Αρχιτεκτονική συστήματος σύστασης ταινιών με χρήση μηχανικής μάθησης

Περιεχόμενα

1. Εισαγωγή

Το σύστημα σύστασης ταινιών (movie recommendation system) που παρουσιάζουμε, στοχεύει σε γρήγορες και ακριβής συστάσεις ταινιών με επίκεντρο την εύκολη πλοήγηση του χρήστη στην πλατφόρμα. Μέσα από την διαδικτυακή πλατφόρμα ο θεατής θα έχει στη διάθεση του μία ευρύ ποικιλία ταινιών, με προτάσεις που έχουν παραχθεί από τον αλγόριθμο μηχανικής μάθησης LightGCN, έναν αλγόριθμο που η ακρίβεια του έχει αποδειχθεί ερευνητικά. Στόχος του παρόν εγγράφου είναι η περιγραφή της αρχιτεκτονικής του συστήματος με σκοπό να τεθούν οι βάσεις για την υλοποίηση του συνόλου των προτάσεων του έργου.

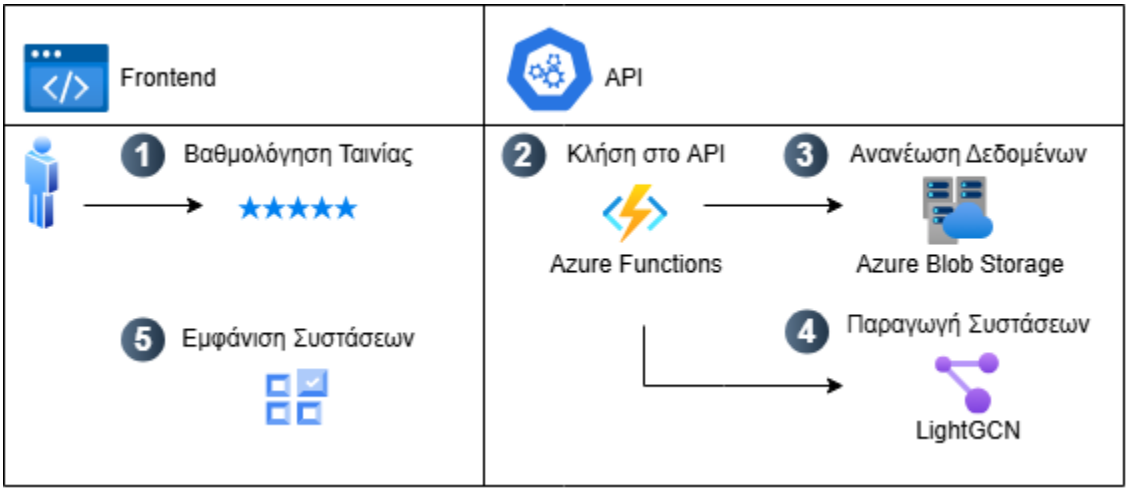
2. Αρχιτεκτονική Συστήματος

Το αρχιτεκτονικό σχέδιο του συστήματος διαχωρίζεται σε 2 βασικά επίπεδα:

- Το **επίπεδο frontend,** που εξασφαλίζει την διεπαφή χρήστη.
- Το **επίπεδο API**, που είναι χωρισμένο σε 2 τμήματα: α) Ανανέωση Συστάσεων και β) Αποθήκευση Δεδομένων.

Όταν ο χρήστης πραγματοποιεί μια ενέργεια που απαιτεί ενημέρωση των προτάσεων, το frontend θα καλέσει το API ώστε να ζητήσει νέες συστάσεις από το backend. Αυτό συμβαίνει, όταν ο χρήστης αλληλεπιδράσει με μία ταινία βαθμολογώντας την. Η διαδικασία έχει ως εξής:

1. **Αξιολόγηση θεατή:** Ο θεατής αξιολογεί την ταινία βαθμολογώντας την από το 1 έως το 5.
2. **Κλήση στο API:** Το frontend, σε απόκριση αυτής της ενέργειας, στέλνει ένα αίτημα HTTP προς ένα καθορισμένο endpoint της Azure Function.
3. **Αποθήκευση δεδομένων:** Τα δεδομένα αναβαθμίζονται μέσα στα CSV αρχεία
4. **Ανανέωση συστάσεων:** Με βάση τα νέα δεδομένα παράγει τη λίστα των προτεινόμενων ταινιών από το image του μοντέλου μηχανικής μάθησης στο cloud.
5. **Εμφάνιση αποτελεσμάτων:** Η νέα λίστα συστάσεων στέλνεται πίσω στο frontend μέσω του API, και εμφανίζεται στην διεπαφή χρήστη.



Εικόνα 1: Λειτουργικό διάγραμμα συστήματος.

Με αυτόν τον τρόπο, κάθε φορά που ο χρήστης χρειάζεται ενημερωμένες συστάσεις, η εφαρμογή frontend επικοινωνεί με το backend μέσω ενός ορισμένου API αντί να προσπαθεί να παράξει προτάσεις τοπικά. Αυτό διατηρεί το frontend απλό και μεταφέρει την απαιτητική επεξεργασία στο serverless backend.

