Πανεπιστήμιο Δυτικής Αττικής
Σχολή Μηχανικών
Τμήμα Βιομηχανικής Σχεδίασης και Παραγωγής

Διπλωματική εργασία

Συστήματα συστάσεων με αλγόριθμους μηχανική μάθησης

ΓΚΡΕΚΑΣ ΔΗΜΗΤΡΙΟΣ

Επιβλέπων Καθηγητής: Νικολάου Γρηγόριος

Αιγάλεω – Αθήνα – 2023

Μέλη Εξεταστικής Επιτροπής:

Νικολάου Γρηγόριος	Δρόσος Χρήστος	Βασιλειάδου Σουλτάνα
GRIGORIOS NIKOLAOU Digitally signed by GRIGORIOS NIKOLAOU Date: 2023.07.21 12:06:43 +03'00'	Christos Drosos Digitally signed by Christos Drosos Date: 2023.07.23 15:36:06 +03'00'	Soultana Vasileiadou Digitally signed by Soultana Vasileiadou Date: 2023.07.25 08:54:05 +03'00'

ο παρακάτω υπογράφων Γκρέκας Δημήτριος του Δανιήλ, με αριθμό μητρώου 71446170, επιβεβαιώνω επισήμως τα εξής: Είμαι ο συγγραφέας της παρούσας Διπλωματικής εργασίας και καταθέτω υπεύθυνη δήλωση ότι, οποιεσδήποτε πηγές χρησιμοποίησα για δεδομένα, ιδέες ή λέξεις, είτε αυτές είναι ακριβείς είτε παραφρασμένες, έχουν αναφερθεί πλήρως με αναφορά στους συγγραφείς, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο.

ΓΚΡΕΚΑΣ ΔΗΜΗΤΡΙΟΣ

Περίληψη

Η εργασία εξετάζει την εξέλιξη, την εφαρμογή και τις διάφορες μεθοδολογίες των συστημάτων συστάσεων.

Ξεκινώντας με την ουσιαστική κατανόηση των συστημάτων συστάσεων, η εργασία παρακολουθεί την ιστορική προέλευσή τους ενώ επισης εξετάζει την εξέλιξη τους. Στη συνέχεια, επεκτείνεται στην διερεύνηση της εφαρμογής των συστημάτων συστάσεων σε διάφορους τομείς, από το ηλεκτρονικό εμπόριο και την ψυχαγωγία, μέχρι τα κοινωνικά δίκτυα και την ενημέρωση, επισημαίνοντας την αυξανόμενη σημασία και την πανταχού παρουσία τους στην καθημερινή μας ζωή.

Επιπλέον, η εργασία εξετάζει διεξοδικά τους διάφορους μηχανισμούς που χρησιμοποιούνται για την υλοποίηση των συστημάτων συστάσεων. Παρουσιάζει τη μεθοδολογία του συνεργατικού φιλτραρίσματος, των συστάσεων βασισμένων στο περιεχόμενο και των συστάσεων βασισμένων στη γνώση, καθώς και την χρήση της βαθιάς μάθησης για την βελτίωση της ακρίβειας των συστάσεων.

Το τελευταίο κεφάλαιο της εργασίας αφιερώνεται στην πρακτική εφαρμογή των συστημάτων συστάσεων, με τη δημιουργία ενός πρακτικού συστήματος συστάσεων χρησιμοποιώντας έναν αλγόριθμο μηχανικής μάθησης. Παρουσιάζεται η διαδικασία εκπαίδευσης του αλγορίθμου, η επιλογή και προετοιμασία των δεδομένων, καθώς και η ανάλυση και η δόμηση της εφαρμογής. Τέλος, προσφέρεται μια κριτική αξιολόγηση του συστήματος, επισημαίνοντας τα περιθώρια για μελλοντική βελτίωση.

Πίνακας περιεχομένων

Εισαγωγή

Στην βάση του ένα σύστημα συστάσεων έχει σκοπό να διευκολύνει την κατάληξη σε μια επιλογή η σε ένα συμπέρασμα όταν οι πληροφορίες είναι πάρα πολλές. Στην σήμερον ημερα τέτοια παραδειγματα είναι πολύ εύκολο να βρει κάνεις, όπως το Netflix που προτείνει μια ταινία μέσα από την γιγαντιαία βιβλιοθήκη του με βάση τις προτιμήσεις του προφίλ που έχει συνδεθεί ο χρήστης ή το Facebook που βρίσκει πιθανούς γνωστούς ανάμεσα σε δισεκατομμύρια χρήστες, χρησιμοποιώντας τα στοιχεία που έχει για τον χρήστη.

Θα έλεγε κάνεις ότι τα συστήματα συστάσεων είναι μια καινούργια τεχνολογία, αλλά δεν είναι και δεν χρησιμοποιούνται μόνο από τον άνθρωπο. Σε μια καινούργια αποικία μυρμηγκιών τα μυρμηγκιά αφού εγκατασταθούν δεν γνωρίζουν πως θα έχουν πρόσβαση σε τροφή, έτσι κάθε εργάτης μυρμήγκι φεύγει από την αποικία ψάχνονται για πρόσβαση σε τροφή όταν ένα μυρμήγκι βρει φαγητό τότε γυρνώντας στην αποικία αφήνει πίσω του μια μυρωδιά που τα αλλά μυρμήγκια θα ακολουθήσουν και αν όντος ισχύ ότι εκεί υπάρχει φαγητό τότε και αυτά με την σειρά τους θα κάνουν το ίδιο, κάνοντας έτσι την μυρωδιά ποιο ισχυρή. Τα μυρμήγκια παρουσιάζουν ένα είδος συστήματος συστάσεων όπου καθένα από τα μυρμήγκια βγαίνει και εξερευνά ένα διαφορετικό μέρος του χώρου και όταν βρίσκουν κάτι που πιστεύουν ότι θα ήθελε η κοινότητα, τότε με τον τρόπο τους κάνουν τους άλλους να το μάθουν. Αυτό δεν διαφέρει από αλγόριθμους που χρησιμοποιούνται στα συστήματα συστάσεων όπως τις μεθόδους Top-N recommendations που θα μελετηθούν παρακάτω[1], [2].

Τα συστήματα συστάσεων χρησιμοποιούνται ευρέος και εφαρμόζονται σε πάρα πολλούς τομείς, υπάρχουν πόλοι αλγόριθμοί για να δημιουργηθεί ένα σύστημα συστάσεων, ο κάθε ένας χρησιμοποιείται για την επίλυση διαφορετικών προβλημάτων και συχνά χρησιμοποιούνται και μαζί. Επισης στην ποιο αναπτυγμένη τους μορφή χρησιμοποιούν αλγορίθμους βαθιάς μάθησης και τεχνητής νοημοσύνης.

Σκοπός

Η παρούσα Διπλωματική εργασία έχει ως σκοπό, την ανάλυση των συστημάτων συστάσεων, να εξεταστεί τι είναι και ποιος είναι ο σκοπός τους και έπειτα να διατυπωθεί με ποιες μεθόδους υλοποιούνται. Τέλος θα γίνει μια δικιά μου υλοποίηση ενός συστήματος συστάσεων με χρήση αλγορίθμων μηχανικής μάθησης.

Κεφάλαιο 1
Συστήματα συστάσεων

Τι είναι τα συστήματα συστάσεων;

Τα Συστήματα Συστάσεων είναι εργαλεία που παρέχουν προτάσεις για αντικείμενα που είναι χρήσιμα σε έναν χρήστη. Οι προτάσεις σχετίζονται με διάφορες διαδικασίες λήψης αποφάσεων, όπως ποια προϊόντα από ένα κατάστημα μπορεί να θέλει να αγοράσει ο χρήστης, ποια μουσική θα προτιμήσει να ακούσει ή ποιο άρθρο θα βρει ενδιαφέρον. Το ‘’αντικείμενο’’ ή ‘’item’’, είναι ο γενικός όρος που χρησιμοποιείται όταν αναφερόμαστε στο τι συνιστά το σύστημα στους χρήστες. Ένα Συστήματα Συστάσεων συνήθως επικεντρώνεται σε έναν συγκεκριμένο τύπο αντικειμένου (π.χ. εξατομικευμένες διαφημίσεις) και με βάση αυτό, καταλήγουμε στο σχεδιασμό του, το περιβάλλον διεπαφή χρήστη και τη βασική τεχνική προτάσεων που προσαρμόζεται για να παρέχει χρήσιμες και αποτελεσματικές προτάσεις για αυτόν τον συγκεκριμένο τύπο αντικειμένου. [3]

Τα Συστήματα Συστάσεων απευθύνονται κυρίως σε άτομα που δεν διαθέτουν την απαραίτητη γνώση ή ικανότητα για την αξιολόγηση του πιθανά συντριπτικού αριθμού εναλλακτικών αντικειμένων που μπορεί να προσφέρει ένας ιστότοπος, για παράδειγμα. Ένα σύστημα σύστασης που βοηθά τους χρήστες να επιλέξουν νεα αντικείμενο για να αγοράσουν όπως αυτό στον ιστότοπο Amazon.com.

Η Amazon χρησιμοποιεί ένα σύστημα σύστασης για την εξατομίκευση του διαδικτυακού καταστήματος για κάθε πελάτη, κάνοντας κλικ στο σύνδεσμο "Οι προτάσεις σας" στο Amazon.com οδηγεί τον κάθε χρήστη σε μια σελίδα γεμάτη από προϊόντα που προτείνονται μόνο για αυτόν. Η Amazon προτείνει μια σειρά προϊόντων από διαφορετικές κατηγορίες στις οποίες περιηγήθηκε ο χρήστης, με σκοπό να προβάλει προϊόντα μπροστά στον χρήστη στα οποία είναι πιθανό να κάνει κλικ, να ενημερωθεί για αυτά και έτσι να αγοράσει ένα ή και πολλά από αυτά. [2], [4]. Δεδομένου ότι οι προτάσεις είναι συνήθως εξατομικευμένες, διαφορετικοί χρήστες ή ομάδες χρηστών λαμβάνουν διαφορετικές προτάσεις. Επιπλέον, υπάρχουν μη εξατομικευμένες προτάσεις. Αυτές είναι πολύ πιο απλές να δημιουργηθούν και εμφανίζονται συνήθως σε περιοδικά ή εφημερίδες. Τυπικά παραδείγματα περιλαμβάνουν τις δέκα κορυφαίες επιλογές βιβλίων, Billboard Hot 100 κ.α.. [3]

Με άλλα λόγια η βασική αρχή για ένα σύστημα σύστασης είναι ότι υπάρχουν σημαντικές εξαρτήσεις μεταξύ τον χρήστη και το αντικείμενο. Για παράδειγμα, ένας χρήστης που ενδιαφέρεται για μια ταινία δράσης είναι πιο πιθανό να ενδιαφέρεται για μια άλλη ταινία δράσης ή ένα θρίλερ, παρά για ένα ντοκιμαντέρ. Σε πολλές περιπτώσεις, διάφορες κατηγορίες αντικειμένων ενδέχεται να παρουσιάζουν σημαντικούς συσχετισμούς, οι οποίοι μπορούν να αξιοποιηθούν για να κάνουν πιο ακριβείς προτάσεις. Εναλλακτικά, οι συσχετισμοί αυτοί μπορεί να υπάρχουν με λεπτομέρεια σε μεμονωμένα αντικείμενα παρά στις κατηγορίες. Αυτοί οι συσχετισμοί μπορούν να εξαχθούν με βάση τα δεδομένα από τον πίνακα βαθμολογίας και το ένα μοντέλο χρησιμοποιείται για την πραγματοποίηση προβλέψεων για τους χρήστες. Όσο μεγαλύτερος είναι ο αριθμός των ονομαστικών στοιχείων που είναι διαθέσιμα για έναν χρήστη, τόσο πιο εύκολο είναι να πραγματοποιηθούν ισχυρές προβλέψεις για τη μελλοντική συμπεριφορά του χρήστη. Πολλά διαφορετικά μοντέλα μάθησης μπορούν να χρησιμοποιηθούν για την ολοκλήρωση αυτής της εργασίας. Για παράδειγμα, η συλλογική συμπεριφορά αγοράς ή αξιολόγησης διαφόρων χρηστών μπορεί να αξιοποιηθεί για τη δημιουργία ομάδων παρόμοιων χρηστών που ενδιαφέρονται για παρόμοια προϊόντα. Τα ενδιαφέροντα και οι ενέργειες αυτών των ομάδων μπορούν να αξιοποιηθούν για να κάνουν συστάσεις σε μεμονωμένα μέλη αυτών των ομάδων.[3]

Ιστορική αναδρομή

Όπως αναφέρθηκε και προηγούμενος τα συστήματα συστάσεων υπάρχουν στην φύση και οι άνθρωποι αλληλοεπιδρούν μαζί τους από σχεδόν πάντα, όπως για παράδειγμα, όταν υπήρχαν βασίλεια γύρω από το κόσμο, ο βασιλιάς δεχόταν προτάσεις από τους υπουργούς του για σχεδόν κάθε θέμα. Προτείνοντας ποιες πολιτικές πρέπει να εφαρμοστούν για τη βελτίωση των μαζών και άλλα θέματα που αφορούν το βασίλειο είναι μερικά παραδείγματα. Όλα αυτά ήταν συστάσεις ο βασιλιάς δεχόταν προτάσεις και στο τέλος έπαιρνε τις αποφάσεις. Ομοίως, οι άνθρωποι που ζητούν απόψεις από άλλους για το πού να αγοράσουν τα καλύτερα αγαθά ή ποιος προορισμός είναι καλός για διακοπές, το συντομότερο μονοπάτι για να φτάσουν σε κάποιον προορισμό. Όλες αυτές είναι συστάσεις που υπήρχαν στην κοινωνία από πολύ παλιά. Στον παλιό κόσμο δεν υπήρχαν υπολογιστές, αλλά ακόμα και τότε οι άνθρωποι δεχόντουσαν συστάσεις. Αλλά με τη Βιομηχανική Επανάσταση, όχι μόνο αυξήθηκαν οι επιλογές σε διάφορους τομείς, αλλά ήρθε και η εποχή του υπολογιστή με αποτέλεσμα την επανάσταση στην παγκόσμια αγορά. Οι άνθρωποι ερχόντουσαν αντιμέτωποι με πάρα πολλές επιλογές,

πράγματα όπως το τι ρούχα θα αγοράσεις συχνά οδηγούσαν σε σύγχυση σχετικά με το ποιο προϊόν θα μπορούσε πραγματικά να καλύψει τις απαιτήσεις του αγοραστή. Έτσι, η ανάγκη για ένα σύστημα που θα μπορούσε να διευκολύνει αυτά τα κριτήρια επιλογής και να εξαλείψει το δίλημμα των μαζών, πραγματοποιήθηκε και έτσι ήρθαν τα πρώτα συστήματα συστάσεων του σύγχρονου κόσμου. [5]

Τα συστήματα συστάσεων εμφανίστηκαν ως ανεξάρτητος ερευνητικός τομέας στα μέσα της δεκαετίας του 1990. Τα τελευταία χρόνια, το ενδιαφέρον για συστήματα σύστασης έχει αυξηθεί δραματικά, διοτι, τα συστήματα συστάσεων έχουν ένα πολύ σημαντικό ρόλο σε ιστότοπους μεγάλης απηχήσεις, όπως Amazon.com, YouTube, Netflix, Spotify, LinkedIn, Facebook, Tripadvisor, Last.fm και IMDb. Επιπλέον, πολλές εταιρείες μέσων αναπτύσσουν συστήματα συστάσεων ως μέρος των υπηρεσιών που παρέχουν στους συνδρομητές τους. Για παράδειγμα, η Netflix τον Οκτώβριο του 2006,αρχισε έναν διαγωνισμό με έπαθλο 1 εκατομμύριο δολάρια σε οποία ομάδα μπορέσει να ανάπτυξη έναν καλύτερο αλγόριθμο συστάσεων από αυτόν που είχε τότε η Netflix . Ο διαγωνισμός κράτησε για 3 χρονιά, αυτά τα χρονιά αυξήθηκε το ενδιαφέρον για τα συστήματα συστάσεων δραματικά. [6] Επισης Υπάρχουν συνέδρια και εργαστήρια αφιερωμένα ειδικά στον τομέα, συγκεκριμένα το Association of Computing Machinery's (ACM) Conference Series on Recommender Systems (RecSys), που ιδρύθηκε το 2007. Το συνέδριο αυτό αποτελεί την κορυφαία ετήσια για την τεχνολογία συστημάτων συστάσεων, την έρευνα και τις εφαρμογές. Επιπλέον, οι συνεδρίες αφιερωμένες στα συστήματα συστάσεων συχνά περιλαμβάνονται σε πιο παραδοσιακά συνέδρια στον τομέα των βάσεων δεδομένων, των πληροφοριακών συστημάτων και των προσαρμοστικών συστημάτων. Πρόσθετα αξιοσημείωτα συνέδρια σε αυτό το πεδίο περιλαμβάνουν: Ομάδα ειδικού ενδιαφέροντος της ACM για την ανάκτηση πληροφοριών (SIGIR) Μοντελοποίηση χρήστη, προσαρμογή και εξατομίκευση (UMAP). Ευφυείς διεπαφές χρήστη (IUI); Παγκόσμιος Ιστός (WWW); και την ομάδα ειδικού ενδιαφέροντος της ACM για τη διαχείριση δεδομένων (SIGMOD). [3]

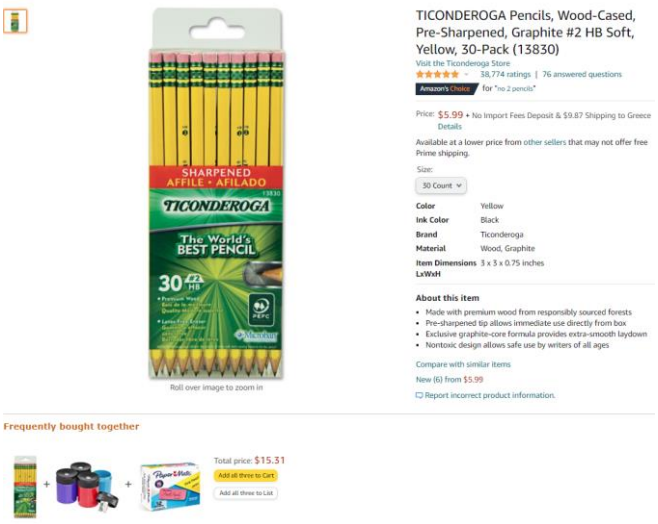
Τα Συστήματα συστάσεων αναπτύχθηκαν για να εξυπηρετήσουν την ταχεία αύξηση των επιλογών μπροστά στις μάζες. Μπορει να είναι μια σχετικά νεα έννοια όσον αφορά τις έρευνες που έχουν πραγματοποιηθεί αλλά έχουν επιτευχθεί πολλά σε πολλούς τον τομείς.

Τομείς Εφαρμογής

Σχεδόν όλοι οι τομείς μπορούν να επωφεληθούν από ένα σύστημα συστάσεών. Υπάρχουν δύο σημαντικές πτυχές, μια είναι το εύρος δεδομένων και η άλλη είναι το βάθος δεδομένων και οι δυο είναι πολύ σημαντικές και καθορίζουν πόσο ωφέλιμο είναι να αναπτυχθεί ένα σύστημα συστάσεων. Το εύρος δεδομένων έχει να κάνει με το τι όγκο δεδομένο θα έρχεται αντιμέτωπο το σύστημα συστάσεων, ενώ το βάθος δεδομένων είναι το πόσο χρήσιμα είναι τα δεδομένα. Αυτά τα δυο μαζί δείχνουν πόσο ωφέλιμο θα είναι ένα σύστημα συστάσεων. Παρακάτω αναφέρονται πως και που εφαρμόζονται σε διάφορους τομείς τα συστήματα συστάσεων.

E-commerce (Ηλεκτρονικό εμπόριο)

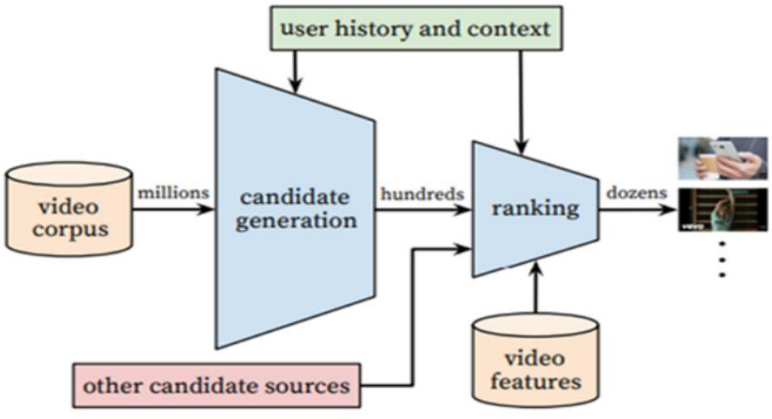
Τις τελευταίες δεκαετίες, στον τομέα του ηλεκτρονικού εμπορίου έχουν αναπτυχθεί διάφορα συστήματα σύστασης για την παροχή ηλεκτρονικής βοήθειας στους καταναλωτές που περιηγούνται σε διαδικτυακούς ιστότοπους αγορών. Τα σχόλια και οι αξιολογήσεις των χρηστών αποτελούν τα κοινά κριτήρια για την υποβολή των συστάσεων. Για παράδειγμα, στο play store της Google, οι χρήστες δίνουν βαθμολογίες στις εφαρμογές που έχουν κατεβάσει και υστέρα αυτές οι βαθμολογίες χρησιμοποιούνται για την υποβολή προτάσεων σε άλλους χρήστες. Έχουν αναπτύχθει πολλοι ιστότοποι αγορών μέχρι σήμερα, όπως Amazon και Ebay. Σε τέτοιους ιστότοπους, μπορούμε πάντα να βλέπουμε προτάσεις με τη μορφή: άτομα που αγόρασαν επίσης, με περισσότερες προβολές, παρόμοια προϊόντα, κορυφαίες πωλήσεις [5]



Εικόνα 1 Παράδειγμα πρότασης στο Amazon.com

Ψυχαγωγία

Στον τομέα της ψυχαγωγίας, τα συστήματα συστάσεων έχουν τεράστια δημοτικότητα. Στην μουσική αλλά και στις ταινίες, τα συστήματα σύστασης έχουν καταφέρει πάρα πολλά. Το Spotify,Netflix και YouTube είναι μερικά παραδείγματα συστημάτων σύστασης στην ψυχαγωγία. Τέτοια συστήματα αναπτύσσουν συνήθως collaborative filtering και content-based filtering. Για παράδειγμα, στο YouTube χρησιμοποιείται σύστημα συστάσεων για να δημιουργήσει εξατομικευμένες προτάσεις, ώστε οι χρήστες να βρίσκουν γρήγορα και εύκολα βίντεο που σχετίζονται με τα ενδιαφέροντά τους. Λόγω της αξίας της διατήρησης των χρηστών, το YouTube προσπαθεί να ενημερώνει τακτικά τις προτάσεις, να αντικατοπτρίζει τη δραστηριότητα κάθε χρήστη στον ιστότοπο και να επισημαίνει ταυτόχρονα το ευρύ φάσμα του διαθέσιμου περιεχομένου. Το σύστημα σύστασης καθοδηγείται από το Google Brain ένα έργο τεχνητής νοημοσύνης βαθιάς μάθησης που αποτελείται από δύο νευρωνικά δίκτυα. Το πρώτο συλλέγει και συγκεντρώνει πληροφορίες σχετικά με το ιστορικό παρακολούθησης των χρηστών και χρησιμοποιεί συνεργατικό φιλτράρισμα (collaborative filtering) για την επιλογή εκατοντάδων βίντεο. Αυτή η διαδικασία, γνωστή ως υποψήφια γενιά (candidate generation), χρησιμοποιεί σχόλια από χρήστες για να εκπαιδεύσει το μοντέλο. Το δεύτερο νευρωνικό δίκτυο κατατάσσει τα επιλεγμένα βίντεο για να κάνει προτάσεις στους χρήστες.[7]



Εικόνα 2 Flow chart για την αρχιτεκτονική πίσω απο το σύστημα συστάσεων του YouTube[8]

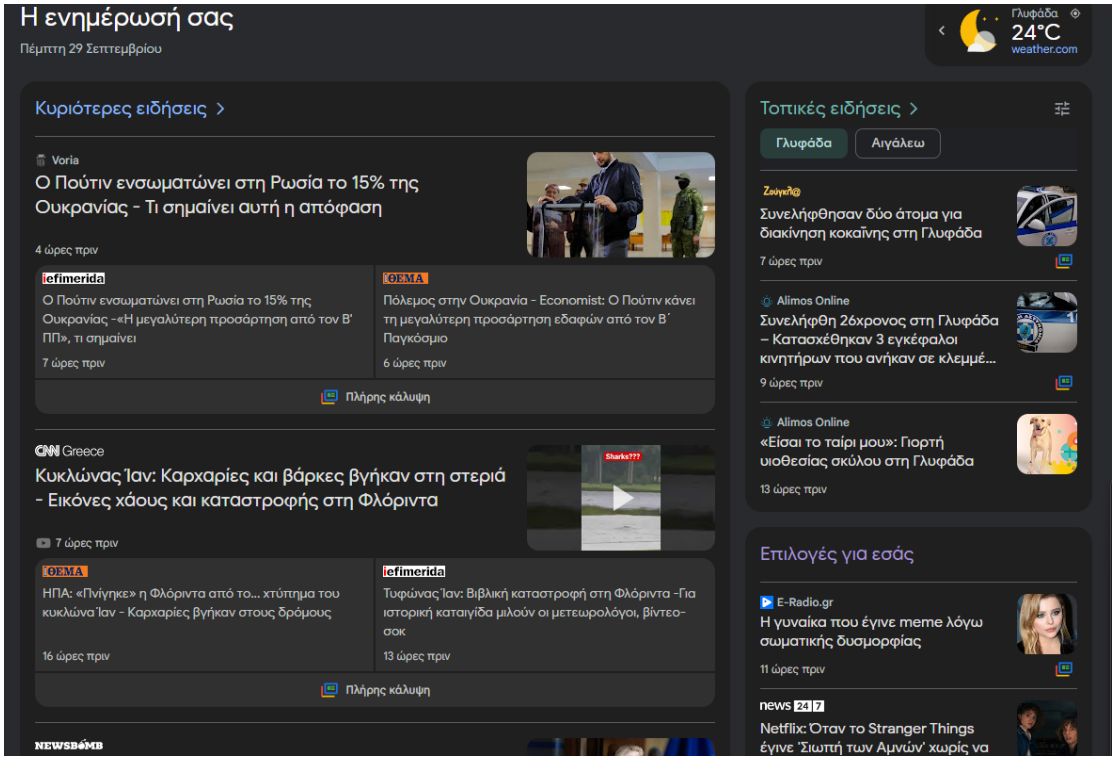
Κοινωνικά δίκτυα

Τα μέσα κοινωνικής δικτύωσης εισήγαγαν πολλούς νέους τύπους περιεχομένου που μπορούν να δημιουργηθούν και να διαμοιραστούν από κάθε χρήστη με τρόπο που δεν ήταν ποτέ πριν εφικτός. Οι χρήστες έγιναν το επίκεντρο κάθε ιστότοπου κοινωνικής δικτύωσης και σε πολλές περιπτώσεις ήταν αυτοί που δημιουργούσαν το πραγματικό περιεχόμενο του ιστότοπου. Περιεχόμενο κειμένου όπως στη Wikipedia και το WordPress, φωτογραφίες όπως στο Flickr και το Facebook και βίντεο όπως στο YouTube. Οι χρήστες έχουν επίσης βασικό ρόλο στην παροχή ανατροφοδότησης και σχολιασμού του υπάρχοντος περιεχομένου στους ιστότοπους κοινωνικής δικτύωσης. Τα σχόλια επιτρέπουν στους χρήστες να προσθέσουν τη δική τους γνώμη- οι ψήφοι και οι αξιολογήσεις τους επιτρέπουν να "αρέσουν" (ή να μην αρέσουν) αγαπημένες αναρτήσεις- και οι ετικέτες τους επιτρέπουν να σχολιάσουν το περιεχόμενο με λέξεις-κλειδιά που αντικατοπτρίζουν τη δική τους άποψη. Αυτοί οι νέοι τύποι μορφών ανατροφοδότησης επιτρέπουν στα συστήματα συστάσεων να συμπεραίνουν σιωπηρά τις προτιμήσεις των χρηστών και τη δημοτικότητα του περιεχομένου αναλύοντας τα σχόλια του πλήθους.[9] Το LinkedIn είναι ένα από τα πιο επιτυχημένα SNS και, ως το μεγαλύτερο επαγγελματικό δίκτυο στον κόσμο, έχει πολλές μοναδικές προκλήσεις σχετικά με τις συστάσεις, όπως των εταιρειών και των επαγγελματικών ομάδων. Ένα άλλο ιδιαίτερα ενδιαφέρον παράδειγμα είναι η σύσταση ευκαιριών απασχόλησης. Μια τέτοια σύσταση μπορεί να έχει τεράστια επιρροή στη ζωή των ανθρώπων, καθώς μπορεί τελικά να οδηγήσει σε αλλαγή καριέρας. Η σύσταση πρέπει να λαμβάνει υπόψη πολλές πτυχές, όπως εναλλακτικές τοποθεσίες, την εμπειρία του υποψηφίου και το χρονοδιάγραμμα.

Στο LinkedIn για παράδειγμα εστιάζουν ιδιαίτερα στο χρονοδιάγραμμα της σύστασης. Το στατιστικό τους μοντέλο εξετάζει τη χρονική διάρκεια μεταξύ δύο διαδοχικών αποφάσεων για να εκτιμήσει την πιθανότητα της απόφασης ενός χρήστη να προβεί σε αλλαγή θέσης εργασίας σε ένα δεδομένο σημείο. [10]

Ενημέρωση

Η ενημέρωση είναι επίσης ένας τομέας στον οποίο οι εταιρείες έχουν εφαρμόσει προσεγγίσεις συστάσεων για την εξατομίκευση και την εστίαση στα ενδιαφέροντα ενός χρήστη. Για παράδειγμα, το Google News τροφοδοτήθηκε από την αρχή με κάποιου είδους συστάσεις για ειδησεογραφικά άρθρα. Η Yahoo! έχει επίσης επενδύσει στην εξατομίκευση ειδήσεων. Για τη σύσταση ειδήσεων, μερικές από τις βασικές προκλήσεις είναι η φρεσκάδα, όπου τα σχετικά άρθρα μπορεί να έχουν πολύ περιορισμένη χρονική διάρκεια, ποικιλομορφία και μπορεί να υπάρχει μεγάλος αριθμός άρθρων για το ίδιο θέμα. Ωστόσο, οι ειδήσεις έχουν το πλεονέκτημα του κειμενικού περιεχομένου, το οποίο επιτρέπει την εφαρμογή τεχνικών από την επεξεργασία φυσικής γλώσσας για τη δημιουργία χαρακτηριστικών που μπορούν να χρησιμοποιηθούν στη σύσταση, γεγονός που είναι ιδιαίτερα χρήσιμο όταν τα δεδομένα συμπεριφοράς των χρηστών είναι αραιά.[11]



Γιατί ένα συστήματος συστάσεων;

Ο σκοπός ενός συστήματος συστάσεων μπορει απλά κάποιος να πει ότι είναι δώσει μια απάντηση στην ερώτηση που θα του κάνει ο χρήσης. Όμως σιγουρά δεν είναι ο λόγος που μια εταιρία θέλει να αναπτύξει ένα τέτοιο σύστημα για τον εαυτό της, εξάλλου αυτό που πρέπει να πραγματοποίηση ένα σύστημα συστάσεων είναι να φέρει κέρδη και αυτό μπορει να το καταφέρει με πόλους τρόπους

- Αύξηση του αριθμού των πωλούμενων ειδών. Αυτή είναι ίσως η πιο σημαντική λειτουργία για ένα εμπορικό σύστημα συστάσεων, δηλαδή να μπορεί να πουλήσει ένα πρόσθετο σύνολο ειδών σε σύγκριση με αυτά που συνήθως πωλούνται χωρίς κανενός είδους σύσταση. Ο στόχος αυτός επιτυγχάνεται επειδή τα συνιστώμενα είδη είναι πιθανό να ταιριάζουν στις ανάγκες και τις επιθυμίες του χρήστη. Πιθανότατα ο χρήστης θα το αναγνωρίσει αυτό αφού δοκιμάσει αρκετές συστάσεις. Οι μη εμπορικές εφαρμογές έχουν παρόμοιους στόχους, ακόμη και αν δεν υπάρχει κόστος για τον χρήστη που να συνδέεται με την επιλογή ενός αντικειμένου. Για παράδειγμα, ένα δίκτυο περιεχομένου στοχεύει στην αύξηση του αριθμού των ειδήσεων που διαβάζονται στον ιστότοπο του. Σε γενικές γραμμές, μπορούμε να πούμε ότι από την πλευρά του παρόχου υπηρεσιών, ο πρωταρχικός στόχος για την εισαγωγή ενός συστήματος συστάσεων είναι η αύξηση του ποσοστού μετατροπής, δηλαδή του αριθμού των χρηστών που αποδέχονται τη σύσταση και καταναλώνουν ένα στοιχείο, σε σύγκριση με τον αριθμό των απλών επισκεπτών που απλώς περιηγούνται στις πληροφορίες.[12]

- Πώληση ποικίλων ειδών. Μια άλλη σημαντική λειτουργία ενός συστήματος συστάσεων είναι να επιτρέπει στο χρήστη να επιλέγει αντικείμενα που μπορεί να είναι δύσκολο να βρει χωρίς ακριβή σύσταση. Για παράδειγμα, σε ένα τουριστικό συστήματος συστάσεων ο πάροχος υπηρεσιών ενδιαφέρεται να προωθήσει όλα τα σημεία ενδιαφέροντος σε μια τουριστική περιοχή και όχι μόνο τα πιο δημοφιλή. Αυτό θα μπορούσε να είναι δύσκολο χωρίς ένα συστήματος συστάσεων, δεδομένου ότι ο πάροχος υπηρεσιών δεν μπορεί να αναλάβει τον κίνδυνο να διαφημίσει μέρη που δεν είναι πιθανό να ταιριάζουν στο γούστο ενός συγκεκριμένου χρήστη. Ως εκ τούτου, ένα συστήματος συστάσεων προτείνει ή διαφημίζει μη δημοφιλή μέρη στους κατάλληλους χρήστες[12], [13]

- Αύξηση της ικανοποίησης των χρηστών. Ένα καλά σχεδιασμένο σύστημα συστάσεων μπορεί επίσης να βελτιώσει την εμπειρία του χρήστη με τον ιστότοπο ή την εφαρμογή. Ο χρήστης θα βρίσκει τις συστάσεις ενδιαφέρουσες, σχετικές και, με μια σωστά σχεδιασμένη αλληλεπίδραση ανθρώπου-υπολογιστή, θα απολαμβάνει επίσης τη χρήση του συστήματος. Ο συνδυασμός αποτελεσματικών, ακριβών συστάσεων και εύχρηστης διεπαφής θα αυξήσει την υποκειμενική αξιολόγηση του συστήματος από τον χρήστη. Αυτό, με τη σειρά του, θα αυξήσει τη χρήση του συστήματος και την πιθανότητα αποδοχής των συστάσεων.[12]

- Αύξηση της πιστότητας του χρήστη. Ο χρήστης θα πρέπει να είναι πιστός σε έναν ιστότοπο τον οποίο, όταν επισκέπτεται, αναγνωρίζει τον παλιό πελάτη και τον αντιμετωπίζει ως πολύτιμο επισκέπτη. Αυτό είναι ένα πρότυπο χαρακτηριστικό ενός συστήματος συστάσεων, δεδομένου ότι πολλά συστήματα συστάσεων υπολογίζουν συστάσεις αξιοποιώντας έτσι τις πληροφορίες που αποκτήθηκαν από τον χρήστη κατά τη διάρκεια προηγούμενων αλληλεπιδράσεων, όπως οι αξιολογήσεις των αντικειμένων από τον χρήστη. Κατά συνέπεια, όσο περισσότερο καιρό ο χρήστης αλληλεπιδρά με το ιστότοπο, τόσο πιο εκλεπτυσμένο γίνεται το μοντέλο του χρήστη. Αναπτύσσεται η αναπαράσταση των προτιμήσεων του χρήστη από το σύστημα και αυξάνεται η αποτελεσματικότητα του αποτελέσματος της σύστασης για την προσαρμογή και την αντιστοίχιση με τις προτιμήσεις του χρήστη.[12]

- Καλύτερη κατανόηση του τι θέλει ο χρήστης. Μια άλλη σημαντική λειτουργία του ενός συστηματως συστασεων που μπορεί να αξιοποιηθεί σε πολλές άλλες εφαρμογές είναι η περιγραφή των προτιμήσεων του χρήστη, οι οποίες συλλέγονται είτε ρητά είτε προβλέπονται από το σύστημα. Ο πάροχος υπηρεσιών μπορεί στη συνέχεια να αποφασίσει να επαναχρησιμοποιήσει αυτή τη γνώση για διάφορους άλλους στόχους, όπως η βελτίωση της διαχείρισης του στοιχείου αποθέματος ή της παραγωγής. Για παράδειγμα, στον τομέα των ταξιδιών, οι οργανισμοί διαχείρισης προορισμών μπορούν να αποφασίσουν να διαφημίσουν μια συγκεκριμένη περιοχή σε νέους τομείς πελατών ή να διαφημίσουν έναν συγκεκριμένο τύπο διαφημιστικού μηνύματος που προέρχεται από την ανάλυση των δεδομένων που συλλέγονται από τα συστήματα συστασεων[12], [14]

Πως όμως ένα σύστημα συστάσεων μπορει να βοηθήσει των χρήστη και ποιες είναι συνήθως η εργασία που πραγματοποίει ένα σύστημα συστάσεων. Έχει γίνει αναφορά σε ένα έγγραφο[15] για έντεκα εργασίες που μπορει ένα σύστημα συστάσεων να βοηθήσει.

