Τεχνητή Νοημοσύνη 2η Εργασία

Μιλτιάδης Καλοδούκας 3190069

Ηλίας Καραγέωργας 3190073

**Αλγόριθμος ID3**

**-** *main()*

Αρχικά δίνονται διάφορες δυνατότητες σχετικά με τις τιμές που μπορούν να λάβουν τα arguments που θα δοθούν από το command line κατά την έναρξη της εκτέλεσης του προγράμματος. Η εντολή της εκτέλεσης θα πρέπει να έχει τη μορφή:

“python id3.py D”, όπου το D θα παίρνει τιμή είτε “Y” σε περίπτωση που πρέπει να γίνουν οι διαγνωστικοί έλεγχοι είτε “N” σε αντίθετη περίπτωση. Σε περίπτωση που το D πάρει τιμή “N” θα γίνει default εκτέλεση του αλγορίθμου και θα εμφανιστούν τα ανάλογα αποτελέσματα.

- *IMDB\_dataset()*

Γίνεται χρήση των συναρτήσεων του keras προκειμένου να αποθηκευτούν τα δεδομένα του dataset που θα χρησιμοποιηθούν κατά την εκτέλεση του αλγορίθμου. Στα tuples train και test αποθηκεύονται μόνο οι λέξεις που βρίσκονται εντός των ορίων που καθορίζουμε (skip\_top, num\_words) με βάση τη συχνότητα τους στο σύνολο των reviews του dataset.

*- create\_properties()*

Τα δεδομένα μετατρέπονται στην επιθυμητή μορφή. Κάθε κείμενο μετατρέπεται δηλαδή σε ένα διάνυσμα (λίστα) ιδιοτήτων με τιμές 0 ή 1 ανάλογα με το αν υπάρχει η κάθε ιδιότητα (λέξη) στο συγκεκριμένο κείμενο. Επιστρέφεται μία λίστα (properties) που περιέχει όλες τις παραπάνω λίστες.

*- calculate\_entropy()*

Γίνεται ο υπολογισμός τη εντροπίας με βάση τον ορισμό των διαφανειών.

*- calculate\_IG()*

Υπολογίζεται το information gain για κάθε ιδιότητα προκειμένου να μπορέσουμε παρακάτω να επιλέγουμε κάθε φορά την καλύτερη. Επιστρέφεται μία λίστα που περιέχει όλα τα information gain τα οποία είναι τοποθετημένα στη σειρά που βρίσκονται τα properties. Δηλαδή το IG[5] αντιστοιχεί στην 6η ιδιότητα (properties[5]).

*- select\_best\_property()*

Επιστρέφεται κάθε φορά ο δείκτης που αντιστοιχεί στην ιδιότητα με το μεγαλύτερο information gain.

*- sub\_tree()*

Αρχικά δημιουργούμε ένα dictionary με δύο στοιχεία. Το ένα από αυτά παίρνει ως key το 0 και value τον αριθμό των κειμένων που η συγκεκριμένη ιδιότητα (best\_property) δεν υπάρχει. Αντίστοιχα συμπληρώνουμε το δεύτερο στοιχείο (key = 1, value = # κειμένων που η ιδιότητα υπάρχει). Στη συνέχεια δημιουργούμε μία λίστα για κάθε τιμή που μπορεί να πάρει η ιδιότητα, άρα στη δική μας περίπτωση 2, και τοποθετούμε τα κείμενα σε αυτές με βάση το αν υπάρχει ή όχι η ιδιότητα σε αυτά. Χρησιμοποιούμε αυτό το διαχωρισμό για να ερευνήσουμε αν η συγκεκριμένη ιδιότητα (σε συνδυασμό με όσες έχουν προηγηθεί (αν δεν είναι η πρώτη ιδιότητα που ελέγχουμε)) καθορίζει την κατηγορία (C) του κειμένου. Αν περισσότερο από το 90% των κειμένων σε μία από τις λίστες έχουν την ίδια τιμή στην ιδιότητα που ελέγχουμε, τότε μπορούμε να υποθέσουμε ότι η τιμή της ιδιότητας που αντιστοιχεί σε αυτή τη λίστα καθορίζει (μαζί με τις προηγούμενες ιδιότητες του δέντρου) την κατηγορία του κειμένου και ανανεώνουμε ανάλογα το δέντρο. Διαφορετικά (<90%) πρέπει να επεκτείνουμε το δέντρο. Στην πρώτη περίπτωση θα αφαιρέσουμε από τη λίστα των διανυσμάτων των ιδιοτήτων (properties) τα κείμενα που έχουν την παραπάνω τιμή στην ιδιότητα που ελέγχουμε.