2.1 Επίπεδο Frontend

Το frontend της εφαρμογής υλοποιείται με React και παρέχεται ως στατικό web περιεχόμενο. Η εφαμογη είναι μονοσέλιδη (Single Page Application, SPA), έχει χτιστεί σε αρχεία HTML, CSS και JavaScript, τα οποία δίνονται απευθείας στον χρήστη χωρίς να απαιτείται δυναμική δημιουργία σε server. Δηλαδή, όλες οι διαδρομές (routes) φορτώνονται από ένα μόνο αρχείο HTML.

Ο ρόλος του frontend είναι να προσφέρει το περιβάλλον διεπαφής χρήστη (UI) όπου ο χρήστης μπορεί να αλληλεπιδρά με το σύστημα. Όλη η λογική των συστάσεων υλοποιείται μέσω κλήσεων API σε ένα backend, γι' αυτό το frontend δεν χρειάζεται να γνωρίζει λεπτομέρειες του μοντέλου μηχανικής μάθησης, απλώς εμφανίζει τα αποτελέσματα και στέλνει αιτήματα.

Το Azure Static Web Apps είναι η υπηρεσία της Azure που χρησιμοποιείται για τη φιλοξενία του frontend. Η υπηρεσία αυτή αναλαμβάνει να δημοσιεύσει τα αρχεία της εφαρμογής στο δίκτυο, εξασφαλίζοντας γρήγορη φόρτωση στους χρήστες. Η χρήση του απλοποιεί σημαντικά τη διαδικασία ανάπτυξης και φιλοξενίας της εφαρμογής. Με μία ενέργεια (ένα push σε ένα GitHub repo), η υπηρεσία μπορεί αυτόματα να κάνει build και deploy τόσο το React app όσο και τις Azure Functions εννιαία.

2.2 Επίπεδο API

Όσο αφόρα το επίπεδο API η κλήση από το frontend φτάνει σε μια Azure Function, η οποία περιέχει τη λογική του μοντέλου LightGCN. Μια Azure Function είναι ουσιαστικά μια serverless λειτουργία δηλαδή εκτελείται στο cloud χωρίς διαχείριση δικού μας διακομιστή, και ενεργοποιείται από ένα εισερχόμενο HTTP αίτημα. Όταν φτάσει το αίτημα με τα στοιχεία του χρήστη (user ID), ο κώδικας της function εκτελείται, φορτώνει το εκπαιδευμένο μοντέλο LightGCN και το χρησιμοποιεί για να παράξει τις νέες συστάσεις.

Αφότου η Azure Function υπολογίσει τις προτεινόμενες επιλογές για τον συγκεκριμένο θεατή με βάση τα δεδομένα εισόδου, διαμορφώνει την απόκριση σε μορφή αρχείου JSON, που περιλαμβάνει τη λίστα των συστάσεων. Στη συνέχεια επιστρέφει αυτή την απόκριση πίσω στο frontend ως αποτέλεσμα του HTTP αιτήματος.

Με αυτόν τον τρόπο, η υπολογιστική εργασία του συστήματος συστάσεων παραμένει στο backend, και το frontend απλώς λαμβάνει έτοιμα τα αποτελέσματα. Επιπλέον, η Azure Function μπορεί κατά τη διάρκεια αυτής της διαδικασίας να ενημερώσει τα αποθηκευμένα δεδομένα για παράδειγμα, να καταγράψει σε ένα αρχείο CSV τη νέα αλληλεπίδραση του χρήστη ώστε η πληροφορία αυτή να χρησιμοποιηθεί αργότερα για ανανέωση του μοντέλου.

Παράλληλα η πλατφόρμα Azure, εκκινεί δυναμικά instances της function όταν υπάρχουν αιτήματα και τερματίζει τα instances που δεν χρησιμοποιούνται, εξοικονομώντας πόρους. Αυτό σημαίνει επίσης ότι το σύστημα μπορεί να κλιμακωθεί αυτόματα, αν πολλοί χρήστες ζητήσουν συστάσεις ταυτόχρονα, η Azure μπορεί να ενεργοποιήσει πολλά παράλληλα instances της function, που το καθένα θα φορτώσει ένα αντίγραφο του μοντέλου, ώστε να εξυπηρετηθούν όλα τα αιτήματα χωρίς σημαντικές καθυστερήσεις.

Για την ανάπτυξη του μοντέλου στο περιβάλλον της Azure Function, το μοντέλο προεκπαιδεύεται και στη συνέχεια οι παράμετροί του διατίθενται στη λειτουργία. Συμπεριλάμβάνεται το αρχείο των βαρών του μοντέλου στο πακέτο ανάπτυξης της function, ώστε να αναπτυχθεί μαζί με τον κώδικα.

Όταν ο θεατής προσθέσει μία νέα βαθμολογία ανανεώνεται η συγκεκρίμενη στήλη που αντιστοιχεί στο user ID του και το μοντέλο επανεκπαιδεύεται σε αυτά τα νέα δεδομένα.

