Efficient Top-k Querying over Social-Tagging Networks

Εισαγωγή

Στην εποχή μας έχουν αναπτυχθεί διάφορες ηλεκτρονικές κοινότητες στις οποίες κοινότητες έχουμε ένα σύνολο από χρήστες που επικοινωνούν μεταξύ τους και διαφόρων ειδών περιεχόμενα τα οποία μπορούν να ανταλλάσουν και να δημοσιεύουν. Το είδος τον περιεχομένων που μπορούν που υπάρχουν είναι φωτογραφίες, video, books κτλ. (για ευκολία ονομάζουμε όλα τα περιεχόμενα αυτά έγγραφα documents). Επίσης υπάρχει δυνατότητα των χρηστών να βάζουν σημειώσεις- ετικέτες (tag) στα διάφορα έγγραφα, εκφράζοντας έτσι το προσωπικό ενδιαφέρων τους προς αυτά. Τέτοιου είδους κοινότητες έχουν ονομαστεί Social-Tagging Networks.

Σκοπός της δουλείας αυτής είναι έχοντας μια ερώτηση από έναν χρήστη να βρεθούν τα top-k αποτελέσματα συμφώνα με την ερώτηση που δόθηκε. Ο συγκεκριμένος αλγόριθμος για τον υπολογισμό των top-k αποτελεσμάτων λαμβάνει υπόψη κάποια στοιχεία που δεν λαμβάνανε προηγούμενες δουλειές που είχαν γίνει. Τα στοιχεία που πρέπει να λάβουμε υπόψη μας είναι οι σχέσεις που υπάρχουν μεταξύ των χρηστών σε ένα social network (γιατί τα αντικείμενα που κάνουν tag οι κοντινότεροι φίλοι μας να είναι αυτά που θα μας ενδιέφεραν περισσότερο (social expansion). Ένας άλλος παράγοντας που επηρεάζει τα αποτελέσματα μας είναι η σχετικότητα (ομοιότητα) των διαφορετικών ετικετών μεταξύ τους (tag expansion).

Social network model

Ένα social network αναπαριστά τις σχέσεις μεταξύ των κόμβων του γράφου. Σαν κόμβους του γράφου έχουμε τους χρήστες (users), τις ετικέτες (tags), και τα έγγραφα (documents). Και οι ακμές του γράφου αναπαριστούν τις διαφορετικές σχέσεις που υπάρχουν μεταξύ των κόμβων του γράφου. Οι σχέσεις που υπάρχουν είναι οι εξής:

- **Friendship(User1, User2, FriendshipStrength) :** Η σχέση αυτή δείχνει τον βαθμό φιλίας μεταξύ δύο χρηστών.
- **TagSimilarity(Tag1, Tag2, TagSim):** Η σχέση αυτή δείχνει τον βαθμό ομοιότητας μεταξύ δύο ετικετών.
- **Linkage(Document1, Document2, Weight) :** Η σχέση δείχνει τον βαθμό σύνδεσης μεταξύ δύο εγγράφων.
- **DocContent(Document, Tag, ContentScore):** Η σχέση δηλώνει κατά πόσο η ετικέτα περιγράφει καλά το συγκεκριμένο document.
- **Tagging(User, Tag, TagScore):** Η σχέση δείχνει την σύνδεση ενός χρήστη με την ετικέτα και το σκορ δείχνει πόσο έντονα η ετικέτα χρησιμοποιείται από έναν χρήστη.

- **Rating(User, Document, RatingScore):** Σχέση που χρησιμοποιείται για την κατάταξη ενός εγγράφου από έναν χρήστη. Η σχέση αυτή δεν χρησιμοποιείται σε όλα τα κοινωνικά μοντέλα.

Social Scoring model (Υπολογισμού του κόστους του μοντέλου)

Έστω έχουμε την ερώτηση Q(u, q_1,…,q_n) όπου u ο χρήστης και q_1,…,q_n το σύνολο από τις ετικέτες. Το αποτέλεσμα πρέπει να είναι ένα σύνολο από document ταξινομημένα σύμφωνα με το σκορ του υπολογισμού. Για τον υπολογισμό του σκορ του κοινωνικού μοντέλου λαμβάνουμε υπόψη μας τρία βασικά στοιχεία:

1. Ο χρήστης u κατά πόσο λαμβάνει υπόψη τους άλλους χρήστες(φίλους) που ανήκουν στο κοινωνικό δίκτυο (social network).
2. Την συχνότητα με την οποία οι χρήστες μπορεί να χρησιμοποιούν μια ετικέτα.
3. Την επέκταση των ετικετών με σημασιολογικά ίδιες ετικέτες.