- Εύρεση μερικών καλών αντικείμενων: Προτείνονται στον χρήστη ορισμένα αντικείμενα ως μια καταταγμένη λίστα μαζί με προβλέψεις για το πόσο θα αρέσουν στον χρήστη (π.χ., σε μια κλίμακα από ένα έως πέντε αστέρια). Αυτό είναι το κύριο έργο σύστασης που αντιμετωπίζουν πολλά εμπορικά συστήματα.[15]

- Εύρεση όλων των καλών αντικειμένων: Προτείνονται όλα τα στοιχεία που μπορούν να ικανοποιήσουν κάποιες ανάγκες του χρήστη. Σε τέτοιες περιπτώσεις δεν αρκεί να βρεθούν μόνο μερικά καλά στοιχεία. Αυτό ισχύει ιδιαίτερα όταν ο αριθμός των αντικειμένων είναι σχετικά μικρός ή όταν το σύστημα συστάσεων έχει να κάνει με κρίσιμα δεδομένα, όπως σε ιατρικές ή οικονομικές εφαρμογές. Σε αυτές τις περιπτώσεις, εκτός από το όφελος που προκύπτει από την προσεκτική εξέταση όλων των δυνατοτήτων, ο χρήστης μπορεί επίσης να επωφεληθεί από την κατάταξη των στοιχείων αυτών με πρόσθετες εξηγήσεις που παράγει το σύστημα συστάσεών.[15]

- Σχολιασμός σε ένα πλαίσιο: Δεδομένου ενός υπάρχοντος πλαισίου π.χ., σε έναν κατάλογο αντικειμένων, τονίζονται ορισμένα από αυτά ανάλογα με τις μακροπρόθεσμες προτιμήσεις του χρήστη. Για παράδειγμα, ένα σύστημα τηλεοπτικών συστάσεων μπορεί να σχολιάσει ποιες τηλεοπτικές εκπομπές από αυτές που εμφανίζονται στον προγράμματα της τηλεόρασης είναι αξίζει να παρακολουθήσει ο χρήστης. [15]

- Σύσταση μιας ακολουθίας: Αντί να γίνει μια μεμονωμένης σύσταση, η ιδέα είναι να προταθεί μια ακολουθία στοιχείων που είναι ευχάριστη στο σύνολό της. Χαρακτηριστικά παραδείγματα περιλαμβάνουν τη σύσταση μιας τηλεοπτικής σειράς, ενός βιβλίου για τα συστήματα συστάσεων μετά από τη σύσταση ενός βιβλίου για την εξόρυξη δεδομένων, ή μιας συλλογής μουσικών κομματιών[15], [16]

- Σύσταση ενός πακέτου: Προτείνετε μια ομάδα αντικειμένων που ταιριάζουν καλά μεταξύ τους. Για παράδειγμα, ένα τουριστικό πακετο μπορεί να αποτελείται από διάφορα αξιοθέατα, προορισμούς και υπηρεσίες διαμονής που βρίσκονται σε μια οριοθετημένη περιοχή.[15], [17]

- Απλή περιήγησή: Σε αυτή την εργασία, ο χρήστης περιηγείται στον κατάλογο χωρίς καμία επικείμενη πρόθεση να αγοράσει κάποιο αντικείμενο. Το έργο του συστήματος εδώ είναι να βοηθήσει τον χρήστη να περιηγηθεί στα αντικείμενα που είναι πιο πιθανό να εμπίπτουν στο πεδίο του ενδιαφέροντος του χρήστη για τη συγκεκριμένη περίοδο περιήγησης. [15], [18]

- Εύρεση αξιόπιστου συστήματος συστάσεων: Κάποιοι χρήστες δεν εμπιστεύονται τα συστήματα συστάσεων, επομένως τα δοκιμάζουν για να δουν πόσο καλά είναι στο να κάνουν συστάσεις. Ως εκ τούτου, ένα σύστημα μπορεί επίσης να προσφέρει συγκεκριμένες λειτουργίες για να μπορούν οι χρήστες να δοκιμάσουν τη συμπεριφορά του, εκτός από εκείνες που απαιτούνται μόνο για την απόκτηση συστάσεων.[15]

- Βελτιώσει προφίλ: Αυτό αφορά την ικανότητα του χρήστη να παρέχει πληροφορίες στο σύστημα συστάσεων σχετικά με το τι του αρέσει και τι δεν του αρέσει. Πρόκειται για ένα θεμελιώδες έργο που είναι απολύτως απαραίτητο για την παροχή εξατομικευμένων συστάσεων. Εάν το σύστημα δεν έχει συγκεκριμένες γνώσεις σχετικά με τον ενεργό χρήστη, τότε μπορεί μόνο να παρέχει τις ίδιες συστάσεις που θα παρέχονταν σε έναν "μέσο" χρήστη.[15]

- Εκφράσει του εγώ: Ορισμένοι χρήστες μπορεί να μην ενδιαφέρονται καθόλου για τις συστάσεις. Αντίθετα, αυτό που τους ενδιαφέρει είναι να μπορούν να συνεισφέρουν με τις αξιολογήσεις τους και να εκφράζουν τις απόψεις και τα πιστεύω τους. Η ικανοποίηση του χρήστη από αυτή τη δραστηριότητα μπορεί να λειτουργήσει θετικά, με αποτέλεσμα τη συνεχή αφοσίωση του χρήστη στην εφαρμογή.[15]

- Παροχή βοήθειας: Κάποιοι χρήστες μπορεί να μην ενδιαφέρονται καθόλου για τις συστάσεις. Αλλά, αυτό που είναι σημαντικό γι' αυτούς είναι να μπορούν να συνεισφέρουν με τις αξιολογήσεις τους και να εκφράσουν τις απόψεις και τις

πεποιθήσεις τους. Αυτό θα μπορούσε να είναι ένα σημαντικό κίνητρο για την εισαγωγή πληροφοριών σε ένα σύστημα συστάσεων που δεν χρησιμοποιείται συνήθως. Για παράδειγμα, σε ένα συστήματος συστάσεων για νέο αυτοκίνητο, ένας χρήστης που έχει ήδη αγοράσει ένα νέο αυτοκίνητο γνωρίζει ότι η βαθμολογία που έχει καταχωρηθεί στο σύστημα θα να είναι χρήσιμη για άλλους χρήστες παρά για αυτόν.[15]

- Επιρροή στους άλλους: Στα διαδικτυακά συστάσεων, υπάρχουν χρήστες των οποίων ο κύριος στόχος είναι να επηρεάσουν άλλους χρήστες ώστε να αγοράσουν συγκεκριμένα προϊόντα. Στην πραγματικότητα, υπάρχουν επίσης κακόβουλοι χρήστες που μπορεί να χρησιμοποιούν το σύστημα απλώς για να προωθήσουν ή να τιμωρήσουν ορισμένα αντικείμενα[15]

Όπως δείχνουν αυτά τα διάφορα σημεία, ο ρόλος ενός συστήματος συστάσεων μπορεί να είναι αρκετά διαφορετικός. Αυτή η ποικιλομορφία απαιτεί την αξιοποίηση μιας σειράς διαφορετικών μεθόδων.

Αξιολόγηση συστημάτων συστάσεων

Ένας σχεδιαστής εφαρμογών που επιθυμεί να προσθέσει ένα σύστημα συστάσεων στην εφαρμογή της έχει μια μεγάλη ποικιλία αλγορίθμων στη διάθεσή της και πρέπει να λάβει μια απόφαση σχετικά με τον καταλληλότερο αλγόριθμο για τους στόχους της. Συνήθως, οι αποφάσεις αυτές βασίζονται σε πειράματα, συγκρίνοντας τις επιδόσεις ενός αριθμού υποψήφιων συστημάτων. Ο σχεδιαστής μπορεί στη συνέχεια να επιλέξει τον αλγόριθμο με τις καλύτερες επιδόσεις, δεδομένων των δομικών περιορισμών, όπως ο τύπος, η επικαιρότητα και η αξιοπιστία των δεδομένων διαθεσιμότητας, η επιτρεπόμενη μνήμη και το αποτύπωμα της CPU. Επιπλέον, οι περισσότεροι ερευνητές που προτείνουν νέους αλγορίθμους σύστασης συγκρίνουν επίσης την απόδοση του νέου τους αλγορίθμου με ένα σύνολο υφιστάμενων προσεγγίσεων. Τέτοιες αξιολογήσεις συνήθως πραγματοποιούνται με την εφαρμογή κάποιας μετρικής αξιολόγησης που παρέχει μια κατάταξη των υποψήφιων αλγορίθμων (συνήθως με αριθμητικές βαθμολογίες)[19]

Υπάρχουν τρεις βασικοί τύποι αξιολόγησης των συστημάτων συστάσεων, που αντιστοιχούν σε μελέτες χρηστών, σε διαδικτυακές αξιολογήσεις και σε αξιολογήσεις εκτός σύνδεσης με ιστορικά σύνολα δεδομένων. Οι δύο πρώτοι τύποι περιλαμβάνουν χρήστες, αν και διεξάγονται με ελαφρώς διαφορετικούς τρόπους. Οι κύριες διαφορές μεταξύ των δύο πρώτων ρυθμίσεων έγκεινται στον τρόπο με τον οποίο προσλαμβάνονται οι χρήστες για τις μελέτες. Αν και οι online αξιολογήσεις παρέχουν χρήσιμες πληροφορίες σχετικά με τα πραγματικά αποτελέσματα ενός αλγορίθμου συστάσεων, συχνά υπάρχουν σημαντικά πρακτικά εμπόδια στην ανάπτυξή τους. Στη συνέχεια, παρέχεται μια επισκόπηση αυτών των διαφορετικών τύπων αξιολόγησης. [20]

Δείκτες του σχεδιασμού αξιολόγησης

Τα συστήματα συστάσεων μπορούν να αξιολογηθούν μέσω διαφόρων μετρήσεων και πειραμάτων εκτός σύνδεσης. Οι δείκτες για τα συστήματα συστάσεων μπορούν να ομαδοποιηθούν σε διάφορες ομάδες. Κάθε ομάδα έχει έναν συγκεκριμένο σκοπό, τον οποίο θα συζητήσουμε σε αυτή την ανάρτηση. Αυτοί οι δείκτες χρησιμοποιούνται για διαφορετικές περιπτώσεις και δεν ισχύει ότι ένας από αυτούς είναι καλύτερος από τους άλλους.

- **Ακρίβεια**: Ο πιο απλός δείκτης είναι το μέσο απόλυτο σφάλμα. Παρακάτω ακολουθεί η μαθηματική εξίσωση για τον υπολογισμό της.

$$\frac{\sum_{i=1}^{n}|y_i - x_i|}{n} \ [21]$$

Ας υποθέσουμε ότι έχουμε n αξιολογήσεις στο σύνολο δοκιμών που θέλουμε να αξιολογήσουμε, για κάθε αξιολόγηση μπορούμε να ονομάσουμε την αξιολόγηση ή το σύστημα προβλέπει y, και την αξιολόγηση που ο χρήστης έδωσε στην πραγματικότητα x. Παίρνουμε απλώς την απόλυτη τιμή της διαφοράς μεταξύ των δύο, για να μετρήσουμε το σφάλμα για την πρόβλεψη της αξιολόγησης. Είναι κυριολεκτικά απλώς η διαφορά μεταξύ της προβλεπόμενης βαθμολογίας και της πραγματικής βαθμολογίας. Αθροίζουμε αυτά τα σφάλματα για όλες τις n αξιολογήσεις στο σύνολο δοκιμών μας και διαιρούμε με το "n" για να πάρουμε το μέσο όρο ή τη μέση τιμή. Έτσι, το μέσο απόλυτο σφάλμα είναι ακριβώς αυτό, ο μέσος όρος ή η μέση απόλυτη τιμή κάθε σφάλματος στις προβλέψεις

αξιολόγησης. Θέλουμε τη χαμηλότερη βαθμολογία MAE, όχι την υψηλότερη.[21]

- **Ποσοστό επιτυχίας (Hit-Rate):** Το ίδιο το ποσοστό επιτυχίας είναι εύκολο να κατανοηθεί, αλλά η μέτρησή του είναι λίγο δύσκολη.

$$\frac{hit}{users}\,[21]$$

Δεν μπορούμε να χρησιμοποιήσουμε την ίδια προσέγγιση εκπαίδευσης/δοκιμής ή διασταυρούμενης επικύρωσης που χρησιμοποιήσαμε για την ακρίβεια, επειδή δεν μετράμε την ακρίβεια σε μεμονωμένες αξιολογήσεις. Μετράμε την ακρίβεια των κορυφαίων λιστών για μεμονωμένους χρήστες. Τώρα θα μπορούσαμε να κάνουμε το προφανές πράγμα και να μην χωρίσουμε καθόλου τα πράγματα και να μετρήσουμε απλώς το ποσοστό επιτυχίας απευθείας στις κορυφαίες συστάσεις που δημιουργήθηκαν από ένα σύστημα συστάσεων που εκπαιδεύτηκε σε όλα τα δεδομένα που διαθέτουμε. Αλλά τεχνικά αυτό είναι εξαπάτηση. Γενικά δεν θέλουμε να αξιολογούμε ένα σύστημα χρησιμοποιώντας δεδομένα με τα οποία εκπαιδεύτηκε. Θα μπορούσαμε απλώς να προτείνουμε τις πραγματικές 10 κορυφαίες ταινίες που αξιολογήθηκαν από κάθε χρήστη χρησιμοποιώντας τα δεδομένα εκπαίδευσης και να επιτύχουμε ποσοστό επιτυχίας 100%. Έτσι, ένας έξυπνος τρόπος για να το παρακάμψουμε αυτό ονομάζεται "Leave-one-out cross validation". Αυτό που κάνουμε είναι να υπολογίζουμε τις κορυφαίες συστάσεις για κάθε χρήστη στα δεδομένα εκπαίδευσής μας και να αφαιρούμε σκόπιμα ένα από αυτά τα στοιχεία από τα δεδομένα εκπαίδευσης αυτού του χρήστη. Στη συνέχεια, δοκιμάζουμε την ικανότητα του συστήματός μας να συστήσει το στοιχείο που έμεινε εκτός των κορυφαίων αποτελεσμάτων που δημιουργεί για τον συγκεκριμένο χρήστη στη φάση δοκιμής. Έτσι, μετράμε την ικανότητά μας να προτείνουμε ένα στοιχείο σε μια κορυφαία λίστα για κάθε χρήστη που είχε παραλειφθεί από τα δεδομένα εκπαίδευσης. Αυτός είναι ο λόγος για τον οποίο ονομάζεται "leave-one-out". Το πρόβλημα είναι ότι είναι πολύ πιο δύσκολο να πετύχουμε μια συγκεκριμένη ταινία σωστά κατά τη διάρκεια της δοκιμής από το να πετύχουμε απλώς μια από τις συστάσεις. Έτσι, το "ποσοστό επιτυχίας" με το "leave-one-out" τείνει να είναι πολύ μικρό και δύσκολο να μετρηθεί, εκτός αν έχετε ένα πολύ μεγάλο σύνολο δεδομένων για να εργαστείτε. Αλλά είναι μια πολύ πιο εστιασμένη στον χρήστη μέτρηση όταν γνωρίζουμε ότι το σύστημα

συστάσεών μας θα παράγει κορυφαίες λίστες στον πραγματικό κόσμο, πράγμα που κάνουν τα περισσότερα από αυτά.[20]

- **Μέσο ποσοστό αμοιβαίων επιτυχιών (ARHR):** Μια παραλλαγή του ποσοστού επιτυχίας είναι το μέσο αμοιβαίο ποσοστό επιτυχίας, ή ARHR. Αυτή η μέτρηση είναι ακριβώς όπως το ποσοστό επιτυχίας, αλλά λαμβάνει υπόψη της το σημείο στη λίστα top-N εμφανίζονται αντικείμενα. Έτσι, ένα στοιχεία έχει περισσότερο βαρύτητα για την επιτυχή σύσταση ενός στοιχείου στην πρώτη θέση από ό,τι στην τελευταία θέση. Και πάλι, αυτός είναι ένας δείκτης που εστιάζει περισσότερο στον χρήστη, καθώς οι χρήστες τείνουν να εστιάζουν στην αρχή των λιστών. Η μόνη διαφορά είναι ότι αντί να αθροίζουμε τον αριθμό των επιτυχιών, εμείς αθροίζουμε την αμοιβαία κατάταξη κάθε επιτυχίας. Έτσι, αν προβλέψουμε επιτυχώς κατι στη θέση 3, αυτό μετράει μόνο ως το 1/3. Αλλά μια επιτυχία στη θέση 1 λαμβάνει την πλήρη βαρύτητα 1,0. Εάν ο χρήστης πρέπει να κάνει κύλιση ή να κάνει σελιδοποίηση για να δει τα χαμηλότερα στοιχεία της λίστας top-N, τότε έχει νόημα να αγνοούνται οι καλές συστάσεις που εμφανίζονται πολύ χαμηλά στη λίστα, όπου ο χρήστης πρέπει να δουλέψει για να τις βρει.[22]

$$\frac{\sum_{i=1}^{n} \frac{1}{rank_j}}{Users}$$

- **Κάλυψη**: Ακόμα και όταν ένα σύστημα συστάσεων είναι εξαιρετικά ακριβές, μπορεί συχνά να μην είναι σε θέση να συστήσει ένα ορισμένο ποσοστό των αντικειμένων, ή να μην είναι σε θέση να συστήσει ποτέ σε ένα ορισμένο ποσοστό των χρηστών. Αυτός ο δείκτης αναφέρεται ως κάλυψη. Αυτός ο περιορισμός των συστημάτων συστάσεων είναι αποτέλεσμα του γεγονότος ότι οι πίνακες αξιολογήσεων είναι αραιοί. Για παράδειγμα, σε έναν πίνακα αξιολόγησης που περιέχει μία μόνο εγγραφή για κάθε γραμμή και κάθε στήλη, τότε δεν μπορούν να γίνουν ουσιαστικές συστάσεις από σχεδόν κανέναν αλγόριθμο. Παρ' όλα αυτά, τα διάφορα συστήματα συστάσεων έχουν διαφορετικά επίπεδα ροπής στην παροχή κάλυψης. Σε πρακτικές ρυθμίσεις, τα συστήματα έχουν συχνά 100% κάλυψη λόγω της χρήσης προεπιλογών για αξιολογήσεις που δεν είναι δυνατόν να προβλεφθούν. Ένα παράδειγμα μιας τέτοιας προεπιλογής θα ήταν η αναφορά του μέσου όρου όλων των αξιολογήσεων ενός χρήστη για ένα στοιχείο, όταν η αξιολόγηση για έναν συγκεκριμένο συνδυασμό χρήστη-στοιχείου δεν μπορεί να

προβλεφθεί. Επομένως, η αντιστάθμιση μεταξύ ακρίβειας και κάλυψης πρέπει πάντα να ενσωματώνεται στη διαδικασία αξιολόγησης. Υπάρχουν δύο τύποι κάλυψης, οι οποίοι είναι αναφέρονται ως κάλυψη χώρου χρήστη και κάλυψη χώρου στοιχείου, αντίστοιχα.[19]

Η κάλυψη user-space υπολογίζει το κλάσμα των χρηστών για τους οποίους μπορούν να γίνουν τουλάχιστον k αξιολογήσεις. Η τιμή του k θα πρέπει να οριστεί στο αναμενόμενο μέγεθος του καταλόγου συστάσεων. Όταν μπορούν να προβλεφθούν λιγότερες από k αξιολογήσεις για έναν χρήστη, δεν είναι πλέον δυνατή η παρουσίαση μια ουσιαστική λίστα συστάσεων μεγέθους k στον χρήστη. Μια τέτοια κατάσταση μπορεί να συμβεί όταν ένας χρήστης έχει καθορίσει πολύ λίγες κοινές αξιολογήσεις με άλλους χρήστες. Ένας εναλλακτικός ορισμός της κάλυψης user-space είναι η ελάχιστη ποσότητα του προφίλ που πρέπει να δημιουργηθεί για έναν χρήστη πριν είναι δυνατόν να γίνουν συστάσεις για τον εν λόγω χρήστη. Για έναν συγκεκριμένο αλγόριθμο, είναι δυνατόν να εκτιμηθεί μέσω πειραμάτων ο ελάχιστος αριθμός παρατηρούμενων αξιολογήσεων οποιουδήποτε χρήστη για τον οποίο θα μπορούσε να γίνει σύσταση. Ωστόσο, είναι συχνά δύσκολο να εκτιμηθεί αυτή η ποσότητα, επειδή η μετρική είναι ευαίσθητη στην ταυτότητα των στοιχείων για τα οποία ο χρήστης καθορίζει αξιολογήσεις.[20]

Η έννοια της κάλυψης item-space είναι ανάλογη με εκείνη της κάλυψης user-space. Η κάλυψη item-space υπολογίζει το κλάσμα των αντικειμένων για τα οποία μπορούν να προβλεφθούν οι αξιολογήσεις τουλάχιστον k χρηστών. Στην πράξη, ωστόσο, αυτή η έννοια χρησιμοποιείται σπάνια, επειδή τα συστήματα συστάσεων παρέχουν γενικά λίστες συστάσεων για τους χρήστες και χρησιμοποιούνται μόνο σπάνια για τη δημιουργία συνιστομένων χρηστών για αντικείμενα.[20]

- **Εμπιστοσύνη και αξιοπιστία**: Η εκτίμηση των βαθμολογιών είναι μια ανακριβής διαδικασία που μπορεί να διαφέρει σημαντικά ανάλογα με τα συγκεκριμένα δεδομένα εκπαίδευσης. Επιπλέον, η αλγοριθμική μεθοδολογία μπορεί επίσης να έχει σημαντικό αντίκτυπο στις προβλεπόμενες βαθμολογίες. Αυτό οδηγεί πάντα σε αβεβαιότητα στον χρήστη σχετικά με την ακρίβεια των προβλέψεων. Πολλά συστήματα συστάσεων μπορούν να αναφέρουν αξιολογήσεις μαζί με εκτιμήσεις εμπιστοσύνης. Για παράδειγμα, μπορεί να

παρέχεται ένα διάστημα αξιοπιστίας για το εύρος των προβλεπόμενων αξιολογήσεων. Γενικά, τα συστήματα συστάσεων που μπορούν να προτείνουν με ακρίβεια μικρότερα διαστήματα εμπιστοσύνης είναι πιο επιθυμητά, επειδή ενισχύουν την εμπιστοσύνη του χρήστη στο σύστημα. Για δύο αλγορίθμους που χρησιμοποιούν την ίδια μέθοδο για την αναφορά της εμπιστοσύνης, είναι δυνατό να μετρηθεί πόσο καλά το προβλεπόμενο σφάλμα ταιριάζει με αυτά τα διαστήματα αξιοπιστίας. Για παράδειγμα, εάν δύο συστήματα συστάσεων παρέχουν 95% διαστήματα αξιοπιστίας για κάθε αξιολόγηση, μπορεί κανείς να μετρήσει το απόλυτο πλάτος των διαστημάτων που αναφέρουν οι δύο αλγόριθμοι. Ο αλγόριθμος με το μικρότερο πλάτος αξιοπιστίας θα κερδίσει εφόσον και οι δύο αλγόριθμοι είναι σωστοί (δηλαδή εντός των καθορισμένων διαστημάτων) τουλάχιστον στο 95% του χρόνου στις κρυφές αξιολογήσεις. Εάν ένας από τους αλγορίθμους πέσει κάτω από το απαιτούμενο ποσοστό ακρίβειας 95%, τότε χάνει αυτόματα. Δυστυχώς, εάν ένα σύστημα χρησιμοποιεί 95% διαστήματα αξιοπιστίας και ένα άλλο 99% διαστήματα αξιοπιστίας, δεν είναι δυνατή η ουσιαστική σύγκρισή τους. Επομένως, είναι δυνατή η χρήση τέτοιων συστημάτων μόνο με τον καθορισμό του ίδιου επιπέδου αξιοπιστίας και στις δύο περιπτώσεις.[20]

- **Ποικιλομορφία**: Ένα ακόμη δείκτης είναι η ποικιλομορφία. Μπορείτε να το θεωρήσετε ως ένα δείκτη που αναφέρει το πόσο διαφέρουν οι συστάσεις μεταξύ τους. Ένα παράδειγμα χαμηλής ποικιλομορφίας θα ήταν ένα σύστημα συστάσεων που συνιστά μόνο τα επόμενα βιβλία μιας σειράς που έχετε αρχίσει να διαβάζετε, αλλά δεν προτείνει βιβλία από διαφορετικούς συγγραφείς ή ταινίες που σχετίζονται με αυτό που που έχετε διαβάσει. Αυτό μπορεί να φαίνεται υποκειμενικό, αλλά είναι μετρήσιμο. Πολλά συστήματα συστάσεων ξεκινούν με τον υπολογισμό κάποιου είδους δείκτη ομοιότητας μεταξύ των στοιχείων, οπότε μπορούμε να χρησιμοποιήσουμε αυτές τις βαθμολογίες ομοιότητας για να μετρήσουμε την ποικιλομορφία. Αν κοιτάξουμε τις βαθμολογίες ομοιότητας κάθε πιθανού ζεύγους σε μια λίστα top-N συστάσεων, μπορούμε να τις υπολογίσουμε κατά μέσο όρο για να λάβουμε ένα μέτρο του πόσο παρόμοια είναι τα συνιστώμενα στοιχεία στη λίστα μεταξύ τους. Μπορούμε να ονομάσουμε αυτό το μέτρο S. Η ποικιλομορφία είναι ουσιαστικά το αντίθετο της μέσης ομοιότητας, οπότε την αφαιρούμε από το 1 για να πάρουμε έναν αριθμό που

σχετίζεται με την ποικιλομορφία. Είναι σημαντικό να συνειδητοποιήσουμε ότι η ποικιλομορφία, τουλάχιστον στο πλαίσιο των συστημάτων συστάσεων, δεν είναι πάντα κάτι καλό. Μπορείτε να επιτύχετε πολύ υψηλή ποικιλομορφία με απλά συνιστώντας εντελώς τυχαία στοιχεία - αλλά αυτά δεν είναι καλά συστάσεις από οποιαδήποτε άποψη της φαντασίας! Ασυνήθιστα υψηλή ποικιλομορφία σημαίνει ότι γίνονται κακές συστάσεις, τις περισσότερες φορές.[22]

- **Καινοτομία**: Η καινοτομία ενός συστήματος συστάσεων αξιολογεί την πιθανότητα ένα σύστημα συστάσεων να δώσει στον χρήστη συστάσεις που δεν γνωρίζει ή που δεν έχει ξαναδεί. Οι πρωτόγνωρες συστάσεις αυξάνουν συχνά τη δυνατότητα του χρήστη να ανακαλύψει σημαντικές πληροφορίες σχετικά με τις προτιμήσεις και τις αντιπάθειές του, τις οποίες δεν γνώριζε προηγουμένως. Αυτό είναι πιο σημαντικό από την ανακάλυψη στοιχείων που ήδη γνώριζαν αλλά δεν έχουν αξιολογήσει. Σε πολλούς τύπους συστημάτων συστάσεων, όπως οι μέθοδοι που βασίζονται στο περιεχόμενο, οι συστάσεις τείνουν να είναι κάπως προφανείς λόγω της τάσης του συστήματος να προτείνει αναμενόμενα στοιχεία. Ενώ ένας μικρός αριθμός τέτοιων συστάσεων μπορεί να βελτιώσει την εμπιστοσύνη του τελικού χρήστη στο υποκείμενο σύστημα, δεν είναι πάντα χρήσιμες όσον αφορά τη βελτίωση των ποσοστών μετατροπής. Ο πιο φυσικός τρόπος μέτρησης της καινοτομίας είναι μέσω διαδικτυακού πειραματισμού στον οποίο οι χρήστες ερωτώνται ρητά αν γνώριζαν προηγουμένως ένα στοιχείο[20]

- **Ανθεκτικότητα και σταθερότητα:** Ένα σύστημα συστάσεων είναι σταθερό και ανθεκτικό όταν οι συστάσεις δεν επηρεάζονται σημαντικά από επιθέσεις όπως οι ψεύτικες αξιολογήσεις ή όταν τα πρότυπα των δεδομένων εξελίσσονται σημαντικά με την πάροδο του χρόνου. Γενικά, υπάρχουν σημαντικά κίνητρα με γνώμονα το κέρδος για ορισμένους χρήστες να εισάγουν ψεύτικες αξιολογήσεις. Για παράδειγμα, ο συγγραφέας ή ο εκδότης ενός βιβλίου μπορεί να εισάγει ψεύτικες θετικές αξιολογήσεις για ένα βιβλίο στο Amazon.com ή μπορεί να εισάγει ψεύτικες αρνητικές αξιολογήσεις για τα βιβλία ενός ανταγωνιστή. Τα αντίστοιχα μέτρα μπορούν να χρησιμοποιηθούν για την εκτίμηση της ανθεκτικότητας και της σταθερότητας τέτοιων συστημάτων έναντι επιθέσεων.[20]

- **Κλιμάκωση**: Τα τελευταία χρόνια, η συλλογή μεγάλου αριθμού αξιολογήσεων και σιωπηρών πληροφοριών ανατροφοδότησης από διάφορους χρήστες γίνεται όλο και πιο εύκολη. Σε τέτοιες περιπτώσεις, τα μεγέθη των συνόλων δεδομένων συνεχίζουν να αυξάνονται με την πάροδο του χρόνου. Ως αποτέλεσμα, έχει καταστεί ολοένα και πιο σημαντικός ο σχεδιασμός συστημάτων συστάσεων που μπορούν να αποδίδουν αποτελεσματικά και αποδοτικά παρουσία μεγάλου όγκου δεδομένων . Για τον προσδιορισμό της κλιμάκωσης ενός συστήματος χρησιμοποιούνται διάφορα μέτρα:[20], [22]

    1. Χρόνος εκπαίδευσης: Τα περισσότερα συστήματα συστάσεων απαιτούν μια φάση εκπαίδευσης, η οποία είναι ξεχωριστή από τη φάση δοκιμής. Για παράδειγμα, ένα συνεργατικό φιλτράρισμα με βάση τη γειτονιά αλγόριθμος μπορεί να απαιτεί τον προ-υπολογισμό της ομάδας ομόιων χρηστών για τον χρήστη και έναν πίνακα σύστημα παραγοντοποίησης πινάκων απαιτεί τον προσδιορισμό των λανθανόντων παραγόντων. Ο συνολικός χρόνος που απαιτείται για την εκπαίδευση ενός μοντέλου χρησιμοποιείται ως ένα από τα μέτρα. Στις περισσότερες περιπτώσεις, η εκπαίδευση γίνεται εκτός σύνδεσης. Επομένως, εφόσον ο χρόνος εκπαίδευσης είναι της τάξης των λίγων ωρών, είναι αρκετά αποδεκτός στις περισσότερες πραγματικές ρυθμίσεις.[22]
    2. Χρόνος πρόβλεψης: Μόλις εκπαιδευτεί ένα μοντέλο, χρησιμοποιείται για τον προσδιορισμό της κορυφαίας συστάσεις για έναν συγκεκριμένο πελάτη. Είναι ζωτικής σημασίας ο χρόνος πρόβλεψης να είναι χαμηλός, διότι καθορίζει την καθυστέρηση με την οποία ο χρήστης λαμβάνει τις απαντήσεις[22]
    3. Απαιτήσεις μνήμης: Όταν οι πίνακες αξιολόγησης είναι μεγάλοι, είναι μερικές φορές μια πρόκληση να συγκρατηθεί ολόκληρος ο πίνακας στην κύρια μνήμη. Σε τέτοιες περιπτώσεις, είναι σημαντικό να σχεδιαστεί ο αλγόριθμος να ελαχιστοποιεί τις απαιτήσεις μνήμης. Όταν οι απαιτήσεις μνήμης γίνονται πολύ υψηλές, είναι δύσκολο να χρησιμοποιηθούν τα συστήματα σε μεγάλης κλίμακας και πρακτικές ρυθμίσεις.[22]

Κεφάλαιο 2
Μέθοδοί υλοποιήσεις για συστήματα συστάσεων

Τα βασικά μοντέλα για τα συστήματα συστάσεων εργάζονται με δύο είδη δεδομένων, τα οποία είναι:[23]

(i) Η αλληλεπιδράσεις user-item, όπως οι αξιολογήσεις ή η αγοραστική συμπεριφορά.

(ii) Οι πληροφορίες χαρακτηριστικών για τους χρήστες και τα αντικείμενα, όπως τα προφίλ κειμένου ή οι σχετικές λέξεις-κλειδιά.

Οι μέθοδοι που χρησιμοποιούν τα πρώτα αναφέρονται ως μέθοδοι συνεργατικού φιλτραρίσματος, ενώ οι μέθοδοι που χρησιμοποιούν τις δεύτερες αναφέρονται ως μέθοδοι συστάσεων βάσει περιεχομένου.

Τα συστήματα που βασίζονται στο περιεχόμενο χρησιμοποιούν τους πίνακες αξιολογήσεων στις περισσότερες περιπτώσεις, αν και το μοντέλο συνήθως επικεντρώνεται στις αξιολογήσεις ενός μεμονωμένου χρήστη και όχι σε εκείνες όλων των χρηστών. [23]

Το συνεργατικό φιλτράρισμα και το φιλτράρισμα βάσει περιεχομένου χρησιμοποιούνται ευρέως στα συστήματα συστάσεων. Και οι περισσότερες από τις διαδικτυακές επιχειρήσεις, όπως η Amazon, το Xbox, το Hulu και το Spotify, χρησιμοποιούν ένα από τα δύο ή και τα δύο.[24]

Πίνακας 1 Διάφορες σε συστήματα συνεργατικών συστάσεων και συστάσεων βάση περιεχομένου[24]

	Collaborative recommendation engine (σύστημα συνεργατικών συστάσεων)	**Content based recommendation system (σύστημα συστάσεων βάσει περιεχομένου)**
1	Ένα σύστημα συνεργατικών συστάσεων δίνει έμφαση στην προτίμηση του χρήστη	Ένα σύστημα συστάσεων βάσει περιεχομένου δίνει έμφαση στα χαρακτηριστικά του περιεχομένου
2	Στο συνεργατικό φιλτράρισμα, μια μηχανή συστάσεων απαιτεί το προφίλ του χρήστη για να προτείνει σχετικό περιεχόμενο.	Στο φιλτράρισμα βάσει περιεχομένου, ένα σύστημα συστάσεων χρησιμοποιεί επίσης το προφίλ περιεχομένου το οποίο περιλαμβάνει τα χαρακτηριστικά του περιεχομένου.
3	Τα συνεργατικά συστήματα συστάσεων τροφοδοτούνται από τις αξιολογήσεις,	Τα συστήματα συστάσεων που βασίζονται στο περιεχόμενο είναι προσανατολισμένα

	τις κριτικές και άλλα σχόλια των χρηστών σχετικά με τα προϊόντα ή υπηρεσίες. Έτσι, τα προϊόντα χωρίς αξιολογήσεις ή σχόλια δεν μπορούν να συστηθούν στον χρήστη. Ούτε ένας νέος χρήστης που δεν έχει δώσει σχόλια ή αξιολογήσεις μπορεί να λάβει κάποια σύσταση από τα συνεργατικά συστήματα συστάσεων συστάσεων. Αυτό ονομάζεται πρόβλημα ψυχρής εκκίνησης.	στα χαρακτηριστικά του προϊόντος και, ως εκ τούτου, δεν αντιμετωπίζουν τέτοια προβλήματα.
4	Μια συνεργατική μηχανή συστάσεων δεν εξασφαλίζει πάντα ακριβείς συστάσεις. Επειδή στους χρήστες με παρόμοιο γούστο μπορεί να μην αρέσουν πάντα τα ίδια προϊόντα	Μια μηχανή συστάσεων βασισμένη στο περιεχόμενο μπορεί να παρέχει πιο ακριβείς συστάσεις, καθώς εστιάζει στα χαρακτηριστικά του περιεχομένου που αρέσει στον χρήστη.

Ένας τρίτος τύπος συστήματος συστάσεων είναι αυτός που χρησιμοποιεί τη γνώση για τους χρήστες και τις προϊόντα για να ακολουθήσει μια προσέγγιση βασισμένη στη γνώση για τη δημιουργία μιας σύστασης, συλλογιζόμενο ποια προϊόντα ανταποκρίνονται στις απαιτήσεις του χρήστη. Το σύστημα συστάσεων PersonalLogic προσφέρει έναν διάλογο που ουσιαστικά καθοδηγεί τον χρήστη σε ένα δέντρο διάκρισης των χαρακτηριστικών των προϊόντων.1 Άλλοι έχουν προσαρμόσει ποσοτικά εργαλεία υποστήριξης αποφάσεων για το έργο αυτό. Το σύστημα σύστασης εστιατορίων Entree κάνει τις συστάσεις του βρίσκοντας εστιατόρια σε μια νέα πόλη παρόμοια με τα εστιατόρια που γνωρίζει ο χρήστης και του αρέσουν. Το σύστημα επιτρέπει στους χρήστες να προηγηθούν δηλώνοντας τις προτιμήσεις τους σε σχέση με ένα συγκεκριμένο εστιατόριο, βελτιώνοντας έτσι τα κριτήρια αναζήτησης.[25] Τα συστήματα συστάσεων με βάση την γνώση χωρίζονται σε δυο υποκατηγορίες:

1. Συστήματα βασισμένα σε περιπτώσεις: Τα CBR βασίζονται στη συλλογιστική με βάση περιπτώσεις, η οποία βασίζεται στην ομοιότητα μεταξύ μιας τρέχουσας περίπτωσης και των λύσεων που υπάρχουν ήδη σε μια βάση δεδομένων. Η αλληλεπίδραση με ένα σύστημα CBR αποτελείται από τέσσερις κύκλους βημάτων: Ανάκτηση, επαναχρησιμοποίηση, αναθεώρηση και διατήρηση.[26]

2. Συστήματα βασισμένα σε περιορισμούς - Με βάση ένα δεδομένο σύνολο προτιμήσεων, τα συστήματα βασισμένα σε περιορισμούς παρέχουν ένα σύνολο πιθανών λύσεων, συμπεριλαμβανομένων εξηγήσεων σχετικά με το γιατί επιλέχθηκαν αυτές οι λύσεις. Ένα συστήματα βασισμένο σε περιορισμούς ορίζεται σε σύνολο περιορισμών που μπορεί να είναι τριών τύπων: Περιορισμοί συμβατότητας, συνθήκες φίλτρου και περιορισμοί προϊόντος. [26]

Εικόνα 3 Ένα παράδειγμα μιας αρχικής διεπαφής χρήστη για μια σύσταση βασισμένη σε περιορισμούς [26]

Τα τρία προαναφερθέντα συστήματα εκμεταλλεύονται διαφορετικές πηγές εισόδου και μπορεί να λειτουργούν καλά σε διαφορετικά σενάρια.