Στον κώδικα της Azure Function καθορίζεται μια αρχικοποίηση που φορτώνει το μοντέλο LightGCN καθώς και όλες τις βιβλιοθήκες που χρησιμοποιήθηκαν. Αυτά δηλώνονται μέσω του αρχείου environment.yaml Το φόρτωμα του μοντέλου πραγματοποιείται μία φορά ανά instance της function και όχι εκ νέου σε κάθε αίτημα, ώστε να παραμείνει χαμηλή η καθυστέρηση. Η πρώτη κλήση προς τη function (cold start) είναι συνήθως πιο αργή διότι τότε γίνεται φόρτωση όλου του image από το cloud, όμως οι επόμενες κλήσεις θα είναι πολύ ταχύτερες καθώς το μοντέλο είναι ήδη φορτωμένο.

2.3 Ανανεώση Συστάσεων

Μόλις το frontend λάβει την απόκριση από το API με τις νέες συστάσεις, προχωρά στην ανανέωση του περιεχομένου που εμφανίζεται στον χρήστη. Συγκεκριμένα, ο κώδικας στο frontend ενημερώνει το συγκεκριμένο component με τη νέα λίστα συστάσεων.

Έτσι, χωρίς να απαιτείται πλήρης ανανέωση της σελίδας, ο χρήστης βλέπει δυναμικά ανανεωμένες προτάσεις με βάση την πιο πρόσφατη ενέργειά του. Για παράδειγμα, εάν ο χρήστης μόλις αξιολόγησε μία ταινία με υψηλή βαθμολογία, το σύστημα αμέσως μετά θα ζητήσει και θα εμφανίσει νέες συστάσεις. Η εμπειρία του θεατή είναι ομαλή, καθώς οι αλλαγές εμφανίζονται άμεσα στη διεπαφή και του επιτρέπουν να συνεχίσει την περιήγησή του αξιοποιώντας τις νέες προτάσεις.

2.4 Αποθήκευση Δεδομένων

Για την αποθήκευση των δεδομένων του συστήματος χρησιμοποιείται το Azure Blob Storage μια υπηρεσία της Azure για αποθήκευση αρχείων στο cloud. Όλες οι πληροφορίες που χρειάζεται το σύστημα συστάσεων (τα ιστορικά αλληλεπίδρασης χρηστών) διατηρούνται σε αρχεία CSV. Κάθε γραμμή περιέχει το user ID, το movie ID και την βαθμολογία του θεατή.

Η επιλογή χρήσης απλών CSV αρχείων στο Azure Blob Storage προσφέρει έναν εύκολο και οικονομικό τρόπο αποθήκευσης των δεδομένων. Η Azure Function έχει πρόσβαση σε αυτά τα αρχεία μέσω του Azure SDK και μπορεί να διαβάζει και να γράφει δεδομένα σε αυτά. Έτσι, το ιστορικό των δεδομένων παραμένει αποθηκευμένο με μόνιμο τρόπο στο cloud, ανεξάρτητα από τις επιμέρους εκτελέσεις των Azure Functions που είναι προσωρινές και κλείνουν μόλις ολοκληρωθεί η επεξεργασία τους.

3. Εργαλεία και Τεχνολογίες Ανάπτυξης

3.1 Γλώσσες προγραμματισμού

Για την ανάπτυξη του συστήματος, χρησιμοποιήθηκαν γλώσσες προγραμματισμού τόσο για την υλοποίηση του αλγορίθμου μηχανικής μάθησης, όσο και για την δημιουργία της διαδικτυακής πλατφόρμας. Χρησιμοποιήθηκε Python για την ανάπτυξη του αλγορίθμου, ενώ για την δημιουργία ιστοσελίδων αξιοποιήθηκε η βιβλιοθήκη React της Javascript.

Python

Για την ανάπτυξη του συστήματος χρησιμοποιήθηκε η Python, μια δυναμική, ερμηνευόμενη γλώσσα υψηλού επιπέδου με απλή, ευανάγνωστη σύνταξη που επιτρέπει ταχύτερη γραφή και συντήρηση κώδικα. Η μεγάλη κοινότητά της και το πλήθος βιβλιοθηκών που υποστηρίζει καλύπτουν όλες τις ανάγκες από την προεπεξεργασία δεδομένων μέχρι την απεικόνιση τους.

Η Python αποτελεί πλέον την κλασσική γλώσσα υλοποίησης αλγορίθμων μηχανικής μάθησης, πράγμα που γίνεται εφικτό μέσω βιβλιοθηκών όπως η Pytorch, που είναι εξειδικευμένες στη βαθιά μάθηση.

React

Η React είναι μία open source βιβλιοθήκη της διαδεδομένης γλώσσας προγραμματισμού Javascript, που αναπτύσσει η εταιρία Meta, και αλλάζει την παραδοσιακή παραγωγή ιστοσελίδων με HTML. Αντί ο προγραμματιστής να γράφει μία μεγάλη σελίδα HTML, με scripts που αλλάζουν το DOM, η React επιτρέπει την δημιούργια διεπαφών χρήστη (UI) ως επαναχρησιμοποιούμενα κομμάτια (components).

