ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΚΑΤΑΝΕΜΗΜΕΝΑ ΣΥΣΤΗΜΑΤΑ

Εαρινό Εξάμηνο 2017-2018

Εργασία μαθήματος

Τα τελευταία χρόνια με την μεγάλη αύξηση της χρήστης των κινητών τηλεφώνων και των κοινωνικών μέσων, ένα νέο ειδών κοινωνικών δικτύων έχει δημιουργηθεί, τα λεγόμενα Location Based Social Networks (**LBSN**) όπως το Foursquare, Swarm, Facebook Places, Google Places κ.τ.λ. Σε αυτά τα κοινωνικά δίκτυα οι χρήστες μπορούν να κάνουν checkin σε διάφορα σημεία ενδιαφέροντος (**POI**) μέσα στην πόλη. Αυτά τα checkin συνήθως δεν έχουν κάποια βαθμολογία αλλά δηλώνουν τη “τάση” προτίμησης για κάποιο POI μέσω του πλήθους των checkin που έχουν γίνει σε αυτό. Έτσι, μέσω των **LBSN** παράγεται ένας μεγάλος όγκος πληροφορίας ο οποίος μπορεί να χρησιμοποιηθεί για διαφορετικούς σκοπούς.

Σε αυτή την εργασία καλείστε να χρησιμοποιήσετε τέτοιου είδους πληροφορία με σκοπό να υλοποιήσετε ένα Collaborative Filtering (**CF**) based Recommendation System. Διαισθητικά, σε αυτά τα συστήματα εκμεταλλευόμαστε το γεγονός ότι υπάρχουν πάρα πολλοί χρήστες με παρόμοια checkins σε POIs. Έτσι για παράδειγμα αν ο χρήστης Α έχει κάνει checkin στα POIs *P1, P2, P3, P4* και ο χρήστης Β στα POIs *P1, P2, P3, P5* τότε ο σύστημα αυτό με μεγάλη πιθανότητα θα προτείνει το POI P4 στον χρήστη Β. Οι τεχνικές για Recommendation χωρίζονται σε 2 βασικές κατηγορίες: Recommendation systems με explicit και implicit feedback. Στην πρώτη κατηγορία οι χρήστες δηλώνουν ξεκάθαρη προτίμηση για κάθε POI που έχουν επισκεφθεί, δηλαδή βαθμολογούν το κάθε POI με ένα βαθμό π.χ. σε κλίμακα από 1-5. Στη δεύτερη κατηγορία, που είναι και αυτή με την οποία θα ασχοληθούμε, δεν έχουμε πλέον βαθμολογίες για το κάθε POI, αλλά τον αριθμό των checkins που έχει κάνει ο κάθε χρήστης σε αυτό. Να σημειωθεί, ότι εάν ένας χρήστης έχει επισκεφτεί μόνο μια φορά ένα POI δε δηλώνει αναγκαστικά ότι το POI αυτό δεν του άρεσε π.χ. μια συναυλία όσο και να αρέσει σε κάποιον, μπορεί να βρεθεί εκεί μόνο μια φορά, καθώς είναι μοναδική!

Μια πολύ διαδεδομένη κατηγορία τεχνικών, που χρησιμοποιείται για **CF** είναι το **Matrix Factorization**. Σε αυτού του τύπου τα συστήματα, τα δεδομένα μας τα αναπαριστούμε σε μορφή πίνακα. Σε αυτόν τον πίνακα, το index του αξονα Χ αντιστοιχεί στα POIs, και το index του αξονα Υ αντιστοιχεί στους χρήστες. Κάθε εγγραφή στον πίνακα αντιστοιχεί στις πόσες φορές έχει κάνει checkin ο κάθε χρήστης σε ένα POI. Πρακτικά, οι περισσότερες εγγραφές έχουν την τιμή μηδέν, καθώς οι χρήστες κάνουν checkin σε ένα πολύ μικρό υποσύνολο από το συνολικό αριθμό των POIs που υπάρχουν διαθέσιμα. Ο σκοπός των τεχνικών αυτών είναι

όλες οι μηδενικές τιμές του πίνακα να γεμίσουν με μη-μηδενικές, όπου όσο μεγαλύτερη είναι η τιμή τόσο πιο πιθανό είναι να αρέσει το συγκεκριμένο POI στον χρήστη. Αυτό επιτυγχάνεται ως εξής: Έστω ότι στο σύστημα μας έχουμε **M** χρήστες και **N POIs**. Αυτό θα μας οδηγήσει σε ένα πίνακα $X \in \Re^{N \times M}$. Ο σκοπός μας είναι με τη χρήση 2 άλλων πινάκων, πολύ μικρότερης διάστασης, να κάνουμε ένα approximation του πίνακα X. Πιο συγκεκριμένα διαλέγουμε ένα $K << max\{M, N\}$ και ορίζουμε τους πίνακες $L \in \Re^{N \times K}$ και $R \in \Re^{M \times K}$, όπου θέλουμε $X \simeq LR^T$. Οι πίνακες L και R αρχικά έχουν μέσα τυχαίες τιμές. Αυτό που θέλουμε να κάνουμε εμείς είναι να τους “εκπαιδεύσουμε” έτσι ώστε να προσεγγίσουν τον πίνακα X όσο το δυνατόν περισσότερο. Λόγω όμως την μειωμένης διάστασης των πινάκων L και R, ο χρήστες και τα POIs θα “μοιραστούν” αυτόν το μειωμένο χώρο, και έτσι θα φανερωθούν κρυφές σχέσεις μεταξύ τους.

