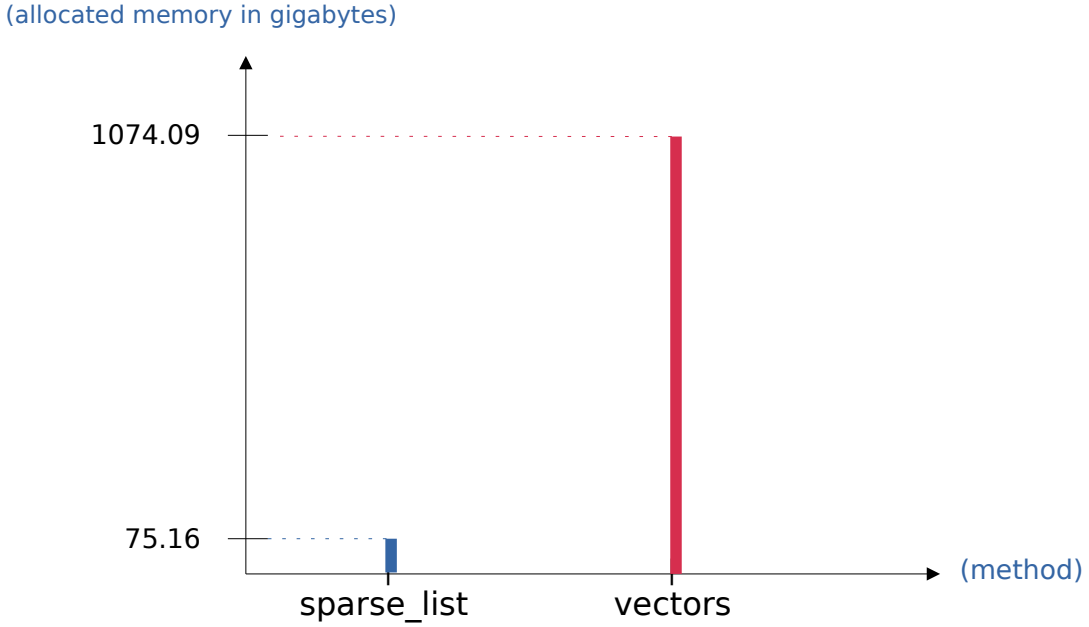
(2): 3m53.456s [training size: 682,437]

(3): 4m 14.521s [training size: 685,405]

Όσον αφορά στα διανύσματα του ζεύγους προϊόντων με τα οποία τροφοδοτείται κάθε φορά το μοντέλο για την εκπαίδευσή του, αναφέρθηκε νωρίτερα ότι είναι σταθερού μήκους διάστασης 1000 το καθένα (όσες και οι σημτικές λέξεις του λεξιλογίου που αποθηκεύονται στο BOW). Δεδομένου ότι το σύνολο του λεξιλογίου αποτελείται από υπερβολικά πολύ περισσότερες λέξεις σε σχέση με τις 1000 σημαντικότερες που επιλέγονται, είναι λογικό για τα περισσότερα πρϊόντα να μην υπάρχουν οι περισσότερες λέξεις και οι τιμές των αντίστοιχων διανυσμάτων για τις λέξεις αυτές να είναι μηδενικές. Για την αποφυγή δέσμευσης περιττής μνήμης για όλες αυτές τις μηδενικές τιμές τα διανύσματα των προϊόντων αποθηκεύονται με τη μορφή sparse πινάκων. Οπότε ουσιαστικά καθ’ όλη τη διάρκεια της εκπαίδευσης του μοντέλου διατηρούνται αποκλειστικά οι μη μηδενικές τιμές των διανυσμάτων των προϊόντων και μόνο για όσο χρειαστεί δεσμεύονται οι πίνακες 2000 θέσεων τη φορά (1000 για κάθε προϊόν σε ένα ζεύγος). Συγκεκριμένα, το μοντέλο κατά την εκπαίδευση του αναμένει μια δομή sparse-list της οποίας κάθε node αποτελείται από τον sparse πίνακα κάθε ζεύγους. Προτιμήθηκε αυτή η υλοποίση, έναντι ενός μεγάλου sparse απο όλα τα ζευγάρια, αφού έτσι η μνήμη που δεσμεύεται ελαχιστοποιείται, καθώς μειώνονται τα allocs και reallocs του προγράμματος. Η δομή του sparse-list, αφού χρησιμοποιείται και κατά το testing του μοντέλου, κρατά επιπλέον μεταβλητές, οι οποίες διευκολύνουν την συλλογή στατιστικών όπως η επιτυχεία του μοντέλου στη συγκεκριμένη λίστα ζευγαριών, αλλά και το λόγο των labels “0”, “1”, που βοηθά στο balancing του threashold του μοντέλου.

