Αναφορά Εργαστηριακής Άσκησης Χειμερινού Εξαμήνου για το μάθημα Ανάκτηση Πληροφορίας

Στοιχεία Φοιτητών:

Δημήτριος Κωστορρίζος AM: 1054419 Έτος Σπουδών: 5ο

Email: up1054419@upnet.gr

Λάμπρος Παπαδόπουλος AM: 1054433 Έτος Σπουδών: 5ο

Email: up1054433@upnet.gr

Περιγραφή Υλοποίησης

Η υλοποίηση έχει γίνει σε C# 8.0. Το περιβάλλον υλοποίησης που έχει χρησιμοποιηθεί είναι Visual Studio 2019. Χρησιμοποιήθηκαν τα εξής NuGet packages: Elasticsearch.Net, NEST,Microsoft.ML.

Κατεβάσαμε την το αρχείο zip που περιέχει τα αρχεία για την Elasticsearch από το link: https://www.elastic.co/downloads/elasticsearch

Τοποθετήσαμε το unwrapped περιεχόμενο του αρχείου στον φάκελο C: και εκτελέσαμε την Elastic search από το CMD. Επιβεβαιώσαμε ότι η Elasticsearch εκτελείται σωστά στην default διεύθυνση localhost:9200.

Ερώτημα 1

Αρχικά, ορίσαμε το index για τα δεδομένα ταινιών, σε πεζά(lowercase) γράμματα. Δημιουργήσαμε ένα στιγμιότυπο του elasticsearch client, ο οποίος χρησιμοποιείται για όλα τα ερωτήματα της άσκησης και χρησιμοποιεί την default επιλογή του Single Node Connection Pooling, καθώς δεν χρειαστήκαμε επιπλέον node την υλοποίηση μας.

Δημιουργήσαμε την κλάση Movies, με πεδία το μοναδικό ID, τον τίτλο της ταινίας και τις κατηγορίες στις οποίες ανήκει η ταινία, για να αναπαραστήσουμε τις εγγραφές του αρχείου movies.csv. Για να ορίσουμε το πεδίο που θα χρησιμοποιείτε ως document Id, χρησιμοποιήσουμε το class attribute:

[ElasticsearchType(IdProperty = nameof(MoviesId))]

Έτσι ορίσαμε το πεδίο MoviesId ως το μοναδικό Id για τις εγγραφές τύπου Movie.

Για να ορίσουμε την μετρική ομοιότητας BM25 για το πεδίο Title, χρησιμοποιήσαμε το property attribute:

[Text(Similarity = "BM25")]

Έχοντας φορτώσει τα αρχεία στην μνήμη, τα εισάγουμε μαζικά στην ElasticSearch μέσα στο index για τα δεδομένα των ταινιών.

Έπειτα, με ένα more like this query, με ελάχιστο αριθμό εμφάνισης της λέξης προς αναζήτησης την τιμή 1, λάβαμε ταξινομημένα κατά φθίνουσα σειρά της εγγραφές που περιέχουν τον όρο αναζήτησης, με βάση την μετρική Score της Elasticsearch. Να σημειωθεί ότι η αναζήτηση του όρου που έθεσε ο χρήστης, γίνεται μόνο στον τίτλο της ταινίας.

Ερώτημα 2

Αρχικά, ορίσαμε το index για τα δεδομένα βαθμολογιών, σε πεζά(lowercase) γράμματα.

Δημιουργήσαμε την κλάση Ratings, με πεδία το μοναδικό ID που δημιουργείται από το σύστημα, τι ID του χρήστη, το ID της ταινίας, την βαθμολογία που έθεσε ο χρήστης και την στιγμή της καταγραφής, για να αναπαραστήσουμε τις εγγραφές του αρχείου ratings.csv. Για να ορίσουμε το πεδίο που θα χρησιμοποιείτε ως document Id, χρησιμοποιήσουμε το class attribute:

[ElasticsearchType(IdProperty = nameof(RatingsId))]

Έχοντας φορτώσει τα αρχεία στην μνήμη, τα εισάγουμε μαζικά στην ElasticSearch μέσα στο index για τα δεδομένα των βαθμολογιών, που έχουν ως MovieId τιμή μικρότερη από 9125, καθώς υπάρχουν εγγραφές με MovieId για ταινίες που δεν υπάρχουν.