- Τα συστήματα συνεργατικού φιλτραρίσματος βασίζονται στις αξιολογήσεις της κοινότητας,
- Οι μέθοδοι που βασίζονται στο περιεχόμενο βασίζονται σε περιγραφές κειμένου και στις αξιολογήσεις του ίδιου του χρήστη-στόχου και
- Τα συστήματα που βασίζονται στη γνώση βασίζονται σε αλληλεπιδράσεις με τον χρήστη στο πλαίσιο βάσεων γνώσης.

Αξίζει να σημειωθεί ότι αυτά τα διαφορετικά συστήματα χρησιμοποιούν διαφορετικούς τύπους εισόδου και έχουν διαφορετικά δυνατά και αδύνατα σημεία. Ορισμένα συστήματα συστάσεων, όπως τα συστήματα βασισμένα στη γνώση, είναι πιο αποτελεσματικά σε περιβάλλοντα ψυχρής εκκίνησης, όπου δεν είναι διαθέσιμος σημαντικός όγκος δεδομένων. Άλλα συστήματα συστάσεων, όπως οι συνεργατικές μέθοδοι, είναι πιο αποτελεσματικά όταν είναι διαθέσιμα πολλά δεδομένα.

Σε πολλές περιπτώσεις όπου είναι διαθέσιμη μια ευρύτερη ποικιλία δεδομένων, υπάρχει η ευελιξία της χρήσης διαφορετικών τύπων συστημάτων συστάσεων για την ίδια εργασία. Σε τέτοιες περιπτώσεις, υπάρχουν πολλές ευκαιρίες για υβριδισμό, όπου οι διάφορες πτυχές από διαφορετικούς τύπους συστημάτων συνδυάζονται για να επιτευχθεί το καλύτερο όλων των κόσμων. Τα υβριδικά συστήματα συστάσεων είναι στενά συνδεδεμένα στον τομέα της ανάλυσης συνόλων, όπου η ισχύς πολλαπλών τύπων μηχανικής μάθησης αλγορίθμων συνδυάζεται για τη δημιουργία ενός πιο ισχυρού μοντέλου. Τα συστήματα συστάσεων που βασίζονται σε σύνολα είναι σε θέση να συνδυάσουν όχι μόνο την ισχύ πολλαπλών πηγών δεδομένων, αλλά και να βελτιώσουν την αποτελεσματικότητα μιας συγκεκριμένης κατηγορίας συστημάτων συστάσεων (π.χ. συνεργατικά συστήματα) συνδυάζοντας πολλαπλά μοντέλα του ίδιου τύπου. [26]

Πίνακας 1 Οι στόχοι των διαφόρων συστημάτων συστάσεων

Προσέγγιση	Σκοπός	Είσοδος
Συνεργατικό φιλτράρισμα	Συστάσεις με βάση μια συνεργατική προσέγγιση που αξιοποιεί τις αξιολογήσεις και τις ενέργειες των ομότιμων του χρήστη και του εαυτού του.	Αξιολογήσεις χρηστών + αξιολογήσεις της κοινότητας
Βάση το περιεχόμενο	Συστάσεις με βάση το περιεχόμενο (χαρακτηριστικά) που έχει προτιμήσει ο χρήστης στις προηγούμενες αξιολογήσεις και ενέργειές του.	Αξιολογήσεις του χρήστη + χαρακτηριστικά στοιχείων
Βάση την γνώση	Συστάσεις με βάση τις σαφείς προδιαγραφές του χρήστη για το είδος του περιεχομένου (χαρακτηριστικά) που θέλει.	Προδιαγραφές χρήστη + χαρακτηριστικά στοιχείου + γνώση τομέα

Collaborative Filtering (Συνεργατικό φιλτράρισμα)

Οι μέθοδοι συστημάτων συστάσεων συνεργατικού φιλτραρίσματος παράγουν συστάσεις αντικειμένων με βάση τα μοτίβα των αξιολογήσεων ή της χρήσης (π.χ. αγορές), χωρίς να χρειάζονται εξωγενείς πληροφορίες είτε για τα αντικείμενα είτε για τους χρήστες. Ενώ οι καθιερωμένες μέθοδοι λειτουργούν επαρκώς για πολλούς σκοπούς, παρουσιάζουμε διάφορες πρόσφατες επεκτάσεις που είναι διαθέσιμες στους αναλυτές που αναζητούν τις καλύτερες δυνατές συστάσεις[27]. Ο διαγωνισμός Netflix Prize που ξεκίνησε τον Οκτώβριο του 2006 έχει τροφοδοτήσει μεγάλη πρόσφατη πρόοδο στον τομέα του συνεργατικού φιλτραρίσματος. Για πρώτη φορά, η ερευνητική κοινότητα απέκτησε πρόσβαση σε ένα μεγάλης κλίμακας, βιομηχανικής ισχύος σύνολο δεδομένων 100 εκατομμυρίων αξιολογήσεων ταινιών - προσελκύοντας χιλιάδες επιστήμονες, φοιτητές, μηχανικούς και λάτρεις του κλάδου. Η φύση του διαγωνισμού ενθάρρυνε την ταχεία ανάπτυξη, όπου οι καινοτόμοι βασίζονταν σε κάθε γενιά τεχνικών για να βελτιώσουν την ακρίβεια και τις πρόβλεψης των συστημάτων. Επειδή όλες οι μέθοδοι κρίνονται με το ίδιο άκαμπτο κριτήριο σε κοινά δεδομένα, η εξέλιξη των ισχυρότερων μοντέλων ήταν ιδιαίτερα αποτελεσματική.[27]

Τα συστήματα συστάσεων βασίζονται σε διάφορους τύπους εισόδου. Το πιο βολικό είναι η υψηλή ποιότητας ρητή ανατροφοδότηση, όπου οι χρήστες αναφέρουν άμεσα το ενδιαφέρον τους για τα προϊόντα. Για παράδειγμα, το Netflix συλλέγει βαθμολογίες αστέρων για ταινίες και οι χρήστες του Youtube δηλώνουν τις προτιμήσεις τους ανάμεσα σε χιλιάδες βίντεο πατώντας τα κουμπιά Like/dislike.[27]

Επειδή η ρητή ανατροφοδότηση δεν είναι πάντα διαθέσιμη, ορισμένα συστήματα συστάσεων συμπεραίνουν τις προτιμήσεις των χρηστών από την πιο άφθονη σιωπηρή ανατροφοδότηση, η οποία αντανακλά έμμεσα αν κάτι θα προτιμηθεί μέσω της παρατήρησης της συμπεριφοράς των χρηστών. Οι τύποι σιωπηρής ανατροφοδότησης περιλαμβάνουν το ιστορικό αγορών, το ιστορικό περιήγησης, τα μοτίβα αναζήτησης ή ακόμη και τις κινήσεις του ποντικιού. Για παράδειγμα, ένας χρήστης που αγόρασε πολλά βιβλία του ίδιου συγγραφέα πιθανόν να του αρέσει ο συγγραφέας αυτός.[27], [28]

Προκειμένου να καταρτίζουν συστάσεις, τα συστήματα με συνεργατικό φιλτράρισμα πρέπει να συσχετίζουν δύο θεμελιωδώς διαφορετικές οντότητες, τα

αντικείμενα και τους χρήστες. Υπάρχουν δύο κύριες προσεγγίσεις για τη διευκόλυνση μιας τέτοιας σύγκρισης, τα neighborhood methods (μοντέλα γειτονιάς) και τα latent factor models (μοντέλα κρυφών παραγόντων) . Οι μέθοδοι γειτονιάς επικεντρώνονται στις σχέσεις μεταξύ αντικειμένων ή εναλλακτικά, μεταξύ χρηστών. Μια προσέγγιση item-item μοντελοποιεί την προτίμηση ενός χρήστη σε ένα αντικείμενο με βάση τις αξιολογήσεις παρόμοιων αντικειμένων από τον ίδιο χρήστη. Τα μοντέλα κρυφού παράγοντα, όπως η matrix factorization (παραγοντοποιήσει πινάκων) (ή αλλιώς SVD), περιλαμβάνουν μια εναλλακτική προσέγγιση μετασχηματίζοντας τόσο τα στοιχεία όσο και τους χρήστες στον ίδιο κρυφό χώρο παραγόντων. Ο κρυφός χώρος προσπαθεί να εξηγήσει τις αξιολογήσεις χαρακτηρίζοντας τόσο τα προϊόντα όσο και τους χρήστες με παράγοντες που προκύπτουν αυτόματα από τα σχόλια των χρηστών.[27]

Η παραγωγή ακριβέστερων μεθόδων πρόβλεψης απαιτεί την εμβάθυνση των θεμελίων τους και τη μείωση της εξάρτησης από αυθαίρετες αποφάσεις. Ωστόσο, η αναζήτηση ακριβέστερων μοντέλων πηγαίνει πέρα από αυτό. Τουλάχιστον εξίσου σημαντική είναι ο προσδιορισμός όλων των χαρακτηριστικών, που είναι διαθέσιμα στα δεδομένα. Οι συμβατικές τεχνικές αντιμετωπίζουν τα αραιά δεδομένα των αξιολογήσεων των χρηστών. Η ακρίβεια βελτιώνεται σημαντικά με την αξιοποίηση και άλλων πηγών πληροφοριών. Ένα χαρακτηριστικό παράδειγμα περιλαμβάνει όλα τα είδη των χρονικών επιδράσεων που αντικατοπτρίζουν τη δυναμική, χρονικά μεταβαλλόμενη φύση των αλληλεπιδράσεων χρήστη με αντικείμενο. Δεν είναι λιγότερο σημαντική η ακρόαση της κρυφής ανατροφοδότησης, όπως τα στοιχεία που επέλεξαν οι χρήστες να αξιολογήσουν (ανεξάρτητα από τις τιμές αξιολόγησης). Τα βαθμολογημένα στοιχεία δεν επιλέγονται τυχαία, αλλά αποκαλύπτουν ενδιαφέρουσες πτυχές των προτιμήσεων των χρηστών, που υπερβαίνουν τις αριθμητικές τιμές των αξιολογήσεων.[27]

Η βασική ιδέα των μεθόδων συνεργατικού φιλτραρίσματος είναι ότι αυτές οι απροσδιόριστες αξιολογήσεις μπορούν να υπολογιστούν, επειδή οι παρατηρούμενες αξιολογήσεις συχνά συσχετίζονται σε μεγάλο βαθμό μεταξύ διαφόρων χρηστών και αντικειμένων. Για παράδειγμα, θεωρήστε δύο χρήστες με τα ονόματα Alice και Bob, οι οποίοι έχουν πολύ παρόμοιες προτιμήσεις. Εάν οι αξιολογήσεις, τις οποίες έχουν καθορίσει και οι δύο, είναι πολύ παρόμοιες, τότε η ομοιότητά τους μπορεί να εντοπιστεί από τον υποκείμενο αλγόριθμο. Σε τέτοιες περιπτώσεις, είναι πολύ πιθανό

ότι οι αξιολογήσεις στις οποίες μόνο ο ένας από τους δύο έχει καθορίσει μια τιμή, είναι επίσης πιθανό να είναι παρόμοιες. [23]

Αυτή η ομοιότητα μπορεί να χρησιμοποιηθεί για την εξαγωγή συμπερασμάτων σχετικά με ελλιπώς καθορισμένες τιμές. Τα περισσότερα μοντέλα για συνεργατικό φιλτράρισμα εστιάζουν στην αξιοποίηση είτε των συσχετίσεων μεταξύ των στοιχείων είτε των συσχετίσεων μεταξύ των χρηστών για τη διαδικασία πρόβλεψης. Ορισμένα μοντέλα χρησιμοποιούν και τους δύο τύπους συσχετίσεων. Επιπλέον, ορισμένα μοντέλα χρησιμοποιούν προσεκτικά σχεδιασμένες τεχνικές βελτιστοποίησης για τη δημιουργία ενός μοντέλου εκπαίδευσης με τον ίδιο περίπου τρόπο με τον οποίο ένας classifier(ταξινομητής) δημιουργεί ένα μοντέλο εκπαίδευσης από τα επισημασμένα δεδομένα. Αυτό το μοντέλο χρησιμοποιείται στη συνέχεια για τον υπολογισμό των ελλιπών τιμών στον πίνακα, με τον ίδιο τρόπο που ένας ταξινομητής υπολογίζει τις ελλιπείς ετικέτες δοκιμής. [29]

Μέθοδοι συνεργατικού φιλτραρίσματος

Υπάρχουν δύο τύποι μεθόδων που χρησιμοποιούνται συνήθως στο συνεργατικό φιλτράρισμα, οι οποίοι αναφέρονται ως μέθοδοι που βασίζονται στη μνήμη και μέθοδοι που βασίζονται στο μοντέλο:

1. Μέθοδοι που βασίζονται στη μνήμη: Οι μέθοδοι που βασίζονται στη μνήμη αναφέρονται επίσης ως neighborhood-based collaborative filtering algorithms.(αλγόριθμοι συνεργατικού φιλτραρίσματος με βάση τη γειτονιά). Αυτές ήταν από τους πρώτους αλγορίθμους συνεργατικού φιλτραρίσματος, στους οποίους οι αξιολογήσεις των συνδυασμών χρήστη-στοιχείου προβλέπονται με βάση τις γειτονιές τους. Αυτές οι γειτονιές μπορούν να οριστούν με έναν από τους δύο τρόπους[23]
    - Συνεργατικό φιλτράρισμα με βάση τον χρήστη: Η βασική ιδέα είναι να προσδιοριστούν οι χρήστες, οι οποίοι είναι παρόμοιοι με τον χρήστη-στόχο A, και να προταθούν αξιολογήσεις για τις μη παρατηρούμενες αξιολογήσεις του A υπολογίζοντας σταθμισμένους μέσους όρους των αξιολογήσεων αυτής της ομάδας. Επομένως, εάν η Alice και ο Bob έχουν αξιολογήσει ταινίες με παρόμοιο τρόπο στο παρελθόν, τότε μπορεί κανείς να χρησιμοποιήσει τις παρατηρούμενες αξιολογήσεις της Alice στην ταινία T για να προβλέψει τις μη παρατηρούμενες αξιολογήσεις του Bob σε αυτή

την ταινία. Γενικά, οι k πιο όμοιοι χρήστες με τον Bob μπορούν να χρησιμοποιηθούν για να γίνουν προβλέψεις βαθμολογίας για τον Bob. Οι συναρτήσεις ομοιότητας υπολογίζονται μεταξύ των γραμμών του πίνακα βαθμολογιών για την ανακάλυψη παρόμοιων χρηστών.[23]

- Συνεργατικό φιλτράρισμα με βάση τα στοιχεία: Προκειμένου να γίνουν οι προβλέψεις αξιολόγησης για το αντικείμενο στόχο B από τον χρήστη A, το πρώτο βήμα είναι να προσδιοριστεί ένα σύνολο S στοιχείων που είναι που μοιάζουν περισσότερο με το αντικείμενο στόχο B. Οι αξιολογήσεις στο σύνολο αντικειμένων S, οι οποίες καθορίζονται από τον A, χρησιμοποιούνται για να προβλέψουν αν το αντικείμενο B θα αρέσει στον χρήστη A. Επομένως, η βαθμολογία του Bob αξιολογήσεις σε παρόμοιες ταινίες επιστημονικής φαντασίας όπως το Alien και το Predator μπορούν να χρησιμοποιηθούν για να προβλέψουμε τη βαθμολογία του στον Terminator. Οι συναρτήσεις ομοιότητας υπολογίζονται μεταξύ των στηλών του πίνακα βαθμολογιών για την ανακάλυψη παρόμοιων αντικειμένων.[23]

Τα πλεονεκτήματα των τεχνικών που βασίζονται στη μνήμη είναι ότι είναι απλές στην εφαρμογή τους. και οι συστάσεις που προκύπτουν είναι συχνά εύκολο να εξηγηθούν. Από την άλλη πλευρά, οι αλγόριθμοι που βασίζονται στη μνήμη δεν λειτουργούν πολύ καλά με αραιούς πίνακες αξιολόγησης. Για παράδειγμα, μπορεί να είναι δύσκολο να βρεθούν αρκετοί παρόμοιοι χρήστες με τον Bob, οι οποίοι έχουν βαθμολογήσει Gladiator. Σε τέτοιες περιπτώσεις, είναι δύσκολο να προβλεφθεί αξιόπιστα η αξιολόγηση του Gladiator από τον Bob. Με άλλα λόγια, τέτοιες μέθοδοι μπορεί να μην έχουν πλήρη κάλυψη των προβλέψεων αξιολόγησης.[23]

2. Μέθοδοι βασισμένες σε μοντέλα: Στις μεθόδους που βασίζονται σε μοντέλα, χρησιμοποιούνται μέθοδοι μηχανικής μάθησης και εξόρυξης δεδομένων στο πλαίσιο προγνωστικών μοντέλων. Σε περιπτώσεις όπου το μοντέλο είναι παραμετροποιημένο, οι παράμετροι αυτού του μοντέλου μαθαίνονται στο πλαίσιο ενός πλαισίου βελτιστοποίησης. Ορισμένα παραδείγματα τέτοιων μεθόδων βασισμένων σε μοντέλα περιλαμβάνουν δέντρα αποφάσεων, μοντέλα βασισμένα σε κανόνες, μεθόδους Bayes και μοντέλα λανθανόντων παραγόντων. Πολλές από αυτές τις μεθόδους, όπως τα μοντέλα λανθανόντων παραγόντων, έχουν υψηλό επίπεδο κάλυψης ακόμη και για αραιούς πίνακες αξιολόγησης.[30]

Content-based Recommender (Σύσταση βάσει περιεχομένου)

Τα συνεργατικά συστήματα που αναφέρθηκαν προηγούμενος χρησιμοποιούν τις συσχετίσεις στα μοτίβα αξιολογήσεων μεταξύ των χρηστών για να κάνουν συστάσεις. Από την άλλη πλευρά, αυτές οι μέθοδοι δεν χρησιμοποιούν τα χαρακτηριστικά των αντικειμένων για τον υπολογισμό των προβλέψεων. Αυτό θα φαινόταν μάλλον σπάταλο, άλλωστε, αν σε καποιων αρέσει μια ταινία δράσης, τότε υπάρχει μια πολύ μεγάλη πιθανότητα να του αρέσει και μια ταινία παρόμοιου είδους. Σε τέτοιες περιπτώσεις, οι αξιολογήσεις άλλων χρηστών μπορεί να μην είναι απαραίτητες για την πραγματοποίηση ουσιαστικών συστάσεων. Τα συστήματα που βασίζονται στο περιεχόμενο έχουν σχεδιαστεί για να αξιοποιούν σενάρια στα οποία τα στοιχεία μπορούν να περιγραφούν με χαρακτηριστικά σύνολα χαρακτηριστικών. Σε τέτοιες περιπτώσεις, οι αξιολογήσεις του ίδιου του χρήστη και οι ενέργειες σε άλλα ταινίες είναι επαρκείς για την ανακάλυψη ουσιαστικών συστάσεων. Αυτή η προσέγγιση είναι ιδιαίτερα χρήσιμη όταν το στοιχείο είναι νέο και υπάρχουν λίγες διαθέσιμες αξιολογήσεις για το στοιχείο αυτό.[30]

Τα συστήματα συστάσεων που βασίζονται στο περιεχόμενο προσπαθούν να αντιστοιχίσουν τους χρήστες με αντικείμενα που είναι παρόμοια με αυτά που έχουν προτιμήσει στο παρελθόν. Αυτή η ομοιότητα δεν βασίζεται απαραίτητα σε συσχετίσεις βαθμολογίας μεταξύ των χρηστών, αλλά με βάση τα χαρακτηριστικά των αντικειμένων που άρεσαν στον χρήστη. Σε αντίθεση με τα συνεργατικά συστήματα, τα οποία αξιοποιούν ρητά τις αξιολογήσεις άλλων χρηστών εκτός από αυτές του χρήστη-στόχου, τα συστήματα που βασίζονται στο περιεχόμενο εστιάζουν σε μεγάλο βαθμό στις αξιολογήσεις του ίδιου του χρήστη-στόχου και τα χαρακτηριστικά των αντικειμένων που αρέσουν στον χρήστη. Ως εκ τούτου, οι άλλοι χρήστες έχουν ελάχιστο, αν όχι καθόλου, ρόλο στα συστήματα που βασίζονται στο περιεχόμενο. Με άλλα λόγια, η μεθοδολογία που βασίζεται στο περιεχόμενο αξιοποιεί μια τελείως διαφορετική πηγή δεδομένων για τη διαδικασία σύστασης.[22]

Με μια απλή ματιά, τα συστήματα που βασίζονται στο περιεχόμενο εξαρτώνται από δύο πηγές δεδομένων:

1. Η πρώτη πηγή δεδομένων είναι η περιγραφή των διαφόρων αντικειμένων με βάση τα χαρακτηριστικά που επικεντρώνονται στο περιεχόμενο. Ένα παράδειγμα τέτοιας αναπαράστασης θα μπορούσε ένα κείμενο που περιγράφει ένα αντικείμενο γραμμένο από τον κατασκευαστή.[26]

2. Η δεύτερη πηγή δεδομένων είναι ένα προφίλ χρήστη, το οποίο δημιουργείται από τα σχόλια των χρηστών σχετικά με διάφορα στοιχεία. Η ανατροφοδότηση του χρήστη μπορεί να είναι ρητή ή σιωπηρή. Η ρητή ανατροφοδότηση μπορεί αντιστοιχηθεί σε αξιολογήσεις, ενώ η σιωπηρή ανατροφοδότηση μπορεί να αντιστοιχεί σε ενέργειες του χρήστη. Το αξιολογήσεις συλλέγονται με τρόπο παρόμοιο με τα συνεργατικά συστήματα.[26]

   Το προφίλ χρήστη συσχετίζει τα χαρακτηριστικά των διαφόρων αντικειμένων με τα ενδιαφέροντα του χρήστη (βαθμολογίες). Ένα πολύ βασικό παράδειγμα ενός προφίλ χρήστη θα μπορούσε να είναι απλά ένα σύνολο επισημασμένων εκπαιδευτικών εγγράφων με περιγραφές αντικειμένων, τις αξιολογήσεις των χρηστών ως ετικέτες, και μια ταξινόμηση ή μοντέλο παλινδρόμησης που συνδέει τα χαρακτηριστικά των αντικειμένων με τις αξιολογήσεις των χρηστών. Το συγκεκριμένο προφίλ χρήστη εξαρτάται σε μεγάλο βαθμό από την εκάστοτε μεθοδολογία. Για παράδειγμα, σε ένα περιβάλλον μπορεί να χρησιμοποιούνται ρητές αξιολογήσεις, ενώ σε ένα άλλο περιβάλλον μπορεί να χρησιμοποιείται σιωπηρή ανατροφοδότηση. Είναι επίσης δυνατό ο χρήστης να καθορίσει το δικό του προφίλ όσον αφορά τις λέξεις-κλειδιά που τον ενδιαφέρουν, και αυτή η προσέγγιση μοιράζεται ορισμένα χαρακτηριστικά με τα συστήματα συστάσεων που βασίζονται στη γνώση.[26]

Τα συστήματα που βασίζονται στο περιεχόμενο χαρακτηρίζονται από την εστίασή τους στην αξιοποίηση των πληροφοριών στις περιγραφές των αντικειμένων, ενώ στα συστήματα που βασίζονται στη γνώση υπάρχει συνήθως κάποιο είδος πρόσθετης γνώσης, όπως μια συνάρτηση χρησιμότητας, για την παραγωγή συστάσεων.[31]

Βασικά τμήματα των συστημάτων βασισμένων στο περιεχόμενο

Τα συστήματα βασισμένα στο περιεχόμενο έχουν ορισμένα βασικά στοιχεία, τα οποία παραμένουν αναλλοίωτα σε διάφορες περιπτώσεις τέτοιων συστημάτων. Δεδομένου ότι τα συστήματα βασισμένα στο περιεχόμενο λειτουργούν με μια μεγάλη ποικιλία περιγραφών στοιχείων και γνώσεων σχετικά με τους χρήστες, πρέπει να μετατρέψει κανείς αυτούς τους διαφορετικούς τύπους αδόμητων δεδομένων σε τυποποιημένες περιγραφές. Στις περισσότερες περιπτώσεις, προτιμάται η μετατροπή των περιγραφών στοιχείων σε λέξεις-κλειδιά. Ως εκ τούτου, τα συστήματα που βασίζονται στο περιεχόμενο λειτουργούν σε μεγάλο βαθμό, αλλά όχι αποκλειστικά, στον τομέα του κειμένου. Πολλές φυσικές εφαρμογές συστημάτων βασισμένων στο περιεχόμενο είναι επίσης κειμενοκεντρικές. Για παράδειγμα, τα συστήματα σύστασης ειδήσεων είναι συχνά συστήματα βασισμένα στο περιεχόμενο, και είναι επίσης κειμενοκεντρικά συστήματα. Σε γενικές γραμμές, οι μέθοδοι ταξινόμησης κειμένου και μοντελοποίησης παλινδρόμησης παραμένουν τα πιο ευρέως χρησιμοποιούμενα εργαλεία για τη δημιουργία βασισμένων σε περιεχόμενο συστημάτων συστάσεων.[32]

Τα κύρια στοιχεία των συστημάτων που βασίζονται στο περιεχόμενο περιλαμβάνουν το τμήμα προεπεξεργασίας (εκτός σύνδεσης), το τμήμα εκμάθησης (εκτός σύνδεσης) και το τμήμα πρόβλεψης στο διαδίκτυο. Τα τμήματα εκτός σύνδεσης χρησιμοποιούνται για τη δημιουργία ενός συνοπτικού μοντέλου, το οποίο συχνά είναι ένα μοντέλο ταξινόμησης ή παλινδρόμησης. Αυτό το μοντέλο χρησιμοποιείται στη συνέχεια για τη διαδικτυακή δημιουργία συστάσεων για τους χρήστες. Τα διάφορα συστατικά στοιχεία των συστημάτων που βασίζονται στο περιεχόμενο είναι τα εξής:

- Προεπεξεργασία και εξαγωγή χαρακτηριστικών: Τα συστήματα βασισμένα στο περιεχόμενο χρησιμοποιούνται σε μια ευρεία ποικιλία, όπως ιστοσελίδες, περιγραφές προϊόντων, ειδήσεις κ.α. . Στις περισσότερες περιπτώσεις, εξάγονται χαρακτηριστικά από αυτές τις διάφορες πηγές για να μετατραπούν σε μια αναπαράσταση διανυσματικού χώρου βασισμένη σε λέξεις-κλειδιά. Αυτό είναι το πρώτο βήμα οποιουδήποτε συστήματος συστάσεων βάσει περιεχομένου. Ωστόσο, η κατάλληλη εξαγωγή των πιο κατατοπιστικών χαρακτηριστικών είναι απαραίτητη για την αποτελεσματική λειτουργία.[32]

- Εκμάθηση προφίλ χρηστών με βάση το περιεχόμενο: Όπως συζητήθηκε προηγουμένως, ένα μοντέλο βασισμένο στο περιεχόμενο είναι συγκεκριμένο για κάθε χρήστη. Ως εκ τούτου, κατασκευάζεται ένα μοντέλο ειδικά για τον χρήστη για την πρόβλεψη των ενδιαφερόντων του χρήστη για αντικείμενα, με βάση το προηγούμενο ιστορικό του είτε στην αγορά είτε στην αξιολόγηση αντικειμένων. Για την επίτευξη αυτού του στόχου, αξιοποιείται η ανατροφοδότηση του χρήστη, η οποία μπορεί να εκδηλώνεται με τη μορφή προηγουμένως καθορισμένων αξιολογήσεων (ρητή ανατροφοδότηση) ή δραστηριότητας του χρήστη (σιωπηρή ανατροφοδότηση). Τέτοιες ανατροφοδοτήσεις χρησιμοποιούνται σε συνδυασμό με τα χαρακτηριστικά των αντικειμένων προκειμένου να κατασκευαστούν τα δεδομένα εκπαίδευσης. Ένα μοντέλο εκπαίδευσης κατασκευάζεται σε αυτά τα δεδομένα εκπαίδευσης δεδομένα. Αυτό το στάδιο συχνά δεν διαφέρει πολύ από τη μοντελοποίηση ταξινόμησης ή παλινδρόμησης, ανάλογα με το αν η ανατροφοδότηση είναι κατηγορική (π.χ. δυαδική πράξη επιλογής ενός στοιχείου), ή αν η ανατροφοδότηση είναι αριθμητική (π.χ. βαθμολογίες ή συχνότητα αγοράς). Το προκύπτον μοντέλο αναφέρεται ως προφίλ χρήστη, επειδή συνδέει εννοιολογικά τον χρήστη ενδιαφέροντα (αξιολογήσεις) με τα χαρακτηριστικά των αντικειμένων.[32]

- Φιλτράρισμα και συστάσεις: Σε αυτό το βήμα, το μοντέλο που μαθαίνεται από το προηγούμενο βήμα χρησιμοποιείται για να κάνει συστάσεις σχετικά με αντικείμενα για συγκεκριμένους χρήστες. Είναι σημαντικό αυτό το βήμα να είναι πολύ αποτελεσματικό, επειδή οι προβλέψεις πρέπει να πραγματοποιούνται σε πραγματικό χρόνο. [32]

Εξαγωγή χαρακτηριστικών

Στη φάση εξαγωγής χαρακτηριστικών, εξάγονται οι περιγραφές των διαφόρων στοιχείων. Παρόλο που είναι δυνατόν να χρησιμοποιηθεί οποιοδήποτε είδος αναπαράστασης, όπως μια πολυδιάστατη αναπαράσταση δεδομένων, η πιο συνηθισμένη προσέγγιση είναι η εξαγωγή λέξεων-κλειδιών από τα υποκείμενα δεδομένα. Η επιλογή αυτή είναι επειδή οι μη δομημένες περιγραφές κειμένου είναι συχνά ευρέως διαθέσιμες σε διάφορους τομείς, και παραμένουν οι πιο φυσικές

αναπαραστάσεις για την περιγραφή στοιχείων. Σε πολλές περιπτώσεις, τα στοιχεία μπορεί να έχουν πολλαπλά πεδία που περιγράφουν διάφορες πτυχές του στοιχείου. Για παράδειγμα, ένας έμπορος που πουλάει βιβλία μπορεί να έχει περιγραφές των βιβλίων και λέξεις-κλειδιά που περιγράφουν διάφορα πεδία , ώστε να διευκολύνεται διαδικασία ταξινόμησης. Η απόδοση βαρύτητας σε χαρακτηριστικά συνδέεται στενά με την επιλογή χαρακτηριστικών, δεδομένου ότι η πρώτη είναι μια ήπια εκδοχή της δεύτερης. Στην τελευταία περίπτωση, τα χαρακτηριστικά είτε περιλαμβάνονται είτε δεν περιλαμβάνονται ανάλογα με τη σημασία τους, ενώ στην πρώτη περίπτωση, τα χαρακτηριστικά λαμβάνουν διαφορετική βαρύτητα ανάλογα με τη σημασία τους. Επειδή η φάση εξαγωγής χαρακτηριστικών είναι ιδιαίτερα συγκεκριμένη για κάθε εφαρμογή, παρέχετε στον αναγνώστη μια γεύση των τύπων χαρακτηριστικών που μπορεί να χρειαστεί να εξαχθούν στο πλαίσιο διαφόρων εφαρμογών.[32]

Συλλογή προτιμήσεων του χρήστη

Εκτός από το περιεχόμενο σχετικά με τα αντικείμενα, είναι επίσης απαραίτητο να συλλέγονται δεδομένα σχετικά με τον χρήστη. συμπάθειες και αντιπάθειες για τη διαδικασία σύστασης. Η συλλογή δεδομένων γίνεται κατά τη διάρκεια της φάση εκτός σύνδεσης, ενώ οι συστάσεις προσδιορίζονται κατά τη διάρκεια της online φάσης, όταν ένας συγκεκριμένος χρήστης αλληλοεπιδρά με το σύστημα. Ο χρήστης για τον οποίο πραγματοποιείται η πρόβλεψη σε κάθε δεδομένη στιγμή αναφέρεται ως ενεργός χρήστης. Κατά τη διάρκεια της διαδικτυακής φάσης, οι δικές του χρήστη προτιμήσεις του χρήστη συνδυάζονται με το περιεχόμενο για τη δημιουργία των προβλέψεων. Τα δεδομένα σχετικά με τον χρήστη συμπάθειες και αντιπάθειες του χρήστη μπορούν να λάβουν οποιαδήποτε από τις ακόλουθες μορφές:

- Αξιολογήσεις: Σε αυτή την περίπτωση, οι χρήστες καθορίζουν βαθμολογίες που υποδεικνύουν την προτίμησή τους για το στοιχείο. Οι αξιολογήσεις μπορεί να είναι δυαδικές, χρονικό διάστημα ή τακτικός αριθμός. Σε σπάνιες περιπτώσεις, οι αξιολογήσεις μπορεί να είναι κάποιος πραγματικός αριθμός . Η φύση της βαθμολογίας έχει σημαντικό αντίκτυπο στο μοντέλο που χρησιμοποιείται για εκμάθηση των προφίλ των χρηστών.[32]

- Σιωπηρή ανατροφοδότηση: Η σιωπηρή ανατροφοδότηση αναφέρεται σε ενέργειες του χρήστη, όπως η αγορά ή η περιήγηση ενός αντικειμένου. Στις περισσότερες περιπτώσεις, μόνο οι θετικές προτιμήσεις ενός χρήστη καταγράφονται με σιωπηρή ανατροφοδότηση, αλλά όχι οι αρνητικές προτιμήσεις.[32]

- Γνώμες κειμένου: Σε πολλές περιπτώσεις, οι χρήστες μπορούν να εκφράσουν τις απόψεις τους με τη μορφή κειμένου. Σε τέτοιες περιπτώσεις, σιωπηρές αξιολογήσεις μπορούν να εξαχθούν από αυτές τις αξιολογήσεις. Αυτή η μορφή εξαγωγής αξιολογήσεων σχετίζεται με τον τομέα της εξόρυξης γνώμης και της ανάλυσης συναισθήματος. [33]

    - Περιπτώσεις: Οι χρήστες μπορούν να καθορίσουν παραδείγματα (ή περιπτώσεις) αντικειμένων που τους ενδιαφέρουν. Τέτοιες περιπτώσεις μπορούν να χρησιμοποιηθούν ως σιωπηρή ανατροφοδότηση με ταξινομητές πλησιέστερου γείτονα. Ωστόσο, όταν η ανάκτηση ομοιότητας χρησιμοποιείται σε συνδυασμό με προσεκτικά σχεδιασμένες συναρτήσεις χρησιμότητας, οι μέθοδοι αυτές σχετίζονται περισσότερο με τα συστήματα συστάσεων που βασίζονται σε περιπτώσεις. Τα συστήματα βασισμένα σε περιπτώσεις είναι μια υποκατηγορία των βασισμένων στη γνώση συστημάτων συστάσεων συστήματα στα οποία η γνώση του τομέα χρησιμοποιείται για την ανακάλυψη αντικειμένων που ταιριάζουν, αντί για αλγορίθμων εκμάθησης. Είναι δύσκολο να οριοθετηθεί που τελειώνουν τα συστήματα συστάσεων βασισμένα στο περιεχόμενο και τα συστήματα συστάσεων βασισμένα στη γνώση.[32]

Σε όλες τις προαναφερθείσες περιπτώσεις, οι συμπάθειες ή αντιπάθειες ενός χρήστη για ένα στοιχείο μετατρέπονται τελικά σε μοναδιαία, δυαδική, χρονικό διάστημα ή πραγματική βαθμολογία. Αυτή η βαθμολογία μπορεί επίσης να θεωρηθεί ως η εξαγωγή μιας ετικέτας κλάσης ή εξαρτημένης μεταβλητής.[32]

Knowledge-Based Recommender (Συστάσεις βασισμένες στη γνώση)