Το React λειτουργεί δημιουργώντας ένα εικονικό DOM στη μνήμη αντί να χειρίζεται απευθείας το DOM του προγράμματος περιήγησης. Πραγματοποιεί τις απαραίτητες μετατροπές σε αυτήν την εικονική αναπαράσταση πριν εφαρμόσει τις αλλαγές στο πραγματικό DOM . Έτσι απλοποιεί τη δημιουργία διαδικτυακών εφαρμογών δίνοντας ιδιαίτερη έμφαση στην απόδοση και τη συντηρησιμότητα του λογισμικού.

Δηλαδή η React δεν αντικαθιστά το HTML, απλά προτείνει έναν πιο βολικό και προγραμματιστικό τρόπο υλοποίησης του HTML κώδικα, χωρίς ο προγραμματιστής να χρειαστεί να γράψει άμεσα HTML.

Το DOM (Document Object Model) είναι το δέντρο που αναπαριστά τη σελίδα HTML (μαζί με το CSS). Κάθε στοιχείο HTML αντιστοιχεί σε έναν κόμβο στο DOM.

3.2 Βασικές Βιβλιοθήκες

Εκτός από τις γλώσσες προγραμματισμού, χρησιμοποιήθηκαν και βιβλιοθήκες που αυξάνουν τις δυνατότητες τους.

NumPy

Στον πυρήνα των αριθμητικών υπολογισμών βρίσκεται η βιβλιοθήκη NumPy, η οποία παρέχει το πολυδιάστατους πίνακες αποδοτικά υλοποιημένους σε C και ένα ευρύ σύνολο συναρτήσεων για γραμμική άλγεβρα, στατιστική και άλλα.

Pandas

Η βιβλιοθήκη Pandas δίνει τα εργαλεία για την προετοιμασία των datasets που χρειάζονται για να εκπαιδευτούν οι αλγόριθμοι μηχανικής μάθησης. Με μεθόδους ομαδοποίησης, καθαρισμού και γενικότερα επεξεργασίας των δεδομένων θεωρείται απαραίτητη.

Pytorch

Η Pytorch, με γνώμονα το Tensor (πολυδιάστατους πίνακες), μπορεί να επιταχυνθεί σε GPU, ενσωματώνει έναν αυτόματο υπολογισμό παραγώγων μέσω υπολογιστικών γραφημάτων, προσφέροντας ευελιξία στο σχεδιασμό μοντέλων. Είναι σχεδιασμένη για τις εφαρμογές της βαθιάς μάθησης παρέχοντας έτοιμα επίπεδα, συναρτήσεις βελτιστοποίησης, κόστους, εκπαίδευσης και

πολλά άλλα. Επιπλέον, η οικογένεια βιβλιοθηκών TorchVision, TorchText και TorchAudio διευκολύνει την πρόσβαση σε datasets και μετασχηματισμούς για εικόνες, κείμενο και ήχο, ενώ παρέχει και εργαλεία που επιτρέπουν την εξαγωγή μοντέλων σε περιβάλλοντα παραγωγής.

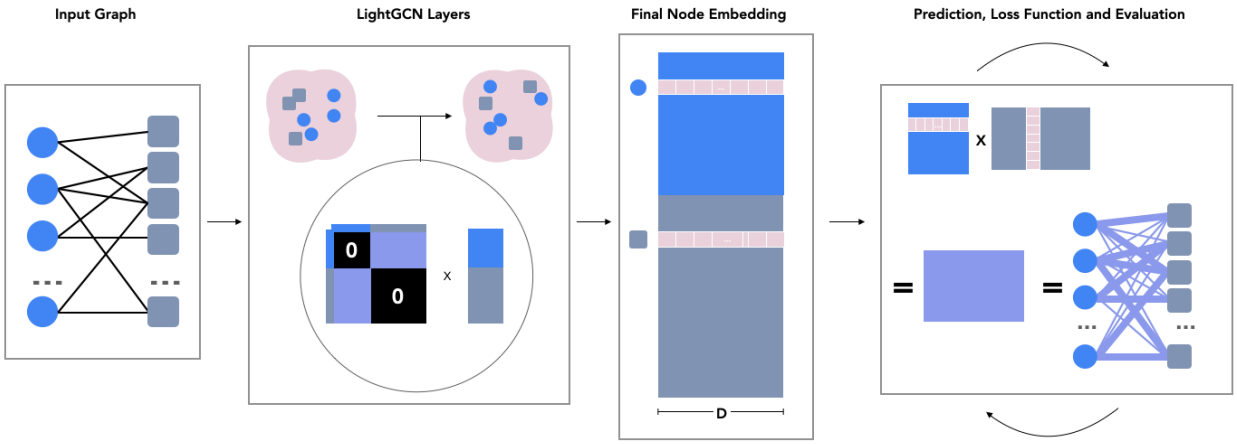
3.3 Εισαγωγή στο LightGCN

Ο LightGCN αποτελεί την λογική με την οποία παράγονται οι προτάσεις των ταινιών, δηλαδή οι προβλέψεις του συστήματος για το ποιά ταινία θα άρεσε περισσότερο στον χρήστη με βάση τις προηγούμενες ταινίες που έχει δει. Όπως οι περισσότεροι αλγόριθμοι σύστασης, βασίζεται σε Graph Neural Network (GNN). Σε αντίθεση όμως με τα συνηθισμένα πιο περίπλοκα δίκτυα γραφημάτων, το LightGCN κρατά μόνο τα απολύτως απαραίτητα στοιχεία, τις σχέσεις που συνδέουν τον χρήστη με τα αντικείμενα, που στην προκειμένη περίπτωση είναι η βαθμολογία που δίνει ο θεατής στις ταινίες που έχει δεί.