*- create\_tree()*

Το δέντρο που θα φτιάξουμε θα είναι της μορφής:

{

Bad:

{

0:

{

Good:

{

…

}

}

1: “0”

}

}

Επομένως χρησιμοποιούμε το δέντρο που προκύπτει κάθε φορά από την sub\_tree() και ανάλογα τις τιμές των φύλλων του αποφασίζουμε αν θα επεκτείνουμε ή όχι το δέντρο. Κάθε φορά θα χρησιμοποιούμε το σύνολο των κειμένων που στις ιδιότητες που βρίσκονται στους προγόνους του τρέχοντα κόμβου έχουν τις κατάλληλες τιμές.

*- id3()*

Αρχικά πραγματοποιούμε τους απαραίτητους ελέγχους προκειμένου να βεβαιωθούμε για την ορθότητα των δεδομένων και στη συνέχεια δημιουργούμε το δέντρο με τη βοήθεια της create\_tree().

*-* *predict()*

Γίνεται ο υπολογισμός της κατηγορίας του κάθε κειμένου με βάση το δέντρο που έχει προκύψει από τον ID3.

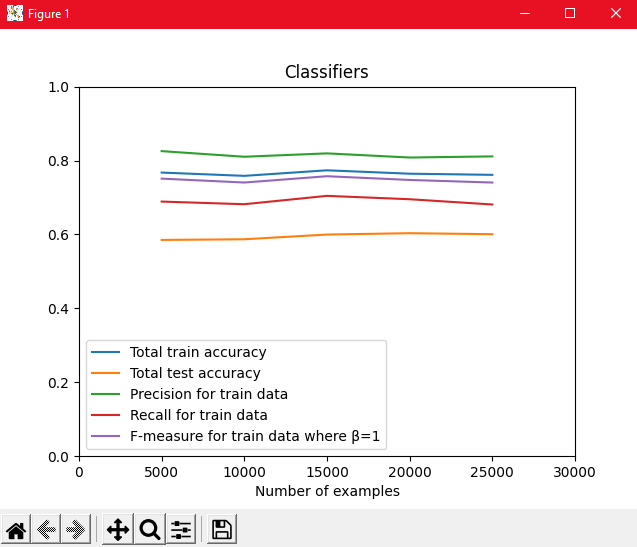
*- calculate\_classifiers()*

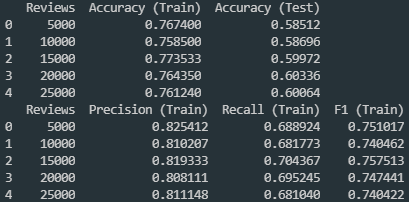
Χρησιμοποιούνται οι τύποι των διαφανειών για να υπολογιστούν τα accuracy, precision, recall και F-measure με β=1.

*- diagnostic\_tests()*

Χωρίζουμε το σύνολο των κειμένων σε 5 τμήματα (π.χ. το πρώτο περιέχει 5000 κείμενα, το δεύτερο 10000 κ.λπ.) και εκτελούμε τον id3 σε καθένα από αυτά. Στη συνέχεια πραγματοποιούμε για κάθε δέντρο που προέκυψε τους διαγνωστικούς ελέγχους και τέλος εμφανίζουμε τα αποτελέσματα που βρήκαμε με τη μορφή καμπυλών και πινάκων.

Αποτελέσματα εκτέλεσης αλγορίθμου χρησιμοποιώντας τις πρώτες 100 λέξεις που προκύπτουν από το IMDB dataset αφού αφαιρέσουμε τις 25 πιο συχνές. Χρησιμοποιήσαμε 5 διαφορετικά σημεία (από τα 5000 reviews μέχρι τα 25000 και κάθε φορά αυξάνεται το πλήθος τους κατά 5000).

**

**

**Αλγόριθμος Naive Bayes**

*- main()*

Αντίστοιχα με τον αλγόριθμο id3, η εντολή που θα δοθεί στο command line πρέπει να έχει τη μορφή:

“python NaiveBayes.py D”, όπου η μεταβλητή D παίρνει τις τιμές: Υ αν θέλουμε να γίνουν διαγνωστικά τεστ και Ν αν δεν θέλουμε να πραγματοποιηθούν διαγνωστικά τεστ. Στη συνέχεια γίνονται οι κατάλληλοι έλεγχοι για την εισαγωγή των δεδομένων και εκτελείται ο αλγόριθμος ανάλογα με το τι επιλέχθηκε.