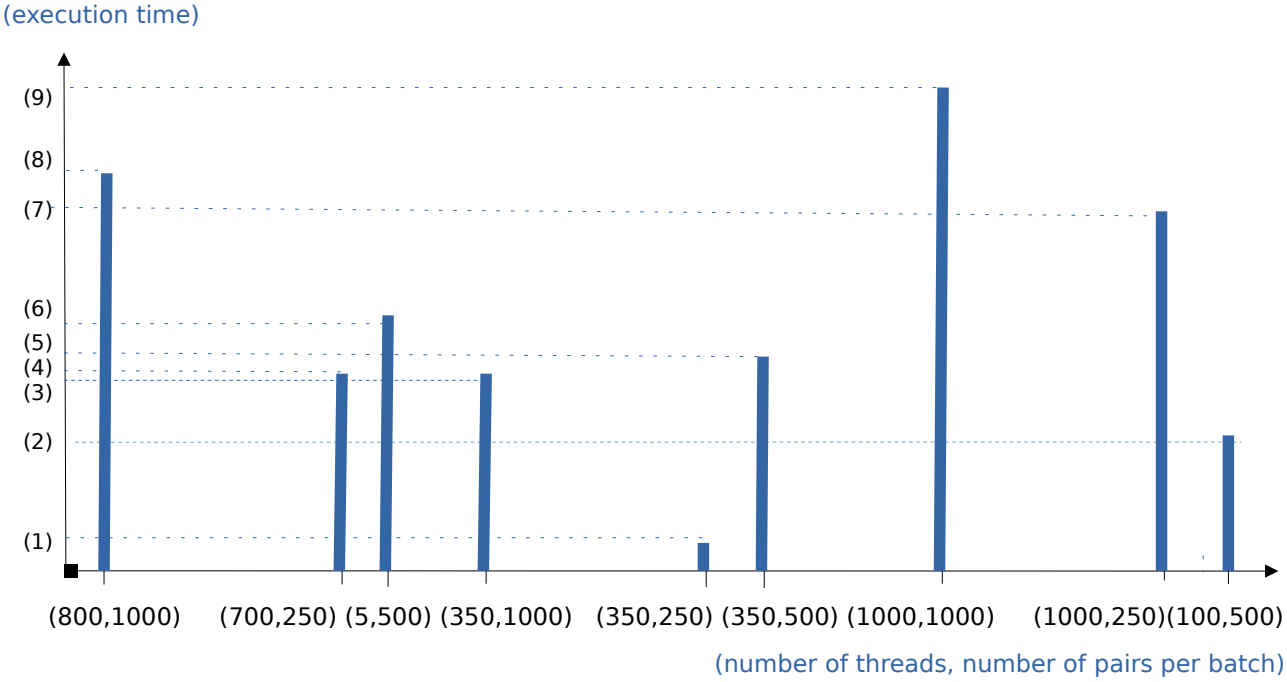
Εκτός από τη μέθοδο μετατροπής των διανυσμάτων σε sparse πίνακες εξετάσθηκε και η περίπτωση διατήρησης των διανυσμάτων ως έχουν (vectors). Παρακάτω φαίνονται πειραματικά οι διαφορές μεταξύ των δύο μεθόδων.





Για την εκπαίδευση του μοντέλου η εφαρμογή υλοποιεί τη μέθοδο επαναληπτικής μάθησης. Ειδικότερα, τα στιγμιότυπα (ζεύγη προϊόντων) εξετάζονται ανά δέσμες (batches) και μετά από κάθε αξιολόγηση ενός τυχαίου batch από το training set πραγματοποιείται επανυπολογισμός των βαρών όλων των συντελεστών. Η διαδικασία αυτή ονομάζεται στοχαστική ανάλυση καθόδου παραγώγου (stochastic gradient descent) και εκτελείται σε συνδυασμό με batch-training.

Ο διαχωρισμός των δυνατών ζευγαριών προϊόντων του training set σε batches επιτρέπει την ενσωμάτωση του πολυνηματισμού (threads) με σκοπό τη μείωση του χρόνου εκτέλεσης της εφαρμογής. Συγκεκριμένα κάθε ένα thread αναλαμβάνει την αξιολόγηση ενός batch για τον υπολογισμό των αντίστοιχων βαρών πρόγνωσης. Όλα τα threads συγχρονίζονται μεταξύ τους έτσι ώστε αφού τελειώσουν όλα με μια αξιολόγηση των batches τους, να γίνεται ο επανυπολογισμός των συντελεστών βάσει των τελευταίων αξιολογήσεων από όλα τα batches σε συνδυασμό και με τον βαθμό εκμάθησης. Μετά τον επανυπολογισμό των βαρών πρόγνωσης τα threads προχωρούν επίσης συγχρονισμένα στην επόμενη επανάληψη της διαδικασίας εκμάθησης. Φυσικά ο αριθμός των batches και κατ’ επέκταση και των threads παίζει σημαντικό ρόλο στο χρόνο εκτέλεσης της εφαρμογής αλλά και στην ακρίβεια του μοντέλου. Συγκεκριμένα, για το βήμα της εκπαίδευσης αποφασίσθηκε να χρησιμοποιούνται 700 threads και batches των 500 ζευγαριών, ύστερα από πειραματικές μετρήσεις που έγιναν, όπως φαίνεται και από το παρακάτω διάγραμμα.