Υπολογισμός της **Friendship Similarity**: Συμβολίζουμε την ομοιότητα αυτή με $F_u(u')$ και υπολογίζεται χρησιμοποιώντας την απόσταση των users στον γράφο και το global measure. Για τον social measure χρησιμοποιώντας επικαλυπτόμενη ομοιότητα από έναν χρήστη u και u' που συνδέονται άμεσα στον γράφο:

$$O(u,u') = \frac{2*|tagset(u \cap u')|}{|tagset(u)|+|tagset(u')|}$$

Στην συνέχεια για να υπολογίζουμε την ομοιότητα μεταξύ χρηστών που απέχουν μια διαδρομή μεταξύ τους, υπολογίζουμε την παραπάνω ομοιότητα για κάθε μονοπάτι και διαλέγουμε την αυτή με την μέγιστη ομοιότητα. Έτσι έχουμε τον παρακάτω τύπο:

$$Pu(u') = \max \prod_{i=0}^{k-1} O(u_i, u_{i+1})$$

Επίσης στην ολική friendship ομοιότητα υπολογίζουμε και ένα μικρό ποσοστό για όλους τους χρηστές που δεν συνδέονται με τον u. Συνεπώς για η τελική ομοιότητα είναι :

$$F_u(u') = \alpha * \frac{1}{|U|} + (1-a)P_u(u')$$

Υπολογισμός του **social frequency:** Η ποσότητα αυτή ορίζει εκτός τον βαθμό ομοιότητας μεταξύ των users αλλά επίσης και το πόσες φορές ένας χρήστης χρησιμοποίησε μια ετικέτα t και για ένα έγγραφο d την ποσότητα αυτή της συμβολίζουμε με $tf_{u'}(d,t)$. Έτσι έχουμε τον εξής τύπο για την social frequency.

$$sf_u(d,t) = \sum_{u' \in U} F_u(u') * tf_{u'}(d,t)$$

Αντικαθιστώντας στο τύπο παραπάνω την τιμή της friendship similarity προκύπτει ο τύπος:

$$sf_u(d,t) = \sum_{u' \in U} \left(\frac{a}{|U|} * tf_{u'}(d,t) + (1-a) * P_u(u') * tf_{u'}(d,t)\right)$$

Στο οποίον το πρώτο μέρος το ονομάζουμε non user specific γιατί δεν εξαρτάτε καθόλου από τον χρήστη που κάνει την ερώτηση επιπλέον κρατάμε και την ποσότητα TF(d,t) που είναι το άθροισμα των συχνοτήτων $tf_{u'}(d, t)$ και δηλώνει το πόσες φορές το έγγραφο d έχει γίνει tag από όλους τους χρήστες)

Υπολογισμός του **Social Score for Tags:** Για τον υπολογισμό της ομοιότητα του εγγράφου d σε σχέση με μια ετικέτα t για τον χρήστη u χρησιμοποιούμε τον απλό τύπο της μεθόδου Okapi (BM25):

$$S_u(d,t) = \frac{(k_1+1) * |U| * sf_u(d,t)}{k_1 + |U| * sf_u(d,t)} * idf(t)$$

Όπου $idf(t)$ η ανάστροφη συχνότητα του εγγράφου (inverse document frequency) για την ετικέτα t παρακάτω δίνεται και ο τύπος του. Επίσης στο τύπο έχουμε την ποσότητα $df(t)$ που ορίζεται ως το πλήθος τον εγγράφων που έχουν την ετικέτα t από τουλάχιστον έναν χρήστη:

$$idf(t) = \log \frac{|D| - df(t) + 0.5}{df(t) + 0.5}$$

Υπολογισμός του **tag similarity**: Σε πολλές περιπτώσεις όπως αναφέραμε παραπάνω πολλοί χρήστες μπορεί να βάζουν ετικέτες σε παρόμοια έγγραφα αλλά χρησιμοποιούν διαφορετικά ονόματα για να τα περιγράψουν. Έτσι στον παραπάνω τύπο του social frequency προσθέτουμε μια άλλη ποσότητα που λαμβάνει υπόψη της το πρόβλημα που αναφέραμε παραπάνω. Έτσι έχουμε :

$$s_u^*(d,t) = \max_{t' \in T} tsim(t,t') * s_u(d,t')$$

Όπου $tsim(t,t')$ η ομοιότητα μεταξύ των δύο ετικετών t και t' και ορίζεται ως:

$$tsim(t,t') = P[t'|t] = \frac{df(t \cap t')}{df(t)}$$

Τέλος το συνολικό σκορ για όλες τις ερωτήσεις (**social Score for Queries**) είναι

$$s_u^*(d, q_1 \dots q_n) = \sum_{q_1 \dots q_n} s_u^*(d, q_i)$$

Περιγραφή του Αλγορίθμου

Ο αλγόριθμος προσπαθεί να βρει τα top-k αποτελέσματα για την ερώτηση Q που του δόθηκε χρησιμοποιώντας το social score που υπολογίσαμε παραπάνω. Ο top-k αλγόριθμος που χρησιμοποιείται είναι ο TA-algorithm (threshold algorithm). Ο αλγόριθμος διατηρεί τέσσερεις λίστες.

- **DOCS(t):** Περιέχει το πλήθος τον φορών που όλοι οι χρήστες έχουν βάλει την ετικέτα t σε ένα έγγραφο (η τιμή TF(d,t) σε φθίνουσα σειρά).
- **USERDOCS(u, t):** Περιέχει το σύνολο τον εγγράφων d που έχουν γίνει tag από τον χρήστη u με μια ετικέτα t και την μεταβλητή $tf_u(d,t)$ (που συνήθως έχει την τιμή 1).
- **FRIENDS(u):** Περιέχει το σύνολο τον φίλων του χρήστη u και την ομοιότητα $Pu(u')$ ταξινομημένη σε φθίνουσα σειρά.

- **SIMTAGS(t):** Περιέχει για την ετικέτα t όλες τις παρόμοιες ετικέτες t' μαζί με την ομοιότητα τους $tsim(t,t')$ ταξινομημένη σε φθίνουσα σειρά και αυτή.

```
Procedure CONTEXTMERGE(user u,query q1…qn,a)
        For i= 1…n do
                FRIENDS[i]=FRIENDS(u);
                DOCS[i]=DOCS(qi);
        End for
        Candidates=0;
        Repeat
                For b=1…batchsize do
                        L=CHOOSENEXTLIST();
                        If l=FRIENDS[i] then
                                Read USERDOCS(FRIENDS[i],q[i])
                                Go to next entry in FRIENDS[i];
                        Else if l=DOCS[i] then
                                Read DOCS[i];
                        End if
                End for
                CHECKRANDOMACCESSES();
                If CHECKTERMINATION() then
                        Break;
                End if
        Until termination
End procedure
```

Ο αλγορίθμου εξετάζει διαδοχικά για όλες τις ετικέτες tag τις λίστες DOCS και USERDOCS των φίλων του χρήστη u και διατηρεί μια λίστα με τα υποψήφια top-k έγγραφα. Ο αλγόριθμος τερματίζει από την στιγμή που κανένα άλλο έγγραφο δεν μπορεί να ξεπεράσει αυτά που είναι ήδη στα top-k αποτελέσματα. (Να σημειώσουμε ότι δεν έχει κανένα ενδιαφέρων να εξετάσουμε τον αλγόριθμο για τις ακραίες τιμές του a. Γιατί στην περίπτωση που το a είναι 0 τότε δεν χρειάζεται να ελέγχουμε την λίστα DOCS και σε περίπτωση που a είναι ένα δεν χρειάζεται να ελέγχουμε την λίστα των φίλων και ο αλγόριθμος συμπεριφέρεται σαν απλός top-k αλγόριθμος.

Για να μειώσουμε την πρόσβαση στον δίσκο ο αλγόριθμος έχει πρόσβαση στην λίστα USERDOCS μόνο όταν είναι απαραίτητο δηλαδή όταν το σκορ από την λίστα είναι μεγαλύτερο από αυτό της DOCS λίστας. Σε κάθε επανάληψη του κύριου βρόγχου υπολογίζουμε το σκορ διαβάζουμε μια παρτίδα (batch) από τις λίστες μας για να υπολογίζουμε το υψηλότερο σκορ. Για να υπολογίσουμε το υψηλότερο σκορ κρατάμε τα εξής στοιχεία :

- Το *high[i]* από την λίστα DOCS[i] και το $s(d,t)$ θέτοντας *TF(d,t)= high[i]* και το user specific μέρος μηδέν.
- Το *highF[i]* από την λίστα FRIENDS[i] και το σκορ υπολογίζεται σύμφωνα με τον τύπο $\frac{(k_1+1)*((1-\alpha)|U|*highF[i]*maxtf(q_i))}{k_1+((1-\alpha)|U|*highF[i]*maxtf(q_i))} * idf(t)$