Τα περισσότερα εμπορικά συστήματα συστάσεων στην πράξη βασίζονται σε τεχνικές συνεργατικού φιλτραρίσματος. Τα συστήματα συνεργατικού φιλτραρίσματος βασίζονται αποκλειστικά στις αξιολογήσεις των χρηστών (και μερικές φορές σε δημογραφικές πληροφορίες) ως τις μόνες πηγές γνώσης για τη δημιουργία προτάσεων στοιχείων για τους χρήστες τους. Έτσι, δεν χρειάζεται να εισαχθεί και να διατηρηθεί στο σύστημα καμία πρόσθετη γνώση - όπως πληροφορίες σχετικά με τις διαθέσιμες ταινίες και τα χαρακτηριστικά τους. Οι τεχνικές συστάσεων που βασίζονται στο περιεχόμενο, χρησιμοποιούν διαφορετικές πηγές γνώσης για να κάνουν προβλέψεις για το αν ένα στοιχείο θα αρέσει σε έναν χρήστη. Οι κύριες πηγές γνώσης που αξιοποιούνται από τα συστήματα που βασίζονται στο περιεχόμενο περιλαμβάνουν πληροφορίες για την κατηγορία και το είδος, καθώς και λέξεις-κλειδιά που συχνά μπορούν να εξαχθούν αυτόματα από τις περιγραφές αντικειμένων σε κείμενο. Παρόμοια με το συνεργατικό φιλτράρισμα, ένα σημαντικό πλεονέκτημα των μεθόδων συστάσεων βάσει περιεχομένου είναι το συγκριτικά χαμηλό κόστος για την απόκτηση και τη συντήρηση της γνώσης. Τόσο οι συνεργατικοί όσο και οι βασισμένοι στο περιεχόμενο αλγόριθμοι σύστασης έχουν τα πλεονεκτήματα και τα δυνατά τους σημεία. Ωστόσο, υπάρχουν πολλές καταστάσεις για τις οποίες αυτές οι προσεγγίσεις δεν αποτελούν την καλύτερη επιλογή. Συνήθως, δεν αγοράζουμε πολύ συχνά ένα σπίτι, ένα αυτοκίνητο ή έναν υπολογιστή. Σε ένα τέτοιο σενάριο, ένα αμιγώς σύστημα με συνεργατικό φιλτράρισμα δεν θα αποδώσει καλά λόγω του χαμηλού αριθμού των διαθέσιμων αξιολογήσεων. Επιπλέον, τα χρονικά διαστήματα παίζουν σημαντικό ρόλο. Για παράδειγμα, αξιολογήσεις πέντε ετών για υπολογιστές μπορεί να είναι μάλλον ακατάλληλες για συστάσεις βάσει περιεχομένου. Το ίδιο ισχύει και για αντικείμενα όπως τα αυτοκίνητα ή τα σπίτια, καθώς οι προτιμήσεις των χρηστών εξελίσσονται με την πάροδο του χρόνου λόγω, για παράδειγμα, αλλαγών στον τρόπο ζωής ή στις οικογενειακές καταστάσεις. Τέλος, σε πιο σύνθετους τομείς προϊόντων, όπως τα αυτοκίνητα, οι πελάτες συχνά θέλουν να καθορίζουν ρητά τις απαιτήσεις τους - για παράδειγμα, "η μέγιστη τιμή του αυτοκινήτου είναι x και το χρώμα πρέπει να είναι μαύρο". Η διατύπωση τέτοιων απαιτήσεων δεν είναι τυπικό για τα καθαρά συνεργατικά πλαίσια συστάσεων και τα πλαίσια συστάσεων βάσει περιεχομένου.[31]

Τα συστήματα συστάσεων που βασίζονται στη γνώση μας βοηθούν να αντιμετωπίσουμε τις προαναφερθείσες προκλήσεις. Το πλεονέκτημα αυτών των συστημάτων είναι ότι δεν υπάρχουν προβλήματα εκκίνησης, επειδή δεν απαιτούνται δεδομένα αξιολόγησης για τον υπολογισμό των συστάσεων. Οι συστάσεις υπολογίζονται ανεξάρτητα από τις μεμονωμένες αξιολογήσεις των χρηστών: είτε με τη μορφή ομοιοτήτων μεταξύ των απαιτήσεων των πελατών και των αντικειμένων είτε με βάση ρητούς κανόνες συστάσεων. Οι παραδοσιακές ερμηνείες του τι είναι ένα σύστημα συστάσεων εστιάζουν στην πτυχή του φιλτραρίσματος πληροφοριών, κατά την οποία φιλτράρονται τα στοιχεία που είναι πιθανό να ενδιαφέρουν έναν συγκεκριμένο πελάτη. Αντίθετα, η διαδικασία σύστασης των βασισμένων στη γνώση εφαρμογών σύστασης είναι σε μεγάλο βαθμό διαδραστική, μια θεμελιώδης ιδιότητα που αποτελεί λόγο για τον χαρακτηρισμό τους ως συνομιλητικά συστήματα. Αυτή η πτυχή της διαδραστικότητας προκάλεσε μια μικρή μετατόπιση από την ερμηνεία ως σύστημα φιλτραρίσματος προς μια ευρύτερη ερμηνεία όπου τα συστηματα ορίζονται ως συστήματα που "καθοδηγούν έναν χρήστη με εξατομικευμένο τρόπο σε ενδιαφέροντα ή χρήσιμα αντικείμενα σε ένα μεγάλο χώρο πιθανών επιλογών ή που παράγουν τέτοια αντικείμενα ως έξοδο. Τα συστήματα συστάσεων που βασίζονται σε πηγές γνώσης που δεν αξιοποιούνται από τις συνεργατικές προσεγγίσεις και τις προσεγγίσεις που βασίζονται στο περιεχόμενο ορίζονται εξ ορισμού ως συστήματα συστάσεων που βασίζονται στη γνώση. [31]

Τα συστήματα συστάσεων που βασίζονται στη γνώση μπορούν να κατηγοριοποιηθούν με βάση τη μεθοδολογία αλληλεπίδρασης με τον χρήστη και τις αντίστοιχες βάσεις γνώσεων που χρησιμοποιούνται για τη διευκόλυνση της αλληλεπίδρασης. Υπάρχουν δύο βασικοί τύποι συστημάτων συστάσεων που βασίζονται στη γνώση [34]

- Συστήματα συστάσεων βασισμένα σε περιορισμούς: Οι περιορισμοί αντιπροσωπεύουν τη γνώση του συγκεκριμένου τομέα που χρησιμοποιείται από το σύστημα. Τέτοιοι κανόνες θα μπορούσαν να έχουν τη μορφή περιορισμών που αφορούν συγκεκριμένο τομέα στα χαρακτηριστικά του στοιχείου (π.χ. "Αυτοκίνητα πριν από το έτος 1970 δεν έχουν cruise control."). Επιπλέον, τα συστήματα που βασίζονται σε περιορισμούς συχνά δημιουργούν κανόνες που συσχετίζουν τα χαρακτηριστικά του χρήστη με τα χαρακτηριστικά του στοιχείου (π.χ. "Οι μεγαλύτεροι σε ηλικία επενδυτές δεν επενδύουν σε προϊόντα εξαιρετικά

υψηλού κινδύνου."). Σε τέτοιες περιπτώσεις, τα χαρακτηριστικά του χρήστη μπορούν επίσης να καθοριστούν κατά τη διαδικασία αναζήτησης. Ανάλογα με τον αριθμό και τον τύπο των επιστρεφόμενων αποτελεσμάτων, ο χρήστης μπορεί να έχει την ευκαιρία να τροποποιήσει τις αρχικές του απαιτήσεις. Για παράδειγμα, ο χρήστης μπορεί να χαλαρώσει ορισμένους περιορισμούς όταν επιστρέφονται πολύ λίγα αποτελέσματα ή να προσθέσει περισσότερους περιορισμούς όταν επιστρέφονται πάρα πολλά αποτελέσματα. Αυτή η διαδικασία αναζήτησης επαναλαμβάνεται διαδραστικά έως ότου ο χρήστης καταλήξει στα επιθυμητά αποτελέσματα. [35], [36]

- Συστήματα συστάσεων βασισμένα σε περιπτώσεις: Στα συστήματα συστάσεων που βασίζονται σε περιπτώσεις, ο χρήστης καθορίζει συγκεκριμένες περιπτώσεις ως στόχους ή σημεία άγκυρας. Οι δείκτες ομοιότητας ορίζονται στα χαρακτηριστικά των αντικειμένων για την ανάκτηση παρόμοιων αντικειμένων με αυτούς τους στόχους. Οι δείκτες ομοιότητας συχνά ορίζονται προσεκτικά με τρόπο που να αφορά συγκεκριμένο τομέα. Ως εκ τούτου, οι δείκτες ομοιότητας αποτελούν τη γνώση του τομέα που χρησιμοποιείται σε τέτοια συστήματα. Τα επιστρεφόμενα αποτελέσματα χρησιμοποιούνται συχνά ως νέες περιπτώσεις στόχων με κάποιες διαδραστικές τροποποιήσεις από τον χρήστη. Για παράδειγμα, όταν ένας χρήστης βλέπει ένα επιστρεφόμενο αποτέλεσμα που είναι σχεδόν παρόμοιο με αυτό που θέλει, μπορεί να επανεκδώσει ένα ερώτημα με αυτόν τον στόχο, αλλά με κάποια από τα χαρακτηριστικά τροποποιημένα κατά την προτίμησή του. Εναλλακτικά, μπορεί να καθοριστεί μια κατευθυνόμενη κριτική για να κλαδέψει στοιχεία με συγκεκριμένες τιμές χαρακτηριστικών μεγαλύτερες ή μικρότερες από αυτές ενός συγκεκριμένου στοιχείου ενδιαφέροντος. Αυτή η διαδραστική διαδικασία χρησιμοποιείται για την καθοδήγηση του χρήστη προς την τελική σύσταση.[25], [34]

Και στις δύο περιπτώσεις, το σύστημα παρέχει τη δυνατότητα στο χρήστη να αλλάξει τις καθορισμένες απαιτήσεις του. Ωστόσο, ο τρόπος με τον οποίο γίνεται αυτό είναι διαφορετικός στις δύο περιπτώσεις. Στα συστήματα που βασίζονται σε περιπτώσεις, παραδείγματα χρησιμοποιούνται ως σημεία άγκυρας για την καθοδήγηση της αναζήτησης σε συνδυασμό με δείκτες ομοιότητας, ενώ στα συστήματα που βασίζονται σε περιορισμούς, χρησιμοποιούνται συγκεκριμένα κριτήρια/κανόνες (ή

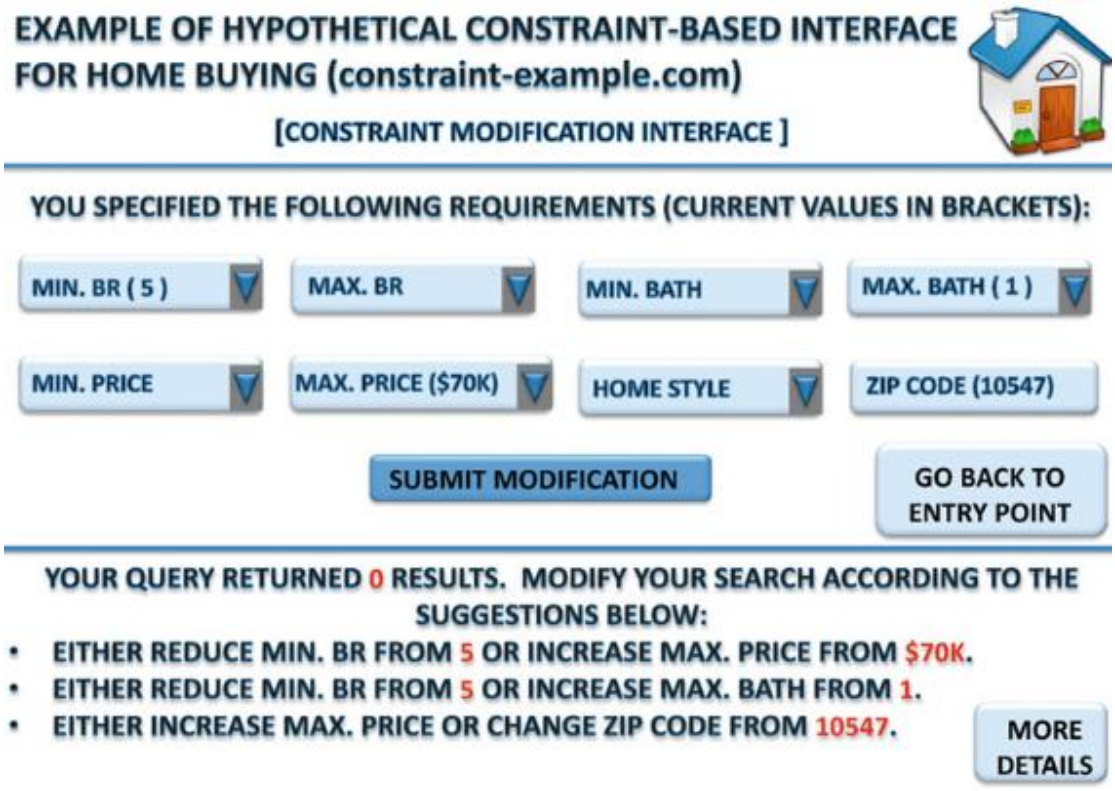
περιορισμοί) για την καθοδήγηση της αναζήτησης. Και στις δύο περιπτώσεις, τα παρουσιαζόμενα αποτελέσματα χρησιμοποιούνται για την τροποποίηση των κριτηρίων για την εύρεση περαιτέρω συστάσεων. Τα συστήματα που βασίζονται στη γνώση αντλούν το όνομά τους από το γεγονός ότι κωδικοποιούν διάφορους τύπους γνώσης του τομέα με τη μορφή περιορισμών, κανόνων, μετρικών ομοιότητας και συναρτήσεων χρησιμότητας κατά τη διαδικασία αναζήτησης. Για παράδειγμα, ο σχεδιασμός ενός δείκτη ομοιότητας ή ενός συγκεκριμένου περιορισμού απαιτεί γνώση ειδικού τομέα, η οποία είναι ζωτικής σημασίας για την αποτελεσματική λειτουργία του συστήματος. Γενικά, τα συστήματα που βασίζονται στη γνώση αντλούν από εξαιρετικά ετερογενείς, ειδικές για τον τομέα πηγές γνώσης, σε σύγκριση με τα συστήματα που βασίζονται στο περιεχόμενο και τα συνεργατικά συστήματα, τα οποία λειτουργούν με κάπως παρόμοιους τύπους δεδομένων εισόδου σε διάφορους τομείς.

Συστήματα συστάσεων βασιζόμενα σε περιορισμούς

Τα συστήματα συστάσεων που βασίζονται σε περιορισμούς επιτρέπουν στους χρήστες να καθορίζουν απαιτήσεις ή περιορισμούς για τα χαρακτηριστικά των αντικειμένων. Επιπλέον, χρησιμοποιείται ένα σύνολο κανόνων για την αντιστοίχιση των απαιτήσεων των πελατών με τα χαρακτηριστικά των ειδών. Ωστόσο, οι χρήστες ενδέχεται να μην καθορίζουν πάντα τα ερωτήματά τους με βάση τα ίδια χαρακτηριστικά που περιγράφουν τα αντικείμενα. Ως εκ τούτου, απαιτείται ένα πρόσθετο σύνολο κανόνων που συσχετίζει τις απαιτήσεις των πελατών με τα προϊόντα χαρακτηριστικά του προϊόντος. Μερικά παραδείγματα χαρακτηριστικά που καθορίζονται από τον χρήστη είναι τα ακόλουθα:
Οικογενειακή κατάσταση (κατηγορηματικά), μέγεθος οικογένειας (αριθμητικά), προάστια ή πόλη (δυαδικά), μέγιστη τιμή (αριθμητικά).[36]
Τα χαρακτηριστικά αυτά μπορεί να αντιπροσωπεύουν είτε εγγενείς ιδιότητες του πελάτη (π.χ. δημογραφικά στοιχεία) είτε να προσδιορίζουν τις απαιτήσεις του πελάτη για το προϊόν. Οι απαιτήσεις αυτές καθορίζονται συνήθως διαδραστικά κατά τη διάρκεια του διαλόγου μεταξύ του χρήστη και του συστήματος συστάσεων. Ενώ οι αντιστοιχίσεις ορισμένων χαρακτηριστικών απαιτήσεων, όπως η μέγιστη τιμή, σε χαρακτηριστικά προϊόντων είναι προφανείς, οι αντιστοιχίσεις άλλων, όπως το προάστιο ή η επαρχία, δεν είναι τόσο προφανείς. Ομοίως, σε μια χρηματοοικονομική

εφαρμογή, ένας χρήστης μπορεί να προσδιορίσει μια απαίτηση προϊόντος, όπως "συντηρητικές επενδύσεις", η οποία πρέπει να αντιστοιχιστεί σε συγκεκριμένα χαρακτηριστικά προϊόντος που περιγράφουν άμεσα τα προϊόντα. Προφανώς, πρέπει με κάποιο τρόπο να είναι σε θέση να αντιστοιχίσει αυτά τα χαρακτηριστικά/απαιτήσεις του χρήστη στα χαρακτηριστικά του προϊόντος προκειμένου να φιλτράρει τα προϊόντα για σύσταση. [36]

Τέτοιοι κανόνες αναφέρονται ως συνθήκες φιλτραρίσματος, επειδή αντιστοιχίζουν τις απαιτήσεις του χρήστη στα χαρακτηριστικά του στοιχείου και χρησιμοποιούν αυτή την αντιστοίχιση για να φιλτράρουν τα αποτελέσματα που ανακτώνται. Ένα άλλο παράδειγμα είναι ο τομέας των αυτοκινήτων, όπου ορισμένα προαιρετικά πακέτα μπορεί να ισχύουν μόνο με ορισμένα άλλα χαρακτηριστικά. Για παράδειγμα, ένας κινητήρας υψηλής ροπής μπορεί να είναι διαθέσιμος μόνο σε ένα σπορ μοντέλο. Τέτοιες συνθήκες αναφέρονται επίσης ως συνθήκες συμβατότητας, επειδή μπορούν να χρησιμοποιηθούν για τη γρήγορη ανακάλυψη ασυνεπειών στις απαιτήσεις που καθορίζονται από τον χρήστη με τον τομέα του προϊόντος. Σε πολλές περιπτώσεις, οι εν λόγω περιορισμοί συμβατότητας μπορούν να ενσωματωθούν στη διεπαφή χρήστη. [36]



Εικόνα 4:Ένα υποθετικό παράδειγμα διεπαφής χρήστη για το χειρισμό κενών αποτελεσμάτων ερωτημάτων στο μια σύσταση με βάση περιορισμούς[34]

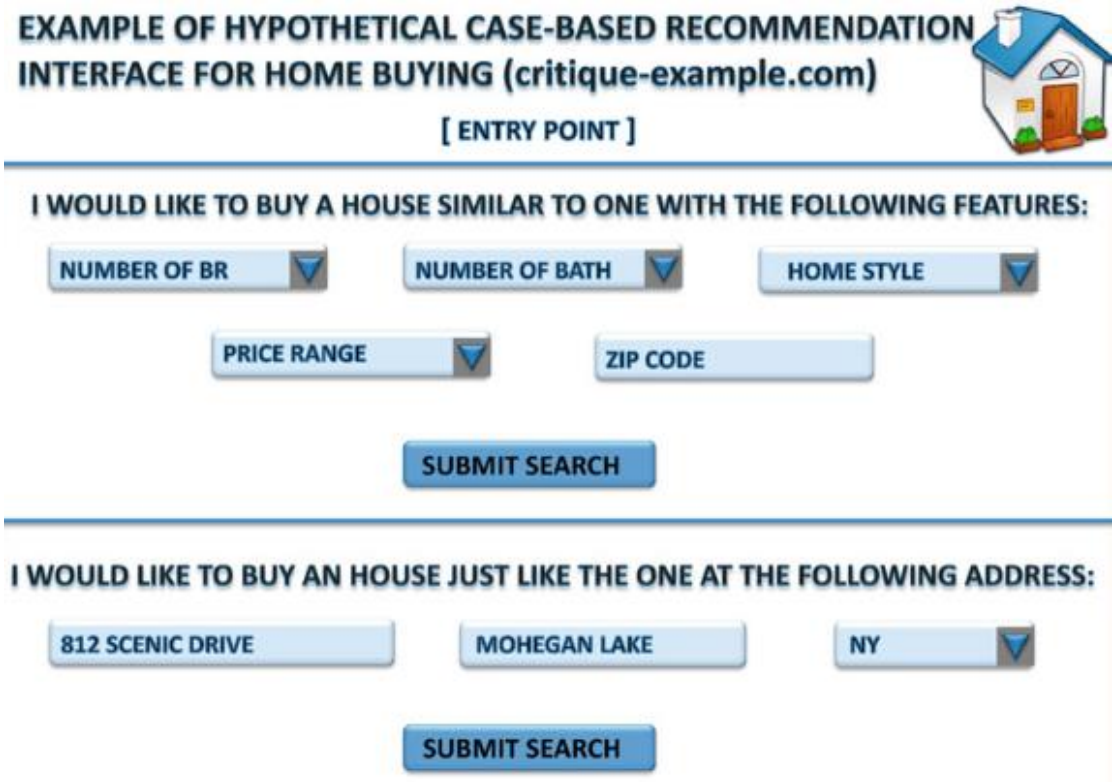
Συστήματα συστάσεων βασισμένα σε περιπτώσεις

Στα συστήματα που βασίζονται σε περιπτώσεις, οι δείκτες ομοιότητας χρησιμοποιούνται για την ανάκτηση παραδειγμάτων που είναι παρόμοια με τους καθορισμένους στόχους (ή περιπτώσεις). Σε αντίθεση με τα συστήματα που βασίζονται σε περιορισμούς, δεν υπάρχουν σκληροί περιορισμοί (π.χ. ελάχιστες ή μέγιστες τιμές). Είναι επίσης δυνατό να σχεδιαστεί μια διεπαφή ερωτήματος στην οποία χρησιμοποιούνται παραδείγματα σχετικών στοιχείων ως στόχοι. Ωστόσο, είναι πιο φυσικό να προσδιορίζονται οι επιθυμητές ιδιότητες στην αρχική διεπαφή ερωτήματος. Μια συνάρτηση ομοιότητας χρησιμοποιείται για την ανάκτηση των παραδειγμάτων που μοιάζουν περισσότερο με τον καθορισμένο από τον χρήστη στόχο. Για παράδειγμα, αν δεν βρεθούν σπίτια που να προσδιορίζουν επακριβώς τις απαιτήσεις του χρήστη, τότε η συνάρτηση ομοιότητας χρησιμοποιείται για την ανάκτηση και την κατάταξη των στοιχείων που είναι όσο το δυνατόν πιο παρόμοια με το ερώτημα του χρήστη. Συνεπώς, σε αντίθεση με τα συστήματα που βασίζονται σε περιορισμούς, το πρόβλημα της ανάκτησης κενών συνόλων δεν αποτελεί ζήτημα στα συστήματα που βασίζονται σε περιπτώσεις.[37]

Υπάρχουν επίσης ουσιαστικές διαφορές μεταξύ μιας σύστασης που βασίζεται σε περιορισμούς και μιας που βασίζεται σε περιπτώσεις όσον αφορά τον τρόπο με τον οποίο βελτιώνετε τα αποτελέσματα. Συστήματα βασισμένα σε περιορισμούς χρησιμοποιούν χαλάρωση, τροποποίηση και αυστηροποίηση των απαιτήσεων για τη βελτίωση των αποτελεσμάτων. Τα πρώτα συστήματα βασισμένα σε περιπτώσεις υποστήριζαν την επαναλαμβανόμενη τροποποίηση των απαιτήσεων του ερωτήματος του χρήστη έως ότου μέχρι να βρεθεί μια κατάλληλη λύση. Στα μεταγενέστερα, αναπτύχθηκε η μέθοδος της κριτικής. Η γενική ιδέα της κριτικής είναι ότι οι χρήστες μπορούν να επιλέξουν ένα ή περισσότερα από τα ανακτηθέντα αποτελέσματα και να καθορίσουν περαιτέρω ερωτήματα της ακόλουθης μορφής [37]
‘’Δώσε μου μου περισσότερα αντικείμενα όπως το X, αλλά να διαφέρουν ως προς το(α) χαρακτηριστικό(α) Υ σύμφωνα με την καθοδήγηση Z‘’

Υπάρχει σημαντική διαφοροποίηση όσον αφορά την επιλογή ενός ή περισσότερων χαρακτηριστικών για τροποποίηση και τον τρόπο με τον οποίο καθορίζεται η καθοδήγηση για την τροποποίηση των χαρακτηριστικών. Ο κύριος στόχος της κριτικής είναι να υποστηρίξει τη διαδραστική περιήγηση στο χώρο των

στοιχείων, όπου ο χρήστης σταδιακά αντιλαμβάνεται τις περαιτέρω επιλογές που έχει στη διάθεσή του. Η διαδραστική περιήγηση στο χώρο των στοιχείων έχει το πλεονέκτημα ότι είναι μια διαδικασία μάθησης για τον χρήστη κατά τη διαδικασία επαναληπτικής διατύπωσης ερωτημάτων. Συχνά είναι δυνατόν ότι μέσω της επαναλαμβανόμενης και διαδραστικής εξερεύνησης, ο χρήστης μπορεί να είναι σε θέση να καταλήξει σε στοιχεία στα οποία δεν θα μπορούσε να φτάσει στην αρχή.[34]



Εικόνα 5 Ένα υποθετικό παράδειγμα μιας διεπαφής χρήστη σε ένα σύστημα συστάσεων βασισμένο σε περιπτώσεις[34]

Για παράδειγμα, αναλογιστείτε το παράδειγμα αγοράς κατοικίας όπως ο παραπάνω πίνακας. Ο χρήστης μπορεί αρχικά να καθορίσει την επιθυμητή τιμή, τον αριθμό των υπνοδωματίων και την επιθυμητή τοποθεσία. Εναλλακτικά, ο χρήστης μπορεί να προσδιορίσει μια διεύθυνση-στόχο για να δώσει ένα παράδειγμα ενός πιθανού σπιτιού που μπορεί να τον ενδιαφέρει. Το επάνω τμήμα της διεπαφής απεικονίζει τον προσδιορισμό των χαρακτηριστικών του στόχου, ενώ το κάτω τμήμα της διεπαφής απεικονίζει τον προσδιορισμό μιας διεύθυνσης στόχου. Η τελευταία προσέγγιση είναι χρήσιμη σε τομείς όπου οι χρήστες δυσκολεύονται περισσότερο να προσδιορίσουν τεχνικά άγνωστα χαρακτηριστικά.. Ένα παράδειγμα θα μπορούσε να

είναι η περίπτωση των ψηφιακών φωτογραφικών μηχανών, όπου είναι πιο δύσκολο να καθοριστούν επακριβώς όλα τα τεχνικά χαρακτηριστικά για έναν μη ειδικό. Ως εκ τούτου, ένας χρήστης μπορεί να προσδιορίσει τη φωτογραφική μηχανή του φίλου του ως περίπτωση-στόχο, αντί να προσδιορίσει όλα τα τεχνικά χαρακτηριστικά.[34]

Μέσω της επαναλαμβανόμενης κριτικής, ο χρήστης μπορεί μερικές φορές να καταλήξει σε ένα τελικό αποτέλεσμα που είναι αρκετά διαφορετικό από την αρχική προδιαγραφή του ερωτήματος. Εξάλλου, είναι συχνά δύσκολο για έναν χρήστη να διατυπώσει όλα τα επιθυμητά χαρακτηριστικά του από την αρχή. Για παράδειγμα, ο χρήστης μπορεί να μην γνωρίζει ένα αποδεκτό σημείο τιμής για τα επιθυμητά χαρακτηριστικά του σπιτιού στην αρχή της διαδικασίας αναζήτησης. Αυτή η διαδραστική προσέγγιση γεφυρώνει το χάσμα μεταξύ της αρχικής κατανόησης και της διαθεσιμότητας των στοιχείων. Μερικές φορές είναι επίσης δυνατό για τον χρήστη να καταλήξει σε ένα κενό σύνολο συστάσεων μέσω επαναλαμβανόμενης μείωσης του συνόλου υποψηφίων. Σημειώστε ότι αυτό διαφέρει από τα συστήματα που βασίζονται σε περιορισμούς, όπου ο χρήστης έχει επίσης τη δυνατότητα να χαλαρώσει το τρέχον σύνολο των απαιτήσεών του για να διευρύνει το σύνολο των αποτελεσμάτων. Ο λόγος για αυτή τη διαφορά είναι ότι τα συστήματα που βασίζονται σε περιπτώσεις μειώνουν γενικά το αριθμό των υποψηφίων από τον ένα κύκλο στον επόμενο, ενώ τα συστήματα που βασίζονται σε περιορισμούς δεν το κάνουν.[34]

Για να λειτουργήσει αποτελεσματικά ένα σύστημα συστάσεων βασισμένο σε περιπτώσεις, υπάρχουν δύο κρίσιμες πτυχές του συστήματος που πρέπει να σχεδιαστούν αποτελεσματικά:

1. Δείκτης ομοιότητας: Ο αποτελεσματικός σχεδιασμός των μετρικών ομοιότητας είναι πολύ σημαντικός για την ανάκτηση σχετικών αποτελεσμάτων. Η σημασία των διαφόρων χαρακτηριστικών πρέπει να ενσωματωθεί κατάλληλα στη συνάρτηση ομοιότητας για να λειτουργήσει το σύστημα αποτελεσματικά.

2. Μέθοδοι κριτικής: Η διαδραστική εξερεύνηση του χώρου των στοιχείων υποστηρίζεται με τη χρήση μεθόδων κριτικής. Μια ποικιλία διαφορετικών μεθόδων κριτικής είναι διαθέσιμες για την υποστήριξη διαφορετικών στόχων εξερεύνησης.

Recommendations with Deep Learning (Συστάσεις με βαθιά μάθηση)

Η βαθιά μάθηση είναι μια τεχνική μηχανικής μάθησης που διδάσκει στους υπολογιστές να κάνουν αυτό που είναι φυσικό για τους ανθρώπους: να μαθαίνουν από το παράδειγμα. Η βαθιά μάθηση είναι μια βασική τεχνολογία πίσω από τα αυτοκίνητα χωρίς οδηγό, που τους επιτρέπει να αναγνωρίζουν ένα σήμα στοπ ή να διακρίνουν έναν πεζό από μια κολόνα. Είναι το κλειδί για τον φωνητικό έλεγχο σε καταναλωτικές συσκευές όπως τηλέφωνα, tablet, τηλεοράσεις και ηχεία hands-free. Η βαθιά εκμάθηση λαμβάνει μεγάλη προσοχή τελευταία και για καλό λόγο. Επιτυγχάνει αποτελέσματα που δεν ήταν εφικτά πριν.[38]

Στη βαθιά μάθηση, ένα μοντέλο υπολογιστή μαθαίνει να εκτελεί εργασίες ταξινόμησης απευθείας από εικόνες, κείμενο ή ήχο. Τα μοντέλα βαθιάς μάθησης μπορούν να επιτύχουν κορυφαία ακρίβεια, που μερικές φορές ξεπερνά τις επιδόσεις σε ανθρώπινο επίπεδο. Τα μοντέλα εκπαιδεύονται χρησιμοποιώντας ένα μεγάλο σύνολο επισημασμένων δεδομένων και αρχιτεκτονικές νευρωνικών δικτύων που περιέχουν πολλά επίπεδα. Η τεχνητή νοημοσύνη και η "βαθιά μάθηση" με τεχνητά νευρωνικά δίκτυα είναι είναι στο επίκεντρο τον περισσοτέρων ερευνών, οπότε δεν αποτελεί έκπληξη το γεγονός ότι υπάρχουν πολλές έρευνες γύρω από την εφαρμογή της βαθιάς μάθησης στα συστήματα συστάσεων.Η παραγοντοποίησης πινάκων μπορεί να υλοποιηθεί με ένα νευρωνικό δίκτυο, όπως επισης μπορει να επιτευχθεί μέσω της βαθιάς μάθησης. Η SVD(Singular Value Decomposition) φαίνεται τρομακτική στο πόσο καλή είναι, με τον ίδιο τρόπο που η βαθιά μάθηση κάνει το ίδιο. Υπάρχει ένας πολύ καλός λόγος γι' αυτό και οι δύο τεχνικές μπορούν να έχουν τα ίδια αποτελέσματα. Όμως, η βαθιά μάθηση ανοίγει επίσης εντελώς νέες προσεγγίσεις για την πραγματοποίηση συστάσεων που αξίζει να εξερευνήσουμε.[22]

Ένα νευρωνικό δίκτυο μπορει να εκπαιδευτεί με αξιολογήσεις ή αγορές χρηστών και να χρησιμοποιηθεί για να κάνει συστάσεις. Η βαθιά μάθηση μπορεί να είναι πολύ καλή στην αναγνώριση μοτίβων παρόμοια με τον εγκέφαλός μας. Είναι καλή σε πράγματα όπως η αναγνώριση εικόνων και η πρόβλεψη ακολουθιών γεγονότων. Τα νευρωνικά δίκτυα είναι θεμελιωδώς λειτουργίες πινάκων, και υπάρχουν ήδη καθιερωμένες λειτουργίες παραγοντοποίησης πινάκων για συστήματα συστάσεων που κάνουν κάτι παρόμοιο. Έτσι, η παραγοντοποίηση πινάκων μπορεί να μοντελοποιεί ως νευρωνικό δίκτυο. Η Amazon, για παράδειγμα, έχει ανοίξει ένα σύστημα που

ονομάζεται "Destiny" (δηλαδή DSSTNE), το οποίο επιτρέπει να αναπτιχθουν τεράστια νευρωνικά δίκτυα που ασχολούνται με αραιά δεδομένα σε ένα cluster αποτελεσματικά. Ισχυρίζονται ότι το χρησιμοποιούν αυτό εσωτερικά για τα δικά τους συστήματα συστάσεων. Υπάρχουν επίσης τρόποι για τη χρήση Tensorflow σε μια cluster και να γινει εκμεταλευσει ένος ολόκληρου στόλου GPU. Επισης υπάρχουν πάντα έρευνες για νέες τοπολογίες νευρωνικών δικτύων που μπορούν να οδηγήσουν σε νέες ιδέες για το πώς να γίνουν καλύτερες συστάσεις με τη χρήση τους. Σε ορισμένες περιπτώσεις, έχει αποδειχθεί ότι οι προσεγγίσεις που χρησιμοποιούν νευρωνικά δίκτυα υπερτερούν σε απόδοση ,έστω και με μικρή διαφορά.[22]

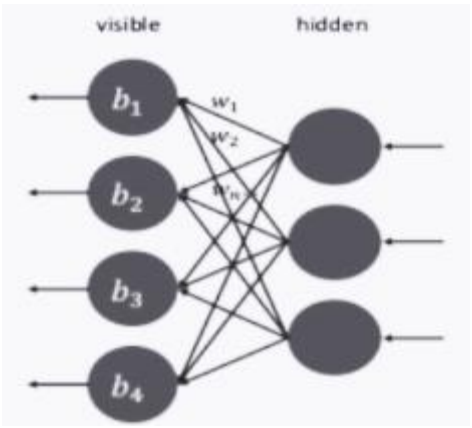
Restricted Boltzmann Machines

Οι περιορισμένες μηχανές Boltzmann (RBM) είναι πιθανότητά γραφικά μοντέλα που μπορούν να ερμηνευθούν ως στοχαστικά νευρωνικά δίκτυα. Η αύξηση της υπολογιστικής ισχύος και η ανάπτυξη ταχύτερων αλγορίθμων μάθησης τις έχουν καταστήσει εφαρμόσιμες σε σχετικά προβλήματα μηχανικής μάθησης. Πρόσφατα προσέλκυσαν μεγάλη προσοχή αφού προτάθηκαν ως δομικά στοιχεία πολυεπίπεδων συστημάτων μάθησης που ονομάζονται βαθιά δίκτυα πεποιθήσεων.[39] Κοιτάζοντας την περίπτωση του Netflix prize, τα κύρια πράγματα που έμαθε το Netflix ήταν ότι η παραγοντοποιήσει πινάκων και τα RBM είχαν την καλύτερη απόδοση, όπως μετρήθηκε από το RMSE(root-mean-square deviation), και οι βαθμολογίες τους ήταν σχεδόν πανομοιότυπες. Και πάλι, αυτό δεν θα πρέπει να μας εκπλήσσει , αφού μπορεί να γίνει παραγοντοποιήσει πινάκων σε ένα νευρωνικό δίκτυο. Όμως, διαπίστωσαν ότι συνδυάζοντας την παραγοντοποιήσει πινάκων με τα RBM, μαζί παρείχαν ακόμη καλύτερα αποτελέσματα, πήγαν από ένα RMSE 8,9 σε 8,8. Πριν από μερικά χρόνια, το Netflix επιβεβαίωσε ότι εξακολουθούν να χρησιμοποιούν τα RBM ως μέρος του συστήματος συστάσεων τους που βρίσκεται σε παραγωγή.

Τα RBM είναι ένα από τα απλούστερα νευρωνικά δίκτυα αποτελουνται απο δύο επίπεδα, ένα ορατό επίπεδο και ένα κρυφό επίπεδο. Εκπαιδεύονται τροφοδοτώντας τα δεδομένα εκπαίδευσής στο ορατό στρώμα σε ένα εμπρόσθιο πέρασμα και εκπαιδεύοντας την βαρύτητα και τις προκαταλήψεις μεταξύ τους κατά τη διάρκεια της backpropagation (οπισθοδιάδοση). Μια συνάρτηση ενεργοποίησης, όπως η ReLU (Rectified Linear Units), χρησιμοποιείται για την παραγωγή της εξόδου από κάθε κρυφό νευρώνα. Ονομάζονται "περιορισμένα" επειδή οι νευρώνες στο ίδιο επίπεδο δεν

μπορούν να επικοινωνήσουν άμεσα μεταξύ τους, υπάρχουν μόνο συνδέσεις μεταξύ των δύο διαφορετικών στιβάδων. Οι μηχανές RBM δεν εφευρέθηκαν από κάποιον Boltzmann το όνομα αναφέρεται στη συνάρτηση κατανομής Boltzmann που χρησιμοποιούσαν για τη συνάρτηση δειγματοληψίας τους. Οι RBM αποδίδονται στην πραγματικότητα στον Geoffrey Hinston, ο οποίος ήταν καθηγητής στο Πανεπιστήμιο Carnegie Mellon εκείνη την εποχή η ιδέα χρονολογείται από το 1985.

Οι RBM εκπαιδεύονται κάνοντας μια πάσα προς τα εμπρός, και στη συνέχεια ένα αντίστροφο πέρασμα, όπου οι είσοδοι ανακατασκευάζονται. Αυτό το κάνουναι επαναληπτικά σε πολλές "εποχές", όπως ακριβώς όταν εκπαιδεύεται ένα βαθύ νευρωνικό δίκτυο, μέχρι να συγκλίνει σε ένα σύνολο βαρυτήτων και προκαταλήψεων που ελαχιστοποιεί το σφάλμα. Ας ρίξουμε μια πιο προσεκτική ματιά σε αυτό το "αντίστροφο" πέρασμα. [39]



Εικόνα 6 Αντίστροφο πέρασμα rbm [22]

Κατά τη διάρκεια του αντίστροφου περάσματος, προσπαθούμε να ανακατασκευάσουμε την αρχική είσοδο ανατροφοδοτώντας την έξοδο του εμπρόσθιου περάσματος μέσω του κρυμμένου στρώματος, και να δούμε τι τιμές θα πάρουμε τελικά από το ορατό στρώμα. Δεδομένου ότι αυτές οι βαρύτητες είναι αρχικά τυχαίες, μπορεί να υπάρχει μεγάλη διαφορά μεταξύ των εισόδων με τις οποίες ξεκινήσαμε και αυτών που ανακατασκευάζουμε. Κατά τη διαδικασία, καταλήγουμε με ένα άλλο σύνολο όρων μεροληψίας, αυτή τη φορά στο ορατό στρώμα. Έτσι, μοιραζόμαστε τα βάρη μεταξύ των δύο περασμάτων προς τα εμπρός και προς τα πίσω, αλλά έχουμε δύο σύνολα μεροληψίας - την κρυφή μεροληψία που χρησιμοποιείται στο εμπρόσθιο πέρασμα και την ορατή μεροληψία που χρησιμοποιείται σε αυτό το αντίστροφο πέρασμα.[22]

Μπορούμε στη συνέχεια να μετρήσουμε το σφάλμα στο οποίο καταλήγουμε και να χρησιμοποιήσουμε αυτές τις πληροφορίες για να προσαρμόσουμε λίγο τα βάρη κατά την επόμενη επανάληψη για να προσπαθήσουμε να ελαχιστοποιήσουμε το σφάλμα.[22]

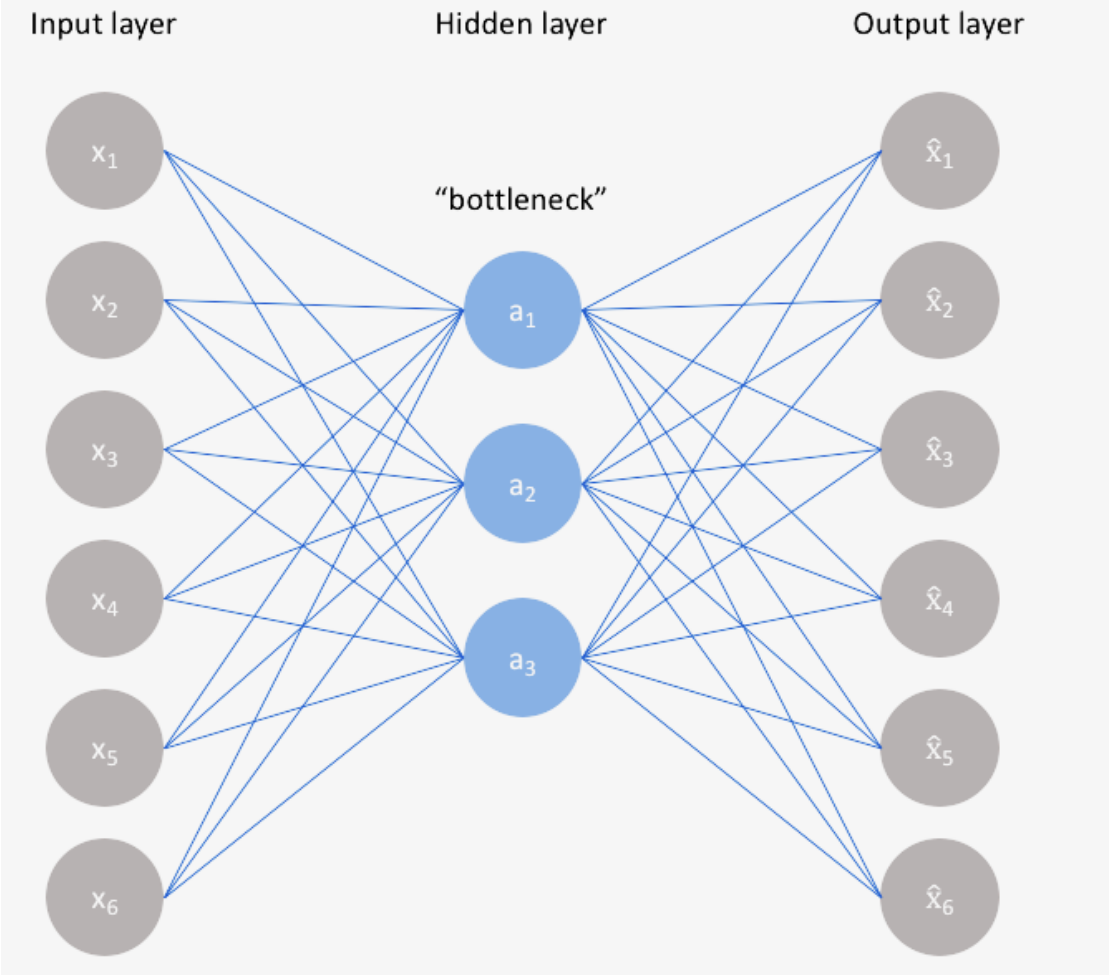
Προσαρμογή ενός RBM σε σύσταση ταινιών με δεδομένα αξιολόγησης 5 αστέρων, απαιτεί μερικές αλλαγές στη γενική αρχιτεκτονική RBM που μόλις περιγράψαμε. Η γενική ιδέα είναι να χρησιμοποιήσουμε κάθε μεμονωμένο χρήστη στα δεδομένα εκπαίδευσής μας ως ένα σύνολο εισόδων στο RBM για να βοηθήσουν στην εκπαίδευσή του. Έτσι, επεξεργαζόμαστε κάθε χρήστη ως μέρος μιας δέσμης κατά τη διάρκεια εκπαίδευσης, εξετάζοντας τις αξιολογήσεις του για κάθε ταινία που βαθμολόγησε. Έτσι, οι ορατοι κόμβοι αντιπροσωπεύουν τις αξιολογήσεις ενός συγκεκριμένου χρήστη για κάθε ταινία, και προσπαθούμε να μάθουμε τα βάρη και προκαταλήψεις για να μπορέσουμε να ανακατασκευάσουμε τις αξιολογήσεις για τα ζεύγη χρήστη/ταινίας που δεν γνωρίζουμε ακόμα. Πρώτα απ' όλα, οι ορατές μονάδες μας δεν είναι απλοί κόμβοι που δέχονται μία μόνο είσοδο. Οι αξιολογήσεις είναι πραγματικά κατηγορικά δεδομένα, οπότε στην πραγματικότητα θέλουμε να αντιμετωπίσουμε κάθε ατομική αξιολόγηση ως πέντε κόμβους, έναν για κάθε πιθανή τιμή αξιολόγησης. Έτσι, ας πούμε ότι η πρώτη βαθμολογία που έχουμε στα δεδομένα εκπαίδευσης είναι μια βαθμολογία 5 αστέρων. αναπαρίσταται ως τέσσερις κόμβοι με τιμή 0 και ένας με τιμή 1, όπως αναπαρίσταται εδώ. Στη συνέχεια έχουμε μερικές αξιολογήσεις που λείπουν, για ζεύγη χρηστών/αντικειμένων που είναι άγνωστα και πρέπει να προβλεφθούν. Στη συνέχεια έχουμε μια βαθμολογία 3 αστέρων, που αναπαρίσταται με ένα 1 στην τρίτη θέση.[22]

Όταν τελειώσουμε με την εκπαίδευση του RBM, θα έχουμε ένα σύνολο βαρών και προκαταλήψεων που θα μας επιτρέψει να ανακατασκευάσουμε τις αξιολογήσεις για κάθε χρήστη. Έτσι, για να το χρησιμοποιήσουμε για να προβλέψουμε τις αξιολογήσεις ενός νέου χρήστη, απλά το εκτελούμε ξανά χρησιμοποιώντας τις γνωστές αξιολογήσεις του χρήστη που μας ενδιαφέρει. Τις εκτελούμε στο εμπρόσθιο πέρασμα, και στη συνέχεια ξανά πίσω στο οπίσθιο πέρασμα, για να καταλήξουμε σε ανακατασκευασμένες τιμές αξιολόγησης για αυτόν τον χρήστη. Στη συνέχεια, μπορούμε να εκτελέσουμε το softmax (κανονικοποιημένη εκθετική συνάρτηση) σε κάθε ομάδα 5 τιμών αξιολόγησης για να μεταφράσουμε την έξοδο σε μια αξιολόγηση 5 αστέρων για κάθε στοιχείο. Αλλά και πάλι, το μεγάλο πρόβλημα είναι ότι τα

δεδομένα που έχουμε είναι αραιά. Εάν εκπαιδεύουμε ένα RBM σε κάθε πιθανό συνδυασμό χρηστών και ταινιών, τα περισσότερα από αυτά τα δεδομένα θα λείπουν, επειδή οι περισσότερες ταινίες δεν έχουν βαθμολογηθεί καθόλου από έναν συγκεκριμένο χρήστη. Θέλουμε όμως να προβλέψουμε τις αξιολογήσεις των χρηστών για κάθε ταινία, οπότε πρέπει να αφήσουμε χώρο για όλες αυτές. Αυτό σημαίνει ότι αν έχουμε N ταινίες, καταλήγουμε σε N * 5 ορατούς κόμβους, και για κάθε δεδομένο χρήστη, οι περισσότεροι από αυτούς είναι απροσδιόριστοι και κενοί. Το αντιμετωπίζουμε αυτό αποκλείοντας από την επεξεργασία τυχόν ελλείπουσες αξιολογήσεις κατά την εκπαίδευση του RBM. Αυτό είναι κάπως δύσκολο πράγμα, επειδή τα περισσότερα πλαίσια που έχουν δημιουργηθεί για βαθιά μάθηση, όπως το tensorflow, υποθέτουν ότι θέλετε να επεξεργάζεστε τα πάντα παράλληλα, όλη την ώρα. Τα αραιά δεδομένα δεν είναι κάτι που φτιάχτηκαν αρχικά. Καθώς εκπαιδεύουμε το RBM με τις γνωστές αξιολογήσεις ενός συγκεκριμένου χρήστη, προσπαθούμε να μάθουμε μόνο τα βάρη και τις προκαταλήψεις που χρησιμοποιούνται για τις ταινίες που ο χρήστης βαθμολόγησε πραγματικά. Καθώς επαναλαμβάνουμε την εκπαίδευση σε όλους τους άλλους χρήστες, συμπληρώνοντας τα άλλα βάρη.[22]

Autoencoders

Οι autoencoders είναι μια τεχνική μάθησης χωρίς επίβλεψη στην οποία αξιοποιούμε τα νευρωνικά δίκτυα για το έργο της μάθησης αναπαράστασης. Συγκεκριμένα, σχεδιάσουμε μια αρχιτεκτονική νευρωσικού δικτύου έτσι ώστε να επιβάλλουμε ένα σημείο συμφόρησης στο δίκτυο το οποίο επιβάλλει μια συμπιεσμένη αναπαράσταση γνώσης της αρχικής εισόδου. Αν τα χαρακτηριστικά εισόδου ήταν το καθένα ανεξάρτητο μεταξύ του, αυτή η συμπίεση και η επακόλουθη ανακατασκευή θα ήταν ένα πολύ δύσκολο έργο. Ωστόσο, εάν υπάρχει κάποιου είδους δομή στα δεδομένα (π.χ. συσχετίσεις μεταξύ των χαρακτηριστικών εισόδου), αυτή η δομή μπορεί να μαθευτεί και, κατά συνέπεια, να αξιοποιηθεί όταν εξαναγκάζεται η είσοδος να περάσει από τη στενωπό του δικτύου.[40]



Μια ομάδα από το Εθνικό Πανεπιστήμιο της Αυστραλίας δημοσίευσε μια εργασία με τίτλο "AutoRec: Autoencoders Meet Collaborative Filtering", και χρησιμοποίησαν την τοπολογία που παραπάνω. Έχει τρία επίπεδα: ένα στρώμα εισόδου στο κάτω μέρος που περιέχει μεμονωμένες αξιολογήσεις, ένα κρυφό στρώμα και ένα στρώμα εξόδου που μας δίνει τις προβλέψεις μας. Ένας πίνακας βαρών μεταξύ των στρωμάτων διατηρείται σε κάθε περίπτωση αυτού του δικτύου, καθώς και ένας κόμβος προκατάληψης τόσο για το κρυφό στρώμα όσο και για το στρώμα εξόδου. Στην εργασία, εκπαιδεύτηκε το δίκτυο μία φορά ανά στοιχείο, τροφοδοτώντας τις αξιολογήσεις από κάθε χρήστη για το συγκεκριμένο στοιχείο στο επίπεδο εισόδου. Μια σιγμοειδής συνάρτηση ενεργοποίησης χρησιμοποιήθηκε στην έξοδο. Συνολικά, πρόκειται για μια αρκετά απλή προσέγγιση καθος ανέφεραν ελαφρώς καλύτερα αποτελέσματα σε σύγκριση με τη χρήση ενός RBM. Αλλά η εφαρμογή είναι λίγο διαφορετική διότι τα RBM είχαν απλώς ξεχωριστούς όρους μεροληψίας για κάθε πέρασμα, ενώ εδώ έχουμε επίσης ένα ολόκληρο ξεχωριστό σύνολο βαρών για να δουλέψουμε.[22]

Αυτό το είδος αρχιτεκτονικής έχει επίσης το πλεονέκτημα ότι είναι πολύ πιο εύκολο να υλοποίηση σε σύγχρονα πλαίσια όπως το Tensorflow ή το Keras. Αλλά υπάρχει ακόμα ένα πρόβλημα - η αραιότητα των δεδομένων με τα οποία εργαζόμαστε. Στην εργασία, αναφέρουν εν συντομία ότι "εξετάζουμε μόνο τη συμβολή των παρατηρούμενων αξιολογήσεων". Έτσι, φρόντισαν να επεξεργαστούν κάθε διαδρομή μέσω αυτού του νευρωνικού δίκτυο ξεχωριστά, διαδίδοντας μόνο πληροφορίες από αξιολογήσεις που πραγματικά υπάρχουν στα δεδομένα εκπαίδευσης, και αγνοώντας τη συμβολή από κόμβους εισόδου, δεν υπάρχει κανένας απλός τρόπος να περιορίσετε την αξιολόγηση. Αυτό εξακολουθεί να είναι δύσκολο να γίνει στο Tensorflow δεν υπάρχει κανένας απλός τρόπος να περιοριστεί η αλυσίδα του πίνακα πολλαπλασιασμών και προσθέσεων που απαιτούνται για την υλοποίηση ενός νευρωνικού δικτύου σε μόνο τους κόμβους εισόδου με τα πραγματικά δεδομένα σε αυτούς. Οποιαδήποτε υλοποίηση του αυτό χρησιμοποιώντας το Tensorflow ή το Keras απλά αγνοεί αυτό το πρόβλημα.[22]

Αυτή η αρχιτεκτονική αναφέρεται ως "autoencoder.". Η πράξη της δημιουργίας των βαρών και των προκαταλήψεων μεταξύ της εισόδου και του κρυφού στρώματος αναφέρεται ως κωδικοποίηση της εισόδου - κωδικοποιούμε τα μοτίβα της εισόδου ως ένα σύνολο βαρών σε αυτό το κρυφό στρώμα. Στη συνέχεια, καθώς ανακατασκευάζουμε την έξοδο στα βάρη μεταξύ του κρυφού στρώματος και του στρώματος εξόδου, την αποκωδικοποιούμε. Έτσι, το πρώτο σύνολο βαρών είναι το στάδιο κωδικοποίησης και το δεύτερο σύνολο είναι το στάδιο αποκωδικοποίησης. Εννοιολογικά, αυτό δεν είναι πραγματικά διαφορετικό από τα RBM.

Κεφάλαιο 3 – Δημιουργία ενός Συστήματος συστάσεων με αλγόριθμο μηχανικής μαθήσεις.

Εισαγωγή

Το Σύστημα Συστάσεων που αναπτύχθηκε αξιοποιεί τεχνολογία μηχανικής μάθησης για να προσφέρει εξατομικευμένες συστάσεις βιβλίων στους χρήστες της έχει σχεδιαστεί για να κατανοεί και να προσαρμόζεται στις προτιμήσεις των ατόμων, βελτιώνοντας έτσι την εμπειρία ανακάλυψης νέων βιβλίων.

Βασικά Χαρακτηριστικά:

- Αλληλεπίδραση με τους Χρήστες: Το σύστημα παρέχει μια φιλική προς τον χρήστη διεπαφή όπου οι χρήστες μπορούν να δημιουργήσουν λογαριασμούς, να βαθμολογήσουν τα βιβλία που έχουν διαβάσει και να δουν τις βαθμολογίες τους.. Κάθε χρήστης λαμβάνει ένα μοναδικό αναγνωριστικό κατά τη δημιουργία λογαριασμού, διατηρώντας ένα εξατομικευμένο χώρο για τις αλληλεπιδράσεις του κάθε χρήστη.

- Εξατομικευμένες Προτάσεις: Στην καρδιά του συστήματος βρίσκεται ένας αλγόριθμος μηχανικής μάθησης που δημιουργεί εξατομικευμένες προτάσεις βιβλίων. Αυτές οι προτάσεις δεν είναι στατικές, αλλά δυναμικές, προσαρμοσμένες στις ατομικές προτιμήσεις του χρήστη και στις βαθμολογίες άλλων χρηστών με παρόμοιες προτιμήσεις..

- Ενημερώσεις του Μοντέλου: Για να εξασφαλίσει την εγκυρότητα των συστάσεων του, το σύστημα περιλαμβάνει έναν μηχανισμό για την περιοδική ενημέρωση του μοντέλου πρότασης. Καθώς προστίθενται νέες βαθμολογίες χρηστών, το μοντέλο εκπαιδεύεται ξανά για να ενσωματώσει αυτά τα νέα δεδομένα, διατηρώντας έτσι την ακρίβεια και την εγκυρότητα των προτάσεων του συστήματος.

Περίληψη

Το Σύστημα συστάσεων που αναπτύχτηκε αποτελείτε ουσιαστικά απο δύο κύρια μέρη, το καθένα αναλαμβάνει ένα συγκεκριμένο κομμάτι του συστήματος: τη διεπαφή του χρήστη με το σύστημα και τη δημιουργία και εκπαίδευση του μοντέλου.

Διεπαφή του χρήστη: Αυτό το μέρος του συστήματος είναι υπεύθυνο για την αλληλεπίδραση με τον χρήστη. Παρέχει μια διεπαφή όπου οι χρήστες μπορούν να αξιολογήσουν βιβλία, να δουν τις αξιολογήσεις τους και να λαμβάνουν εξατομικευμένες προτάσεις βιβλίων. Αυτό υλοποιείται κυρίως από το πλαίσιο Flask, μια δημοφιλής επιλογή για την ανάπτυξη εφαρμογων σε Python. Το κύριο αρχείο εφαρμογής (flask_app.py) δημιουργεί την εφαρμογή, ορίζει διαδρομές για τις ιστοσελίδες και χειρίζεται αιτήσεις και απαντήσεις. Η διεπαφή χρήστη αναπτύσσεται στον φάκελο templates/, οπου υπάρχουν αρχεία html τα οποία αναπαράγονται από το Flask και εμφανίζονται στους χρήστες. Όλες οι δραστηριότητες των χρηστών, όπως η δημιουργία λογαριασμού, η αξιολόγηση βιβλίων και η προβολή προτάσεων, λαμβάνουν χώρα μέσω αυτής της διεπαφής ιστοσελίδας.

Δημιουργία και Εκπαίδευση Μοντέλου: Αυτό το μέρος βρίσκεται στον πυρήνα του συστήματος συστάσεων. Περιλαμβάνει τη δημιουργία και την εκπαίδευση ενός μοντέλου μηχανικής μάθησης που μπορεί να δημιουργήσει προτάσεις βιβλίων βασιζόμενο στις αξιολογήσεις των χρηστών. Αυτή η διαδικασία γίνεται μέσο του notebook Jupyter, ένα εργαλείο που χρησιμοποιείται ευρέως στην επιστήμη δεδομένων και τη μηχανική μάθηση για τη διαδραστικότητα του και την ευκολία της απεικόνισης. Το notebook περιέχει κώδικα Python για την επεξεργασία δεδομένων, τη δημιουργία του μοντέλου και την εκπαίδευση του. Το αρχειο που περιεχει τον κωδικα για την δημιουργια του μοντελου είναι το book-rec_DL_gr.ipynb. Στο οποιο γινεται η εκπαιδεύσει σε ένα μοντέλο για να προβλέψει τις αξιολογήσεις των χρηστών και αποθηκεύει το καλύτερο μοντέλο για μελλοντική χρήση. Αυτά τα μοντέλα αποθηκεύονται στον φάκελο models/ και χρησιμοποιούνται αργότερα από την εφαρμογή Flask για τη δημιουργία προτάσεων βιβλίων.

Κύριες Βιβλιοθήκες

Διάφορες βιβλιοθήκες της Python είναι απαραίτητες για την επιτυχή λειτουργία του συστήματος:

- Python 3: Αυτή είναι η βασική γλώσσα με την οποία έχει αναπτυχθεί το σύστημα. Η Python είναι ευρέως αναγνωρισμένη για την απλότητά της, την αναγνωσιμότητα και την εκτεταμένη υποστήριξη που παρέχει για επιστημονικό υπολογισμό και μηχανική μάθηση.
- Flask: Το Flask είναι ένα ελαφρύ πλαίσιο εφαρμογής ιστού που χρησιμοποιείται για τη δημιουργία του φιλικού προς τον χρήστη διεπαφή του συστήματος. Χειρίζεται την κατεύθυνση των αιτημάτων και εξυπηρετεί τις HTML προτύπους για τη διεπαφή χρήστη.
- Keras: Η Keras είναι μια ανώτερη διεπαφή νευρωνικών δικτύων (API) που έχει γραφεί σε Python και μπορεί να λειτουργήσει πάνω από το TensorFlow. Χρησιμοποιείται σε αυτό το σύστημα για τη δημιουργία και εκπαίδευση του μοντέλου συστασεων. Το Keras παρέχει μια εύχρηστη διεπαφή για την ανάπτυξη και εκπαίδευση μοντέλων βαθιάς μάθησης, που είναι ιδανική για τον αλγόριθμο συνεργατικής φιλτράρισες που χρησιμοποιείται στο σύστημα συστασεων.
- Pandas: Η Pandas χρησιμοποιείται για την επεξεργασία και ανάλυση δεδομένων. Είναι ιδιαίτερα χρήσιμη για τη διαχείριση αρχείων CSV που περιέχουν δεδομένα χρηστών, βιβλίων και αξιολογήσεων.
- Numpy: Η Numpy είναι μια βιβλιοθήκη για τη γλώσσα προγραμματισμού Python που προσθέτει υποστήριξη για μεγάλους, πολυδιάστατους πίνακες και πίνακες, μαζί με μια μεγάλη συλλογή από υψηλού επιπέδου μαθηματικές λειτουργίες. Χρησιμοποιείται για αριθμητικές λειτουργίες κατά την εκπαίδευση του μοντέλου.
- Scikit-learn: Το scikit-learn χρησιμοποιείται για εργασίες μηχανικής μάθησης, όπως Προεπεξεργασία δεδομένων, αξιολόγηση μοντέλου και άλλες εργασίες. Χρησιμοποιείται για εργασίες όπως η διαίρεση των δεδομένων σε σύνολα εκπαίδευσης και δοκιμής..

Μοντέλο Μηχανικής Μάθησης

Το μοντέλο ανήκει στην κατηγορία των συστημάτων συστάσεων συνεργατικής φιλτράρισες(collaborative- filtering) ποιο συγκεκριμένα το μοντέλο ανηκει στην υποκατηγορία (Embedding-based Recommender Systems). Αυτό το είδος μοντέλων χρησιμοποιεί ενσωματώσεις για να αναπαραστήσει τα αντικείμενα (στην περίπτωσή μας, χρήστες και βιβλία) σε έναν χώρο υψηλής διάστασης, έτσι ώστε να μπορούν να αντιληφθούν και να αποθηκεύσουν πληροφορίες σχετικά με τις προτιμήσεις και τα χαρακτηριστικά των αντικειμένων.

Ο αλγόριθμος που χρησιμοποιείται στο μοντέλο είναι ο "Dot Product" (πολλαπλασιασμός πινάκων). Αυτός ο αλγόριθμος υπολογίζει το εσωτερικό γινόμενο μεταξύ των διανυσμάτων των ενσωματώσεων των χρηστών και των βιβλίων. Ουσιαστικά, πολλαπλασιάζει τα αντίστοιχα στοιχεία των διανυσμάτων και συνολικά προκύπτει ένας αριθμός που αντιπροσωπεύει την πρόβλεψη του μοντέλου για την αξιολόγηση που θα έδινε ο χρήστης σε ένα συγκεκριμένο βιβλίο.

Με αυτόν τον τρόπο, το μοντέλο χρησιμοποιεί τις ενσωματώσεις των χρηστών και των βιβλίων για να προβλέψει τις αξιολογήσεις των χρηστών για τα βιβλία. Η εκπαίδευση του μοντέλου στηρίζεται στην ελαχιστοποίηση της απόκλισης μεταξύ των προβλεπόμενων αξιολογήσεων και των πραγματικών αξιολογήσεων στο σύνολο εκπαίδευσης.

Για την εκπαίδευση του μοντέλου, χρησιμοποιείται ένα σύνολο εκπαίδευσης που περιέχει πραγματικές αξιολογήσεις χρηστών για βιβλία. Οι παράμετροι του μοντέλου ενημερώνονται κατά τη διάρκεια της εκπαίδευσης χρησιμοποιώντας έναν αλγόριθμο βελτιστοποίησης. Στην περίπτωση αυτου του μοντέλου, χρησιμοποιείται ο αλγόριθμος Adam ή RMSprop ως βελτιστοποιητής. Ο στόχος είναι να ελαχιστοποιηθεί η απόκλιση (loss) μεταξύ των πραγματικών αξιολογήσεων και των προβλεπόμενων αξιολογήσεων.

Για να παρακολουθείται η επίδοση του μοντέλου κατά την εκπαίδευση και να αποφεύγεται η υπερεκπαίδευση (overfitting), χρησιμοποιείται η μέθοδος του Early Stopping. Με τη βοήθεια του Early Stopping, το μοντέλο παρακολουθείται σε ένα σύνολο επικύρωσης (validation set) και η εκπαίδευση σταματάει όταν η απόκλιση στο σύνολο επικύρωσης αυξάνεται ή δε βελτιώνεται πλέον. Αυτός ο μηχανισμός βοηθάει να επιλεγεί το βέλτιστο σημείο για την εκπαίδευση του μοντέλου, προκειμένου να επιτευχθεί η καλύτερη απόδοση στα ανεξάρτητα δεδομένα.

Δεδομένα εκπαιδεύσεις του αλγορίθμου

Το σύνολο δεδομένων Goodreads, και ειδικότερα το υποσύνολο "goodbooks-10k[41], χρησιμοποιήθηκε για την εκπαίδευση του μοντέλου για το Σύστημα συστάσεων Βιβλίων. Αυτό το σύνολο δεδομένων χρησιμοποιείται ευρέως στον τομέα των συστημάτων προτάσεων βιβλίων και περιλαμβάνει εκτενείς πληροφορίες για τα βιβλία, τις αξιολογήσεις και τις ετικέτες που δημιουργούν οι χρήστες. Αποτελεί μια πολύτιμη πηγή για την εκπαίδευση μοντέλων μηχανικής μάθησης προκειμένου να παρέχονται ακριβείς προτάσεις βιβλίων.

Τα δεδομένα αξιολογήσεων στο σύνολο δεδομένων καταγράφουν τις προτιμήσεις των χρηστών μέσω των αξιολογήσεών τους για διάφορα βιβλία. Κάθε αξιολόγηση περιλαμβάνει αναγνωριστικά για τον χρήστη και το βιβλίο, καθώς και την ίδια την αξιολόγηση σε μια κλίμακα που κυμαίνεται από 1 έως 5. Αναλύοντας αυτά τα δεδομένα, το σύστημα προτάσεων μπορεί να κατανοήσει τις προτιμήσεις του χρήστη και να προσφέρει εξατομικευμένες προτάσεις.

Data scraping(Απόσπαση δεδομένων)

Η τεχνική της απόσπασης δεδομένων (data scraping) χρησιμοποιήθηκε για να εμπλουτιστεί το σύνολο δεδομένων με φωτογραφίες και συνδέσμους ιστοσελίδων, προκειμένου να χρησιμοποιηθούν αργότερα στην εφαρμογή. Με αυτήν την τεχνική, πραγματοποιήθηκε αναζήτηση σε μηχανές αναζήτησης και χρησιμοποιήθηκε η πλατφόρμα Google Custom Search API για να ανακτηθούν φωτογραφίες και συνδέσμοι ιστοσελίδων που σχετίζονται με κάθε βιβλίο στο σύνολο δεδομένων. Με την εμπλουτισμένη αυτή πληροφορία, το σύνολο δεδομένων είναι πλέον έτοιμο να χρησιμοποιηθεί στην εφαρμογή. Γίνεται ανάλυση στην τεχνική στην συνέχεια.

Ανάλυση της εφαρμογής

Η εφαρμογή είναι αναρτημένη στο GitHub: https://github.com/Grecrack/book_rec και είναι πρόσβασιμο απο όλους.

Δόμηση της εφαρμογής

- /Data: Αυτός ο φάκελος περιέχει τα απαραίτητα δεδομένα για την εκπαίδευση του συστήματος προτάσεων και τη φόρτωση των προφίλ χρηστών. Οι περισσότερες από τις αρχειοθετημένες ενότητες είναι σε μορφή CSV.
- /Models: Αυτός ο φάκελος περιέχει τα εκπαιδευμένα μοντέλα που χρησιμοποιούνται για τις προτάσεις βιβλίων. Αυτά τα μοντέλα έχουν εκπαιδευτεί με βάση τα δεδομένα που παρέχονται και χρησιμοποιούνται για τη δημιουργία εξατομικευμένων προτάσεων για τους χρήστες.
- /templates: Αυτός ο φάκελος περιέχει τα πρότυπα HTML που χρησιμοποιούνται από την εφαρμογή Flask για την απεικόνιση του χρήστη. Αυτά τα πρότυπα καθορίζουν τη δομή και τη διάταξη των ιστοσελίδων που οι χρήστες θα δουν κατά την αλληλεπίδρασή τους με το σύστημα προτάσεων.
- flask_app.py: Αυτό είναι το κύριο αρχείο εφαρμογής που δημιουργεί το Flask app και χειρίζεται τις διαδρομές και τη λογική για την παραγωγή προτάσεων βιβλίων. Εισάγει τις απαιτούμενες βιβλιοθήκες όπως το Flask, το Pandas και το μοντέλο προτάσεων. Μέσα σε αυτό το αρχείο, καθορίζετε τις διαδρομές για διάφορες ιστοσελίδες, διαχειρίζεστε τα αιτήματα των χρηστών και χρησιμοποιείτε το μοντέλο προτάσεων για την δημιουργία εξατομικευμένων προτάσεων βιβλίων.
- Το αρχείο `/models/book-rec_DL.ipynb` είναι υπεύθυνο για τη δημιουργία του μοντέλου πρότασης χρησιμοποιώντας μια προσέγγιση βαθιάς μάθησης. Ο παρεχόμενος κώδικας παρουσιάζει τη διαδικασία εκπαίδευσης του μοντέλου. Εδώ εξηγούνται τα βασικά στοιχεία, Συνολικά, πραγματοποιεί μια αναζήτηση υπερπαραμέτρων, δοκιμάζοντας διάφορους συνδυασμούς βελτιστοποιητών, συναρτήσεων απώλειας και μεγεθών δέσμης για να βρει το μοντέλο με την καλύτερη απόδοση ως προς την απώλεια επικύρωσης. Οι λεπτομέρειες του καλύτερου μοντέλου, όπως ο βελτιστοποιητής, η συνάρτηση απώλειας, το μέγεθος δέσμης και η διαδρομή αποθήκευσης του μοντέλου, παρακολουθούνται και ενημερώνονται κατά τη διάρκεια που γίνεται η εκπαίδευση.

- Το αρχείο /data/Preprocess.ipynb είναι υπεύθυνο για τη φόρτωση των δεδομένων από τον φάκελο /data και την εκτέλεση επεξεργασίας προεπεξεργασίας για τη βελτίωση της ποιότητας των δεδομένων πριν από την εκπαίδευση του μοντέλου πρότασης. Αυτό το αρχείο επικεντρώνεται στο φιλτράρισμα χρηστών που δεν έχουν αξιολογήσει πολλά βιβλία, με σκοπό να διασφαλίσει ότι το μοντέλο λαμβάνει αξιόπιστα και ενημερωτικά δεδομένα κατά την εκπαίδευση.

Ανάλυση κώδικα

Η εφαρμογή αποτελείτε απο δύο βασικά μέρη: τη δημιουργία του μοντέλου συστάσεων και το Web interface για την αλληλεπίδραση με τους χρήστες.

Για τη δημιουργία του μοντέλου πρότασης, χρειάστηκαν τα παρακάτω:

1. Σύνολο δεδομένων: Χρησιμοποιήθηκε ένα σύνολο δεδομένων που περιλαμβάνει αξιολογήσεις χρηστών για βιβλία.

2. Python: Χρησιμοποιήθηκε η γλώσσα προγραμματισμού Python για την υλοποίηση του μοντέλου πρότασης.

3. Βιβλιοθήκες Python: Χρησιμοποιήθηκαν διάφορες βιβλιοθήκες Python, όπως το Keras, το Pandas, το NumPy και το scikit-learn, για τη διαχείριση και επεξεργασία των δεδομένων, καθώς και για την υλοποίηση και εκπαίδευση του μοντέλου πρότασης.

4. Εργαλεία ανάπτυξης: Χρησιμοποιήθηκε ένα περιβάλλον ανάπτυξης, όπως το Jupyter Notebook, για την ανάπτυξη και εκτέλεση του κώδικα.

Οπος εχει αναφερθει για την δημιουργεια αυτου του μοντελου χρησιμοποιθηκε το συνολο δεδομενων good-book10k[41]. Τα δεδομενα στην αρχικη του μορφη ειχαν χωρο για βελτιοση και γιαυτο στον παρακατω κωδικα γινετε βελτιοσει των δεδωμενων

Preprocess_gr.ipynb

Εισαγωγή των απαραίτητων βιβλιοθηκών

- Pandas: Χρησιμοποιείται για την επεξεργασία και ανάλυση δεδομένων.

- Requests: Χρησιμοποιείται για την αποστολή HTTP αιτημάτων και την αλληλεπίδραση με διαδικτυακές διεπαφές προγραμματισμού εφαρμογών (APIs).

- Re: Παρέχει υποστήριξη για τις κανονικές εκφράσεις στην Python.

```
1. import pandas as pd
2. import requests, re
```

Φόρτωση δεδομένων

Το απόσπασμα κώδικα διαβάζει τα δεδομένα από δύο αρχεία CSV, το `ratings.csv` και το `books.csv`, και δημιουργεί pandas DataFrames.

1. `pd.read_csv('training/ratings.csv')`: Αυτή η γραμμή διαβάζει τα δεδομένα από το αρχείο `ratings.csv` που βρίσκεται στη διαδρομή `'.training/ratings.csv'` χρησιμοποιώντας τη συνάρτηση `pd.read_csv()` από τη βιβλιοθήκη pandas. Τα δεδομένα φορτώνονται σε ένα DataFrame με όνομα `ratings`.

2. `pd.read_csv('training/books.csv')`: Αντίστοιχα, αυτή η γραμμή διαβάζει τα δεδομένα από το αρχείο `books.csv` που βρίσκεται στη διαδρομή `'training/books.csv'` και τα αποθηκεύει σε ένα DataFrame με όνομα `books`.

Με την εκτέλεση αυτού του κώδικα, το DataFrame `ratings` γεμίζει με τα δεδομένα από το αρχείο `ratings.csv`, το DataFrame `books` γεμίζει με τα δεδομένα από το αρχείο `books.csv`

```
1. ratings = pd.read_csv('training/ratings.csv')
2. books = pd.read_csv('training/books.csv')
```

Εκτύπωση Σύνοψης Δεδομένων

```
1. ratings.head()
```

	book_id	user_id	rating
0	1	314	5
1	1	439	3
2	1	588	5
3	1	1169	4
4	1	1185	4

```
1. books.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 23 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   id                         10000 non-null  int64
 1   book_id                    10000 non-null  int64
 2   best_book_id               10000 non-null  int64
 3   work_id                    10000 non-null  int64
 4   books_count                10000 non-null  int64
 5   isbn                       9300 non-null   object
 6   isbn13                     9415 non-null   float64
 7   authors                    10000 non-null  object
 8   original_publication_year  9979 non-null   float64
 9   original_title             9415 non-null   object
 10  title                      10000 non-null  object
 11  language_code              8916 non-null   object
 12  average_rating             10000 non-null  float64
 13  ratings_count              10000 non-null  int64
 14  work_ratings_count         10000 non-null  int64
 15  work_text_reviews_count    10000 non-null  int64
 16  ratings_1                  10000 non-null  int64
 17  ratings_2                  10000 non-null  int64
 18  ratings_3                  10000 non-null  int64
 19  ratings_4                  10000 non-null  int64
 20  ratings_5                  10000 non-null  int64
 21  image_url                  10000 non-null  object
 22  small_image_url            10000 non-null  object
dtypes: float64(3), int64(13), object(7)
memory usage: 1.8+ MB
```

Η πρώτη εκτύπωση αφορά το dataframe "ratings" και μας παρέχει μια επισκόπηση των πρώτων πέντε γραμμών του. Αναφέρει τις στήλες "book_id", "user_id" και "rating", οι οποίες περιέχουν πληροφορίες σχετικά με το αναγνωριστικό του βιβλίου, το αναγνωριστικό του χρήστη και την αξιολόγηση που έχει δώσει ο χρήστης για το βιβλίο αντίστοιχα.