Για κάθε ταινία του πρώτου ερωτήματος, ορίζουμε το νέο score ως το άθροισμα του score που επέστρεψε η ElasticSearch, του μέσου όρου των βαθμολογιών που έχει η ταινία και της βαθμολογίας του χρήστη, αν έχει βαθμολογήσει την δεδομένη ταινία.

Έπειτα, ξανά ταξινομούμε την λίστα, κατά φθίνουσα σειρά, με βάση το νέο score.

Ερώτημα 3

Εγκαταστήσαμε το package Microsoft.ML. Δημιουργήσαμε ένα νέο στιγμιότυπο της κλάσης MLContext και θέσαμε το την τιμή του seed σε μία σταθερή τιμή, ώστε να διασφαλίσουμε ότι τα αποτελέσματα θα είναι ντετερμινιστικά. Το στιγμιότυπο αυτό χρησιμοποιείται στα ερωτήματα 3 και 4.

Αρχικά, βρίσκουμε τα ID των ανά κατηγορία ταινιών και όλα τα user IDs. Έπειτα, για κάθε χρήστη, υπολογίζουμε το μέσο score ανά είδος ταινίας που έχει βαθμολογήσει. Αν ο χρήστης δεν έχει βαθμολογήσει καμία ταινία από ένα ορισμένο είδος ταινίας, τότε το score που του ορίζουμε είναι 0. Επίσης, αν μία ταινία ανήκει σε περισσότερα από 1 ένα είδη, τότε προσμετράται στον υπολογισμό του μέσο score για κάθε είδος στο οποίο ανήκει.

Ορίζουμε έναν K-Means trainer, με 4 clusters. Χρησιμοποιώντας τον trainer, εκπαιδεύουμε το μοντέλο με βάση τα δεδομένα των μέσων βαθμολογιών για κάθε χρήστη και είδος ταινίας. Έπειτα, χρησιμοποιούμε το ίδιο σύνολο δεδομένων για να ολοκληρώσουμε το clustering. Εξάγουμε τα αποτελέσματα του clustering και υπολογίζουμε την μέση βαθμολογία ανά ταινία για κάθε cluster χρηστών.

Τέλος, χρησιμοποιώντας τα δεδομένα αυτά, συγκεντρώσαμε τις ήδη υπάρχουσες εγγραφές και δημιουργήσαμε τις εγγραφές που λείπουν για κάθε χρήστη και για κάθε ταινία.

Ερώτημα 4

Αρχικά, διαβάσαμε τους τίτλους των ταινιών από την ElasticSearch. Αρχικοποιήσαμε, ένα στιγμιότυπο της κλάσης WordEmbeddingsTokenizer. Έπειτα, εισήγαμε τους τίτλους των ταινιών ως είσοδο στο στιγμιότυπο αυτό. Όταν ολοκληρώθηκε η διαδικασία του prediction, χρησιμοποιήσαμε τα word embeddings ανά τίτλο για περαιτέρω επεξεργασία.

Επόμενο βήμα, ήταν η εύρεση της One Hot Encoding κωδικοποίησης ανά ταινία. Για κάθε ταινία, συνενώσαμε τα Genres της, με τον χαρακτήρα κόμμα. Χρησιμοποιήσαμε τα ενωμένα πλέον, Genres, για να υπολογίσουμε τα διανύσματα της One Hot Encoding κωδικοποίησης ανά ταινία. Δημιουργήσαμε ένα στιγμιότυπο της κλάσης OneHotEncoding και εισήγαμε τα ενωμένα Genres. Ως έξοδο, λάβαμε το διάνυσμα One Hot Encoding ανά ταινία.

Χρησιμοποιήσαμε το KerasModel νευρωνικό δίκτυο, που υπάρχει στο NuGet Package Keras.NET. Στην συνάρτηση Fit, δώσαμε ως είσοδο τις λίστες με τις τιμές των word embeddings και one hot encoding που υπολογίσαμε νωρίτερα. Τέλος, καλέσαμε την μέθοδο Predict, στην οποία δώσαμε ως είσοδο τα id των χρηστών που επιθυμούμε και η μέθοδος επέστρεψε τις τιμές των βαθμολογιών.

Τα αποτελέσματα του 3^{ου} και του 4^{ου} ερωτήματος έχουν μία μικρή απόκλιση, το οποίο είναι λογικό λόγο της διαδικασίας υπολογισμού. Πιο ακριβή δείχνουν τα αποτελέσματα του 4^{ου} ερωτήματος.