Καρέλης Μάριος ΑΜ 1067399

Λιούμη Κυριακή ΑΜ 1067410

**Αναφορά Υλοποιητικού Πρότζεκτ Ανάκτησης**

**Περιεχόμενα:**

* Αρχικές πληροφορίες περί υλοποίησης…………..Σελίδα 2
* Γενική εξήγηση λειτουργίας του προγράμματος …………..Σελίδα 3
* Αναλυτική επεξήγηση ερωτήματος 1 …………..Σελίδα 4
* Αναλυτική επεξήγηση ερωτήματος 2 …………..Σελίδες 5-6
* Αναλυτική επεξήγηση ερωτήματος 3 …………..Σελίδα 7-9
* Αναλυτική επεξήγηση ερωτήματος 4 …………..Σελίδα 10
* Προβλήματα που προέκυψαν κατά την διάρκεια υλοποίησης …………..Σελίδα 11
* Σχολιασμός τελικών αποτελεσμάτων…………..Σελίδα 11

Αρχικές πληροφορίες περί υλοποίησης:

Γλώσσα προγραμματισμού: Python

Εκδόσεις : Python 3.7.9 , Python 3.8.5

Περιβάλλον Υλοποίησης : Visual studio code

Βιβλιοθήκες που χρησιμοποιήθηκαν :

* Numpy
* Pandas
* Scikit-learn
* Keras
* Tensorflow
* Elasticsearch
* Csv
* rank\_bm25
* Matplotlib

Για την εκτέλεση του προγράμματος χρειάζεται να αλλάξουν τα paths των movies.csv και ratings.csv.

Απαιτείται η σύνδεση σε Elasticsearch και Κibana πριν την εκτέλεση του προγράμματος.

Ερωτήματα που τρέχουν κανονικά : 1 ,2 ,3

Ερωτήματα που δεν τρέχουν : 4

Γενικά:

Η γενική μορφή των ερωτημάτων είναι η εξής:

1. Import τις βιβλιοθήκες που θα χρειαστούν
2. Πραγματοποιούμε την σύνδεση με την Elasticsearch και φτιάχνουμε ένα στιγμιότυπο με όνομα es.
3. Φτιάχνουμε μια συνάρτηση με όνομα start,στην οποία:
   1. Ο χρήστης δίνει τα δεδομένα, δηλαδή τον τίτλο της ταινίας και το id του χρήστη, όπου χρειάζεται.
   2. Στα ερωτήματα 2,3 και 4 ελέγχουμε αν υπάρχει χρήστης με το συγκεκριμένο id και αν δεν υπάρχει, έχουμε την δυνατότητα να δώσουμε ξανά id.
   3. Φτιάχνουμε ένα dictionary στην μορφή που θα ήταν ένα ερώτημα στην kibana, για να πάρουμε τις αντίστοιχες ταινίες ανάλογα με το input του χρήστη.
4. Φτιάχνουμε μια συνάρτηση με όνομα index\_and\_data, η οποία δημιουργεί ένα index στην Elasticsearch αν δεν υπάρχει ήδη και περνάμε στο index αυτό τα δεδομένα του movies.csv.
5. Φτιάχνουμε μια συνάρτηση με όνομα search\_es, στην οποία:
   1. Διαβάζουμε το ratings.csv
   2. Η Elasticsearch επιστρέφει τις σχετικές ταινίες ανάλογα με τον τίτλο της ταινίας που εισάγει ο χρήστης.
   3. Σε κάθε ερώτημα γίνεται κάποιο ranking για την σειρά των αποτελεσμάτων.
6. Τέλος, καλούμε τις συναρτήσεις που έχουμε φτιάξει και με μια try δοκιμάζουμε αν η ταινία που αναζητούμε υπάρχει στο database.

Ερώτημα 1:

Στο ερώτημα αυτό χρησιμοποιούμε την μετρική BM-25 για να ταξινομήσουμε τα αποτελέσματα της Elasticsearch.

Αρχικά, κάνουμε split στους τίτλους των ταινιών και στο δικό μας query, έτσι ώστε να χωρίσουμε τις προτάσεις σε λέξεις.

Στις λέξεις αυτές εφαρμόζουμε την μετρική BM-25, η οποία επιστρέφει κάποια scores, ανάλογα με την ομοιότητα των λέξεων. Όσο πιο όμοιες είναι οι λέξεις των τίτλων των ταινιών με τις λέξεις του query μας, τόσο πιο μεγάλο score θα έχουν.

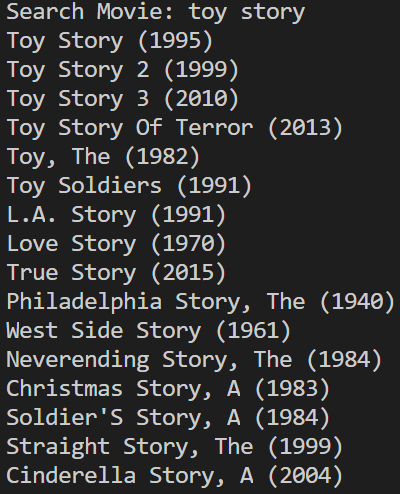
Ανάλογα με τα scores αυτά ταξινομούμε τις ταινίες με φθίνουσα σειρά και τις εκτυπώνουμε.

* Οι υπόλοιπες λειτουργίες του κώδικα αναφέρονται στην σελίδα 3

Παράδειγμα εκτέλεσης :

Εισάγουμε ως query το : toy story

Έχουμε ως αποτέλεσμα τα ακόλουθα :



\*Δείγμα των αποτελεσμάτων, υπάρχουν κι άλλα\*

Ερώτημα 2:

Από το αρχείο ratings.csv παίρνουμε τις στήλες movieId και rating.

Μέσος Όρος Ταινιών:

Φτιάχνουμε ένα defaultdict, στο οποίο για κάθε ταινία ξεχωριστά αποθηκεύουμε τα ratings όλων των χρηστών.

Για τις ταινίες που επιστρέφει η Elasticsearch σύμφωνα με το query μας, ψάχνουμε στο dictionary τα values των ταινιών αυτών.

Αν υπάρχουν values υπολογίζουμε τον μέσο όρο τους, αλλιώς αν κανένας χρήστης δεν έχει βαθμολογήσει την ταινία, βάζουμε 0.

Ratings χρήστη για τις ταινίες:

Φτιάχνουμε ένα map με τις ταινίες και τα ratings του συγκεκριμένου χρήστη που αναζητούμε.

Ελέγχουμε αν ο χρήστης έχει βαθμολογήσει τις επιστρεφόμενες από την Elasticsearch ταινίες και αν τις έχει βαθμολογήσει κρατάμε αύτη την βαθμολογία, αλλιώς βάζουμε σαν rating το 0.

BM-25:

Η επεξήγηση δίνεται στο ερώτημα 1.

Ένωση μετρικής:

Φτιάχνουμε ένα defaultdict στο οποίο βάζουμε για κάθε ταινία :

1. Τον μέσο όρο όλων τον χρηστών
2. Το rating του χρήστη που θέλουμε
3. Το score από την BM-25

Για κάθε ένα από αυτά τα 3, δίνουμε ένα συγκεκριμένο βάρος .

10% είναι ο μέσος όρος των ταινιών

60% είναι το rating του χρήστη

30% είναι το bm-25

και τα προσθέτουμε για να προκύψει η νέα μετρική.

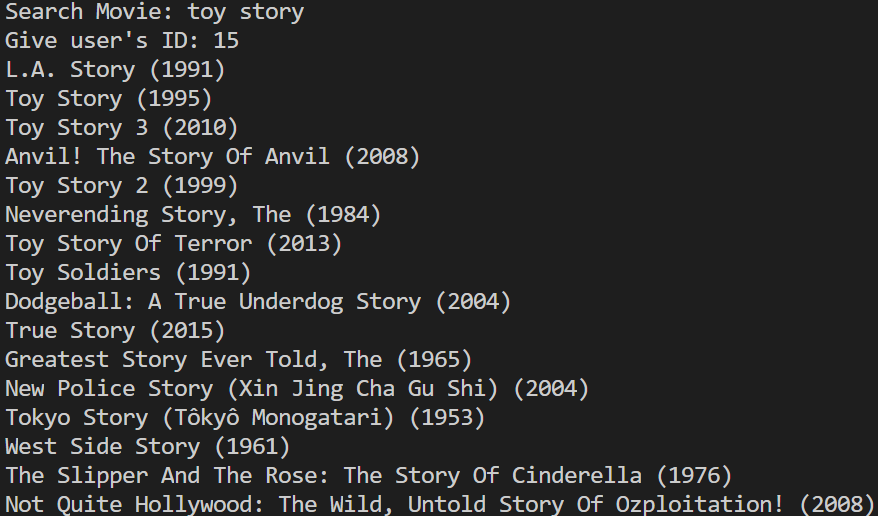
Τέλος, αντιστοιχίζουμε τους τίτλους των ταινιών με τα scores της μετρικής. Τα ταξινομούμε με φθίνουσα σειρά και εκτυπώνουμε τα αποτελέσματα.