Η δεύτερη εκτύπωση αφορά το dataframe "books" και παρέχει μια επισκόπηση των δεδομένων του. Περιγράφει τις στήλες που περιλαμβάνονται στο dataframe, όπως το "id", "book_id", "authors", "original_title", "average_rating" κ.λπ. Επιπλέον, παρέχει πληροφορίες για τον τύπο δεδομένων κάθε στήλης και τον αριθμό των μη-κενών τιμών σε κάθε στήλη. Τέλος, παρουσιάζει τη συνολική χρήση μνήμης από το dataframe.

Με αυτές τις εκτυπώσεις, μπορούμε να πάρουμε μια γενική εικόνα των δεδομένων που χρησιμοποιούνται

Ανάλυση Αξιολογήσεων Χρηστών και Ορίζοντες Αριθμού Αξιολογήσεων

- **Ανάλυση Αξιολογήσεων Χρηστών**

Αυτό το τμήμα κώδικα αναλύει τις αξιολογήσεις των χρηστών μετρώντας τον αριθμό των αξιολογήσεων ανά χρήστη και υπολογίζοντας ορίζοντες αριθμού αξιολογήσεων. Έπειτα, μετρά τον αριθμό των χρηστών με λιγότερες από X αξιολογήσεις και υπολογίζει το ποσοστό του συνόλου των χρηστών για κάθε όριο. Τέλος, δημιουργεί και εκτυπώνει ένα DataFrame που περιλαμβάνει τα αποτελέσματα.

```
user_ratings = ratings.groupby('user_id')['rating'].count()
user_rating_counts = ratings['user_id'].value_counts()
users_with_ratings = user_rating_counts.groupby(user_ratings).count()
rating_thresholds = list(range(5, 100, 5))

count_per_threshold = []
previous_count = 0
total_users = 53424
for threshold in rating_thresholds:
    count = user_ratings[user_ratings < threshold].count() - previous_count
    count_per_threshold.append(count)
    previous_count += count

percent_per_threshold = [round((count / total_users) * 100) for count in
count_per_threshold]

df = pd.DataFrame({"fewer than X": rating_thresholds, "count": count_per_threshold,
"percent": percent_per_threshold})

(df)
```

- **Φιλτράρισμα Χρηστών με βάση τον Αριθμό Αξιολογήσεων**

Το παρακάτω τμήμα κώδικα πραγματοποιεί μια σειρά επιχειρήσεων για την εμπλουτισμένη προετοιμασία του συνόλου δεδομένων, με σκοπό να προσθέσει φωτογραφίες και συνδέσμους ιστοσελίδων, τα οποία θα χρησιμοποιηθούν αργότερα στην εφαρμογή. Ακολουθεί η επεξήγηση του κώδικα:

1. Η γραμμή `filtered_ratings=ratings[~ratings['user_id'].isin(user_rating_counts[user_rating_counts<filter_out].index.tolist())].copy()` φιλτράρει τις αξιολογήσεις (ratings) αποκλείοντας τους χρήστες με αριθμό αξιολογήσεων λιγότερο από το `filter_out`. Το αποτέλεσμα αποθηκεύεται στο πλαίσιο δεδομένων `filtered_ratings`.
2. Η γραμμή `filtered_ratings.loc[:, 'user_id'] = filtered_ratings.groupby('user_id').ngroup()` αντιστοιχεί νέους αριθμούς στο πεδίο "user_id" του πλαισίου δεδομένων `filtered_ratings` με βάση τον ομαδοποιημένο αριθμό κάθε μοναδικού χρήστη.

3. Η γραμμή `filtered_ratings['cold_start'] = False` προσθέτει μια νέα στήλη "cold_start" στο πλαίσιο δεδομένων `filtered_ratings` και ορίζει τιμή False για κάθε εγγραφή. Αυτή η στήλη θα χρησιμοποιηθεί για την αντιμετώπιση του προβλήματος του "cold start" στο σύστημα συστάσεων.
4. Το πλαίσιο δεδομένων `rating_counts` δημιουργείται για να αποθηκευτεί ο αριθμός των αξιολογήσεων ανά μοναδικό αναγνωριστικό χρήστη.
5. Το πλαίσιο δεδομένων `users` δημιουργείται βασισμένο στο `rating_counts`. Η στήλη 'new_data' αρχικοποιείται με τιμή False για κάθε εγγραφή.

Με αυτές τις επιχειρήσεις πραγματοποιείται η επεξεργασία και ο εμπλουτισμός του συνόλου δεδομένων `filtered_ratings` με φωτογραφίες, συνδέσμους ιστοσελίδων και επιπλέον πληροφορίες που θα χρησιμοποιηθούν στην εφαρμογή.

```
 1. filter_out = 10
 2. filtered_ratings =
ratings[~ratings['user_id'].isin(user_rating_counts[user_rating_counts <
filter_out].index.tolist())].copy()
 3. filtered_ratings.loc[:, 'user_id'] = filtered_ratings.groupby('user_id').ngroup()
 4.
 5.
 6. filtered_ratings['cold_start'] = False
 7.
 8.
 9. rating_counts =
filtered_ratings.groupby('user_id').size().reset_index(name='rating_count')
10.
11. users = pd.DataFrame(rating_counts)
12. ratings = filtered_ratings
13. users['new_data'] = False
14. users.head()
```

Ο παρακατω κώδικας πραγματοποιεί ανάκτηση δεδομένων (data scraping) για την εμπλουτισμένη προετοιμασία του συνόλου δεδομένων. Ακολουθεί η επεξήγηση του κώδικα:

1. Οι παράμετροι `api_key` και `search_engine_id` ανάγονται από τα αρχεία '../api_key' και '../search_engine_id' αντίστοιχα. Αυτά τα αρχεία περιέχουν τα απαραίτητα κλειδιά για την επικοινωνία με την Google Custom Search API.
2. Για κάθε βιβλίο στο πλαίσιο δεδομένων `books`, ελέγχεται αν ο σύνδεσμος του Amazon ξεκινά με 'https://www.amazon.com/'. Αν όχι, εκτελούνται επιπλέον ενέργειες.
3. Δημιουργείται η μεταβλητή `search_term` που περιέχει τον όρο αναζήτησης για την Google Custom Search API. Ο όρος αναζήτησης περιλαμβάνει τον τίτλο του βιβλίου, τη λέξη "book", τη λέξη "cover" και το "amazon".

4. Δημιουργείται ο σύνδεσμος αναζήτησης `search_url` με τη χρήση του `api_key`, `search_engine_id` και `search_term`.
5. Εκτελείται η αίτηση HTTP με τη χρήση της `requests.get()` για να αναζητηθούν φωτογραφίες και συνδέσμοι ιστοσελίδων από την Google Custom Search API.
6. Τα αποτελέσματα της αναζήτησης αποθηκεύονται στη μεταβλητή `search_results`.
7. Από τα αποτελέσματα, εξάγονται οι φωτογραφίες και οι συνδέσμοι ιστοσελίδων που σχετίζονται με το βιβλίο.
8. Οι φωτογραφίες εκχωρούνται στην μεταβλητή `image_url`.
9. Αν εντοπιστεί σύνδεσμος Amazon, εκχωρείται στην μεταβλητή `amazon_link`.
10. Οι τιμές των `image_url` και `amazon_link` αντιστοιχίζονται στο πλαίσιο δεδομένων `books` στις αντίστοιχες θέσεις του δείκτη.

Με τον παρακατω κώδικα πραγματοποιείται η ανάκτηση δεδομένων από την Google Custom Search API προκειμένου να προστεθούν φωτογραφίες και συνδέσμοι ιστοσελίδων στο σύνολο δεδομένων των βιβλίων.

```
with open('../api_key', 'rb') as key_file:
    api_key = key_file.read().decode()
with open('../search_engine_id', 'rb') as key_file:
    search_engine_id = key_file.read().decode()
for index, row in books.iterrows():
    amazon_link = str(row['amazon_link'])
    if not amazon_link.startswith('https://www.amazon.com/'):
        book_title = row['title']
        search_title = re.sub(r'[^\w\s-]', '', book_title)
        search_title = re.sub(r'\s', '+', search_title)
        search_title = search_title.lower()
        search_term = f"{book_title}+book+cover+amazon"
        search_url = f"https://www.googleapis.com/customsearch/v1?key={api_key}&cx={search_engine_id}&q={search_term}"
        response = requests.get(search_url)
        search_results = response.json()
        items = search_results.get("items", [])
        image_url = None
        for item in items:
            pagemap = item.get("pagemap", {})
            scraped = pagemap.get("scraped", [])
            if scraped:
                image_link = scraped[0].get("image_link")
                if image_link:
                    image_url = image_link
                link = item.get("link")
                if link and link.startswith('https://www.amazon.com/'):
                    amazon_link = link
                    break
        books.at[index, 'image_url'] = image_url
        books.at[index, 'amazon_link'] = amazon_link

```

book-rec_DL_gr.ipynb

Αυτός ο κώδικας πραγματοποιεί τη δημιουργία, την εκπαίδευση και την αξιολόγηση ενός συστήματος συστάσεων βιβλίων, χρησιμοποιώντας τα δεδομένα χρηστών, αξιολογήσεων και βιβλίων. Πιο συγκεκριμένα:

- Φόρτωση δεδομένων: Φορτώνει τα δεδομένα από δυο αρχεία CSV, ένα για τους ένα για τις αξιολογήσεις και ένα για τα βιβλία.

- Προεπεξεργασία δεδομένων: Μετατρέπει τα μοναδικά αναγνωριστικά των χρηστών και των βιβλίων σε συνεχείς αριθμητικές τιμές, για να μπορεί να τα επεξεργαστεί το εκπαιδευτικό μοντέλο.

- Διαχωρισμός δεδομένων: Διαχωρίζει τα δεδομένα σε εκπαιδευτικά και δοκιμαστικά σετ.

- Δημιουργία μοντέλου: Δημιουργεί έναν συνδετήρα προβλέψεων βαθιάς μάθησης που χρησιμοποιεί embeddings για χρήστες και βιβλία.

- Εκπαίδευση μοντέλου: Εκπαιδεύει το μοντέλο συστάσεων με χρήση διάφορων βελτιστοποιητών, συναρτήσεων απώλειας, μεγέθη δέσμης (batch size) και εποχών. Αυτό γίνεται με την χρήση μιας συνάρτησης που εκπαιδεύει το μοντέλο και αποθηκεύει το μοντέλο με την καλύτερη απόδοση.

- Αξιολόγηση μοντέλου: Φορτώνει κάθε μοντέλο που αποθηκεύτηκε κατά την εκπαίδευση και αξιολογεί την απόδοσή τους στα δεδομένα δοκιμής, χρησιμοποιώντας την τιμή απώλειας ως μετρική. Τέλος, εμφανίζει τα αποτελέσματα από την καλύτερη στην χειρότερη απόδοση.

Το παρακάτω απόσπασμα κώδικα εισάγει τις απαραίτητες βιβλιοθήκες και ενότητες. Εδώ περιγράφονται οι λειτουργίες κάθε εισαγωγής:

- `numpy` (εισάγεται ως `np`): Μια βιβλιοθήκη για αριθμητικές λειτουργίες και διαχείριση πινάκων στην Python.

- `pandas` (εισάγεται ως `pd`): Μια βιβλιοθήκη για την διαχείριση και ανάλυση δεδομένων, παρέχοντας δομές δεδομένων και λειτουργίες για την εργασία με δομημένα δεδομένα.
- `scipy.stats`: Ένα αρθρωτό από το SciPy, μια επιστημονική βιβλιοθήκη υπολογισμού στην Python, παρέχοντας στατιστικές λειτουργίες και κατανομές.
- `sklearn.model_selection`: Μια ενότητα από την scikit-learn, μια δημοφιλής βιβλιοθήκη μηχανικής μάθησης στην Python, που χρησιμοποιείται για την διαίρεση των δεδομένων σε σύνολα εκπαίδευσης και ελέγχου.
- `keras.models`: Μια ενότητα από το Keras, μια βιβλιοθήκη βαθιάς μάθησης, που χρησιμοποιείται για τον καθορισμό και την εκπαίδευση μοντέλων.
- `keras.layers`: Μια ενότητα από το Keras, που χρησιμοποιείται για την κατασκευή των στρωμάτων ενός μοντέλου νευρωνικού δικτύου.
- `keras.optimizers`: Μια ενότητα από το Keras, που παρέχει διάφορους αλγορίθμους βελτιστοποίησης για την εκπαίδευση νευρωνικών δικτύων.
- `keras.callbacks`: Μια ενότητα από το Keras, που περιέχει κλήσεις πίσω που μπορούν να χρησιμοποιηθούν κατά την διάρκεια της εκπαίδευσης του μοντέλου.
- `keras.losses`: Μια ενότητα από το Keras, που παρέχει διάφορες συναρτήσεις απώλειας για προβλήματα παλινδρόμησης και ταξινόμησης.

Αυτές οι βιβλιοθήκες παρέχουν την απαραίτητη λειτουργικότητα για τη διαχείριση των δεδομένων, την κατασκευή μοντέλων, τη βελτιστοποίηση και την αξιολόγηση των μοντέλων.

```
1. import numpy as np
2. import pandas as pd
3. from scipy import stats
4. from sklearn.model_selection import train_test_split
5. from keras.models import Model, load_model
6. from keras.layers import Input, Embedding, Flatten, Dense, Concatenate
7. from keras.optimizers import Adam, RMSprop, SGD
8. from keras.callbacks import ModelCheckpoint
9. from keras.losses import MeanAbsoluteError, MeanSquaredError, Huber, LogCosh
```

- **Φόρτωση δεδομένων**

Εδώ γίνεται η φόρτωση των επεξεργασμένων δεδομένων.

```
1. ratings=pd.read_csv('../Data/ratings.csv')
2. books=pd.read_csv('../Data/books.csv')
```

- **Δημιουργία αντιστοίχισης user-id και book-id**

Το παρακάτω απόσπασμα κώδικα εξηγεί τον σκοπό κάθε γραμμής:

1. Λήψη μοναδικών αναγνωριστικών χρηστών: Ανακτά τα μοναδικά αναγνωριστικά χρηστών από την στήλη 'user_id' του DataFrame ratings και τα μετατρέπει σε λίστα χρησιμοποιώντας τη συνάρτηση `tolist()`. Τα αναγνωριστικά χρηστών αποθηκεύονται στη μεταβλητή `user_ids`.

2. Δημιουργία user-id σε δείκτη: Δημιουργεί ένα λεξικό, `user2user_encoded`, για να αντιστοιχίσει κάθε user-id στον αντίστοιχο δείκτη του. Η συνεπαγόμενη σύνταξη του λεξικού επαναλαμβάνει τη λίστα `user_ids`, αναθέτοντας έναν δείκτη (ξεκινώντας από το 0) σε κάθε user-id χρησιμοποιώντας τη συνάρτηση `enumerate()`. Το προκύπτον λεξικό αντιστοιχίζει κάθε user-id στον αντίστοιχο δείκτη του.

3. Δημιουργία αντιστοίχισης δείκτη σε user-id: Δημιουργεί ένα λεξικό, `userencoded2user`, για να αντιστοιχίσει κάθε δείκτη στο αντίστοιχο user-id. Η συνεπαγόμενη σύνταξη του λεξικού εκτελεί την αντίστροφη αντιστοίχιση, επαναλαμβάνοντας τους αριθμημένους δείκτες και αντιστοιχίζοντας τον κάθε δείκτη στο αντίστοιχο user-id.

4. Λήψη μοναδικών book-id: Ανακτά τα μοναδικά book-id από την στήλη 'book_id' του DataFrame ratings και τα μετατρέπει σε λίστα χρησιμοποιώντας τη συνάρτηση `tolist()`. Τα book-id αποθηκεύονται στη μεταβλητή `book_ids`.

5. Δημιουργία αντιστοίχισης book-id σε δείκτη: Δημιουργεί ένα λεξικό, `book2book_encoded`, για να αντιστοιχίσει κάθε book-id στον αντίστοιχο δείκτη του. Παρόμοια με την αντιστοίχιση των user-id, η συνεπαγόμενη σύνταξη του λεξικού αναθέτει έναν δείκτη (ξεκινώντας από το 0) σε κάθε book-id χρησιμοποιώντας τη συνάρτηση `enumerate()`.

6. Δημιουργία αντιστοίχισης δείκτη σε book-id: Δημιουργεί ένα λεξικό, `book_encoded2book`, για να αντιστοιχίσει κάθε δείκτη στο αντίστοιχο book-id.

Η συνεπαγόμενη σύνταξη του λεξικού εκτελεί την αντίστροφη αντιστοίχιση, αντιστοιχίζοντας τον κάθε δείκτη στο αντίστοιχο book-id.

Αυτές οι αντιστοιχίσεις μεταξύ user-id και book-id και των αντίστοιχων δεικτών τους είναι ουσιαστικές όταν εργάζεστε με επίπεδα ενσωμάτωσης (embedding layers) σε συστήματα συστάσεων. Παρέχουν έναν βολικό τρόπο για τη μετατροπή μεταξύ αρχικών αναγνωριστικών και δεικτών κατά τη διάρκεια της εκπαίδευσης και της πρόβλεψης του μοντέλου.

```
1. user_ids = ratings['user_id'].unique().tolist()
2. user2user_encoded = {x: i for i, x in enumerate(user_ids)}
3. userencoded2user = {i: x for i, x in enumerate(user_ids)}
4. book_ids = ratings['book_id'].unique().tolist()
5. book2book_encoded = {x: i for i, x in enumerate(book_ids)}
6. book_encoded2book = {i: x for i, x in enumerate(book_ids)}
```

- **Αντιστοίχιση user-id και book-id σε δείκτες χρηστών και βιβλίων**

1. Αντιστοίχιση user-id σε ΔείκτεςΠροσθέτει μια νέα στήλη με την ονομασία 'user' στο DataFrame των αξιολογήσεων (ratings DataFrame), αντιστοιχίζοντας την στήλη 'user_id' στους αντίστοιχους δείκτες των χρηστών χρησιμοποιώντας τη συνάρτηση `map()` και το λεξικό `user2user_encoded`. Κάθε αναγνωριστικό χρήστη στη στήλη 'user_id' αντικαθίσταται με την αντίστοιχη τιμή δείκτη.

2. Αντιστοίχιση book-id σε ΔείκτεςΠροσθέτει μια νέα στήλη με την ονομασία 'book' στο DataFrame των αξιολογήσεων, αντιστοιχίζοντας την στήλη 'book_id' στους αντίστοιχους δείκτες των βιβλίων χρησιμοποιώντας τη συνάρτηση `map()` και το λεξικό `book2book_encoded`. Κάθε αναγνωριστικό βιβλίου στη στήλη 'book_id' αντικαθίσταται με την αντίστοιχη τιμή δείκτη.

Με αυτές τις αντιστοιχίσεις, τα user-id και book-id μετατρέπονται στους αντίστοιχους δείκτες τους, οι οποίοι είναι απαραίτητοι για την παροχή των δεδομένων στα επίπεδα ενσωμάτωσης (embedding layers) του μοντέλου συστάσεων. Με αυτόν τον τρόπο, το μοντέλο λειτουργεί με δείκτες αντί για αρχικά αναγνωριστικά, επιτρέποντας αποδοτικούς υπολογισμούς και βελτιωμένη απόδοση.

```
1. ratings['user'] = ratings['user_id'].map(user2user_encoded)
2. ratings['book'] = ratings['book_id'].map(book2book_encoded)
```

- **Διαίρεση των δεδομένων σε σύνολα εκπαίδευσης και ελέγχου**

1. Διαίρεση των Δεδομένων σε Σύνολα Εκπαίδευσης και Ελέγχου: Διαιρεί το DataFrame των αξιολογήσεων (ratings DataFrame) σε σύνολα εκπαίδευσης και ελέγχου χρησιμοποιώντας τη συνάρτηση `train_test_split()` από το scikit-learn. Η συνάρτηση `train_test_split()` δέχεται τα δεδομένα εισόδου (ratings) και τα διαιρεί σε δύο υποσύνολα με βάση την παράμετρο `test_size` που καθορίζεται. Σε αυτήν την περίπτωση, το σύνολο ελέγχου θα έχει μέγεθος 20% του συνόλου δεδομένων.

2. Ανάθεση των Συνόλων Εκπαίδευσης και Ελέγχου: Αναθέτει τα σύνολα εκπαίδευσης και ελέγχου στις μεταβλητές `train` και `test` αντίστοιχα. Το σύνολο εκπαίδευσης θα περιέχει το 80% των δεδομένων, ενώ το σύνολο ελέγχου θα περιέχει το 20% των δεδομένων.

Με τη διαίρεση των δεδομένων σε σύνολα εκπαίδευσης και ελέγχου, δημιουργούνται ξεχωριστά υποσύνολα που μπορούν να χρησιμοποιηθούν για την εκπαίδευση και την αξιολόγηση του μοντέλου συστάσεων. Το σύνολο εκπαίδευσης χρησιμοποιείται για την εκπαίδευση του μοντέλου, ενώ το σύνολο ελέγχου χρησιμοποιείται για την αξιολόγηση της απόδοσής του και την γενίκευσή του σε μη ορατά δεδομένα.

```
1. train, test = train_test_split(ratings, test_size=0.2, random_state=42)
```

- **Ανάκτηση του αριθμού των χρηστών και βιβλίων**

1. Ανάκτηση του Αριθμού των Χρηστών: Υπολογίζει τον αριθμό των μοναδικών χρηστών στο σύνολο δεδομένων, παίρνοντας το μήκος του λεξικού `user2user_encoded`. Η συνάρτηση `len()` επιστρέφει τον αριθμό των στοιχείων στο λεξικό, που αντιπροσωπεύει τον συνολικό αριθμό των μοναδικών χρηστών.

2. Ανάκτηση του Αριθμού των Βιβλίων: Υπολογίζει τον αριθμό των μοναδικών βιβλίων στο σύνολο δεδομένων, παίρνοντας το μήκος του λεξικού `book_encoded2book`. Αντίστοιχα, η συνάρτηση `len()` επιστρέφει τον αριθμό των στοιχείων στο λεξικό, που αντιπροσωπεύει τον συνολικό αριθμό των μοναδικών βιβλίων.

Μέσω της ανάκτησης του αριθμού των χρηστών και των βιβλίων, μπορούμε να προσδιορίσουμε τις διαστάσεις των ενσωματώσεων (embedding layers) στο μοντέλο συστάσεων. Ο αριθμός των χρηστών αντιστοιχεί στον αριθμό των μοναδικών δεικτών χρηστών, ενώ ο αριθμός των βιβλίων αντιστοιχεί στον αριθμό των μοναδικών δεικτών βιβλίων. Αυτές οι τιμές είναι ουσιώδεις για τον καθορισμό των σωστών διαστάσεων των ενσωματώσεων για την αναπαράσταση των χρηστών και των βιβλίων.

```
1. num_users = len(user2user_encoded)
2. num_books = len(book_encoded2book)
```

- **Ορισμός της Διάστασης των Ενσωματώσεων**: Καθορίζει τον αριθμό των διαστάσεων των διανυσμάτων ενσωμάτωσης στο μοντέλο συστάσεων. Η μεταβλητή `embedding_dim` ορίζεται σε 10, που σημαίνει ότι κάθε χρήστης και βιβλίο θα αναπαρίσταται από ένα διάνυσμα μήκους 10 στον χώρο των ενσωματώσεων.

Με τον ορισμό της διάστασης των ενσωματώσεων, καθορίζουμε το μέγεθος των αναπαραστάσεων με διανύσματα για τους χρήστες και τα βιβλία. Μια υψηλότερη διάσταση ενσωμάτωσης μπορεί να επιτρέπει πιο εκφραστικές αναπαραστάσεις, αλλά μπορεί επίσης να αυξήσει την πολυπλοκότητα του μοντέλου και τις απαιτήσεις σε πόρους. Από την άλλη πλευρά, μια χαμηλότερη διάσταση ενσωμάτωσης μπορεί να οδηγήσει σε πιο συμπαγείς αναπαραστάσεις, αλλά μπορεί επίσης να περιορίσει την ικανότητα του μοντέλου να αντιληφθεί περίπλοκες αλληλεπιδράσεις χρήστη-αντικειμένου.

```
1. embedding_dim=10
```

Κατασκευή μοντέλου

Αυτός ο κώδικας δημιουργεί ένα μοντέλο συνεργατικής φιλτράρισης (collaborative filtering) στην Python, χρησιμοποιώντας τη βιβλιοθήκη Keras.

1. Η συνάρτηση create_model() δημιουργεί και επιστρέφει το μοντέλο.
2. Ορίζεται η είσοδος του χρήστη με το όνομα "user_input". Η είσοδος αυτή έχει μορφή ενός αριθμού (ένας χρήστης).
3. Η είσοδος του χρήστη περνά από ένα επίπεδο "Embedding" που μετατρέπει τον αριθμό του χρήστη σε ένα πυκνό διάνυσμα (embedding). Αυτό το πυκνό διάνυσμα αντιπροσωπεύει τον χρήστη στο μοντέλο.
4. Με τη χρήση της συνάρτησης "Flatten()", το πυκνό διάνυσμα του χρήστη μετατρέπεται από πολυδιάστατη μορφή σε μία γραμμική μορφή.
5. Αντίστοιχα με τα βήματα 2-4, ορίζεται η είσοδος για το βιβλίο, με το όνομα "book_input", και μετατρέπεται σε ένα πυκνό διάνυσμα (embedding).
6. Το πυκνό διάνυσμα του βιβλίου επίσης περνά από το επίπεδο "Flatten()" για να μετατραπεί σε μία γραμμική μορφή.
7. Το γινόμενο (dot product) ανάμεσα στα γραμμικά διανύσματα του βιβλίου και του χρήστη υπολογίζεται με τη χρήση της συνάρτησης "Dot(axes=1)".
8. Το μοντέλο ορίζεται με τις εισόδους του χρήστη και του βιβλίου, και με την έξοδο του γινομένου.
9. Το τελικό μοντέλο επιστρέφεται.

Συνοψίζοντας, ο κώδικας αυτός δημιουργεί ένα μοντέλο συνεργατικής φιλτράρισης. Αυτό το μοντέλο χρησιμοποιεί τον αλγόριθμο ενσωμάτωσης (embedding) για την αναπαράσταση των χρηστών και των βιβλίων σε πυκνά διανύσματα. Το γινόμενο των πυκνών διανυσμάτων αντιπροσωπεύει την πρόβλεψη της αξιολόγησης ή προτίμησης ενός χρήστη για ένα συγκεκριμένο βιβλίο.

```
 1. def create_model():
 2.
 3.     user_input = Input(shape=(1,), name="user_input")
 4.     user_embedding = Embedding(num_users, embedding_dim,
name="user_embedding")(user_input)
 5.     user_vec = Flatten()(user_embedding)
 6.
 7.     book_input = Input(shape=[1], name="book_input")
 8.     book_embedding = Embedding(num_books, embedding_dim,
name="book_embedding")(book_input)
 9.     book_vec = Flatten()(book_embedding)
10.
11.     product = Dot(axes=1)([book_vec, user_vec])
12.     model = Model(inputs=[user_input, book_input], outputs=product)
13.
14.     return model
```

Η συνάρτηση `train_model` πραγματοποιεί τα ακόλουθα βήματα για την εκπαίδευση του μοντέλου συστάσεων:

1. Συντακτική Ανάλυση του Μοντέλου: Μεταγλωττίζει το μοντέλο συστάσεων με την καθορισμένη `συνάρτηση απώλειας` και τον `βελτιστοποιητή`. Αυτό το βήμα ρυθμίζει το μοντέλο για εκπαίδευση καθορίζοντας τη συνάρτηση απώλειας που πρέπει να βελτιστοποιηθεί και τον αλγόριθμο βελτιστοποίησης που πρέπει να χρησιμοποιηθεί.

2. Εκπαίδευση του Μοντέλου: Προσαρμόζει το μοντέλο συστάσεων στα δεδομένα εκπαίδευσης χρησιμοποιώντας τη μέθοδο `fit`. Καθορίζει τα είσοδα εκπαίδευσης (`[train.user.values, train.book.values]`), τους στόχους εκπαίδευσης (`train.rating.values`) και άλλες παράμετρους όπως το `batch_size`, τα `epochs` και το `verbose`. Αυτό το βήμα εκπαιδεύει το μοντέλο στα δεδομένα εκπαίδευσης για τον καθορισμένο αριθμό εποχών.

3. Επικύρωση του Μοντέλου: Αξιολογεί το εκπαιδευμένο μοντέλο στα δεδομένα επικύρωσης (`[test.user.values, test.book.values]` και `test.rating.values`) κατά τη διάρκεια της διαδικασίας εκπαίδευσης. Αυτό παρέχει πληροφορίες για την απόδοση του μοντέλου σε μη διαθέσιμα δεδομένα και βοηθά στην παρακολούθηση της προόδου του.
4. Αποθήκευση του Μοντέλου: Αποθηκεύει το εκπαιδευμένο μοντέλο στην καθορισμένη `διαδρομή αποθήκευσης του μοντέλου` χρησιμοποιώντας τον `Συντονιστή Έλεγχου Μοντέλου (ModelCheckpoint)`. Αυτό εξασφαλίζει ότι μόνο το καλύτερο μοντέλο βάσει της απώλειας επικύρωσης θα αποθηκευτεί.

5. Επιστροφή της Απώλειας Επικύρωσης: Επιστρέφει την απώλεια επικύρωσης (`val_loss`) από το ιστορικό του μοντέλου. Η απώλεια επικύρωσης παρέχει μια ένδειξη για το πόσο καλά το μοντέλο γενικεύει σε μη διαθέσιμα δεδομένα.

```
 1. def train_model(batch_size, optimizer, loss_function, model_save_path, num_epochs):
 2.     checkpoint = ModelCheckpoint(model_save_path, monitor='val_loss',
save_best_only=True, mode='min')
 3.     model.compile(loss=loss_function, optimizer=optimizer)
 4.     history = model.fit(x=[train.user.values, train.book.values],
y=train.rating.values,
 5.                         batch_size=batch_size, epochs=num_epochs, verbose=1,
```

```
 6.                         validation_data=([test.user.values, test.book.values],
test.rating.values),
 7.                         callbacks=[checkpoint])
 8.     model.save(model_save_path)
 9.     checkpoint = ModelCheckpoint(model_save_path, monitor='val_loss',
save_best_only=True, mode='min')
10.     return history.history['val_loss'][-1]
```

- **Ρύθμιση του Μοντέλου Συστάσεων**

1. (Optimizers): Μια λίστα που περιέχει παραδείγματα διαφορετικών βελτιστοποιητών, συμπεριλαμβανομένων των Adam και RMSprop. Οι βελτιστοποιητές είναι υπεύθυνοι για την ενημέρωση των βαρών του μοντέλου κατά τη διάρκεια της εκπαίδευσης για την ελαχιστοποίηση της συνάρτησης απώλειας.

2. (Loss Functions): Μια λίστα που περιέχει παραδείγματα διαφορετικών συναρτήσεων απώλειας, όπως οι MeanAbsoluteError, MeanSquaredError, Huber και LogCosh. Οι συναρτήσεις απώλειας μετρούν τη διαφορά μεταξύ των προβλεπόμενων βαθμολογιών και των πραγματικών βαθμολογιών.

3. (Batch Sizes): Μια λίστα που περιέχει διάφορα μεγέθη δέσμης, όπως τα 32 και 64, τα οποία καθορίζουν τον αριθμό των δειγμάτων που επεξεργάζονται πριν από την ενημέρωση των βαρών του μοντέλου.

4. (Model Paths): Μια κενή λίστα που θα αποθηκεύει τις διαδρομές των καλύτερων μοντέλων που βρέθηκαν κατά τη διάρκεια της διαδικασίας ρύθμισης.

5. Best Validation Loss): Μια μεταβλητή που αρχικοποιείται με ένα μεγάλο αριθμό (`float('inf')`) για να παρακολουθεί την καλύτερη απώλεια επικύρωσης που επιτεύχθηκε κατά τη διάρκεια της εκπαίδευσης.

6. (Best Optimizer): Μια μεταβλητή που θα αποθηκεύει τον βελτιστοποιητή που παρουσιάζει την καλύτερη απώλεια επικύρωσης.

7. (Best Loss Function): Μια μεταβλητή που θα αποθηκεύει τη συνάρτηση απώλειας που παρουσιάζει την καλύτερη απώλεια επικύρωσης.

8. (Best Batch Size): Μια μεταβλητή που θα αποθηκεύει το μέγεθος δέσμης που παρουσιάζει την καλύτερη απώλεια επικύρωσης.

9. (Best Number of Epochs): Μια μεταβλητή που θα αποθηκεύει τον αριθμό εποχών που αντιστοιχεί στην καλύτερη απώλεια επικύρωσης.

10. (Best Model Path): Μια μεταβλητή που θα αποθηκεύει τη διαδρομή του μοντέλου με την καλύτερη απώλεια επικύρωσης.

Αυτές οι μεταβλητές και λίστες θα χρησιμοποιηθούν για την καταγραφή και ενημέρωση των βέλτιστων υπερπαραμέτρων και της απόδοσης του μοντέλου κατά τη διάρκεια της διαδικασίας ρύθμισης.

```
optimizers = [Adam(), RMSprop()]
loss_functions = [MeanAbsoluteError(), MeanSquaredError(), Huber(), LogCosh()]
batch_sizes = [32, 64]
model_paths = []

best_val_loss = float('inf')
best_optimizer = None
best_loss_function = None
best_batch_size = None
best_num_epochs = None
best_model_path = None
```

- **Ρύθμιση των υπερπαραμέτρων και εκπαίδευση του μοντέλου συστάσεων**

1. Βρόχος Βελτιστοποιητών: Ο εξωτερικός βρόχος επανάληψης διατρέχει τους βελτιστοποιητές στη λίστα `optimizers`.

2. Βρόχος Συνάρτησης Απώλειας: Ο δεύτερος βρόχος επανάληψης διατρέχει τις συναρτήσεις απώλειας στη λίστα `loss_functions`.

3. Βρόχος Μεγέθους Δέσμης: Ο εσωτερικός βρόχος επανάληψης διατρέχει τα μεγέθη δέσμης στη λίστα `batch_sizes`.

4. Αριθμός Εποχών: Η μεταβλητή `num_epochs` ορίζεται σε 20, που υποδηλώνει τον αριθμό των εποχών εκπαίδευσης για κάθε συνδυασμό υπερπαραμέτρων.

5. Δημιουργία Μοντέλου: Δημιουργείται μια νέα περίπτωση του μοντέλου συστάσεων χρησιμοποιώντας την συνάρτηση `create_model()`.

6. Διαδρομή Αποθήκευσης Μοντέλου: Η μεταβλητή `model_save_path` δημιουργείται για να καθορίσει τη διαδρομή για την αποθήκευση του εκπαιδευμένου μοντέλου με βάση τον τρέχοντα συνδυασμό βελτιστοποιητή, συνάρτησης απώλειας και μεγέθους δέσμης.

7. Εκτύπωση Υπερπαραμέτρων: Ο βελτιστοποιητής, η συνάρτηση απώλειας και το μέγεθος δέσμης εκτυπώνονται για να παρακολουθηθεί η πρόοδος της διαδικασίας ρύθμισης.

8. Εκπαίδευση του Μοντέλου: Η συνάρτηση `train_model()` καλείται για να εκπαιδεύσει το μοντέλο με τις τρέχουσες υπερπαραμέτρους. Η συνάρτηση παίρνει ως είσοδο το μέγεθος δέσμης, τον βελτιστοποιητή, τη συνάρτηση απώλειας, τη διαδρομή αποθήκευσης μοντέλου και τον αριθμό εποχών. Επιστρέφει την απώλεια επικύρωσης του εκπαιδευμένου μοντέλου.

9. Ενημέρωση των Καλύτερων Υπερπαραμέτρων: Εάν η απώλεια επικύρωσης που έχει επιτευχθεί με τις τρέχουσες υπερπαραμέτρους είναι μικρότερη από την προηγούμενη καλύτερη απώλεια επικύρωσης (`best_val_loss`), ενημερώνονται οι καλύτερες υπερπαράμετροι και πληροφορίες του μοντέλου.

10. Διαδρομή Καλύτερου Μοντέλου: Η μεταβλητή `best_model_path` αποθηκεύει τη διαδρομή του μοντέλου με τη χαμηλότερη απώλεια επικύρωσης μεταξύ όλων των συνδυασμών.