Πρακτικά ο αλγόριθμος προσπαθεί να αναθέσει σε κάθε θεατή και κάθε ταινία ένα διανυσματικό “προφίλ” (embedding), έτσι ώστε παρόμοιοι θεατές ή ταινίες να έχουν κοντινά διανύσματα. Τελικός σκοπός του αλγορίθμου είναι για κάθε θεατή να βαθμολογήσει νοητά όλες τις ταινίες που δεν έχει δει, και να τις κατατάξει με αυτήν την βαθμολογία ξεκινώντας από αυτές που δεν πιστεύει ότι αυτός ο χρήστης θα έβλεπε και καταλήγοντας σε αυτές που πιστεύει ότι θα ήθελε να δει πολύ.

Το πρόβλημα της σύστασης έτσι μετατρέπεται σε ένα πρόβλημα πρόβλεψης συνδέσμων σε έναν γράφο. Το μοντέλο προσπαθεί να προβλέψει ποιες ακμές (συνδέσεις θεατή-ταινίας) που δεν υπάρχουν ακόμη θα έπρεπε να υπάρχουν, δηλαδή ποιες ταινίες θα άρεσαν σε ποιον χρήστη.

Εδώ συνδέεται και η συνέλιξη (convolution) που προσθέθηκε στην κλασσική αρχιτεκτονική των GNN ώστε να μετατραπεί σε GCN (Graph Convolutional Network). Η μαθηματική πράξη της συνέλιξης, σε πρακτικό επίπεδο εντοπίζει συγκεκριμένα χαρακτηριστικά ανάλογα με τους αριθμούς που υπάρχουν στον πίνακα της συνέλιξης. Για το LightGCN τα χαρακτηριστικά είναι οι σχέσεις μεταξύ θεατών και ταινιών. Αυτή η μινιμαλιστική σχεδίαση μειώνει την πολυπλοκότητα και το κόστος υπολογισμού, ενώ παράλληλα συλλαμβάνει αποτελεσματικά τη δομή του γράφου.



Εικόνα 2: Λειτουργικό διάγραμμα του LightGCN

3.4 Πώς διαφέρει από άλλες μεθόδους σύστασης

Το LightGCN προέκυψε από την παρατήρηση ότι οι κλασικές αρχιτεκτονικές GNN ήταν υπερβολικά πολύπλοκες και περιττές για το πρόβλημα του φιλτραρίσματος αντικειμένων. Στις συστάσεις, οι κόμβοι (χρήστες και αντικείμενα) δεν έχουν πλούσια αρχικά χαρακτηριστικά και ουσιαστικά αναπαρίστανται μόνο από το μοναδικό τους ID. Με άλλα λόγια, κάθε χρήστης/αντικείμενο ξεκινά με ένα αναγνωριστικό χωρίς νόημα πέρα από την ταυτότητά του. Σε μια τέτοια περίπτωση, το να εφαρμόσουμε πολλαπλά στρώματα με μη γραμμικούς μετασχηματισμούς κάνει το μοντέλο πιο δύσκολο να εκπαιδευτεί. Ερευνητές διαπίστωσαν πειραματικά ότι αφαιρώντας αυτά τα στοιχεία η απόδοση όχι μόνο δεν πέφτει, αλλά βελτιώνεται σημαντικά.