όπου

(1): 3m 35.395s [training size 678,318]

(2): 4m 3.021s [training size 682,355]

(3): 4m 23.427s [training size 682,318]

(4): 4m 24.626s [training size 678,318]

(5): 4m25.265s [training size 682,359]

(6): 4m 40.498s [training size 682,362]

(7): 5m 10.369s [training size 678,318]

(8): 5m 21.291s sec [training size 682,329]

(9): 5m 44.977s [training size 682,336]

6. ΜΟΝΤΕΛΟ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ – ΕΛΕΓΧΟΣ - ΑΚΡΙΒΕΙΑ

Μετά τη διαδικασία εκπαίδευσης του μοντέλου ακολουθεί το βήμα ελέγχου της απόδοσής του. Σε αυτό το βήμα το μοντέλο τροφοδοτείται με το testing set που είχε φυλαχθεί στην αρχή της εφαρμογής, κατά τρόπο ανάλογο με το training. Ειδικότερα, ελέγχεται κατά πόσο οι προβλέψεις του μοντέλου για τα δυνατά ζεύγη προϊόντων πλησιάζουν στο 0 (για διαφορά) ή στο 1 (για ομοιότητα) και εάν χρειαστεί γίνονται αυξομειώσεις στις τιμές παραμέτρων που επηρεάζουν την απόδοση του μοντέλου μέχρις ότου τα αποτελέσματα να είναι ικανοποιητικά. Όταν αυτό συμβεί θα έχουν οριστικοποιηθεί και οι τιμές των βαρών πρόγνωσης του μοντέλου.

Όπως και κατά εκμάθηση, έτσι και στην περίπτωση του testing set τα δυνατά ζεύγη εξετάζονται σε batches. Κάθε batch του testing set εκτελείται και από ένα thread, αλλά σε αυτό το βήμα ο συγχρονισμός των threads απαιτείται μόνο κάθε φορά που ολοκληρώνεται η επεξεργασία ενός ολόκληρου testing set και όχι μετά από κάθε αξιολόγηση των batches. Και πάλι ο αριθμός των batches αλλά και των threads επηρεάζουν τον τελικό χρόνο εκτέλεσης της εφαρμογής. Μετά από δοκιμές διάφορων τιμών καταλήξαμε σε batches των 100 ζευγαριών προϊόντων και 700 threads. Οι μετρήσεις για τις υπόλοιπες δοκιμαστικές τιμές είναι οι εξής:

Threads	Pairs Per Batch	Time
350	70	3m 42.444s
700	70	3m 44.497s
1000	70	4m 34.850s
350	100	3m 38.832s
700	100	4m 3.719s
1000	100	4m 23.769s
350	200	4m 56.102s
700	200	5m 30.454s
1000	200	5m 19.641s

Η ακρίβεια που λαμβάνουμε από την εκτέλεση του testing αντικατοπτρίζει και την απόδοση του μοντέλου, το οποίο είναι έτοιμο να χρησιμοποιηθεί για άγνωστα δεδομένα με αποτελεσματικό και αξιόπιστο τρόπο.

8. ΤΕΛΙΚΑ ΑΠΟΤΕΛΕΣΜΑΤΑ & ΠΡΟΔΙΑΓΡΑΦΕΣ

Με βάση τις πειραματικές δοκιμές που παρουσιάστακαν παραπάνω μέσω διαγραμμάτων και χρησιμοποιώντας τις σχεδιαστικές επιλογές που αναφέρθηκαν ως βέλτιστες, η εφαρμογή εκτελείται σε χρόνο 5m 57s με κατανάλωση μνήμης 80,796,177,031 bytes. Το αποτέλεσμα της εκτέλεσης αυτής είναι το μοντέλο, μετά την εκπαίδευσή του (με 680540 προϊόντα), να επιτυγχάνει 0.8388 ακρίβεια στις προβλέψεις του.

Όλες οι εκτελέσεις πραγματοποιήθηκαν σε περιβάλλον Linux και συγκεκριμένα:

operating system information: GNU/Linux
kernel release: 5.4.0-60-generic
kernel version: #67~18.04.1-Ubuntu
processor type: x86_64
cpus: 4