- *create\_properties(), IMDB\_dataset(), calculate\_classifiers()*

Οι συναρτήσεις αυτές είναι ίδιες με αυτές που χρησιμοποιούνται στον id3.

- *calculate\_possibilty\_properties()*

Σε αυτή τη συνάρτηση υπολογίζουμε τις πιθανότητες να υπάρχει μία λέξη από το δυαδικό διάνυσμα που αναπαριστά το λεξιλόγιο (properties) σε ένα θετικό ή ένα αρνητικό review καθώς και την πιθανότητα να είναι θετικό ή αρνητικό το review που ελέγχουμε. Δηλαδή τα:

* P(X=0|C=1), P(X=1|C=1),P(X=0|C=0) και P(X=1|C=0) όπου Χ είναι η κάθε λέξη και C η κατηγορία.
* P(C=1) και P(C=0) όπου C η κατηγορία.

Αναλυτικότερα, αρχίζουμε υπολογίζοντας το PC0 (δηλαδή την πιθανότητα P(C=0)) καθώς και το PC1 (δηλαδή την πιθανότητα P(C=1)). Επίσης υπολογίζουμε πόσες φορές βρίσκεται κάθε λέξη του λεξιλογίου μας (properties) σε αντικείμενο που ανήκει στην κατηγορία 1 και στη κατηγορία 0.

Στη συνέχεια, υπολογίζουμε τις πιθανότητες:

1. κάθε λέξη να υπάρχει σε αντικείμενο κατηγορίας 1 (possibility\_pos)
2. κάθε λέξη να μην υπάρχει σε αντικέιμενο κατηγορίας 1 (possibility\_not\_pos)
3. κάθε λέξη να υπάρχει σε αντικείμενο κατηγορίας 0 (possibility\_neg)
4. κάθε λέξη να μην υπάρχει σε αντικείμενο κατηγορίας 0 (possibility\_not\_neg)

Στον υπολογισμό κάθε πιθανότητας από αυτές χρησιμοποιούμε και εξομάλυνση πιθανοτήτων με εκτιμήτρια Laplace στην οποία προσθέτουμε δύο ακόμη ψευτο-αντικείμενα εκπαίδευσης για κάθε κατηγορία όπου το ένα περιέχει την λέξη Χ και το άλλο δεν την περιέχει. Ο τρόπος που το κάνουμε αυτό είναι να προσθέτουμε δύο στον αριθμό των αντικειμένων κατηγορίας 0 (negative\_count), να προσθέτουμε δύο στον αριθμό των αντικειμένων κατηγορίας 1 (len(properties) - negative\_count ), να προσθέτουμε 1 στον αριθμό των αντικειμένων κατηγορίας 0 που βρέθηκε η κάθε λέξη (properties\_count\_neg) καθώς και να προσθέτουμε 1 στον αριθμό των αντικειμένων κατηγορίας 1 που βρέθηκε η κάθε λέξη (properties\_count\_pos).

Τέλος επιστρέφουμε τις πιθανότητες που υπολογίσαμε.

*- NaiveBayes()*

Στην συνάρτηση αυτή χρησιμοποιούμε τις πιθανότητες που βρήκαμε στην *calculate\_possibilty\_properties()* και κάνουμε μια εκτίμηση στις κατηγορίες των αντικειμένων που μας δίνονται. Έχουμε τον πίνακα categories στον οποίο αποθηκεύουμε την εκτίμηση κατηγορίας κάθε αντικειμένου που διαβάζουμε. Ο τρόπος που κάνουμε την εκτίμηση είναι να υπολογίζουμε το γινόμενο του τύπου του Bayes και για τις δύο κατηγορίες και στη συνέχεια να επιλέγουμε το μεγαλύτερο γινόμενο.

- *diagnostic\_tests()*

Αντίστοιχα με τα τεστ του id3, χωρίζουμε τα δεδομένα σε 5 partitions και τρέχουμε τον NaiveBayes προσθέτοντας κάθε φόρα ένα partition. Έτσι βρίσκουμε αποτελέσματα για διαφορετικά μεγέθη εισαγωγών ώστε να σχεδιάσουμε τα ζητούμενα διαγράμματα.

Αποτελέσματα εκτέλεσης αλγορίθμου χρησιμοποιώντας τις πρώτες 200 λέξεις που προκύπτουν από το IMDB dataset αφού αφαιρέσουμε τις 25 πιο συχνές. Χρησιμοποιήσαμε 5 διαφορετικά σημεία (από τα 5000 reviews μέχρι τα 25000 και κάθε φορά αυξάνεται το πλήθος τους κατά 5000).