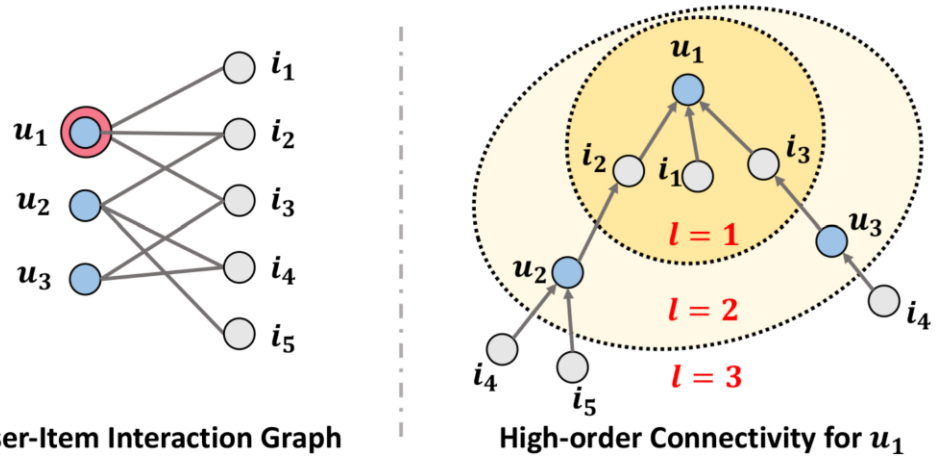
3.5 Αρχιτεκτονική του LightGCN

Ο γράφος του LightGCN ονομάζεται διμερές γράφημα (bipertite graph), και το βασικό του χαρακτηριστικό είναι ότι οι κόμβοι του γράφου ανοίκουν αναγκαστικά σε 2 κατηγορίες χρήστες και αντικείμενα (στη προκειμένη περίπτωση θεατές και ταινίες), σχηματίζοντας ένα γράφημα όπου ακμές υπάρχουν μόνο μεταξύ χρήστη και αντικειμένου και όχι μεταξύ 2 χρηστών ή 2 αντικειμένων. Κάθε χρήστης και κάθε αντικείμενο ξεκινά με ένα αρχικό διάνυσμα που είναι ουσιαστικά τα μόνα παραμετροποιήσιμα μεγέθη του μοντέλου. Δηλαδή, δεν χρειάζεται κανένα επιπλέον γνώρισμα για τους κόμβους πέρα από αυτά τα ενσωματωμένα διανύσματα τα οποία μαθαίνονται κατά τη διάρκεια της εκπαίδευσης του μοντέλου, όπως σε όλους τους αλγορίθμους μηχανικής μάθησης.

Σε επίπεδο αρχιτεκτονικής, το LightGCN πραγματοποιεί “K” επίπεδα συνελικτικών βημάτων διάδοσης (propagation layers) πάνω στο γράφημα. Σε κάθε τέτοιο επίπεδο, μεταδίδεται

η πληροφορία από τους γείτονες ενός κόμβου προς τον ίδιο. Αντίστοιχα, το embedding ενός αντικειμένου ενημερώνεται από τα embeddings των χρηστών που το έχουν ως γείτονές τους.

Μετά την ολοκλήρωση όλων των επιπέδων διάδοσης, υπολογίζει το τελικό embedding κάθε κόμβου (χρήστη ή αντικειμένου) ως έναν συνδυασμό όλων των επιπέδων. Με αυτό τον συνδυασμό, το τελικό διάνυσμα του χρήστη περιλαμβάνει τόσο το δικό του αρχικό embedding, όσο και τις πληροφορίες από τους γείτονες κάθε βαθμίδας. Διατηρούμε έτσι την πληροφορία τόσο από κοντινές όσο και από πιο μακρινές σχέσεις, αποφεύγοντας την πλήρη αλλοίωση της μοναδικότητας κάθε κόμβου



Εικόνα 3: Αριστερά, διμερές γράφημα, Δεξιά, τα διαφορετικά επίπεδα με τους κόμβους και γείτονες τους

4. Αναμενόμενη είσοδος και έξοδος

Για τη δημιουργία προτάσεων μέσω του LightGCN, η είσοδος είναι το user ID του θεατή για τον οποίο θέλουμε συστάσεις. Με αυτό το δεδομένο, το μοντέλο ανακτά το διάνυσμα που αντιστοιχεί σε αυτόν τον θεατή και παράχθηκε κατά την εκπαίδευση.

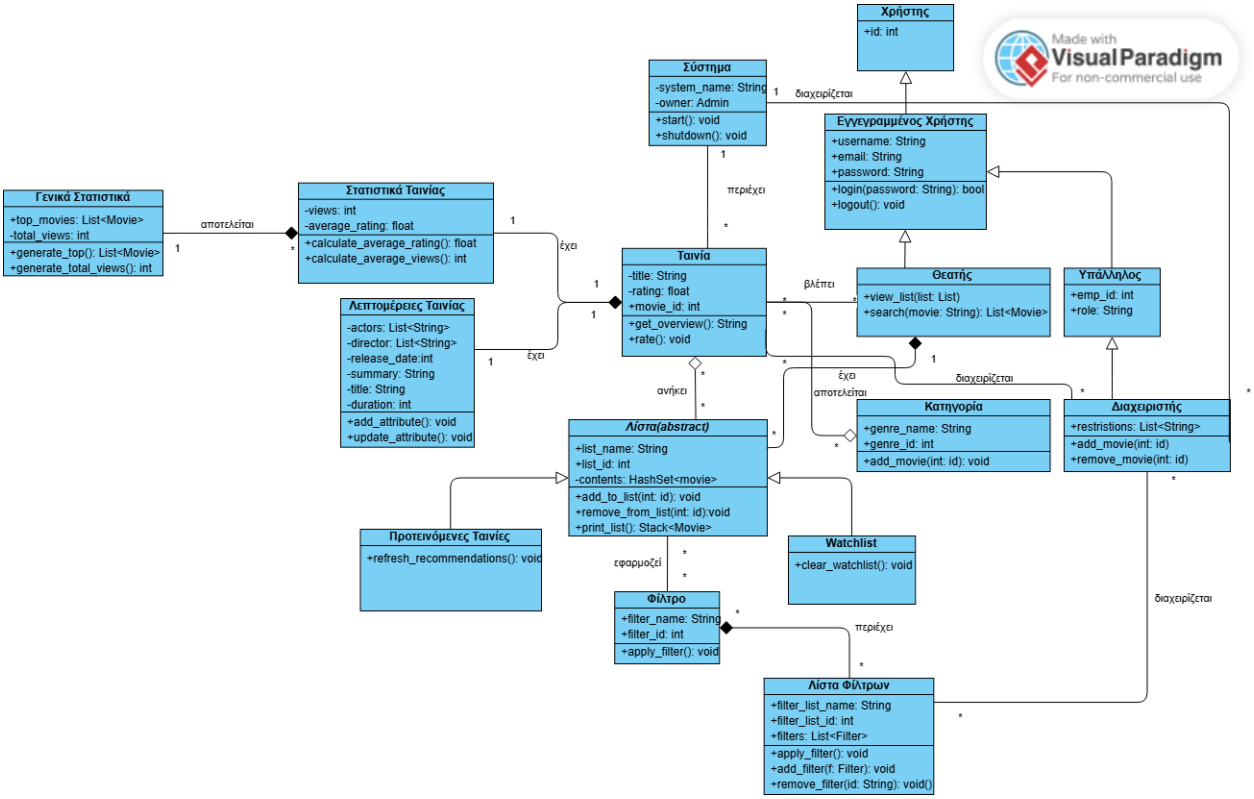
Πρακτικά το σύστημα λαμβάνει αυτόματα το user ID του συνδεδεμένου χρήστη στη συσκευή και κάθε φορά που θεατής βαθμολογεί μια νέα ταινία, το API αναβαθμίζει τα αντίστοιχα στοιχεία του πίνακα csv, προσθέτοντας μια νέα γραμμή με user ID την νεά βαθμολογία και το movie ID της βαθμολογημένης ταινίας