Με την επανάληψη σε διάφορους συνδυασμούς βελτιστοποιητών, συναρτήσεων απώλειας και μεγεθών δέσμης, τον καλύτερο συνδυασμό υπερπαραμέτρων για την εκπαίδευση του μοντέλου συστάσεων.

```
for optimizer in optimizers:
    for loss_function in loss_functions:
        for batch_size in batch_sizes:
            num_epochs = 20
            model = create_model()
            model_save_path = 
f"model_{optimizer.__class__.__name__}_{loss_function.__class__.__name__}_batch{batch_si
ze}.tf"
            model_paths.append(model_save_path)
            print(optimizer, loss_function, batch_size)
            val_loss = train_model(batch_size, optimizer, loss_function, 
model_save_path, num_epochs)
            if val_loss < best_val_loss:
                best_val_loss = val_loss
                best_optimizer = optimizer
                best_loss_function = loss_function
                best_batch_size = batch_size
                best_num_epochs = num_epochs
                best_model_path = model_save_path
```

Έπειτα με τον παρακάτω κώδικα βλέπουμε ποιος συνδυασμός είχε την καλύτερη απόδοση

```
print(f"Best optimizer: {best_optimizer.__class__.__name__}")
print(f"Best loss function: {best_loss_function.__class__.__name__}")
print(f"Best batch size: {best_batch_size}")
print(f"Best number of epochs: {best_num_epochs}")
print(f"Best model path: {best_model_path}")
```

```
Best optimizer: RMSprop
Best loss function: LogCosh
Best batch size: 32
Best number of epochs: 20
Best model path: model_RMSprop_LogCosh_batch32.h5
```

Μπορούμε επισης να δούμε τα μοντέλα σε κατάταξη από το καλύτερο προς το χειρότερο.

```
import glob
model_paths = []
model_paths = glob.glob("*.tf")
results = []
for model_path in model_paths:
    model = load_model(model_path)
    evaluation = model.evaluate([test.user.values, test.book.values],
test.rating.values,verbose=0)
    results.append((model_path, evaluation))


results.sort(key=lambda x: x[1])


for i, (model_path, evaluation) in enumerate(results, start=1):
    print(f'Rank {i}: Test MSE for {model_path}: {evaluation}')
```

```
Rank 1: Test MSE for model_Adam_LogCosh_batch32.tf: 0.31020718812942505
Rank 2: Test MSE for model_Adam_LogCosh_batch16.tf: 0.32118740677833557
Rank 3: Test MSE for model_Adam_Huber_batch32.tf: 0.35046201944351196
Rank 4: Test MSE for model_Adam_Huber_batch16.tf: 0.35643431544303894
Rank 5: Test MSE for model_RMSprop_LogCosh_batch16.tf:
0.4972408413887024
Rank 6: Test MSE for model_RMSprop_Huber_batch16.tf: 0.5370606184005737
Rank 7: Test MSE for model_RMSprop_LogCosh_batch32.tf:
0.6359776854515076
Rank 8: Test MSE for model_RMSprop_Huber_batch32.tf: 0.6430408954620361
Rank 9: Test MSE for model_Adam_MeanAbsoluteError_batch32.tf:
0.6962500214576721
Rank 10: Test MSE for model_Adam_MeanAbsoluteError_batch16.tf:
0.706304669380188
Rank 11: Test MSE for model_Adam_MeanSquaredError_batch32.tf:
0.8042389154434204
Rank 12: Test MSE for model_RMSprop_MeanAbsoluteError_batch16.tf:
0.838294506072998
Rank 13: Test MSE for model_Adam_MeanSquaredError_batch16.tf:
0.8691396713256836
Rank 14: Test MSE for model_RMSprop_MeanAbsoluteError_batch32.tf:
0.8888751268386841
```

```
Rank 15: Test MSE for model_RMSprop_MeanSquaredError_batch16.tf:
1.3519461154937744
Rank 16: Test MSE for model_RMSprop_MeanSquaredError_batch32.tf:
1.4482344388961792
```

Από τα αποτελέσματα της αξιολόγησης των μοντέλων, βλέπουμε ότι το μοντέλο με την καλύτερη απόδοση είναι το "model_Adam_LogCosh_batch32.tf" με την χαμηλότερη τιμή απώλειας 0.271904319524765. Αυτό το μοντέλο χρησιμοποιεί τον αλγόριθμο βελτιστοποίησης RMSprop, την συνάρτηση απώλειας LogCosh και το μέγεθος δέσμης (batch size) 32.

Επίσης, βλέπουμε μια γενική τάση όπου τα μοντέλα που χρησιμοποιούν τη συνάρτηση απώλειας LogCosh ή Huber είναι αυτά με την καλύτερη απόδοση, ενώ τα μοντέλα που χρησιμοποιούν την MeanSquaredError είναι αυτά με την χειρότερη απόδοση.

Τέλος, παρατηρούμε ότι τα μοντέλα που εκπαιδεύτηκαν με μικρότερο μέγεθος δέσμης (32) είχαν γενικά καλύτερη απόδοση σε σχέση με τα μοντέλα που εκπαιδεύτηκαν με μεγαλύτερο μέγεθος δέσμης (64).

Για τη δημιουργία του Web interface που ο χρηστης μπορει να παιρνει συστασεις για βιλαι χρειάστηκαν τα παρακάτω:

1. Flask: Το Flask είναι ένα ελαφρύ web framework για Python. Χρησιμοποιήθηκε για τη δημιουργία του back-end της εφαρμογής και για τη διαχείριση των HTTP αιτημάτων και απαντήσεων.

2. Keras: Η βιβλιοθήκη Keras χρησιμοποιήθηκε για τη φόρτωση του μοντέλου συστασεων που είχε εκπαιδευτεί στην Python.

3. Pandas και NumPy: Οι βιβλιοθήκες Pandas και NumPy χρησιμοποιήθηκαν για τη διαχείριση, επεξεργασία και μετασχηματισμό των δεδομένων.

4. Python os: Χρησιμοποιήθηκε για την διαχείριση των αρχείων και των διαδρομών του συστήματος αρχείων.

5. re: Η βιβλιοθήκη re της Python, που επιτρέπει τη χρήση των regular expressions, χρησιμοποιήθηκε για την επεξεργασία και τον καθαρισμό των δεδομένων.

Η εφαρμογή χρησιμοποιεί το μοντέλο συστάσεων (`model_RMSprop_LogCosh_batch32.h5`) για να παρέχει προτάσεις στους χρήστες μέσω μιας web διεπαφής. Οι χρήστες μπορούν να κάνουν αιτήματα μέσω της εφαρμογής , που στη συνέχεια διαχειρίζεται τα αιτήματα, εφαρμόζει το μοντέλο συστασεων και επιστρέφει τα αποτελέσματα στους χρήστες.

Το κομμάτι αυτό αποτελείτε από το Flask_app.py και τα .html αρχεία που βρίσκονται στον /templates φάκελο

Flask_app.py

```
from flask import Flask, request, render_template, Response, url_for, redirect
from flask_login import LoginManager, UserMixin, login_user, login_required,
logout_user, current_user
from keras.models import Model,load_model
from keras.optimizers import Adam
from keras.losses import LogCosh
from keras.layers import Input, Embedding, Flatten,Dot
from sklearn.model_selection import train_test_split
from apscheduler.schedulers.background import BackgroundScheduler
import numpy as np
import pandas as pd
import os,  math, threading, queue, signal, sys

app = Flask(__name__)
app.secret_key = os.environ.get('SECRET_KEY', 'default-secret-key')
scheduler = BackgroundScheduler()
delay_minutes = 5
max_q_size = 2
stop_threads = False
update_ratings_executed = False
login_manager = LoginManager()
login_manager.init_app(app)
login_manager.login_view = "login"
task_queue = queue.Queue()
class User(UserMixin):
    def __init__(self, id):
        self.id = id

    @classmethod
    def get_user(cls, user_id):
        users = pd.read_csv('data/users.csv')
        if str(user_id) in users['user_id'].astype(str).values:
            return cls(user_id)
        return None
def create_model(model,num_users,num_books):

def retrain_model():

def recommend_books(user_id, num_books=5):

def update_rating(user_id, book_id, rating):

def get_unrated_books(user_id):

def sort_books_by_rating(books):
```

```
def give_rating(user_id):

def create_user():

def get_user_ratings(user_id):

def load_user(user_id):

def background_task():

def schedule_retrain():

def stop_background_task(signal, frame):

task_thread = threading.Thread(target=background_task)
task_thread.start()

timer = threading.Timer(delay_minutes * 60, schedule_retrain)
timer.start()

#-------------------------------------------------------------------------------------#

@app.route('/')
def home():

@app.route('/login', methods=['POST','GET'])
def login():

@app.route('/main', methods=['POST','GET'])
@login_required
def main():

@app.route('/new_user', methods=['GET'])
def new_user():

@app.route('/train_user', methods=['POST','GET'])
@login_required
def train_user():

@app.route('/recommend', methods=['POST','GET'])
@login_required
def recommend():

@app.route('/view_profile', methods=['GET','POST'])
@login_required
def view_profile():

@app.route('/logout', methods=['POST','GET'])
@login_required
def logout():

@app.route("/admin")
@login_required
def admin_page():

@app.route('/enqueue_task', methods=['GET'])
def enqueue_task():

@login_manager.user_loader
def load_user(user_id):

if __name__ == "__main__":
    app.run(debug=True,port=5000)
```

Το παραπάνω κομμάτι κώδικα περιέχει την εφαρμογή Flask που υλοποιεί το σύστημα συστάσεων βιβλίων. Η εφαρμογή έχει την δυνατότητα εγγραφής και σύνδεσης

χρηστών, αξιολόγησης βιβλίων, εκτέλεσης συστάσεων βιβλίων και προβολής προφίλ χρήστη. Ο κώδικας είναι οργανωμένος σε διάφορες λειτουργίες και υποστηρίζει τη χρήση της βιβλιοθήκης Keras για την εκπαίδευση ενός μοντέλου συστάσεων και την πρόταση βιβλίων στους χρήστες. Επιπλέον, χρησιμοποιεί τη βιβλιοθήκη apscheduler για τον προγραμματισμό επανεκπαίδευσης του μοντέλου και τη χρήση ενός background thread για την εκτέλεση αρχείων tasks στο παρασκήνιο. Θα γινει παρακατω εκτενεις αναλυση των συναρτησεων και ολου του κωδικα

```
app = Flask(__name__)
app.secret_key = os.environ.get('SECRET_KEY', 'default-secret-key')
scheduler = BackgroundScheduler()
delay_minutes = 5
max_q_size = 2
stop_threads = False
update_ratings_executed = False
login_manager = LoginManager()
login_manager.init_app(app)
login_manager.login_view = "login"
task_queue = queue.Queue()
class User(UserMixin):
    def __init__(self, id):
        self.id = id

    @classmethod
    def get_user(cls, user_id):
        users = pd.read_csv('data/users.csv')
        if str(user_id) in users['user_id'].astype(str).values:
            return cls(user_id)
        return None
```

Σε αυτήν την ενότητα του κώδικα συμβαίνουν τα παρακάτω:

1. Δημιουργία της εφαρμογής Flask: Η γραμμή `app = Flask(__name__)` δημιουργεί μια νέα περίπτωση της κλάσης Flask, που αντιπροσωπεύει την εφαρμογή Flask που τρέχει. Το `__name__` είναι η μεταβλητή που αναφέρεται στο τρέχον όνομα του αρχείου.
2. Ορισμός του μυστικού κλειδιού: Η γραμμή `app.secret_key = os.environ.get('SECRET_KEY', 'default-secret-key')` ορίζει το μυστικό κλειδί της εφαρμογής. Το μυστικό κλειδί χρησιμοποιείται για την ασφάλεια των συνεδριών.
3. Δημιουργία του BackgroundScheduler: Η γραμμή `scheduler = BackgroundScheduler()` δημιουργεί μια περίπτωση της κλάσης BackgroundScheduler από τη βιβλιοθήκη apscheduler. Αυτός ο προγραμματιστής φόντου χρησιμοποιείται για να προγραμματίσει και να εκτελέσει φόντο εργασίες, όπως η επανεκπαίδευση του μοντέλου.

4. Ρυθμίσεις εργασιών και ουρά εργασιών: Οι μεταβλητές `delay_minutes`, `max_q_size`, `stop_threads` και `update_ratings_executed` χρησιμοποιούνται για τη ρύθμιση και τον έλεγχο των εργασιών και της ουράς εργασιών. Αυτές οι μεταβλητές ελέγχουν τον χρόνο καθυστέρησης πριν από την εκτέλεση της επανεκπαίδευσης του μοντέλου, τον μέγιστο αριθμό εργασιών που μπορούν να προστεθούν στην ουρά εργασιών και την κατάσταση των νημάτων εργασίας.
5. Αρχικοποίηση του LoginManager: Η γραμμή `login_manager = LoginManager()` δημιουργεί μια περίπτωση της κλάσης LoginManager από το Flask-Login. Ο LoginManager χρησιμοποιείται για τη διαχείριση της επιτυχούς σύνδεσης χρήστη και της πρόσβασης.
6. Αρχικοποίηση της ουράς εργασιών: Η γραμμή `task_queue = queue.Queue()` δημιουργεί μια ουρά εργασιών χρησιμοποιώντας την κλάση Queue από τη βιβλιοθήκη queue. Αυτή η ουρά χρησιμοποιείται για την προσθήκη και την εκτέλεση εργασιών στο φόντο.

```
def create_model(model,num_users,num_books):
    embedding_dim = 10
    user_embedding_layer = model.get_layer('user_embedding')
    user_embedding_config = user_embedding_layer.get_config()
    user_id_max = user_embedding_config['input_dim']
    print(user_id_max)

    old_weights = model.get_layer('user_embedding').get_weights()[0]
    new_weights = np.zeros((num_users , embedding_dim))
    new_weights[:user_id_max,0 :] = old_weights
    new_user_embedding = Embedding(num_users, embedding_dim, input_length=1)

    new_user_embedding.build((None,))
    new_user_embedding.set_weights([new_weights])
    new_user_embedding._name = "user_embedding"

    user_input = Input(shape=(1,), name="user_input")
    user_embedding = new_user_embedding(user_input)
    user_vec = Flatten()(user_embedding)

    book_input = Input(shape=[1], name="book_input")
    book_embedding = Embedding(num_books, embedding_dim,
name="book_embedding")(book_input)
    book_vec = Flatten()(book_embedding)

    product = Dot(axes=1)([book_vec, user_vec])

    new_model = Model(inputs=[user_input, book_input], outputs=product)
    for layer_index, layer in enumerate(model.layers):
        if layer_index >= 2 and layer.name != 'user_embedding':
            new_model.layers[layer_index].set_weights(layer.get_weights())
    new_model.compile(loss=model.loss, optimizer=model.optimizer)
    return new_model

def retrain_model():
    print('Starting - Retrain session')
    global update_ratings_executed
    if update_ratings_executed == True:
        update_ratings_executed = False
```

```
        print('next scheduled retrain will be skipped')
    ratings=pd.read_csv('Data/ratings.csv')
    optimizer = Adam()
    loss_function = LogCosh()
    batch_size = 32
    model_path = ('models/model_Adam_LogCosh_batch32.tf')
    model = load_model(model_path)
    user2user_encoded = {x: i for i, x in enumerate(ratings['user_id'].unique())}
    book2book_encoded = {x: i for i, x in enumerate(ratings['book_id'].unique())}
    book_ids = ratings['book_id'].unique().tolist()
    book_encoded2book = {i: x for i, x in enumerate(book_ids)}
    num_users = len(user2user_encoded)
    num_books = len(book_encoded2book)
    ratings['user'] = ratings['user_id'].map(user2user_encoded)
    ratings['book'] = ratings['book_id'].map(book2book_encoded)
    train, test = train_test_split(ratings, test_size=0.2, random_state=42)
    new_model = create_model(model,num_users,num_books)
    for layer_index, layer in enumerate(model.layers):
        if layer_index >= 2 and layer.name != 'user_embedding':
            new_model.layers[layer_index].set_weights(layer.get_weights())
    new_model.compile(loss=loss_function, optimizer=optimizer)
    new_model.fit(
        x=[train['user'].values, train['book'].values],y=train['rating'].values,
        batch_size=batch_size,
        epochs=1,
        verbose=0,
        validation_data=([test['user'].values, test['book'].values],
test['rating'].values))
    print('Done')
    new_model.save(model_path)
    return
```

Στο παραπάνω τμήμα κώδικα, υπάρχουν δύο συναρτήσεις: `**create_model**` και `**retrain_model**`. Ας τις αναλύσουμε μία προς μία στα ελληνικά.

- Συνάρτηση `**create_model**`

Αυτή η συνάρτηση χρησιμοποιείται για τη δημιουργία ενός νέου μοντέλου πρόβλεψης βαθμολογιών. Παίρνει ως είσοδο ένα υπάρχον μοντέλο `model`, τον αριθμό των χρηστών `num_users` και τον αριθμό των βιβλίων `num_books`.

Η συνάρτηση δημιουργεί ένα νέο μοντέλο με τα ίδια επίπεδα με το αρχικό μοντέλο, εκτός από το επίπεδο `user_embedding`. Αντικαθιστά το `user_embedding` με ένα νέο επίπεδο `Embedding`, ώστε να μπορεί να αντιστοιχίσει τους χρήστες με ένα διάνυσμα χαρακτηριστικών (`user_vec`). Αυτό γίνεται για να ληφθούν υπόψη οι νέοι χρήστες που μπορεί να εισέλθουν στο σύστημα μετά την αρχική εκπαίδευση του μοντέλου.

Στη συνέχεια, δημιουργείται ένα νέο μοντέλο με είσοδο τα διανύσματα `user_vec` και `book_vec` που παράγονται από τα επίπεδα `user_embedding` και `book_embedding` αντίστοιχα. Ο πολλαπλασιασμός γίνεται με τη χρήση του επιπέδου `Dot` για να προβλέψει τ

η βαθμολογία ενός χρήστη για ένα βιβλίο. Τέλος, το νέο μοντέλο μεταγλωττίζεται με την αρχική συνάρτηση και επιστρέφεται.

Συνάρτηση `**retrain_model**`

Αυτή η συνάρτηση χρησιμοποιείται για την επανεκπαίδευση του μοντέλου πρόβλεψης βαθμολογιών. Ξεκινά διαβάζοντας τα δεδομένα βαθμολογιών από το αρχείο `ratings.csv`. Στη συνέχεια, δημιουργείται ένα νέο μοντέλο από το αρχικό μοντέλο που έχει φορτωθεί από το αρχείο `model_Adam_LogCosh_batch32.tf`. Υπολογίζονται οι κωδικοποιημένες αντιστοιχίες των χρηστών και των βιβλίων. Τα δεδομένα εκπαίδευσης χωρίζονται σε σύνολο εκπαίδευσης και σύνολο επικύρωσης με τη χρήση της συνάρτησης `train_test_split`. Έπειτα, το νέο μοντέλο εκπαιδεύεται στα δεδομένα εκπαίδευσης για έναν καθορισμένο αριθμό εποχών. Τέλος, το νέο μοντέλο αποθηκεύεται στο αρχείο `model_Adam_LogCosh_batch32.tf` για μελλοντική χρήση.

```
def recommend_books(user_id, num_books=5):
            model = load_model('models/model_Adam_LogCosh_batch32.tf')
            ratings = pd.read_csv('data/ratings.csv')
            books = pd.read_csv('data/books.csv')
            book_id_to_name = pd.Series(books.title.values, index =
books.index).to_dict()
            user_ids = ratings['user_id'].unique().tolist()
            user2user_encoded = {x: i for i, x in enumerate(user_ids)}
            book_ids = ratings['book_id'].unique().tolist()
            book2book_encoded = {x: i for i, x in enumerate(book_ids)}

            user_encoded = user2user_encoded[user_id]
            # Getting the book ids in the encoding order
            book_ids = list(book2book_encoded.keys())
            book_ids = np.array(book_ids) - 1
            # Repeating the user id to match the shape of book ids
            user_array = np.array([user_encoded for _ in range(len(book_ids))])

            # Making the prediction
            pred_ratings = model.predict([user_array, np.array(book_ids)])

            # Getting the indices of the top num_books ratings
            top_indices = pred_ratings.flatten().argsort()[-num_books:][::-1]

            # Returning the corresponding book names
            recommended_books = []
            for i in top_indices:
                book_id = book_ids[i] + 1
                book_title = book_id_to_name[book_id]
                book_image_url = books.loc[books['title'] == book_title,
'image_url'].values[0]
                amazon_link = books.loc[books['title'] == book_title,
'amazon_link'].values[0]
                recommended_books.append({
                    "title": book_title,
                    "image": book_image_url,
                    "amazon_link": amazon_link
                })
            return recommended_books
```

Η παραπάνω συνάρτηση `**recommend_books**` χρησιμοποιείται για τη σύσταση βιβλίων σε έναν συγκεκριμένο χρήστη με βάση ένα εκπαιδευμένο μοντέλο πρόβλεψης βαθμολογιών.

Η συνάρτηση παίρνει ως είσοδο το `user_id` του χρήστη και τον αριθμό των `num_books` βιβλίων που θα συσταθούν (προεπιλεγμένη τιμή: 5). Αρχικά, φορτώνεται το εκπαιδευμένο μοντέλο από το αρχείο `model_Adam_LogCosh_batch32.tf` και διαβάζονται τα δεδομένα βαθμολογιών από το αρχείο `ratings.csv` και τα δεδομένα των βιβλίων από το αρχείο `books.csv`. Στη συνέχεια, δημιουργούνται λεξικά αντιστοίχισης μεταξύ των αναγνωριστικών των χρηστών και των βιβλίων με τις αντίστοιχες κωδικοποιημένες τιμές για τη χρήση στο μοντέλο. Στη συνέχεια, η συνάρτηση προβλέπει τις βαθμολογίες που θα έδινε ο συγκεκριμένος χρήστης σε όλα τα βιβλία. Οι προβλεπόμενες βαθμολογίες αποθηκεύονται στη μεταβλητή `pred_ratings`. Έπειτα, επιλέγονται οι κορυφαίες `num_books` βαθμολογίες με βάση τις προβλεπόμενες βαθμολογίες και αποθηκεύονται οι αντίστοιχοι δείκτες στη μεταβλητή `top_indices`.

Τέλος, επιστρέφονται οι αντίστοιχοι τίτλοι, εικόνες και συνδέσμοι από τα βιβλία που συστάθηκαν στη μορφή μιας λίστας λεξικών με τα αντίστοιχα στοιχεία.

```
def update_rating(user_id, book_id, rating):
    ratings = pd.read_csv('data/ratings.csv')
    books = pd.read_csv('data/books.csv')
    users = pd.read_csv('data/users.csv')
    print('user', user_id, 'rated', book_id, 'with', rating)
    if (ratings['book_id'].isin([int(book_id)]) &
ratings['user_id'].isin([int(user_id)])).any():
        print('Rating already exists, updating the existing rating')
        old_rating = ratings.loc[
            (ratings['book_id'] == int(book_id)) & (ratings['user_id'] == int(user_id)),
'rating'].values[0]
        ratings.loc[(ratings['book_id'] == int(book_id)) & (ratings['user_id'] ==
int(user_id)), 'rating'] = int(
            rating, 'cold_start' == [bool(False)])
        books.loc[books['id'] == int(book_id), 'average_rating'] = ratings.loc[
            ratings['book_id'] == int(book_id), 'rating'].mean()
        books.loc[books['id'] == int(book_id), f'ratings_{rating}'] += 1
        books.loc[books['id'] == int(book_id), f'ratings_{old_rating}'] -= 1

        print('rating', old_rating, 'replaced with', rating)
    else:
        new_rating = pd.DataFrame({'book_id': [int(book_id)], 'user_id': [int(user_id)],
'rating': [int(rating)], 'cold_start':[bool(False)]})
        ratings = pd.concat([ratings, new_rating], ignore_index=True)
        books.loc[books['id'] == int(book_id), 'ratings_count'] += 1
        books.loc[books['id'] == int(book_id), 'average_rating'] = ratings.loc[
            ratings['book_id'] == int(book_id), 'rating'].mean()
        books.loc[books['id'] == int(book_id), f'ratings_{rating}'] += 1
        users.loc[ratings['user_id'] == int(user_id), f'rating_count'] += 1
        print('rating', new_rating, 'added to', book_id, 'for',user_id)
```

```
    if len(ratings[(ratings['cold_start'] == False) & (ratings['user_id'] ==
int(user_id))]) > 10:
        ratings = ratings[(ratings['cold_start'] == False) | (ratings['user_id'] !=
int(user_id))]

    users.loc[users['user_id'] == int(user_id), 'new_data'] = True
    ratings.to_csv('data/ratings.csv', index=False)
    books.to_csv('data/books.csv', index=False)
    users.to_csv('data/users.csv', index=False)
    print('update rating.csv and book.csv')
    global update_ratings_executed
    if update_ratings_executed == False:
        update_ratings_executed = True
        print('scheduled a model retrain session')
    return give_rating(user_id)
```

Αυτό το τμήμα κώδικα διαχειρίζεται την ενημέρωση των βαθμολογιών των βιβλίων από τους χρήστες. Αρχικά φορτώνει τα δεδομένα από τα αρχεία `ratings.csv`, `books.csv` και `users.csv`.

Η συνάρτηση `**update_rating(user_id, book_id, rating)**` λαμβάνει ως είσοδο το αναγνωριστικό του χρήστη, το αναγνωριστικό του βιβλίου και τη βαθμολογία που θέλει να δώσει ο χρήστης. Αν η βαθμολογία υπάρχει ήδη (ο ίδιος χρήστης έχει βαθμολογήσει το ίδιο βιβλίο), η συνάρτηση ενημερώνει την υπάρχουσα βαθμολογία και αλλάζει την μέση βαθμολογία του βιβλίου. Αν η βαθμολογία δεν υπάρχει, η συνάρτηση προσθέτει μια νέα εγγραφή στο `ratings.csv` και ενημερώνει τη σχετική στήλη βαθμολογίας στο `books.csv` καθώς και τον αριθμό των βαθμολογιών που έχει δώσει ο χρήστης. Αν ο χρήστης έχει δώσει περισσότερες από 10 βαθμολογίες, τότε όλες οι παλιές βαθμολογίες (με `cold_start=False`) από αυτόν τον χρήστη αφαιρούνται από το DataFrame.Τέλος, η συνάρτηση αποθηκεύει τα ενημερωμένα DataFrame πίσω στα αρχεία `ratings.csv`, `books.csv` και `users.csv`. Εάν η συνάρτηση `update_rating` εκτελείται για πρώτη φορά, προγραμματίζει μια νέα συνεδρία εκπαίδευσης μοντέλου (`model retrain session`).

```
def get_unrated_books(user_id):
    ratings = pd.read_csv('data/ratings.csv')
    books = pd.read_csv('data/books.csv')
    rated_books = ratings[(ratings['user_id'] == user_id) & (ratings['cold_start'] ==
False)]['book_id']
    all_book_ids = books['id']
    unrated_books = all_book_ids[~all_book_ids.isin(rated_books)]
    return list(unrated_books)

def sort_books_by_rating(books):
    ratings= pd.read_csv('data/ratings.csv')
    book_ratings = ratings.groupby('book_id')['rating'].mean()
    sorted_books = book_ratings.loc[books].sort_values(ascending=False)
    return sorted_books.index.tolist()

def give_rating(user_id):
    ratings= pd.read_csv('data/ratings.csv')
    books = pd.read_csv('data/books.csv')
```

```
    unrated_books = books[~books['id'].isin(ratings[ratings['user_id'] ==
int(user_id)]['book_id'])]
    sorted_books = unrated_books.sort_values(by='ratings_count', ascending=False)
    top_10_percent = sorted_books.head(int(len(sorted_books) * 0.1))
    book_id = top_10_percent.sample()['id'].values[0]
    book_title = books.loc[books['id'] == book_id, 'title'].values[0]
    image_url = books.loc[books['id'] == book_id, 'image_url'].values[0]
    print(book_id, book_title, image_url)
    return render_template('rate_book.html', user_id=user_id, book_id=book_id,
book_title=book_title,image_url=image_url)
```

Οι παραπάνω συναρτήσεις παρέχουν λειτουργίες που σχετίζονται με την ανάκτηση αβαθμολόγητων βιβλίων και την προτεραιότητα τους βάσει βαθμολογίας. Ας εξετάσουμε κάθε συνάρτηση πιο λεπτομερώς:

1. `**get_unrated_books(user_id)**`: Αυτή η συνάρτηση διαβάζει τα δεδομένα από τα αρχεία `ratings.csv` και `books.csv`. Στη συνέχεια, εντοπίζει τα βιβλία που έχουν βαθμολογηθεί από τον χρήστη και δεν είναι μέρος της αρχικής "cold start" συλλογής βαθμολογιών. Από τη λίστα αυτή, αφαιρεί τα βιβλία που έχουν ήδη βαθμολογηθεί από τον χρήστη, αφήνοντας μόνο τα βιβλία που δεν έχουν βαθμολογηθεί ακόμη. Τέλος, επιστρέφει τη λίστα με τα αναγνωριστικά των αβαθμολόγητων βιβλίων.
2. `**sort_books_by_rating(books)**`: Αυτή η συνάρτηση δέχεται μια λίστα βιβλίων και τα ταξινομεί βάσει της μέσης βαθμολογίας τους. Χρησιμοποιεί το αρχείο `ratings.csv` για να υπολογίσει τη μέση βαθμολογία για κάθε βιβλίο και στη συνέχεια επιστρέφει μια λίστα με τα αναγνωριστικά των βιβλίων, ταξινομημένη με φθίνουσα σειρά βαθμολογίας.
3. `**give_rating(user_id)**`: Αυτή η συνάρτηση βρίσκει ένα βιβλίο για να βαθμολογήσει ο χρήστης. Πρώτα, φιλτράρει τα βιβλία που δεν έχουν βαθμολογηθεί ακόμη από τον χρήστη και τα ταξινομεί βάσει της συνολικής τους βαθμολογίας (το πλήθος των βαθμολογιών που έχουν λάβει). Επιλέγει το 10% των βιβλίων με την υψηλότερη συνολική βαθμολογία και στη συνέχεια επιλέγει τυχαία ένα από αυτά τα βιβλία. Επιστρέφει τα στοιχεία του βιβλίου (το αναγνωριστικό, τον τίτλο και την εικόνα), προκειμένου να προβληθούν στη σελίδα `rate_book.html`.

```
def create_user():
    ratings=pd.read_csv('data/ratings.csv')
    books=pd.read_csv('data/books.csv')
    users = pd.read_csv('data/users.csv')
    filtered_books5 = books[books['average_rating'] >= 4]
    filtered_books4 = books[(books['average_rating'] >= 3.0) & (books['average_rating']
< 4)]
    filtered_books3= books[(books['average_rating'] >= 2.0) & (books['average_rating'] <
3.0)]
    filtered_books2= books[(books['average_rating'] >= 1.0) & (books['average_rating'] <
2.0)]
```

```
    filtered_books1= books[(books['average_rating'] >= 0) & (books['average_rating'] <
1)]
    top_books_5 = filtered_books5[['id', 'ratings_5']]
    top_books_4 = filtered_books4[['id', 'ratings_4']]
    top_books_3 = filtered_books3[['id', 'ratings_3']]
    top_books_2 = filtered_books2[['id', 'ratings_2']]
    top_books_1 = filtered_books1[['id', 'ratings_1']]
    filtered_top_books_5 = top_books_5.nlargest(20, 'ratings_5').tail(10)
    filtered_top_books_4 = top_books_4.nlargest(20, 'ratings_4').tail(10)
    filtered_top_books_3 = top_books_3.nlargest(20, 'ratings_3').tail(10)
    filtered_top_books_2 = top_books_2.nlargest(20, 'ratings_2').tail(10)
    filtered_top_books_1 = top_books_1.nlargest(20, 'ratings_1').tail(10)
    last_user_id = users['user_id'].max()

    if ratings[(ratings['user_id'] == last_user_id) & (ratings['cold_start'] ==
False)].empty:
        new_user_id = last_user_id
        return(new_user_id)
    else:
        new_user_id = last_user_id + 1

    print('check')
    new_user = pd.DataFrame({'user_id': [new_user_id]})

    users = pd.concat([users, new_user], ignore_index=True)

    users.loc[users['user_id'] == int(new_user_id), 'rating_count'] = 0



    filtered_top_books_5['rating'] = 5
    filtered_top_books_4['rating'] = 4
    filtered_top_books_3['rating'] = 3
    filtered_top_books_2['rating'] = 2
    filtered_top_books_1['rating'] = 1


    filtered_top_books_5['user_id'] = new_user_id
    filtered_top_books_4['user_id'] = new_user_id
    filtered_top_books_3['user_id'] = new_user_id
    filtered_top_books_2['user_id'] = new_user_id
    filtered_top_books_1['user_id'] = new_user_id

    coldstart_df = pd.concat([filtered_top_books_5[['id', 'user_id', 'rating']],
                              filtered_top_books_4[['id', 'user_id', 'rating']],
                              filtered_top_books_3[['id', 'user_id', 'rating']],
                              filtered_top_books_2[['id', 'user_id', 'rating']],
                              filtered_top_books_1[['id', 'user_id', 'rating']]])
    coldstart_df = coldstart_df.rename(columns={"id":"book_id"})
    coldstart_df['cold_start'] = True

    ratings = pd.concat([ratings, coldstart_df], ignore_index=True)
    users.to_csv('data/users.csv', index=False)
    ratings.to_csv('data/ratings.csv', index=False)
    task_queue.put(retrain_model)
    return(new_user_id)
```

Η συνάρτηση `**create_user()**` δημιουργεί έναν νέο χρήστη στο σύστημα διαβάζοντας πρώτα τα δεδομένα από τα `ratings.csv`, `books.csv` και `users.csv`. Στη συνέχεια, φιλτράρει τα βιβλία βάσει των μέσων αξιολογήσεών τους και επιλέγει βιβλία με διάφορα εύρη αξιολογήσεων (5, 4, 3, 2 και 1).

Αποκτά το τελευταίο ID χρήστη από το αρχείο `users.csv` και ελέγχει αν αυτός ο χρήστης υπάρχει ήδη στο αρχείο `ratings.csv`. Εάν δεν υπάρχει, χρησιμοποιεί αυτό το

ID ως το νέο ID χρήστη. Διαφορετικά, δημιουργεί ένα νέο ID χρήστη αυξάνοντας το τελευταίο ID χρήστη κατά 1.Δημιουργεί ένα νέο DataFrame χρήστη και το συγχωνεύει με το υπάρχον DataFrame χρηστών. Επίσης, αρχικοποιεί τον αριθμό των αξιολογήσεων του νέου χρήστη στο 0. Στη συνέχεια, ορίζει την αξιολόγηση για κάθε εύρος αξιολογήσεων (5, 4, 3, 2, 1) και προσθέτει έναν στήλο για το ID χρήστη. Συγχωνεύει όλα τα DataFrames σε ένα, μετονομάζει τη στήλη "id" σε "book_id" και προσθέτει μια στήλη "cold_start" με τιμή True.

Στο τέλος, ενημερώνει τα αρχεία `users.csv` και `ratings.csv` με τις νέες πληροφορίες και προσθέτει μια εργασία για να εκπαιδεύσει ξανά το μοντέλο.

Η συνάρτηση επιστρέφει το νέο ID χρήστη.

```
def get_user_ratings(user_id):
    ratings = pd.read_csv('data/ratings.csv')
    users = pd.read_csv('data/users.csv')
    books = pd.read_csv('data/books.csv')

    user_ratings = ratings[(ratings['user_id'] == user_id) & (ratings['cold_start'] ==
False)]
    rating_count = users[users['user_id'] == user_id]['rating_count'].values[0]

    if not user_ratings.empty:
        books = books[['id', 'title']]
        user_ratings = user_ratings.merge(books, left_on='book_id', right_on='id')
        user_ratings = user_ratings.drop(columns=['user_id'])

    return user_ratings, int(rating_count)
```

Η συνάρτηση `**get_user_ratings(user_id)**` δέχεται ως είσοδο ένα αναγνωριστικό χρήστη (`user_id`) και επιστρέφει τις αξιολογήσεις που έχει κάνει ο χρήστης αυτός, καθώς και τον συνολικό αριθμό των αξιολογήσεων που έχει κάνει.

Η συνάρτηση αρχικά φορτώνει τα δεδομένα από τα αρχεία `ratings.csv`, `users.csv` και `books.csv`. Στη συνέχεια, λαμβάνει τις αξιολογήσεις του χρήστη από το DataFrame `ratings`, αγνοώντας τις αξιολογήσεις που δημιουργήθηκαν κατά το (cold start). Επίσης, λαμβάνει τον συνολικό αριθμό αξιολογήσεων του χρήστη από το DataFrame `users`. Αν ο χρήστης έχει κάνει αξιολογήσεις, τότε η συνάρτηση συνενώνει τις αξιολογήσεις του χρήστη με το DataFrame `books` για να λάβει τους τίτλους των βιβλίων που έχει αξιολογήσει ο χρήστης, και στη συνέχεια αφαιρεί τη στήλη 'user_id'.

Τέλος, επιστρέφει τις αξιολογήσεις του χρήστη και τον συνολικό αριθμό αξιολογήσεων που έχει κάνει ο χρήστης.

```
 def background_task():
    while not stop_threads:
        task = task_queue.get()
        if task is None:
            break
        print("Running background task:", task.__name__)
        task()
        task_queue.task_done()

def schedule_retrain():
    if update_ratings_executed:
        update_ratings_executed=False
        task_queue.put(retrain_model)
        print("Retrain model executed")
    else:
        print("Update ratings function not executed yet. Skipping retrain model")

def stop_background_task(signal, frame):
    global stop_threads
    stop_threads = True
    try:
        print("Terminating the application...")
        sys.exit(0)
    finally:
        sys.exit(0)

task_thread = threading.Thread(target=background_task)
task_thread.start()

timer = threading.Timer(delay_minutes * 60, schedule_retrain)
timer.start()
```

Οι παραπάνω συναρτήσεις είναι υπεύθυνες για τη δημιουργία ενός παράλληλου thread (δηλαδή ενός διεργασίας που εκτελείται παράλληλα με την κύρια διεργασία) και για την προγραμματισμένη εκτέλεση κάποιων συναρτήσεων.

1. Η `background_task()` είναι η συνάρτηση που εκτελείται από το παράλληλο thread. Περιμένει συναρτήσεις να προστεθούν στην ουρά εργασιών και τις εκτελεί.

2. Η `schedule_retrain()` ελέγχει αν έχει εκτελεστεί η συνάρτηση `update_ratings` και αν ναι, προσθέτει την `retrain_model` στην ουρά εργασιών.

3. Η `stop_background_task(signal, frame)` είναι η συνάρτηση που καλείται όταν ο χρήστης προσπαθεί να διακόψει την εφαρμογή. Σταματά την εκτέλεση του παράλληλου thread.

