ΑΝΑΚΤΗΣΗ ΠΛΗΡΟΦΟΡΙΑΣ

Υλοποιητική εργασία : Μηχανή αναζήτησης κινηματογραφικών ταινιών

Πορτοκάλογλου Ανδρονίκη 1067539 (CEID)

Κοντογιάννης Μύρων 1059526 (ECE)

Η εργασία υλοποιήθηκε στη γλώσσα προγραμματισμού Python,

στο περιβάλλον του jupyter notebook.

**Τρόπος εγκατάστασης**

Εργαστήκαμε στην python v3.8 ενώ οι βιβλιοθήκες που χρειάστηκαν εγκαταστάθηκαν μέσω του anaconda prompt**.**

Παρακάτω βρίσκονται οι βιβλιοθήκες που χρησιμοποιήθηκαν κατά την διάρκεια του project.

1. elasticsearch
2. pandas
3. numpy
4. json
5. sklearn
6. tensorflow
7. genism
8. ntlk
9. keras
10. re

Για το upload στην elasticsearch σημειώνεται ότι πρέπει να τρέξουμε στον τερματικό τις εντολές :

elasticsearch.bat

kibana.bat

αφού γίνει η πλοήγηση στους αντίστοιχους φάκελους \bin.

Αναλυτικά τα βήματα που ακολουθήσαμε για την πραγματοποίηση της εργασίας ανά ερώτημα παρουσιάζονται στην συνέχεια.

**ΕΡΩΤΗΜΑ 1**

Αρχικά, διαβάσαμε το αρχείο movies.csv σε ένα dataframe με την βοήθεια της βιβλιοθήκης pandas.

Έπειτα το μετατρέψαμε σε json και μέσω του setting βάλαμε τους απαραίτητους συντελεστές για την BM25 η οποία θα είναι η μετρητική μας για το ερώτημα αυτό. Στη συνέχεια, κάναμε upload το json στη elasticsearch.

Ρωτήσαμε το χρήστη για το τι ψάχνει να βρει και φροντίσαμε σε περίπτωση που δεν ψάξει με βάση το τίτλο άλλα το είδος ταινιών να γίνει match και στα δύο αυτά fields.Τέλος επεξεργαστήκαμε το dictionary για να εμφανίζονται τα αποτελέσματα σε πιο φιλική για το χρήστη μορφή.

**ΕΡΩΤΗΜΑ 2**

Διαβάσαμε τα αρχεία movies.csv και ratings.csv σε δυο dataframes με την βοήθεια της βιβλιοθήκης pandas.Έπειτα με την χρήση του .means και του .groupby αποθηκεύσαμε σε ένα dataframe το μέσο όρο βαθμολογίας κάθε ταινίας σύμφωνα με αυτές που έχουν βάλει οι χρήστες και το ενσωματώσαμε στο αρχικό dataframe.Ρωτήσαμε το χρήστη για το id του και αναλόγως με το ποιος είναι προσθέσαμε διπλά στη κάθε ταινία που ο συγκεκριμένος έχει βαθμολογήσει τη αντίστοιχη προσωπική του βαθμολογία. Το dataframe το μετατρέψαμε σε json, το ανεβάσαμε στη elasticsearch και όπως ακριβώς στο προηγούμενο ερώτημα ρωτήσαμε για το τι ψάχνει να βρει ο χρήστης .

Η διαφορά όμως με το προηγούμενο ερώτημα είναι πως φτιάξαμε μια δική μας μετρητική. Χρησιμοποιούμε τον μέσο όρο τριών βαθμολογιών : του χρήστη, όλων των χρηστών για τη συγκεκριμένη ταινία αλλά και της μετρητικής ΒΜ25.

Τα αποτελέσματα τα εμφανίσαμε και πάλι πιο φιλικά για το χρήστη.