Η έξοδος του αλγορίθμου είναι μια βαθμολογία συνάφειας για κάθε ταινία, που υποδεικνύει πόσο πιθανό είναι ο χρήστης να προτιμήσει αυτή τη ταινία. Το LightGCN χρησιμοποιεί το εσωτερικό γινόμενο του διανύσματος του θεατή και του διανύσματος της ταινία που δίνει ως αποτέλεσμα τη βαθμολογία προτίμησης. Αυτές οι βαθμολογίες στη συνέχεια να ταξινομούνται για να δημιουργήσουν μια λίστα.

Οι προτεινόμενες ταινίες είναι αυτές με τις υψηλότερες βαθμολογίες. Έπειτα μέσω του API στέλνονται οι συστάσεις στο frontend για να τις λάβει ο θεατής.

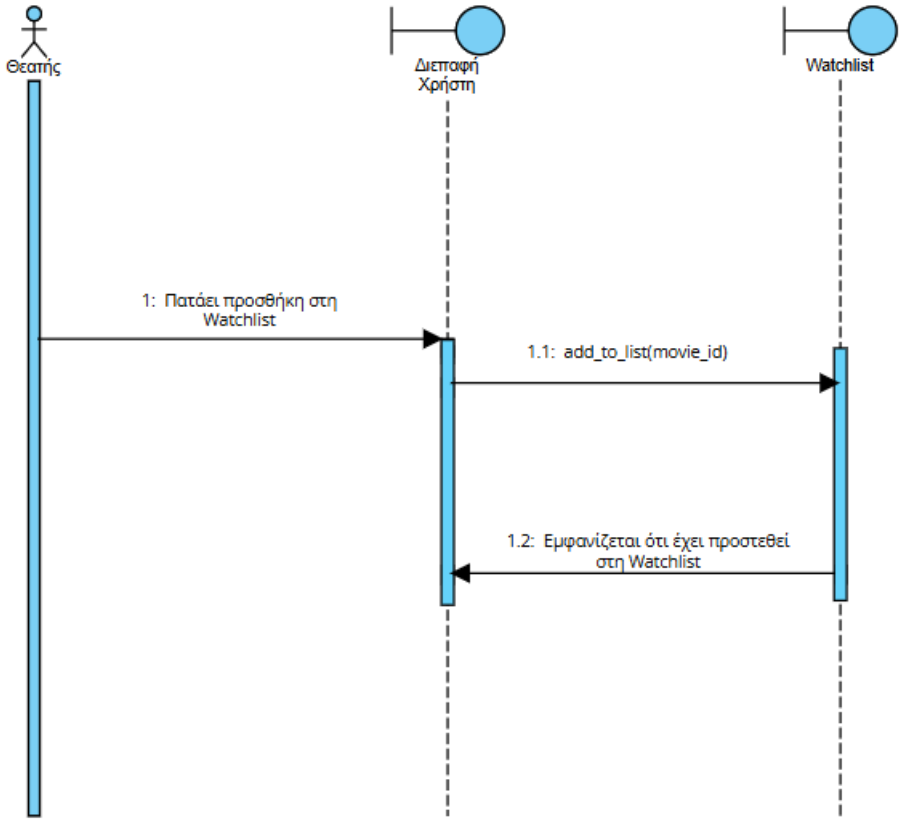
5. Σχεδίαση Πλατφόρμας

5.1 Διάγραμμα Κλάσεων



5.2 Διαγράμματα Ακολουθίας

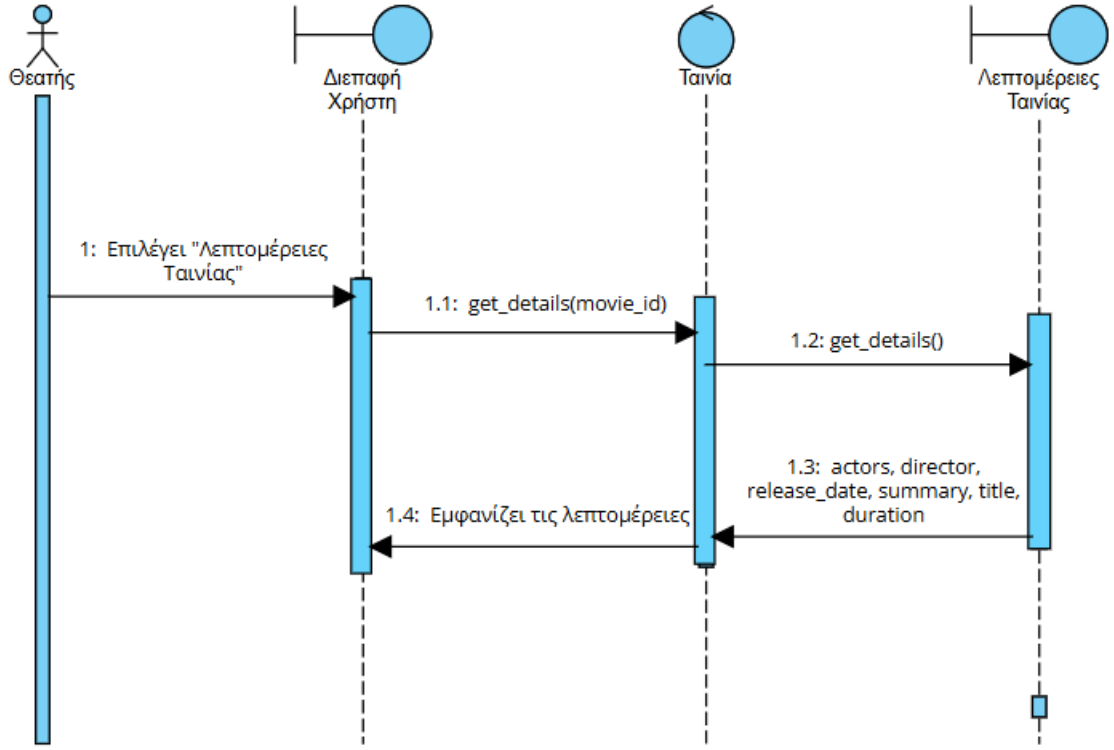
Διάγραμμα ακολουθίας για την περίπτωση «Προσθήκη στη Watchlist»



Το παραπάνω διάγραμμα ακολουθίας παρουσιάζει τη διαδικασία με την οποία ένας θεατής προσθέτει μια ταινία που επιθυμεί στη λίστα Watchlist. Αρχικά, ο θεατής πατάει την επιλογή "Προσθήκη στη Watchlist" (βήμα 1). Η Διεπαφή Χρήστη καλεί τη συνάρτηση add_to_list(movie_id) (βήμα 1.1), περνώντας το αναγνωριστικό της ταινίας. Η ενέργεια αυτή αποστέλλεται στην κλάση Watchlist, η οποία χειρίζεται την αποθήκευση της ταινίας στη λίστα

