Υλοποιητική Εργασία

Ανάκτηση Πληροφορίας

Τσάκωνας Κωνσταντίνος ΑΜ: 1059666 email: st1059666@ceid.upatras.gr
Παρλαπάνης Αντώνιος ΑΜ: 1059709 email: st1059709@ceid.upatras.gr

Περιβάλλον Υλοποίησης

Η εργασίας υλοποιήθηκε στη γλώσσα προγραμματισμού python και οι βιβλιοθήκες που χρησιμοποιήθηκαν είναι:

- **Pandas** (για την διαχείριση των csv αρχείων)
- **Elasticsearch** (το api για την χρήση της elasticsearch από το προγραμμά μας)
- **Requests** (για αποστολή http requests στην elasticsearch με τα queries)
- **Progress** (μπάρα φόρτωσης για το ανέβασμα δεδομένων)
- **Numpy** (για τη δυνατότητα εργασίας με arrays)
- **Scikit-learn** (χρήση K-Means και One-hot encoding αλγορίθμους)
- **Tensorflow** (προετοιμασία δεδομένων και δημιουργία μοντέλων machine learning)
- **Keras** (api για τα μοντέλα machine learning)

LINUX:

- Εγκατασταση Elasticsearch
  - Ubuntu και Debian-based Distributions

    sudo apt install elasticsearch
  - Arch-based Distributions

    sudo pacman -S elasticsearch

- Ενεργοποίηση Elasticsearch

  sudo systemctl start elasticsearch

- Εγκατάσταση της Python 3.8.x.
    - Ubuntu και Debian-based Distributions

      sudo apt install python3.8

    - Για Arch-based Distributions χρειάζεται να εγκαστήσετε έναν AUR(Arch User Repository) helper (π.χ yay) και να εγκαταστήσετε την python ως εξής

      yay -S python38

- Εγκατασταση του PIP.
    - Ubuntu και Debian-based Distributions

      sudo add-apt-repository universe

      sudo apt install python3-pip

- Arch-based Distributions

  sudo pacman -S python-pip

- Χρειάζεται να δημιουργήσουμε ένα Virtual Enviroment στο οποίο θα γίνει η εγκατάσταση των βιβλιοθηκών που χρησιμοποιήσαμε
    - **python3 -m pip install --user virtualenv**

      python3 -m venv <enviroment_name>

- Σε περίπτωση που έχετε δύο εκδόσεις python3 στον υπολογιστή σας(συνήθως arch-based distributions) τότε δημιουργήστε το Virtual Enviroment ως εξής

  virtualenv --python=/usr/bin/python3.8 <enviroment_name>

- Τώρα πρέπει να ενεργοποιήσουμε το Virtual Enviroment.

  source <enviroment_name>/bin/activate

- Εγκατάσταση Βιβλιοθηκών.

  pip install elasticsearch pandas numpy scikit-learn progress tensorflow keras requests

- Εκτέλεση προγράμματος.

  python program_menu.py

WINDOWS:

- Αρχικά κατεβάζουμε την **elasticsearch** από την επίσημη σελίδα.
- Για την ενεργοποίηση της **elasticsearch** ανοίγουμε το cmd και τρέχουμε το

  C:\<path_to_elasticsearch_folder>\bin\elasticsearch.bat

- Στη συνέχεια χρειάζεται να εγκαταστήσουμε την python στον υπολογιστή μας. Επιλέγουμε από τα versions 3.5-3.8 διότι αυτά μόνο υποστηρίζονται από το tensorflow.
- Έπειτα χρειάζεται ένα περιβάλλον ανάπτυξης python. Συνίσταται το **PyCharm** λόγω του πολύ καλού package management.Για την εγκατάσταση βιβλιοθηκών πρέπει να κατεβάσουμε το **pip** (σύστημα διαχείρισης πακέτων).
- Για να κατεβάσουμε κάποια βιβλιοθήκη γράφουμε στο terminal του PyCharm

  pip install <ονομα_βιβλιοθήκης>

- Διαφορετικά πηγαίνουμε **File -> Settings -> Python Interpreter -> pip** , κάνουμε αναζήτηση της επιθυμητής βιβλιοθήκης και πατάμε install package.

- Για να κατεβάσουμε κάποια βιβλιοθήκη γενικά στο σύστημα μας ανοίγουμε το cmd των Windows και τρέχουμε την εντολή.

  pip install <ονομα βιβλιοθήκης>

- Για την υλοποίηση του project χρησιμοποιήθηκαν οι εξής βιβλιοθήκες και μπορούν να γίνουν εγκατάσταση ως εξής:

  pip install elasticsearch pandas numpy scikit-learn progress tensorflow keras requests

---

Περιγραφή Διαδικασίας Υλοποίησης

upload_data.py:

- Στο αρχείο αυτό περιέχεται ο κώδικας για την εισαγωγή των δεδομένων μας στην Elasticsearch. Αρχικά το πρόγραμμα ελέγχει αν υπάρχουν τα απαραίτητα αρχεία στο στον ίδιο φάκελο που υπάρχει το αρχείο, αυτά τα αρχεία είναι το movies.csv και το settings.json που περιέχει τις ρυθμίσεις για την Elasticsearch.
- Αν δεν υπάρχουν τα αρχεία, ζητάει από τον χρήστη να μεταφέρει τα αρχεία στο σωστό directory ή να ορίσει το path προς αυτά τα αρχεία.
- Η συνάρτηση upload_data(es) ανεβάζει τα αρχεία που διαβάζει από το csv, ο χρήστης μπορεί να βλέπει την εξέλιξη του ανεβάσματος από το progress bar που έχουμε προσθέσει.

simple_search.py:

- Ζητείται από τον χρήστη να κάνει αναζήτηση για μία ταινία.
- Στη συνέχεια κάνουμε ένα query στην Elasticsearch και μας επιστρέφονται τα αποτελέσματα σε μορφή json, τα οποία τα φιλτράρουμε και επιστρέφουμε στον χρήστη τους τίτλους ταινιών με τα αντίστοιχα score.

custom_search.py:

- Ομοίως με την simple search , μόνο που τα score συνδυάζονται κατά περίπτωση με τη βαθμολογία των cluster και της πρόβλεψης του νευρωνικού δικτύου.
- Πιο συγκεκριμένα παίρνουμε κάθε ταινία που επέστρεψε η elasticsearch και τη δίνουμε στην συνάρτηση unseen_movies.Η συνάρτηση αυτή μας επιστρέφει την βαθμολογία που έχει δώσει ο χρήστης στην ταινία , αν την έχει δει, η τον μέσο όρο βαθμολογίας του cluster στο οποίο ανήκει η ταινία.
- Επιπλέον , αν ο χρήστης δεν έχει δει την ταινία , με τη συνάρτηση predict_rating , μας επιστρέφεται από το νευρωνικό δίκτυο μια προβλεπόμενη βαθμολογία που θα έβαζε ο χρήστης.
- Οι βαθμολογίες για κάθε ταινία μαζί με το score της elasticsearch τροφοδοτούνται στη μετρική που έχουμε δημιουργήσει ,και προκύπτει το τελικό score.
- Τέλος , με ένα lambda function αντιστρέφουμε το dictionary βαθμολογιών-ταινιών, ώστε να κάνουμε sort με βάση τα score.

clusters.py:

- Περιέχονται τρεις συναρτήσεις στο αρχείο αυτό η create_mean_file(), create_clusters(), unseen_movies(user_id, movie_id, ratings_with_clusters, mean_with_clusters).
- Πρώτο πράγμα που κάνουμε είναι να αποθηκεύσουμε τα δεδομένα μας από το ratings.csv και το movies.csv σε μεταβλητές. Στη συνέχεια βρίσκουμε των αριθμών των χρηστών και τις κατηγοριές ταινιών που έχουμε.
- Στη συνέχεια δημιουργούμε ένα dataframe το οποίο περιέχει ότι και το ratings.csv συν τη κατηγορία ή κατηγορίες που ανήκει η κάθε ταινία. Αφου επεξεργαστούμε τη μορφή που είναι οι κατηγορίες για να μπορούμε να κάνουμε τις πράξεις που χρειαζόμαστε, δημιουργούμε ένα αρχείο το οποίο περιέχει το μέσο όρο των βαθμολογιών για κάθε χρήστη σε κάθε κατηγορία.
- Έχοντας αυτά τα δεδομένα μπορούμε να δημιουργήσουμε τα clusters και να τοποθετήσουμε το κάθε χρήστη σε αυτά με τους παραπάνω μέσους όρους.
- Τελειώνοντας τη δημιουργία των clusters ακολουθεί η unseen_movies η οποία επιστρέφει ένα από τα εξής:
    - Αν ο χρήστης έχει βαθμολογήσει την ταινία επιστρέφει τη βαθμολογία που έχει δώσει.
    - Αν δεν την έχει βαθμολογήσει, ελέγχει αν οι υπόλοιποι χρήστες του cluster που ανήκει έχουν δει την ταινία και αν ισχύει επιστρέφει τον μέσο όρο των βαθμολογιών που έχουν δώσει.
    - Αν δεν ισχύει τίποτα από τα παραπάνω, τότε επιστρέφει 0.

Σημείωση: Μαζί με τους κώδικες περιέχεται και ένα αρχείο που ονομάζεται mean.csv όπου περιέχει τους μέσους όρους των βαθμολογιών για κάθε χρήστη σε κάθε κατηγορία, επειδή ο υπολογισμός αυτών είναι χρονοβόρα διαδικασία

wemb_model.py:

- Στο αρχείο αυτό ορίζονται τρεις συναρτήσεις, οι convert_text,create_model και predict_rating.
- Ορίζουμε αρχικά ένα dataframe που περιέχει τα στοιχεία του ratings.csv συν τα genres, όμως το movieId έχει αντικατασταθεί από τον τίτλο της ταινίας. Τα id των χρηστών τα μετατρέψαμε σε string, δηλαδή το userId του χρήστη 1 έγινε ‘user1’. Στη συνέχει καθαρίσαμε τους τίτλους των ταίνιων από τις χρονολογίες χρησιμοποιόντας Regular Expressions και απαλείψαμε τα σημεία στίξης.
- Για τις κατηγορίες των ταινιών, πηράμε όλες τις κατηγορίες που μπορεί να ανήκει μια ταινία και χρησιμοποιόντας συναρτήσεις του scikit-learn, δημιουργήσαμε τις one-hot κωδικοποιήσεις όλων των κατηγοριών. Για επιπλέον διευκόλυνση φτίαξαμε ένα dictionary που περιέχει τις κατηγοριίες και όλους του πιθανούς συνδιασμού κατηγοριών και την αντίστοιχη κωδικοποίηση τους, η οποία είναι στη περίπτωση των περισσότερων της μιας κατηγορίας, το bitwise or των επιμέρος κατηγοριών.
- Για τα userId και τους τίτλους των ταινίων τα οπόια είναι πλεον ένα sting, τα κάναμε split, έχοντας δημιουργήσει μια λίστα που περιέχει λίστες με τις λέξεις που προέκυψαν από το διαχωρισμό των αλφαριθμητικών. Παράλληλα με τη διαδικασία αυτή αποθηκεύσαμε όλες τις λέξεις ώστε να γνωρίζουμε ποιό είναι το πλήθος λέξεων που έχουμε.
- Έχοντας τα παραπάνω με το Tokenizer του Keras δημιουργούμε tokens για κάθε λέξη που έχουμε και μετατρέπουμε τα userId και τους τίτλους σε ένα διάνυσμα με τα αντίστοιχα tokens με την συνάρτηση convert_text. Επίσης δημιουργούμε ένα dictionary το οποίο περιέχει τις λέξεις και τη τιμή που αντιστοιχεί σε καθε μία από αυτές.