**ΕΡΩΤΗΜΑ 3**

Αρχικά διαβάσαμε τα αρχεία csv. Xχρησιμοποιήσαμε το str.get\_dummies (“|”) για να μπορέσουμε να πάρουμε τα genres στα οποία ανήκει κάθε ταινία. Έτσι φτιάξαμε μια συνάρτηση για να υπολογίζει ανά χρήστη το μέσο ορό κάθε genre και στα αποτελέσματα αλλάξαμε τα ΝΑΝ σε 0.Καταλήξαμε σε ένα dataframe (data4kmeans )τo οποίο θα χρησιμοποιήσουμε στη kmeans.Πειραματικά επιλέξαμε ο αριθμός των cluster να είναι 8 καθώς με τα δεδομένα που είχαμε εμφάνιζε τις πιο κοντινές τιμές. Έπειτα συμπληρώσαμε στις τιμές που μας έλλειπαν το prediction που δόθηκε από την kmeans. Επεξεργαστήκαμε τα στοιχεία αυτά και φτιάξαμε ένα dataframe το οποίο περιέχει movieid,title, genres average per genre, user\_rating ,rating kai ένα prediction rating το οποίο είναι ο μέσος ορός των genre average που ανήκει κάθε ταινία. Φροντίσαμε αυτό το dataframe ανάλογα με τον user που ψάχνει να έχει τα δικά του ratings.Μετατρέψαμε τα dataframe σε json,φτιάξαμε το mapping ,τα settings και τα ανεβάσαμε στη elasticsearh. Με τα νέα στοιχεία βελτιώσαμε την μετρητική του προηγουμένου ερωτήματος, εισάγοντας τον μέσο όρο όλων των κατηγοριών που ανήκει μια ταινία σύμφωνα με τον cluster του χρήστη. Όπως και στο προηγούμενο ερώτημα τα αποτελέσματα εμφανίζονται σύμφωνα με τον μέσο όρο του παραπάνω σε συνδυασμό με την βαθμολογία BM25,βαθμολογία χρήστη και τον μέσο όρο βαθμολογιών για την εκάστοτε ταινία .

**ΕΡΩΤΗΜΑ 4**

Στο ερώτημα 4 καλούμαστε να χρησιμοποιήσουμε ένα νευρωνικό δίκτυο για την κατηγοριοποίηση του χρήστη. Για να γίνει αυτό είναι αναγκαίο να μετατραπούν οι τίτλοι των ταινιών σε vectors.Η υλοποίηση αυτή έγινε μεσω της τεχνικής word2vec. Έπειτα πρέπει να μετατρέψουμε τα genres σε vectors. Η υλοποίηση έγινε με την τεχνική onehotencoding.Αφού μετατράπηκαν τα δεδομένα σε vectors , δημιουργήθηκε το νευρωνικό δίκτυο το οποίο εκπαιδεύτηκε πάνω στα δεδομένα και έπειτα έγινε η κατηγοριοποίηση του χρήστη. Τα αποτελέσματα δεν είναι τόσο καλά όσο το ερώτημα 3. Γι’ αυτό αφού έγινε upload όλων των δεδομένων στην elasticsearch , και έγινε το query από τον χρήστη, τα αποτελέσματα επιστράφηκαν σύμφωνα με μια καινούρια μετρητική. Η μετρητική αυτή δημιουργεί έναν μέσο όρο όλων των προηγούμενων κατηγοριοποιήσεων ( BM25, k-means clustering, neuron-classification), για την όσο το δυνατόν καλύτερη επιστροφή των αποτελεσμάτων. Η διάταξη των τελικών αποτελεσμάτων θεωρούμε πως βρίσκεται κοντά στις προτιμήσεις του χρήστη.

Βιβλιογραφία:

1. <http://ethen8181.github.io/machine-learning/search/bm25_intro.html#Gaining-Intuition-for-Okapi-BM25>
2. <https://www.datacamp.com/community/tutorials/pandas-tutorial-dataframe-python>
3. <https://www.elastic.co/guide/en/elasticsearch/reference/current/index-modules-similarity.html>
4. <https://www.elastic.co/blog/better-than-average-sort-by-best-rating-with-elasticsearch>
5. <https://halimatusyak.medium.com/introduction-to-k-means-clustering-algorithm-in-python-ee6ddf6ef2c0>
6. <https://www.tensorflow.org/tutorials/text/word_embeddings>
7. <https://towardsdatascience.com/machine-learning-word-embedding-sentiment-classification-using-keras-b83c28087456>
8. <https://www.geeksforgeeks.org/>