του χρήστη. Μόλις ολοκληρωθεί η προσθήκη, η Διεπαφή Χρήστη λαμβάνει επιβεβαίωση και εμφανίζει μήνυμα που ενημερώνει ότι η ταινία έχει προστεθεί επιτυχώς στη Watchlist (βήμα

Διάγραμμα ακολουθίας για την περίπτωση «Προβολή Λεπτομερειών Ταινίας»



Το παραπάνω διάγραμμα ακολουθίας απεικονίζει τη διαδικασία προβολής των λεπτομερειών μιας ταινίας. Ο θεατής ουσιαστικά, πατάει στο κουμπί "Λεπτομέρειες Ταινίας" (βήμα 1), και η Διεπαφή Χρήστη καλεί τη συνάρτηση get_details(movie_id) (βήμα 1.1), περνώντας το αναγνωριστικό της επιλεγμένης ταινίας. Η κλάση Ταινία, λειτουργώντας ως ενδιάμεσο controller, ζητά τις λεπτομέρειες από την κλάση Λεπτομέρειες Ταινίας μέσω της get_details() (βήμα 1.2). Αυτή επιστρέφει όλα τα απαραίτητα δεδομένα της ταινίας, όπως ηθοποιούς, σκηνοθέτη, ημερομηνία κυκλοφορίας, περίληψη, τίτλο και διάρκεια (βήμα 1.3). Τέλος, τα δεδομένα αποστέλλονται πίσω στη Διεπαφή Χρήστη, όπου και προβάλλονται στον θεατή (βήμα 1.4).

Βιβλιογραφία

Xiangnan He, Kuan Deng, Xiang Wang, "LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation"