Εργασία στην Ανάκτηση Πληροφορίας

**Ονοματεπώνυμο: ΑΕΜ:**

Νικόλαος Καράμπινας 2355

Αστέριος Χουλιάρας 2428

**Περιγραφή**

Αρχικά το project έγινε κάτω από τη διαχείριση του maven για να υπάρχει οργάνωση στο project με τη βιβλιοθήκη Lucene. Η βιβλιοθήκη Lucene αποτελεί το σκελετό του project καθώς χρησιμοποιήθηκε για τη δημιουργία του αντεστραμμένου καταλόγου (inverted index), για τη χρήση του test data ως queries και τη βαθμολόγηση τους για την ομοιότητα τους με τα data του train set.

Χρησιμοποιήθηκαν 4 τεχνικές, μια unsupervised και 3 supervised. Η unsupervised τεχνική είναι η Knn (K nearest neighbors), ενώ η supervised είναι η Naive Bayes, η Rocchio και χρησιμοποιώντας το LingPipe API ο Polarity Classifier. Αναλυτικότερη περιγραφή της υλοποίησης παρακάτω.

**Υλοποίηση**

* **Lucene**

Η βιβλιοθήκη Lucene της Apache χρησιμοποιήθηκε για τη δημιουργία του καταλόγου από τα δεδομένα για training που μας δόθηκαν. Χρησιμοποιεί συναρτήσεις για το stemming και τα σωστά stopwords και δημιουργεί σε αρχείο τον inverted index ο οποίος χρησιμοποιείται από τις τεχνικές που υλοποιούνται για την εκτίμηση των δεδομένων στο test set. Επίσης με τη Lucene χρησιμοποιούμε τα δεδομένα στο τεστ ως ερωτήματα (queries) και με σωστή προ επεξεργασία για το κάθε ερώτημα παίρνουμε ένα score για την ομοιότητα του εγγράφου στο test με τα έγγραφα του train από τον inverted index. Για την ομοιότητα χρησιμοποιήθηκε ο TF-IDF και το cosine similarity μέσα από τις συναρτήσεις που παρέχει η βιβλιοθήκη.

* **Naïve Bayes**
* **KNN (K Nearest Neighbor)**

Ο Knn αποτελεί μια unsupervised τεχνική η οποία κάνει τις εκτιμήσεις ανάλογα με τους γείτονες του εγγράφου που επεξεργαζόμαστε. Για να λειτουργήσει ο Knn απαιτεί το κατάλληλο HashMap structure που περιέχει όλα τα έγγραφα και την ομοιότητα του με τα υπόλοιπα.

Αρχικά με τη μέθοδο findNeighbors() ελέγχονται όλα τα έγγραφα ώστε να δημιουργηθεί μια νέα δομή neighborsMap το οποίο περιέχει για κάθε έγγραφο μια λίστα με τους γείτονες του. Στην υλοποίηση του κώδικα για το αντικείμενο Document ή έγγραφο όπως αναφέρεται, χρησιμοποιείται η κλάση Review η οποία περιέχει το ID του document και την κλάση του, δηλαδή αν είναι θετικό ή αρνητικό. Έτσι για την εύρεση γειτόνων εντοπίζονται τα έγγραφα με τις μεγαλύτερες ομοιότητες ως προς το έγγραφο που εξετάζεται και δημιουργούνται ενδιάμεσες δομές για να αποθηκευτούν οι top-k γείτονες και να δημιουργηθεί τελικά το neighborsMap που αναφέρθηκε παραπάνω.

Έπειτα εκτελείται η μέθοδος findClass() η οποία δέχεται τον neighborsMap και για κάθε έγγραφο ελέγχει αν οι top-k γείτονες έχουν κλάση 1 (positive) ή 0 (negative). Αθροιστικά ελέγχουμε ποια τιμή είναι η μεγαλύτερη και έτσι επιλέγεται η κλάση του εγγράφου.

Τα αποτελέσματα αποθηκεύονται στη λίστα results η οποία επιστρέφεται στη main, ταξινομείται βάση του document id και γράφει τα αποτελέσματα στο αρχείο predictions.txt.

* **Rocchio**
* **Polarity Classifier (LingPipe API)**

Για να μελετήσουμε περαιτέρω τεχνικές classification, μέσα από [papers](http://www.cs.cornell.edu/home/llee/papers/cutsent.pdf), ασχοληθήκαμε με τη βιβλιοθήκη LingPipe που παρέχει πολλά εργαλεία για ανάλυση πληροφορίας. Χρησιμοποιώντας τα classifiers που παρέχει πετύχαμε πολύ καλή ακρίβεια σε σχετικά σύντομο χρόνο για τα δεδομένα μας.   
Έχουμε 3 βασικές μεθόδους και τη Main. Η Main για το train και για την ακρίβεια καλεί την train() και testTrain() (Ο κώδικας έχει μπει σε σχόλια για να γίνει απευθείας εκτέλεση με το test set), ενώ για το τελικό testing καλεί την train() και την test().

Η train() ξεκινάει και διαβάζει τα δεδομένα από το path που δίνουμε στη μεταβλητή directory. Επειδή έχουμε 2 κατηγορίες χρησιμοποιούμε το categories για να διαβάσουμε και από τις 2. Όταν κάνουμε το evaluation με το train set επιλέγουμε για train όλα τα data που το id τους στο 2ο χαρακτήρα δεν έχουν το 9 (Στον κώδικα είναι σε σχόλιο γιατί δε χρειάζεται για το τελικό τεστ). Έτσι στο testTrain() χρησιμοποιούμε αυτά τα δεδομένα για να ελέγξουμε το μοντέλο μας. Χρησιμοποιώντας το API και τα classifiers κάνουμε train το μοντέλο και είναι έτοιμο για να γίνει evaluation ή το testing.

Η test() επιλέγει τα αρχεία από το σωστό path σε ένα File που δημιουργούμαι και ξεκινάει να διαβάζει στη μνήμη ένα-ένα τα test reviews. Για κάθε αρχείο μέσω του Classification.classify() επιλέγει τη σωστή κλάση για το αρχείο και εκτυπώνει τα αποτελέσματα στο predictions.txt.

( Η testTrain() είναι ίδια με τη διαφορά ότι επιλέγει τα files από το train set)

Υπάρχουν όλες οι πληροφορίες για τη βιβλιοθήκη και τις μεθόδους της στο παρακάτω Link:

<http://alias-i.com/lingpipe/index.html>

**Σύγκριση αποτελεσμάτων**

* **KNN**
* **Naïve Bayes**
* **Rocchio**
* **Polarity Classifier (LingPipe API)**

Κάνοντας training στο μοντέλο και evaluation με τις train() και testTrain(), όπου επιλέγουμε για train 22618 reviews και για testing 2382 reviews. Το μοντέλο βρίσκει 1974 reviews σωστά, δηλαδή το ποσοστό ακρίβειας είναι:

**82.87%.**

Για το testing στο test data set τα αποτελέσματα είναι αποθηκευμένα στο:

predictions\_lpc.txt