Οι τελευταίες γραμμές κώδικα δημιουργούν το παράλληλο thread και τον χρονοδιακόπτη, ο οποίος καλεί τη συνάρτηση `schedule_retrain` μετά από τον καθορισμένο αριθμό λεπτών (`delay_minutes`).

```
@app.route('/')
def home():
    return render_template('home.html')
```

Αυτός το Flask route δημιουργεί μια διεύθυνση URL για την αρχική σελίδα της εφαρμογής σας. Όταν ο χρήστης επισκέπτεται τον ιστότοπο στη διεύθυνση "/", ο server επιστρέφει την αρχική σελίδα, που δημιουργείται από το αρχείο template 'home.html'.

```
@app.route('/login', methods=['POST','GET'])
def login():
    try:
        user_id = request.form['user_id']
        user = User.get_user(user_id)
        if user_id == ('0'):
            login_user(user)
            return redirect(url_for('admin_page'))
        if user is not None:
            login_user(user)
            return redirect(url_for('main'))
        else:
            return 'Invalid user_id'
    except KeyError:
        if current_user.is_authenticated:
            user_id = current_user.id
            user = User.get_user(user_id)
            if user == ('0'):
                login_user(user)
                return redirect(url_for('admin_page'))
            else:
                return redirect(url_for('main'))
        else:
            return 'No user_id provided and no user logged in'
```

Αυτή η λειτουργία είναι για τη σελίδα εισόδου στην εφαρμογή Flask σας. Όταν ο χρήστης στέλνει αίτηση POST με το 'user_id' στη διεύθυνση "/login", η εφαρμογή προσπαθεί να συνδεθεί ως ο χρήστης με αυτό το ID.

Αν το 'user_id' είναι ίσο με '0', ο χρήστης συνδέεται ως διαχειριστής και ανακατευθύνεται στην 'admin_page'. Αν ο χρήστης υπάρχει στη βάση δεδομένων, τότε ο χρήστης συνδέεται και ανακατευθύνεται στην κύρια σελίδα. Αν το 'user_id' δεν παρέχεται ή δεν υπάρχει χρήστης με αυτό το ID, τότε επιστρέφεται ένα μήνυμα λάθους.

Εάν παρουσιαστεί κάποιο σφάλμα κατά την προσπάθεια ανάκτησης του 'user_id' από τη φόρμα, τότε η εφαρμογή ελέγχει αν υπάρχει ήδη συνδεδεμένος χρήστης. Εάν υπάρχει, τότε ανακατευθύνεται στην κύρια σελίδα, ειδάλλως επιστρέφει ένα μήνυμα λάθους.

```
@app.route('/main', methods=['POST','GET'])
@login_required
def main():
    user_id = int(current_user.id)
    if user_id is not None:
        return render_template('user.html', user_id=user_id)
    return "No user id provided", 400

@app.route('/new_user', methods=['GET'])
def new_user():
    user_id = create_user()
    user = User.get_user(user_id)
    if user:
        login_user(user)

        return render_template('user.html', user_id=user_id)

    return render_template('home.html', error="Failed to create new user")
```

Αυτές οι δύο συναρτήσεις προωθούν σε διαφορετικές σελίδες της εφαρμογής Flask σας.

1. `@app.route('/main', methods=['POST','GET'])`: Αυτή η συνάρτηση προωθεί στη σελίδα "user.html" και απαιτεί τον χρήστη να έχει συνδεθεί (authenticated) με χρήση της λειτουργίας `@login_required`. Αν ο χρήστης έχει παράσχει έγκυρο user_id, η σελίδα παρουσιάζει τα δεδομένα του χρήστη με βάση το συγκεκριμένο user_id. Αν δεν παρέχεται έγκυρο user_id, επιστρέφεται το μήνυμα "No user id provided" με status code 400.

2. `@app.route('/new_user', methods=['GET'])`: Αυτή η συνάρτηση προωθεί στη σελίδα "user.html" και δεν απαιτεί σύνδεση (authentication) του χρήστη. Καλεί την συνάρτηση `create_user()` για να δημιουργήσει ένα νέο χρήστη και μετά πραγματοποιεί την σύνδεση του χρήστη με χρήση της λειτουργίας `login_user(user)`. Αν η δημιουργία και η σύνδεση του χρήστη είναι επιτυχείς, επιστρέφεται η σελίδα "user.html" με το σχετικό user_id. Αν δεν είναι επιτυχείς η δημιουργία του νέου χρήστη, επιστρέφεται η σελίδα "home.html" με ένα μήνυμα λάθους.

```
@app.route('/train_user', methods=['POST','GET'])
@login_required
def train_user():
    users = pd.read_csv('data/users.csv')
    user_id = int(current_user.id)
    book_id = None
    rating = None
    if 'book_id' in request.form:
        book_id = int(request.form.get('book_id'))
    if 'rating' in request.form:
        rating = int(request.form.get('rating'))
    if (book_id!=None) & (rating!=None):
        return update_rating(user_id, book_id, rating)
    if str(user_id) in users['user_id'].astype(str).values:
        return give_rating(user_id)
    else:
        return render_template('user.html', error=f"User ID {user_id} not found.")
```

Η παραπάνω συνάρτηση αντιστοιχεί στο (`route`) "/train_user".

Όταν γίνεται μια αίτηση GET ή POST στο "/train_user", η συνάρτηση αναλαμβάνει να εκτελέσει τις εξής ενέργειες:

1. Διαβάζει το αρχείο "users.csv" και το αποθηκεύει στη μεταβλητή `users`.

2. Ανακτά το ID του χρήστη που έχει συνδεθεί από την `current_user` και το μετατρέπει σε ακέραιο αριθμό.

3. Αρχικοποιεί τις μεταβλητές `book_id` και `rating` σε `None`.

4. Έλεγχος αν υπάρχουν οι παράμετροι "book_id" και "rating" στη φόρμα της αίτησης. Αν ναι, τότε αντιστοιχεί τις τιμές τους στις αντίστοιχες μεταβλητές `book_id` και `rating`.

5. Αν έχουν οριστεί τόσο οι `book_id` όσο και ο `rating`, τότε καλεί τη συνάρτηση `update_rating(user_id, book_id, rating)` για να ενημερώσει τη βαθμολογία ενός βιβλίου από τον συγκεκριμένο χρήστη.

6. Αν το ID του χρήστη υπάρχει στο αρχείο "users.csv", τότε καλεί τη συνάρτηση `give_rating(user_id)` για να δώσει στον χρήστη ένα βιβλίο για βαθμολόγηση.

7. Αν κανένα από τα παραπάνω δεν ισχύει, επιστρέφει τη σελίδα "user.html" με ένα μήνυμα λάθους που αναφέρει ότι το συγκεκριμένο ID χρήστη δεν βρέθηκε.

```
@app.route('/recommend', methods=['POST','GET'])
@login_required
def recommend():
        users = pd.read_csv('data/users.csv')
        try:
                user_id = int(current_user.id)
                if int(user_id) in users['user_id'].astype(int).values:
                    recommended_books = recommend_books(user_id)
                    return render_template('recommended_books.html',
books=recommended_books)
                else:
                    return render_template('user.html', error=f"User ID {user_id} not
found.")

        except:
            return render_template('user.html', error=f"User ID {user_id} not found in
Training Data")
```

Η παραπάνω συνάρτηση αντιστοιχεί στο(`route`) "/recommend

Όταν γίνεται μια αίτηση GET ή POST στο "/recommend", η συνάρτηση αναλαμβάνει να εκτελέσει τις εξής ενέργειες:

1. Διαβάζει το αρχείο "users.csv" και το αποθηκεύει στη μεταβλητή `users`.

2. Αρχικοποιεί τη μεταβλητή `user_id` με το ID του συνδεδεμένου χρήστη που ανάκτησε από την `current_user`.

3. Έλεγχος αν το `user_id` υπάρχει στο αρχείο "users.csv" ως ακέραιος αριθμός. Αν ναι, τότε συνεχίζει στο επόμενο βήμα.

4. Καλεί τη συνάρτηση `recommend_books(user_id)` για να ανακτήσει τα συνιστώμενα βιβλία για τον συγκεκριμένο χρήστη.

5. Επιστρέφει τη σελίδα "recommended_books.html" με τα συνιστώμενα βιβλία ως παράμετρο `books`.

6. Αν το `user_id` δεν υπάρχει στο αρχείο "users.csv" ως ακέραιος αριθμός, επιστρέφει τη σελίδα "user.html" με ένα μήνυμα λάθους που αναφέρει ότι το συγκεκριμένο ID χρήστη δεν βρέθηκε.

```
@app.route('/view_profile', methods=['GET','POST'])
@login_required
def view_profile():
    user_id = int(current_user.id)
    user_ratings, rating_count = get_user_ratings(user_id)

    page = request.args.get('page', default=1, type=int)
    per_page = 5
    total_pages = math.ceil(len(user_ratings) / per_page)
    paginated_ratings = user_ratings[(page-1)*per_page: page*per_page]

    return render_template('profile.html', user_id=user_id,
ratings=paginated_ratings.to_dict('records'),
                           rating_count=rating_count, page=page,
total_pages=total_pages)
```

Η παραπάνω συνάρτηση αποτελεί τον το route ("/view_profile")

1. Ανάκτηση του ID του συνδεδεμένου χρήστη από τη μεταβλητή `current_user` και μετατροπή του σε ακέραιο αριθμό.

2. Κλήση της συνάρτησης `get_user_ratings(user_id)` για να ανακτήσει τις βαθμολογίες του χρήστη και τον αριθμό των βαθμολογιών.

3. Ανάκτηση της τρέχουσας σελίδας από την παράμετρο "page" της αίτησης, με προεπιλεγμένη τιμή 1.

4. Ορισμός του αριθμού των εγγραφών που θα εμφανίζονται ανά σελίδα (per_page) και υπολογισμός του συνολικού αριθμού σελίδων (total_pages) με βάση τον αριθμό των βαθμολογιών.

5. Δημιουργία ενός υποσυνόλου των βαθμολογιών για την τρέχουσα σελίδα (paginated_ratings) με βάση τον αριθμό της σελίδας και τον αριθμό εγγραφών ανά σελίδα.

6. Επιστροφή της σελίδας "profile.html" με τις απαιτούμενες παραμέτρους, όπως το ID του χρήστη, οι βαθμολογίες της τρέχουσας σελίδας, ο αριθμός των βαθμολογιών, ο αριθμός της τρέχουσας σελίδας και ο συνολικός αριθμός σελίδων.

```
@app.route("/admin")
@login_required
def admin_page():
    user_id = int(current_user.id)
    print(user_id)
    if user_id == 0:
        print('success')
        return render_template("admin.html")
    else:
        return "Access denied"
```

Η παραπάνω συνάρτηση αντιστοιχεί στο route ("/admin")

Όταν γίνεται μια αίτηση στο "/admin", η συνάρτηση αναλαμβάνει να εκτελέσει τις εξής ενέργειες:

1. Ανάκτηση του ID του συνδεδεμένου χρήστη από τη μεταβλητή `current_user` και μετατροπή του σε ακέραιο αριθμό.

2. Έλεγχος εάν το ID του χρήστη είναι μηδέν. Αυτό υποδηλώνει ότι ο συνδεδεμένος χρήστης είναι ο διαχειριστής.

3. Εάν ο συνδεδεμένος χρήστης είναι ο διαχειριστής, επιστρέφεται η σελίδα "admin.html".

4. Αλλιώς, επιστρέφεται το μήνυμα "Access denied", υποδηλώνοντας ότι ο συνδεδεμένος χρήστης δεν έχει πρόσβαση στη σελίδα διαχειριστή.

```
@app.route('/enqueue_task', methods=['GET'])
def enqueue_task():
    if task_queue.qsize() >= max_q_size:
        return "Task queue is full. Retry later."

    while task_queue.qsize() >= max_q_size:
        task_queue.get()
    task_queue.put(retrain_model)
    return render_template('admin.html', error=f"Train started")
```

Η παραπάνω συνάρτηση αντιστοιχεί στο route "/enqueue_task".

Όταν γίνεται μια αίτηση GET στο "/enqueue_task", η συνάρτηση εκτελεί τις εξής ενέργειες:

1. Έλεγχος εάν το μέγεθος της ουράς task_queue είναι μεγαλύτερο ή ίσο από το max_q_size. Αν είναι, επιστρέφεται το μήνυμα "Task queue is full. Retry later." για να υποδείξει ότι η ουρά έχει γεμίσει και πρέπει να δοκιμάσετε ξανά αργότερα.
2. Ελέγχεται επαναληπτικά αν το μέγεθος της ουράς είναι μεγαλύτερο ή ίσο από το max_q_size. Εάν είναι, αφαιρείται το πρώτο στοιχείο της ουράς με την `task_queue.get()` μέχρι η ουρά να είναι κάτω από το max_q_size.

3. Το retrain_model προστίθεται στην ουρά task_queue με την `task_queue.put(retrain_model)`.
4. Επιστρέφεται η σελίδα "admin.html" με ένα μήνυμα λάθους που υποδεικνύει ότι η εκπαίδευση ξεκίνησε.

Αυτή η συνάρτηση επιτρέπει την εκκίνηση μιας εκπαιδευτικής διαδικασίας (retrain) προσθέτοντας το retrain_model στην ουρά task_queue.

```
@app.route('/logout', methods=['POST','GET'])
@login_required
def logout():
    logout_user()
    return redirect(url_for('home'))

@login_manager.user_loader
def load_user(user_id):
    users = pd.read_csv('data/users.csv')

    if str(user_id) in users['user_id'].astype(str).values:
        return User(user_id)
    else:
        return None

if __name__ == "__main__":
    app.run(debug=True,port=5000)
```

Η συνάρτηση `logout()` είναι ένα route που αντιστοιχεί στη διεύθυνση "/logout" στην εφαρμογή Flask. Αυτή η διαδρομή υποστηρίζει μεθόδους POST και GET. Όταν γίνεται αίτηση GET ή POST στη διεύθυνση "/logout", ο χρήστης αποσυνδέεται με τη χρήση της συνάρτησης `logout_user()` και ανακατευθύνεται στη σελίδα "home" με τη χρήση της συνάρτησης `redirect(url_for('home'))`.

Η συνάρτηση `load_user(user_id)` είναι μια συνάρτηση φόρτωσης χρήστη που χρησιμοποιείται από το Flask-Login για να φορτώσει έναν χρήστη με βάση τον παρεχόμενο αναγνωριστικό χρήστη (user_id). Αυτή η συνάρτηση αναζητά τον χρήστη στο αρχείο "users.csv" και επιστρέφει ένα αντικείμενο τύπου User αν βρεθεί ο χρήστης, διαφορετικά επιστρέφει None.

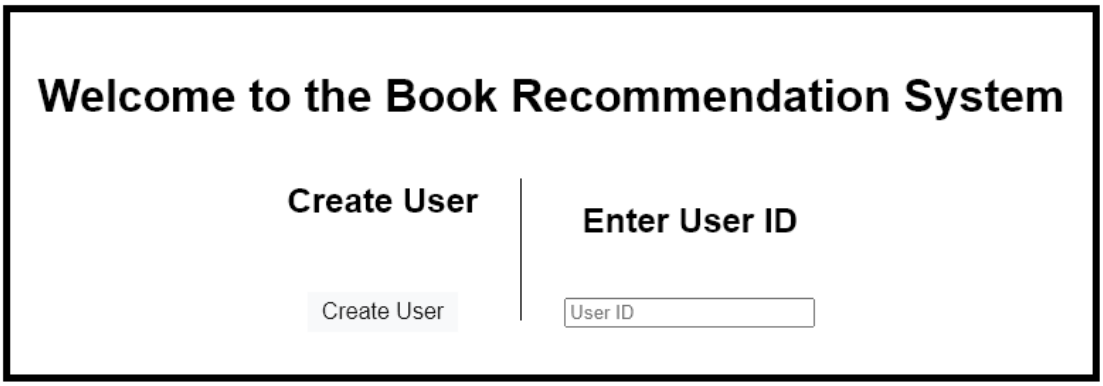
Τέλος, η συνάρτηση `app.run()` χρησιμοποιείται για να ξεκινήσει τον εξυπηρετητή Flask και να ακούει για εισερχόμενα αιτήματα στη θύρα 5000.

/Templates

Ο φακελος templates περιέχει διάφορα αρχεία HTML που καθορίζουν τη δομή και τη διάταξη διάφορων σελίδων στην εφαρμογή. Αυτά περιλαμβάνουν τα `admin.html`, `home.html`, `profile.html`, `rate_book.html`, `recommended_books.html`, και `user.html`. Κάθε ένα από αυτά τα αρχεία αντιστοιχεί σε ένα διαφορετικό τμήμα της ιστοσελίδας, όπως η αρχική σελίδα, τα προφίλ των χρηστών, το σύστημα βαθμολόγησης βιβλίων, οι προτάσεις βιβλίων.

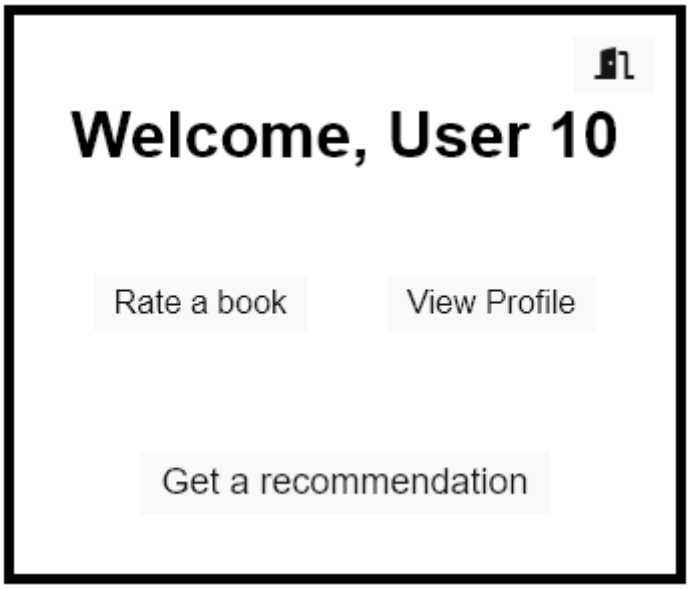
Home.html

```
<html>
<head>
<title>Book Recommendation System</title>
 <style>
  (……)
 </style>
</head>
<body>
  <div class="container">
    <h1>Welcome to the Book Recommendation System</h1>
    <div class="option-row">
      <div class="option-item">
        <h2>Create User</h2>
        <form action="/new_user" method="GET">
          <button type="submit">Create New User</button>
        </form>
      </div>
      <div class="separator"></div>
      <div class="option-item">
        <h2>Enter User ID</h2>
        <form method="POST" action="/login">
            <input type="text" name="user_id" placeholder="User ID">
            <input type="submit" value="Login">
        </form>
     </div>
    </div>
    {% if error %}
    <p class="error-message">{{ error }}</p>
    {% endif %}
  </div>
</body>
</html>
```



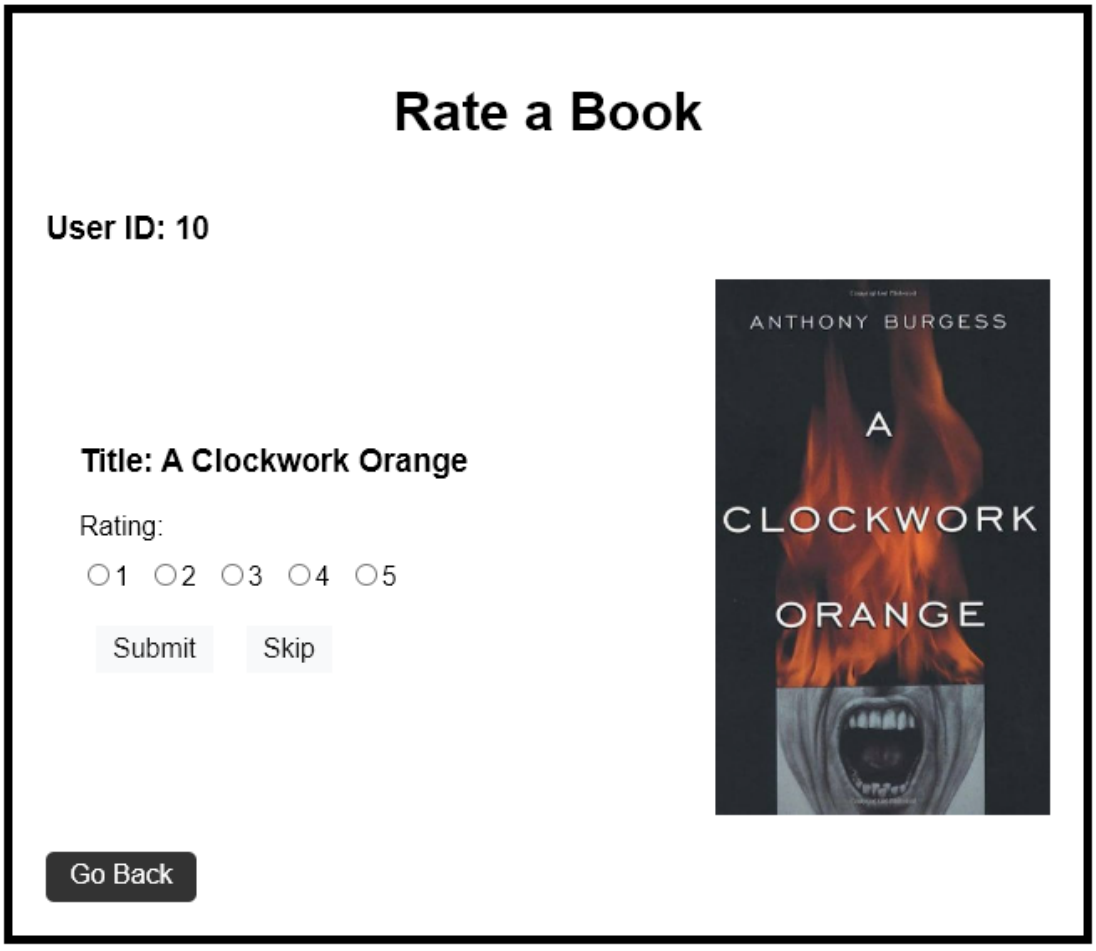
user.html

```
<html>
<head>
<title>User Page</title>
 <style>
  (……)
 </style>
</head>
<body>
  <div class="container">
    <h1>Welcome, User {{user_id}}</h1>
    <form class="logout-button" action="/logout" method="POST">
      <button type="submit">Logout</button>
    </form>
    <div class="option-row">
      <div class="option-item">
        <form action="/train_user" method="post">
          <button type="submit">Rate a book</button>
        </form>
      </div>
      <div class="option-item">
        <form action="/view_profile" method="POST">
          <button type="submit">View Profile</button>
        </form>
      </div>
    </div>
    <div class="option-row">
      <div class="option-item">
        <form action="/recommend" method="POST">
          <button class="recommend-button" type="submit">Get a recommendation</button>
        </form>
      </div>
    </div>
    {% if error %}
    <p class="error-message">{{ error }}</p>
    {% endif %}
  </div>
</body>
</html>
```



rate_book.html

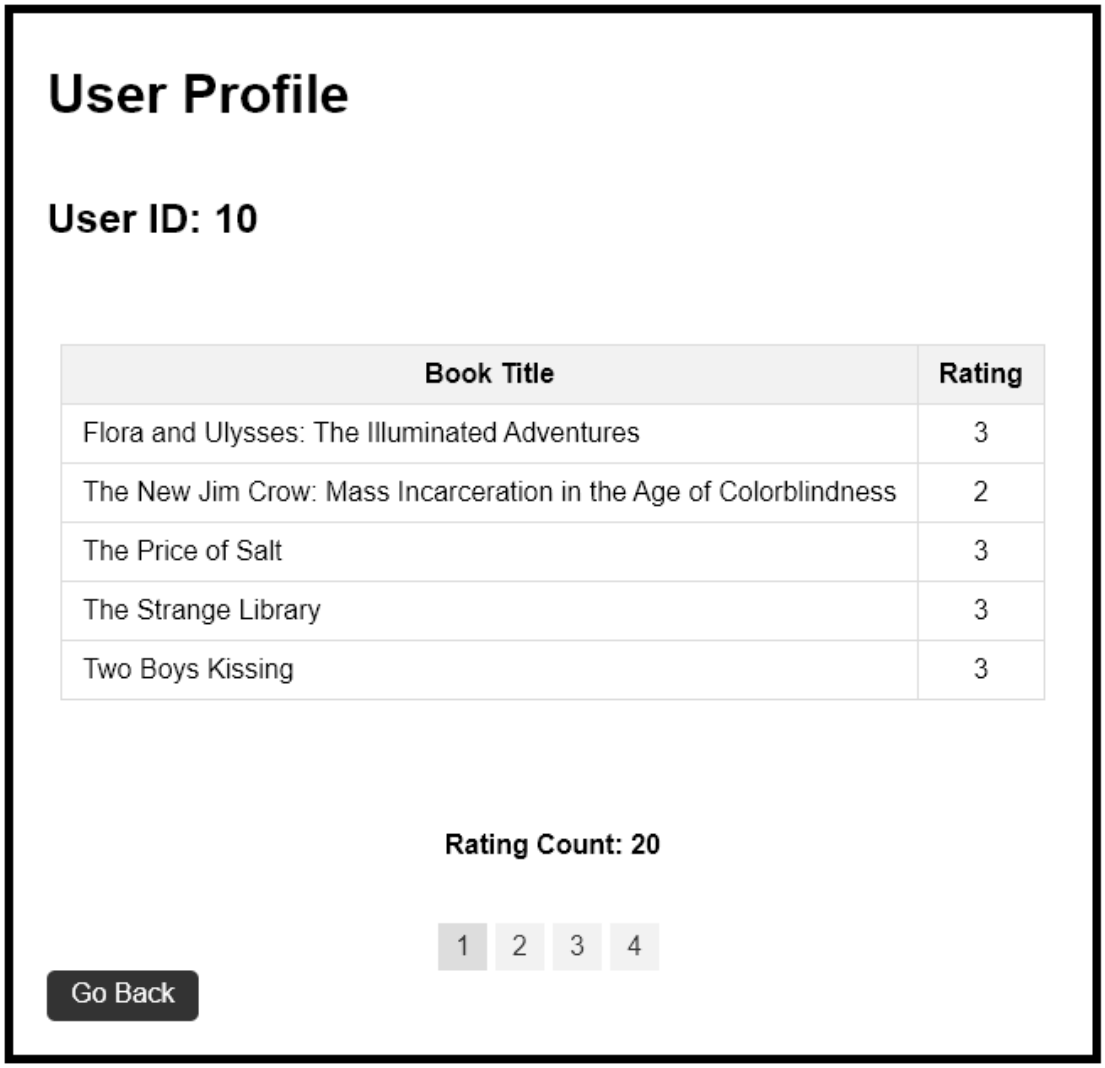
```
<html>
<head>
 <style>
  (……)
</style>
</head>
<body>
  <div class="container">
    <div class="Rate-a-book">
     <h1>Rate a Book</h1>
    </div>
    <h3>User ID: {{ user_id }}</h3>
      <div class="book-container">
        <div class="book-details">
          <h3>Title: {{ book_title }}</h3>
          <form id="rating-form" action="/train_user" method="POST">
            <input type="hidden" name="user_id" value="{{ user_id }}">
            <input type="hidden" name="book_id" value="{{ book_id }}">
            <input type="hidden" name="book_title" value="{{ book_title }}">
            <input type="hidden" name="rating_count" value="{{ rating_count }}">
            <label for="rating">Rating:</label>
            <ul class="rating-list">
              <li><input type="radio" name="rating" id="rating1" value="1"><label
for="rating1">1</label></li>
              <li><input type="radio" name="rating" id="rating2" value="2"><label
for="rating2">2</label></li>
              <li><input type="radio" name="rating" id="rating3" value="3"><label
for="rating3">3</label></li>
              <li><input type="radio" name="rating" id="rating4" value="4"><label
for="rating4">4</label></li>
              <li><input type="radio" name="rating" id="rating5" value="5"><label
for="rating5">5</label></li>
            </ul>
            <div class="button-container">
              <button type="submit">Submit</button>
              <form action="/train_user" method="POST">
                <input type="hidden" name="user_id" value="{{ user_id }}">
                <button type="submit">Skip</button>
              </form>
            </div>
          </form>
        </div>
        <div class="book-image">
          <img src="{{ image_url }}" alt="Book Cover">
        </div>
      </div>
      <br>
      <a href="/main" class="button">Go Back</a>
  </div>
</body>
</html>
```



profile.html

```
<html>
<head>
<title>User Profile</title>
 <style>
  (……)
</style>
</head>
<body>
  <div class="container">
    <h1>User Profile</h1>
     <h2>User ID: {{ user_id }}</h2>
    <table>
      <tr>
        <th>Book Title</th>
        <th>Rating</th>
      </tr>
      {% for rating in ratings %}
      <tr>
        <td>{{ rating.title }}</td>
        <td style="text-align: center;">{{ rating.rating }}</td>
      </tr>
      {% endfor %}
     </table>
    <h4 style="text-align: center;"><p>Rating Count: {{ rating_count }}</p></h4>
    <div class="pagination">
      {% for num in range(1, total_pages + 1) %}
        {% if num == page %}
          <a class="active" href="{{ url_for('view_profile', page=num) }}">{{ num }}</a>
```

```
        {% else %}
          <a href="{{ url_for('view_profile', page=num) }}">{{ num }}</a>
        {% endif %}
      {% endfor %}
    </div>
    <a href="/main" class="button">Go Back</a>
  </div>
</body>
</html>
```



User Profile

User ID: 10

Book Title	Rating
Flora and Ulysses: The Illuminated Adventures	3
The New Jim Crow: Mass Incarceration in the Age of Colorblindness	2
The Price of Salt	3
The Strange Library	3
Two Boys Kissing	3

Rating Count: 20

1 2 3 4

Go Back

recommended_books.html

```
<html>
<head>
<title>Recommended Books</title>
 <style>
  (……)
  </style>
</head>
<body>
  <div class="container">
  <h2>Recommended Books</h2>
  <ul>
    {% for book in books %}
      <li>
        <img src="{{ book.image }}" alt="Book Cover">
        <div>
          <a href="{{ book.amazon_link }}" target="_blank" class="book-title">{{
book.title }}</a>
        </div>
      </li>
    {% endfor %}
  </ul>

  <a href="/main" class="button">Go Back</a>
</div>
</body>
</html>
```

admin.html

```
<html>
<head>
<title>Recommended Books</title>
 <style>
  (……)
 </style>
</head>
<body>
  <div class="container">
    <h1>Admin - Page</h1>

    <form class="logout-button" action="/logout" method="POST">
      <button type="submit">Logout</button>
    </form>

    <div class="option-row">
        <div class="option-item">
          <form action="/enqueue_task" method="GET">
            <button class="force-retrain-button" type="submit">Force Retrain
Model</button>
          </form>
        </div>
      </div>

    {% if error %}
    <p class="error-message">{{ error }}</p>
    {% endif %}
  </div>
</body>
</html>
```

Βιβλιογραφία

[1] P. Bedi, R. Sharma, και H. Kaur, 'Recommender System Based on Collaborative Behavior of Ants', στο *Journal of Artificial Intelligence*, 2022.

[2] J. A. Konstan και M. D. Ekstrand, 'Introduction to Recommender Systems', *COURSERA*, 2016. https://www.coursera.org/learn/recommender-systems-introduction (ημερομηνία πρόσβασης 18 Απρίλιος 2021).

[3] F. Ricci, L. Rokach, B. Shapira, και P. B. Kantror, *Recommender Systems Handbook*. 2011.

[4] REJOINER, 'The Amazon Recommendations Secret to Selling More Online'. https://rejoiner.com/resources/amazon-recommendations-secret-selling-online/ (ημερομηνία πρόσβασης 27 Μάιος 2021).

[5] R. K. Singh και R. Sharma, 'Evolution of Recommender Systems from Ancient Times to Modern Era: A Survey', *researchgate.net*, Μάιος 2016. https://www.researchgate.net/publication/303953909_Evolution_of_Recommender_Systems_from_Ancient_Times_to_Modern_Era_A_Survey (ημερομηνία πρόσβασης 1 Ιούνιος 2021).

[6] Y. Koren, R. Bell, και C. Volinksy, 'Matrix factorization techniques for recommender system', *IEEE Computer Society*, Αύγουστος 2009. https://datajobs.com/data-science-repo/Recommender-Systems-[Netflix].pdf (ημερομηνία πρόσβασης 29 Σεπτέμβριος 2022).

[7] C. Underwood, 'Use Cases of Recommendation Systems in Business – Current Applications and Methods', *emerj.com*, 4 Μάρτιος 2020. https://emerj.com/ai-sector-overviews/use-cases-recommendation-systems/ (ημερομηνία πρόσβασης 24 Ιούνιος 2021).

[8] P. Covington, J. Adams, και E. Sargin, *Deep Neural Networks for YouTube Recommendations*.

[9] I. Guy, 'Social Recommender Systems', στο *Recommender Systems Handbook*, Second edition.Boston: Springer, 2015.

[10] J. Wang, Y. Zhang, C. Posse, και A. Bhasin, 'Is It Time for a Career Switch?', στο *Proceedings of the 22nd International Conference on World Wide Web*, στο WWW '13. New York, NY, USA: Association for Computing Machinery, 2013, σσ. 1377–1388. doi: 10.1145/2488388.2488509.

[11] X. Amatriain και J. Basilico, 'Recommender Systems in Industry', στο *Recommender Systems Handbook*, Second edition.2015.

[12] F. Ricci, R. Lior, και B. Shapira, 'Recommender Systems: Introduction and Challenges', στο *Recommender Systems Handbook*, Second edition.2015.

[13] F. Ricci, *Travel recommender systems*. 2002.

[14] J. Borràs, A. Moreno, και A. Valls, 'Intelligent tourism recommender systems: A survey', στο *Expert Systems with Applications*, 2014.

[15] J. Herlocker, J. Konstan, L. Terveen, και J. Riedl, ‘Evaluating collaborative filtering recommender systems’, στο *ACM Transaction on Information Systems*, 2004.

[16] M. Kaminskas και F. Ricci, ‘ Contextual music information retrieval and recommendation: State of the art and challenges’, στο *Computer Science Review*, 2012.

[17] F. Ricci, D. Cavada, N. Mirzadeh, και A. Venturini, ‘Case-based travel recommendations’, στο *Destination Recommendation Systems: Behavioural Foundations and Applications*, 2006.

[18] P. Brusilovsky, ‘ Methods and techniques of adaptive hypermedia’, στο *User Modeling and User-Adapted Interaction*, 1996.

[19] A. Gunawardana και G. Shani, ‘Evaluating Recommender Systems’, στο *Recommender Systems Handbook,* Second edition.2015.

[20] C. C. Aggarwal, ‘Evaluating Recommender Systems’, στο *Recommender Systems The Textbook*, 2016, σσ. 246–275.

[21] S. Saumyadeepta, ‘Evaluating Recommender Systems. Part 5 of Evaluation Metrics Series | by Saumyadeepta Sen | The Owl | Medium’, 21 Ιούλιος 2020. https://medium.com/the-owl/evaluating-recommender-systems-749570354976 (ημερομηνία πρόσβασης 14 Οκτώβριος 2022).

[22] F. Kane, *Building Recommender Systems with Machine Learning and AI*. 2018.

[23] C. C. Aggarwal, *Recommender Systems The Textbook*. Springer, 2016.

[24] Muvi, ‘The Difference Between Collaborative and Content Based Recommendation Engines | by Muvi | Medium’, 20 Μάιος 2021. https://muvi-com.medium.com/the-difference-between-collaborative-and-content-based-recommendation-engines-ade24b5147f2 (ημερομηνία πρόσβασης 13 Οκτώβριος 2022).

[25] R. Burke, ‘Knowledge-Based Recommender Systems’, *Encyclopedia of library and information systems*, τ. 69, Μαΐου 2000.

[26] C. C. Aggarwal, ‘An Introduction to Recommender Systems’, στο *Recommender Systems The Textbook*, 2016, σσ. 23–50.

[27] Y. Koren και R. Bell, ‘Advances in Collaborative Filtering’, στο *Recommender Systems Handbook,* Second edition.2015.

[28] D. W. Oard και J. Kim, ‘Implicit Feedback for Recommender Systems’, στο *Workshop on Filtering and Collaborative Filtering*, 1998.

[29] P. Mishra και S. N. Mohanty, ‘Collaborative Filtering Techniques’, στο *Recommender Systems algorithms and applications*, 2021.

[30] A. Mishra, M. K. Gourisaria, και S. S. Patra, ‘Model-Based Filtering Systems Using a Latent-Factor Technique’, στο *Recommender Systems algorithms and applications*, 2021.

[31] D. Jannach, M. Zanker, A. Felfernig, και G. Friedrich, *Recommender Systems An Introduction*. Cambridge University Press, 2011.

[32] C. C. Aggarwal, ‘Content-Based Recommender Systems’, στο *Recommender Systems The Textbook*, 2016, σσ. 161–188.

[33] B. Liu, *data mining: exploring hyperlinks, contents, and usage data*. New York: Springer, 2007.

[34] C. C. Aggarwal, ‘Knowledge-based recommendation’, στο *Recommender Systems The Textbook*, 2016, σσ. 189–221.

[35] A. Felfernig, G. Friedrich, D. Jannach, και M. Zanker, ‘Developing constraint-based recommenders’, στο *Recommender Systems Handbook*, First edition.Springer, 2011, σσ. 187–216.

[36] A. Felfernig και R. Burke, ‘Constaint-based recommender systems’, στο *International conference on Electronic Commerce*, 2008.

[37] D. BRIDGE, M. H. GÖKER, L. McGINTY, και B. SMYTH, ‘Case-based recommender systems’, *Knowl Eng Rev*, τ. 20, τχ. 3, σσ. 315–320, 2005, doi: DOI: 10.1017/S0269888906000567.

[38] ‘What Is Deep Learning? | How It Works, Techniques & Applications - MATLAB & Simulink’. https://www.mathworks.com/discovery/deep-learning.html (ημερομηνία πρόσβασης 3 Νοέμβριος 2022).

[39] A. Fischer και C. Igel, ‘LNCS 7441 - An Introduction to Restricted Boltzmann Machines’.

[40] ‘Introduction to autoencoders.’ https://www.jeremyjordan.me/autoencoders/ (ημερομηνία πρόσβασης 3 Νοέμβριος 2022).

[41] Zygmunt Zając, ‘goodbooks-10k’. https://github.com/zygmuntz/goodbooks-10k (ημερομηνία πρόσβασης 12 Ιούνιος 2023).