Real-time, Location-aware Collaborative Filtering of Web Content

Thomas Sandholm and Hang Ung
HP Labs
Social Computing Group
thomas.e.sandholm,hang.ung@hp.com

ABSTRACT
In this paper we describe the collaborative filtering feature of a location-aware, Web content recommendation service, called *Gloe*. The main purpose of our collaborative filtering solution is to increase the diversity of recommendations and to thereby mitigate popularity bias. The key challenge is to filter candidate suggestions in real-time, with minimal data mining and model building overhead. There is an apparent trade-off between building general purpose reusable models with contributions from a large user base on one hand and efficient on-line evaluation and recommendation in real-time on the other hand. Our solution is to apply item-based, top-N collaborative filtering within a hierarchical folksonomy structure in a Geohash pre-partitioned geographic locale. We demonstrate that these recommendations can be, on average, as fast to compute as aggregate rating-based recommendations, while offering a more diverse as well as personalized set of recommendations.

Author Keywords
Item-based Collaborative Filtering, Location-Based Service

ACM Classification Keywords
H.3.3 Information Storage and Retrieval: Information filtering

INTRODUCTION

Recommender systems are becoming increasingly popular to filter out relevant information in the era of social media where content is co-created and shared at an accelerating pace. Having rich filtering capabilities is particularly important in the case of content for mobile devices with limited screen real estate, bandwidth and input capabilities. A key outstanding challenge of recommender systems is how to efficiently combine general and user-centric contextual information to provide more targeted and personal recommendations. Mobile devices such as smartphones are well-suited to carry and track such contextual information as they are rarely shared among users and they have advanced sensing and communication capabilities.

A related challenge for mobile, social media recommendation services is to maintain diversity of top-N recommendations in order to avoid *popularity bias*. N is typically smaller on mobile devices exacerbating the issue. Popularity bias refers to the effect when a small set of items has a tendency to "get stuck" at the top of the recommendations due to increased exposure. This effect is sometimes also referred to as *preferential attachment* or *rich get richer*. The result of this effect is that users have no incentive to revisit the social media site, because they mostly get recommendations that they are already familiar with. To simplify our evaluation later we assume here that a higher diversity of items recommended across users is an indication of a lower popularity bias.

In this paper we present a method to combine general location context and user-centric context for making *real-time* recommendations. Additionally, we propose techniques to increase diversity and promote serendipitous discovery. Providing recommendations in real-time, referred to here as an algorithm that does not require off-line model-building (cf. lazy loading), helps with managing the huge amount of possible user-item-geography combinations.

All of the algorithms have been implemented and tested in our live local Web content recommender system, *Gloe*[1]. Gloe is a recommendation service targeted at mobile devices with Web browsing capabilities. Web pages are recommended based on your location as sensed by the device to avoid explicitly searching and then browsing through large amounts of information to find relevant content, which may be counter productive on small mobile devices.

The paper is organized as follows. First, we review Collaborative Filtering (CF) approaches relevant to our work, and then we discuss related work on location-based content retrieval. We present our algorithm and our implementation of this algorithm in the live Gloe system. Finally, we evaluate the performance of our system and show that it is feasible for real-time recommendations.

CaRR 2011, February 13, 2011, Stanford, CA, USA.

[1] http://hpgloe.com

COLLABORATIVE FILTERING

Collaborative filtering techniques have been very successful in recommending items to users based on various similarity metrics. Similarity could either be measured between users or between items. Assuming an e-commerce scenario with many users and relatively few items, the algorithms tend to scale better if one focuses on item based similarities. However, in a social media system like YouTube, with a very large number of items being continuously contributed by users, item-based algorithms will also result in prohibitively large user-item matrices. One solution is to parallelize the computation, in particular the model building phase, in a compute farm framework such as MapReduce. However, the recommendations will still suffer from being potentially stale and requiring a big investment in computational power. Another typical issue of CF recommendations is that they may lead to a popularity bias, and suffer from a lack of diversity. A direct consequence of this problem is that recommendations tend to exhaust the number of *recommendable* items, which in turn leads to a degradation of recommendation quality over time.

Because of the beneficial scalability features of item-based CF, we decided to apply it to our problem. However, we extend the approach to fit in situations with a large number of user-contributed items. The main idea is to create randomly sampled, small, ad-hoc item models of the top viewed and rated items within a folksonomy-based tag scope. These models are then directly applied to user-specific models for one-shot recommendations aimed at providing both serendipity and personalization.

Go *et al.* [4] have recently studied context-based recommendations in terms of implicit signals such as favorite bookmarking. We use very similar item-based models but leverage implicit views as well as explicit ratings in our model, and we apply the model in a geo-aware context.