- Τα τελικά μας διανύσματα προκύπτουν αφού προσθέσουμε στα παραπάνω το one-hot κωδικό της καταγηροιάς ή των κατηγοριών που ανήκει η ταινία.
- Η συνάρτηση predict_rating παίρνει τα userId, τίτλο ταινίας και την/ις κατηγορία/ες, τα μετατρέπει στη μόρφη των δεδομένων που εκπαιδεύσαμε το νευρωνικό μας και επιστρέφει το prediction για την βαθμολογία που θα έβαζε ο χρήστης σε κάποια ταινία.
- Λόγω του διαφορετικού μήκους των διανυσμάτων στο final_vectors χρησιμοποιούμε τη συνάρτηση του keras pad_sequences ώστε να μετατρέψουμε τα τελικά διανύσματα σε ένα 2D numpy array με ισομήκη διανύσματα.
- Με το attribute 'post' δηλώνουμε ότι τα επιπλέον μηδενικά όπου χρειάζονται θα προστεθούν στο τέλος του διανύσματος.
- Στη συνέχεια μετατρέπουμε τα ratings(labels του μοντέλου) σε float και τα κανονικοποιούμε ανάμεσα στο 0 και στο 1 για τη βελτιστοποίηση των αποτελεσμάτων.
- Πριν φτιάξουμε το μοντέλο όπου θα γίνει το training χωρίζουμε τα δεδομένα σε training και test, ώστε να μπορούμε να επιβεβαιώσουμε τη σωστή λειτουργία του μοντέλου σε μη trained data.
- Για το μοντέλο του keras χρησιμοποιούμε την μορφή και τις παραμετροποιήσεις που υπάρχουν στο documentation του tensorflow για τα Word Embeddings.
- Για την αποθήκευση του μοντέλου χρησιμοποιούμε την εντολή model.save ώστε να μη γίνετε εκ νέου εκπαίδευση του μοντέλου κάθε φορά που τρέχει το πρόγραμμα.
- Με την εντολή keras.models.load_model φορτώνουμε το πρόγραμμα από τη μνήμη.

program_menu.py:

Περιέχει το CLI μενού για την εύκολη χρήση του προγράμματος.

Σχολιασμός Τελικών Αποτελεσμάτων.

Η μετρική που έχουμε δημιουργήσει υπολογίζει το score συνδυάζοντας το score που επιστρέφει η elasticsearch , την βαθμολογία που έχει δώσει ο χρήστης ή τον μέσο όρο βαθμολογίων της ταινίας στο cluster που ανήκει ο χρήστης , το μέσο όρο όλων των βαθμολογίων για αυτή την ταινία και την βαθμολογία που θα προβλέψει το νευρωνικό. Οι συντελεστές για κάθε ένα από τα προαναφερθέντα είναι οι εξής: 30% στο score της elasticsearch, 40% στη βαθμολογία του χρήστη η του μέσου όρου του cluster, 20% στο μέσο όρο όλων των βαθμολογιών για τη συγκεκριμένη ταινία και 10% για τη βαθμολογία του νευρωνικού δικτύου.

Ο λόγος που έχουμε επιλέξει αυτά τα βάρη για τα παραπάνω δεδομένα ειναί ότι η μετρική μας έχει ως γνώμονα το χρήστη, οπότε θέλαμε μεγαλύτερη βαθμολογία να έχουν ταινίες που μπορεί να τον ενδιαφέρουν και όχι αυτές που είναι πιο κοντά σε αυτό που αναζήτησε. Έτσι είναι λογικό η βαθμολογία που έχει δώσει ο χρήστης ή ο μέσος όρος της βαθμολογίας του cluster να έχουν το μεγαλύτερο βάρος. Από την άλλη μεριά, η βαθμολογία που θα προβλέψει το νευρωνικό μας έχει τόσο χαμηλό συντελεστή, διότι με τα δέδομενα που μας ζητήθηκε να εκπαιδεύσουμε το μοντέλο μας δεν επιτεύχθηκε υψηλό accuracy όποτε οι βαθμολογίες που επιστρέφει δεν είναι αρκετά αξιόπιστες. Τέλος για tests που κάναμε στο τελικό μας πρόγραμμα τα απότελεσματα ήταν τα αναμενόμενα.