* Οι υπόλοιπες λειτουργίες του κώδικα αναφέρονται στην σελίδα 3

Παράδειγμα εκτέλεσης :

Εισάγουμε ως query το : toy story

Εισάγουμε ως user\_id το : 15



Ερώτημα 3:

Στο ερώτημα αυτό έχουμε προσθέσει 4 καινούριες συναρτήσεις:

1. Την συνάρτηση hasWatched, στην οποία ελέγχουμε για τον κάθε χρήστη αν υπάρχει μέσος όρος για όλες τι κατηγορίες, δηλαδή αν έχει βαθμολογήσει ταινίες από όλες τις κατηγορίες. Αν κάποιος χρήστης δεν έχει βαθμολογήσει καμία ταινία για κάποια κατηγορία τότε βάζουμε ότι ο μέσος όρος είναι 0.
2. Την συνάρτηση add\_df η οποία βάζει σε ένα dataframe για κάθε χρήστη, τον μέσο όρο της κάθε κατηγορίας.
3. Την συνάρτηση find\_cluster, η οποία βρίσκει σε ποιόν cluster ανήκει ο χρήστης που αναζητούμε.
4. Η συνάρτηση genre\_div, η οποία χωρίζει τις ταινίες ανά κατηγορία για κάθε χρήστη και ύστερα υπολογίζει τον μέσο όρο για κάθε κατηγορία.

Χρήση συναρτήσεων:

Φτιάχνουμε ένα dataframe στο οποίο βάζουμε τους χρήστες.

Καλούμε τις συναρτήσεις genre\_div , hasWatched και add\_df για κάθε κατηγορία (δηλ. 19 φορές όσες και οι κατηγορίες).

Μετά το τέλος όλων των κλήσεων των συναρτήσεων, έχουμε ένα τελικό dataframe, το οποίο περιέχει για κάθε χρήστη, τον μέσο όρο της κάθε κατηγορίας.

K-Means:

Μετατρέπουμε το dataframe που αναφέραμε από πάνω σε ένα μητρώο.

Εφαρμόζουμε τον αλγόριθμο K-Means, χωρίζοντας τους χρήστες σε 6 clusters.

Δείχνουμε σε διάγραμμα τους clusters.

Για κάθε cluster φτιάχνουμε μια λίστα, η οποία περιέχει τους χρήστες που υπάρχουν σε κάθε cluster.

Καλούμε την συνάρτηση which\_cluster.

Έχοντας πλέον βρει τον cluster από την συνάρτηση, πάμε στον πίνακα του cluster αυτού, οποίος περιέχει όλους τους χρήστες που ανήκουν στον cluster. Και βρίσκουμε τον μέσο όρο όλων των χρηστών για κάθε ταινία. Στην περίπτωση που δεν υπάρχει μέσος όρος βάζουμε μηδέν.

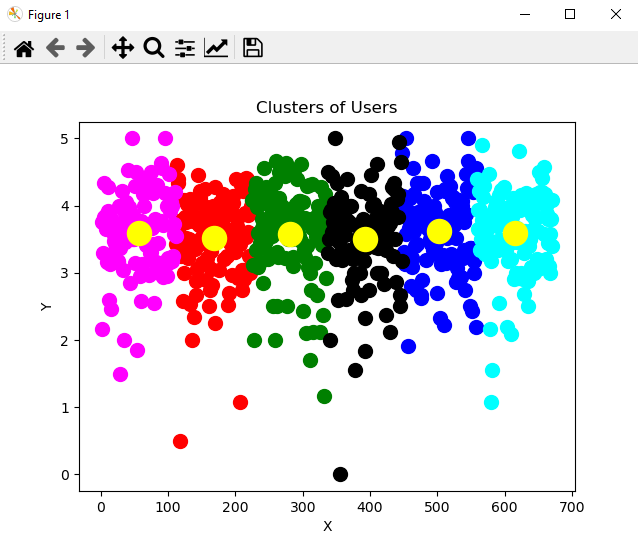
Στην συνέχεια, βρίσκουμε για τον χρήστη που αναζητούμε ποιες ταινίες δεν έχει βαθμολογήσει και σε αυτές βάζουμε τον μέσο όρο του cluster.