Στην βιβλιογραφία υπάρχουν διάφοροι αλγόριθμοι για matrix factorization, εμείς θα ασχοληθούμε με τον αλγόριθμο που παρουσιάζεται στο paper [1]. Προκειμένου να μπορέσουμε να εκπαιδεύσουμε τους πίνακες, χρησιμοποιούμε μια τεχνική που λέγεται Alternating Least Squares. Σε αυτήν την τεχνική, σε κάθε “βήμα” της εκπαίδευσης θεωρούμε γνωστό τον ένα από τους δύο πίνακες L, R και εκπαιδεύουμε τον άλλο. Αυτό γίνεται παίρνοντας την παράγωγο του X=LR, μια φορά θεωρώντας γνωστό το L, και μια θεωρώντας γνωστό το R. Περισσότερες λεπτομέρειες για το πως θα γίνει αυτή διαδικασία θα παρουσιαστούν στο αντίστοιχο εργαστήριο του μαθήματος και μπορούν να διαβαστούν και από το paper που σας έχει παρατεθεί.

Η διαδικασία του Matrix Factorization είναι μια “βαριά” διαδικασία από θέμα CPU, αλλά κυρίως από θέμα διαθέσιμης RAM. Επίσης, είναι μια διαδικασία που σε ένα single μηχάνημα μπορεί να διαρκέσει ώρες ή ακόμα και μέρες για να ολοκληρωθεί. Στην εργασία αυτή, καλείστε να υλοποιήσετε μια κατανεμημένη έκδοση αυτού του αλγορίθμου, όπου ο κάθε processing node θα λαμβάνει φόρτο ανάλογα με την διαθέσιμη CPU και RAM που διαθέτει, έτσι ώστε το σύστημα να παρουσιάζει την μέγιστη δυνατή επίδοση. Στη συνέχεια, καλείστε να υλοποιήσετε μια εφαρμογή Android στην οποία, ανάλογα με την τοποθεσία του χρήστη, θα εμφανίζονται τριγύρω του, τα top-k σημεία ενδιαφέροντος τα οποία έχει να προτείνει το recommendation system που έχετε υλοποιήσει. Πιο συγκεκριμένα, έχετε να υλοποιήσετε τα παρακάτω σε 2 φάσεις:

Παραδοτέο Α - (Ημερομηνία Παράδοσης: 15/4/2018 - 23:55)

Ένα πλήρες, λειτουργικό και κατανεμημένο Recommendation System. Αυτό το σύστημα θα αποτελείται από ένα Master node και από ένα πλήθος Worker nodes. Ο Master node θα είναι υπεύθυνος για να διαμοιράζει κάθε φορά τον φόρτο εργασίας στους Worker nodes καθώς θα επίσης θα είναι αυτός που θα παράγει τα Recommendations κάθε φορά που θα του ζητούνται. Οι worker nodes, όταν θα εισέρχονται στο σύστημα θα ενημερώνουν τον Master Node για τα χαρακτηριστικά

τους (αριθμό CPU nodes/ διαθέσιμη RAM) και θα περιμένουν να λάβουν εργασία από τον Master Node.

Bonus(+10%): Ο Master Node, όταν αντιληφθεί ότι κάποιος Worker καθυστερεί το σύστημα, κάνει rebalance το σύστημα έτσι ώστε να μεγιστοποιηθεί η απόδοση.

Παραδοτέο Β - (Ημερομηνία Παράδοσης: 27/5/2018 - 23:55)

Μια εφαρμογή Android, η οποία ανάλογα με την τοποθεσία του χρήστη, θα κάνει ένα ερώτημα στο Master Node, έτσι ώστε αυτός να του στείλει τα top-k recommenations γύρω από την τοποθεσία του χρήστη. Τα προτεινόμενα **POIs** θα πρέπει εμφανίζονται στην οθόνη του κινητού του χρήστη μαζί με τα στοιχεία του κάθε **POI** σε μορφή marker.

Γλώσσα υλοποίησης: Java 8

Paper:

[1] Hu, Yifan, et al. “Collaborative Filtering for Implicit Feedback Datasets.” *2008 Eighth IEEE International Conference on Data Mining*, 2008, doi:10.1109/icdm.2008.22.