Ο αλγόριθμος κρατάει δυο λίστες στην μία αποθηκεύει τους top-k μέχρι στιγμής και μια άλλη για τους υποψήφιους που ακόμα μπορεί να διεκδικήσουν μια θέση στους top-k. Για κάθε υποψήφιο d_j υπολογίζει το ανω σκορ (best score) και το κάτω σκορ (worst score) . Για να υπολογίσουμε το χειρότερο σκορ

($sf_u(d,t)$) του d_j αρκεί να θέσουμε τον $TF(d_j,q_i)$ με μηδέν για όταν το έγγραφο δεν το έχουμε δει ακόμα στις λίστες για τις ετικέτες ι. Έτσι το σκορ υπολογίζεται μόνο χρησιμοποιώντας την ποσότητα $uf(d_j,q_i) = \sum_{USERDOCS(u')read\ for\ qi} P_u(u') * tf_{u'}(d_j,q_i)$. Για το best score ($sf_u(d,t)$) θέτουμε το $TF(d_j,q_i) = high[i]$ στην συνέχεια υπολογίζουμε την ποσότητα $uf(d_j,q_i)$+C όπου C είναι μια μεταβλητή για την συνεισφορά των χρηστών που δεν έχουμε δει ακόμα. Θέτουμε σαν C=(1-$mass_i$) *maxtf* όπου $mass_i = \sum_{USERDOCS(u')read\ for\ qi} P_u(u')$ και *maxtf* το μέγιστο πλήθος tag για οποιοδήποτε έγγραφο από οποιονδήποτε χρήστη.

Για τον τερματισμό του αλγορίθμου κρατάμε σε κάθε βήμα τον worst score (min-k) από τα τρέχων (current) top-k έγγραφα το οποίο είναι και το κατώφλι (threshold) για τον αλγόριθμο και το best score των υποψήφιων και εγγράφων. Ο αλγόριθμος τερματίζει όταν το best score των υποψήφιων εγγράφων είναι μικρότερο από το min-k.

Τέλος στον αλγόριθμο υπάρχει και μια μορφή τυχαίας προσπέλασης (random access) στην λίστα DOCS για την εύρεση του πλήθος των φορών ένα αντικείμενο έχει γίνει tag. Αυτό θα μας βοηθήσεις για τον πιο ακριβή στον υπολογισμό του best score και worst score.

Σημείωση ότι ο αλγόριθμος που αναφέραμε πάνω είναι χωρίς tag expansion και να συμπεριλάβουμε και παράμετρο αυτή προσθέτουμε και την λίστα SIMTAGS(q_i).

Πειράματα

Στα πειράματα χρησιμοποιήθηκαν τα δεδομένα από τα παρακάτω social networks

- Del.icio.us
- Flickr
- LibraryThing

Το δύσκολο κατά την διαδικασία τον πειραμάτων είναι να βγάλουν κατάλληλα ερωτήματα. Για τον σκοπό αυτό χρησιμοποιήθηκε η μέθοδος user-specific ground truth στην οποία έχοντας μια ερώτηση q(u,q1,…,qn) επιλέγουμε το σύνολο τον εγγράφων από έχουν γίνει tagged από τον u και από τους άμεσους φίλους του. Στην συνέχει επιλέγουμε τυχαία τις ερωτήσεις και χρήστη u. Η άλλη μέθοδος επιλογής ήταν η user-study στην οποία χρησιμοποιηθήκαν κάποια άτομα τα οποία είχαν κάνει tag κάποια έγγραφα και δημιουργήθηκαν δεσμοί φιλίας. Οι ίδιοι χρήστε αυτοί έφτιαξαν ερωτήσεις συμφώνα με τα tags που είχα κάνει.

Έτσι έχουμε αύξηση του NDCG όταν αυξάνεται το a και για τα 3 dataset και αύξηση της ακρίβειας (precision) άλλα πτώση των δύο τιμών όταν αγνοούμαι το a (a=1) και στις δυο περιπτώσεις. Επίσης τα πειράματα έδειξαν ότι το κόστος (χρόνος εκτελέσεις και πρόσβαση στον δίσκο) για όλα τα δίκτυα που χρησιμοποιήσαμε είναι μικρότερο σε σχέση με τον αρχικό απλό αλγόριθμος (baseline).