Tέλος προσαρμόζουμε τα νέα δεδομένα στην μετρική και εκτυπώνουμε τα αποτελέσματα.

\*Λεπτομέρειες για την μετρική υπάρχουν στο ερώτημα 2.\*

Παράδειγμα εκτέλεσης :

Διάγραμμα με clusters:



\*Για να συνεχίσει να τρέχει το πρόγραμμα κλείστε το διάγραμμα.\*

Εισάγουμε ως query το : toy story

Εισάγουμε ως user\_id το:15

Ερώτημα 4:

Στο ερώτημα αυτό χρησιμοποιούμε την συνάρτηση genre\_div από το προηγούμενο ερώτημα, ωστόσο με κάποιες αλλαγές:

* Χωρίζει τις ταινίες ανά κατηγορία και φτιάχνει μια λίστα για κάθε κατηγορία, όπου η κάθε λίστα περιέχει τους τίτλους των ταινιών.

Καλούμε την συνάρτηση αυτή 19 φορές, δηλαδή όσες και οι κατηγορίες.

Προετοιμασία για Word Embeddings:

1. Δημιουργούμε μια συνάρτηση με όνομα cleantext, η οποία αφαιρεί τα σημεία στίξης από τους τίτλους των ταινιών.
2. Δημιουργούμε μια συνάρτηση με όνομα new, στην οποία:
   1. Χρησιμοποιούμε την συνάρτηση lower(), έτσι ώστε να μην υπάρχουν κεφαλαία γράμματα.
   2. Χρησιμοποιούμε την συνάρτηση split(), έτσι ώστε οι προτάσεις μας να χωριστούν σε λέξεις.
   3. Αφαιρούμε τα stop words.

Δημιουργούμε ένα dataframe, στο οποίο οι στήλες είναι τα genres και τα values είναι οι τίτλοι των ταινιών.

Επειδή οι στήλες του dataframe δεν έχουν το ίδιο μέγεθος , γεμίζουμε όπου χρειάζεται με τιμές NaN.

Δημιουργούμε ένα μητρώο Word2Vec, με τα περιεχόμενα του αρχείου test.txt .

Μετά φτιάχνουμε άλλο ένα μητρώο με τα genres, με την τεχνική του one hot encoding.

Έχουμε βάλει σε σχόλια ένα παράδειγμα για την δημιουργία του μοντέλου για το νευρωνικό δίκτυο, καθώς και δύο διαγράμματα για την ακρίβεια και την απώλεια , που προκύπτουν από την εκπαίδευση του.

Είναι σε σχόλια καθώς ο κώδικας δεν λειτουργεί απόλυτα.

Προβλήματα που υπήρξαν:

1. Η σύνδεση μεταξύ Word2Vec και One hot encoding. Δεν μπορούσαμε να κάνουμε την αντιστοίχιση ταινιών με τη σωστή κατηγορία, μετά την μετατροπή σε μητρώα.
2. Οπότε δεν μπορούσαμε να βάλουμε τα τελικά δεδομένα στο νευρωνικό δίκτυο.
3. Ούτε να συνδυάσουμε με τα αποτελέσματα των προηγούμενων ερωτημάτων.

Σχολιασμός τελικών αποτελεσμάτων:

Παρατηρούμε ότι στο ερώτημα 1 εφόσον δεν υπάρχουν ακόμα χρήστες τα αποτελέσματα που παίρνουμε είναι με βάση την ομοιότητα των λέξεων, ανάλογα με αυτό που αναζητούμε.

Στο δεύτερο ερώτημα αλλάζει η σειρά των αποτελεσμάτων για την ίδια αναζήτηση, καθώς τώρα υπολογίζουμε και τις προτιμήσεις του χρήστη. Οπότε πιο πάνω στα αποτελέσματα θα είναι οι ταινίες που έχει βαθμολογήσει ο χρήστης. Υπάρχουν όμως ταινίες που ο χρήστης δεν έχει βαθμολογήσει και βρίσκονται πιο χαμηλά στην κατάταξη, τις οποίες μπορεί να προτιμά από άλλες ταινίες που βρίσκονται πιο πάνω.

Αυτό φτιάχνεται στο ερώτημα 3, που συμπληρώνουμε τις ταινίες που δεν έχουν βαθμολογηθεί, σύμφωνα με χρήστες που έχουν κοινά ενδιαφέροντα με τον χρήστη μας. Έτσι παρατηρούμε ότι η σειρά ξανά αλλάζει και έχουμε καλύτερα αποτελέσματα.