Αλγόριθμος Matrix Factorization

Έστω ο πίνακας P ο οποίος περιέχει τις τιμές των Checkin σε POIs. Ο στόχος μας είναι να βρούμε τους πίνακες X και Y οι οποίοι θα ελαχιστοποιήσουν την παρακάτω cost function.

$$\min_{x_*,y_*} \sum_{u,i} c_{ui}(p_{ui} - x_u^T y_i)^2 + \lambda \left(\sum_u \|x_u\|^2 + \sum_i \|y_i\|^2 \right)$$

Ο πίνακας P είναι ένας binary πίνακας όπου παίρνει την τιμή 1 όπου ο αρχικός πίνακας μας έχει τιμή μεγαλύτερη του μηδενός και ο πίνακας C είναι ένας πίνακας που προκύπτει από την παρακάτω σχέση:

$$c_{ui} = 1 + \alpha r_{ui}$$

όπου α=40.

Η ποσότητα $\lambda \left(\sum_u \|x_u\|^2 + \sum_i \|y_i\|^2 \right)$ είναι απαραίτητη για την κανονικοποίηση του μοντέλου έτσι ώστε να αποφύγουμε το overfitting. Το overfitting είναι ένα γνωστό πρόβλημα που συναντάμε στα Recommendation Systems. Πιο αναλυτικά, αν θέλαμε να κάνουμε απλά minimize την ποσότητα

$$\min_{x_*, y_*} \sum_{u,i} c_{ui}(p_{ui} - x_u^T y_i)^2$$

θα γινόταν το εξής: Οι πίνακες X και Y θα προσπαθούσαν απλά να πλησιάσουν τις τιμές του πίνακα P και για να επιτύχουν αυτό θα πάρουν ακραίες τιμές στο εσωτερικό τους. Αυτό συνεπάγεται ότι οι υπόλοιπες μη γνωστές τιμές, θα πάρουν τιμές πολύ κοντά στο 0. Τέλος, είναι πολύ πιθανό οι τιμές στους πίνακες X και Y να πάρουν τόσο ακραίες τιμές που το πρόγραμμα που θα φτιάξουμε να κολλήσει καθώς οι τιμές μέσα στους πίνακες μπορεί να περάσουν το όριο που χωράει ένας double. Για να λύσουμε το πρόβλημα αυτό, προσθέτουμε την ποσότητα $\lambda \left(\sum_u \|x_u\|^2 + \sum_i \|y_i\|^2 \right)$ στην cost function. Αν προσέξουμε, αυτό που πρακτικά συμβαίνει είναι ότι βάζουμε ένα penalty στο μέγεθος των τιμών που μπορούν να πάρουν οι πίνακες X και Y.

Όπως είπαμε πριν, ο τρόπος με τον οποίο γίνεται το Matrix Factorization, λέγεται Alternating Least Squares. Αυτό που κάνουμε για να ελαχιστοποιήσουμε την cost function είναι να παραγωγίζουμε τη συνάρτηση αυτό μια φορά ως προς X θεωρώντας γνωστό το Y και μια ως προς Y θεωρώντας γνωστό το X. Έτσι, μετά από ένα μεγάλο πλήθος επαναλήψεων θα παρατηρήσουμε ότι το κόστος της συνάρτησης (1) θα σταθεροποιηθεί. Οι κανόνες με οι οποίοι προκύπτουν μετά την παραγώγιση είναι οι παρακάτω:

$$x_u = (Y^T C^u Y + \lambda I)^{-1} Y^T C^u p(u)$$

και

$$y_i = (X^T C^i X + \lambda I)^{-1} X^T C^i p(i)$$

Χρησιμοποιώντας αυτούς τους κανόνες, υπολογίζουμε τους πίνακες για τα U (Users) και όλα τα I (POIs). Πιο συγκεκριμένα, θεωρώντας γνωστό τον πίνακα Y, υπολογίζουμε όλα τα x_u και με το που τελειώσει αυτή η διαδικασία υπολογίζουμε όλα τα y_u θεωρώντας γνωστό τον πίνακα X. Με το που γίνει αυτή η διαδικασία μια φορά, θεωρούμε ότι ο αλγόριθμος εκπαίδευσης έχει κάνει μια επανάληψη. Μέτα από κάθε επανάληψη μπορούμε να μετρήσουμε την cost function και να δούμε κατά πόσο έχει μειωθεί. Η εκπαίδευση συνήθως σταματάει με 2 κριτήρια. Πρώτο, όταν η cost function μεταξύ δύο επαναλήψεων δώσει διαφορά μικρότερη ενός threshold που έχουμε ορίσει και δεύτερο, όταν ο αλγόριθμος ολοκληρώσει ένα συγκεκριμένο αριθμό επαναλήψεων που έχουμε ορίσει εμείς.
Προκειμένου να υπολογίσουμε κάθε recommendation χρησιμοποιούμε τον παρακάτω τύπο:

$$\hat{r}_{ui} = x_u^T y_i$$