Gantner *et al.* [3] have investigated context-based recommendations in terms of episodic patterns. Their approach is to use assemblies of overlapping weekly models to capture daily and weekly patterns. From their evaluation it seems that the most recent trend had a very big impact, which is the general idea behind economic time-discount models based on the theory of interest and time-value of money [5]. Given that our rating model already uses a currency (see next section), we can apply these interest-rate like methods directly to normalize for elapsed time.

An important element of making the model computationally feasible for on-line use is geographic pre-partitioning, discussed next.

LOCATION-BASED CONTENT RETRIEVAL

The recent boost in availability and use of GPS and 3G enabled smartphones has led to a surge in Location Based Services (LBS), which are providing you with location relevant information either in the form of location-scoped keyword search or location-aware content feeds. There are two main types of LBS, those that focus on automatically mining and mapping existing content (e.g. [1]), and those that rely on explicit user contributions, e.g. Yelp [2] restaurant reviews, or Foursquare [3] venue check-ins. Many systems also provide some hybrid of these two models, e.g. Google local search. Purely basing the mapping of content on text analysis, in general, saves users from entering the location in keyword searches in traditional search engines. The fact that many users today publish on-line content and have GPS enabled phones means that much more reliable and precise location mappings can be obtained directly from users in the form of tagged Flickr photos or annotated Wikipedia pages. However, when aggregating end-user contributions directly one has to pay extra attention to behavioral interactions, such as incentives to avoid issues including free-riding, spamming and mechanism gaming. To address these behavioral issues we developed an economics and prediction market inspired, location-aware recommender system that we call Gloe [6], where users rate content by placing bids, using a virtual currency. A full description of Gloe is outside the scope of this paper. For a more detailed information we refer to [6] and the on-line documentation at `http://www.hpgloe.com`. The most novel aspect of the system is that it uses a budget-based model to encourage truthful ratings.

Gloe allows retrieval of the most popular Web pages in a location based on a hierarchical folksonomy tagging structure. The result may be filtered on recommendations from a select set of friends, e.g. Facebook friends, to obtain a more personalized view of what is popular. Gloe uses explicit, real-time, aggregation-based metrics of popularity (highest sum of bids on an item), which makes the evaluation very fast. Another key design feature of Gloe is location partitioning. All recommendations are partitioned according to the Geohash [4] algorithm, which converts a two-dimensional geographic coordinate into a one dimensional string that is alphabetically sortable by location vicinity. One issue with this approach is that it is hard for users to discover long-tail content not captured by the immediate social network. This is in part an extension of the popularity bias issue. In, Gloe, this problem is arguably mitigated by allowing, location, friend as well as folksonomy tag filtering. However, diversity and discovery of long tail content outside of your circle of friends are still concerns.

To address these problem we implemented ad-hoc top-N item-based collaborative filtering on a per-location, per tag-scope basis including some controlled randomness to allow for more long-tail content and diversity. Tag scope here refers to a branch in the tree of user-contributed (folksonomy) tags. Tag scopes are also called *channels* in the Gloe system.

The Web UI can be seen in Figure 1. We describe the model we implemented in some more detail next.

[2] http://yelp.com
[3] http://foursquare.com
[4] http://wikipedia.org/wiki/Geohash

Figure 1. View of the Web UI (partial). User enters address in (*A*), sets the radius (*B*), and picks a channel/tag scope from a dynamic geo-dependent list (*C*). There are 4 tabs, showing (from left to right): aggregate ratings (*D*, "popular here"), item-based CF suggestions (selected tab *E*,"suggestions for you"), a tag cloud and a list of top contributing users in the considered area ("local experts"). Each recommendation (*F*) is displayed with the current number of views and votes received, as well as a vote button. Not shown here are: an interactive map with the positions of the recommendations; lists of recently voted and viewed items; and a list of users who contributed top-ranked content.

MODEL

We want to recommend N out of m items (represented by a URL pointing to Web content) to n users. The state of the system from which to make a recommendation can be represented by the $n \times m$ user-item matrix R, where the value of $R_{i,j}$ is the current value of the money spent by user i on item j stemming from explicit ratings or views of Web content. There is a value decay according to the theory of time-value of money to account for item popularity and novelty decay over time. This value decay takes the following form:

$$v_t = \frac{v_{t+T}}{(1+r)^T} \tag{1}$$

where v_t is the value at time t, r is the decay rate per time step, and T is the number of time steps from time t (e.g. the time of producing a recommendation) **into the past** to the time when the rating or view took place.

The top-N recommendation problem is then defined as (adapted from [2]): Given R and a set of items, U that have been either rated or viewed by the user, provide an ordered set of items X, such that $|X| \leq N$ and $X \cap U = \emptyset$.

What constitutes the top, i.e. how the items are ordered, depends on the definition of a similarity metric between items. The position of a candidate item in the list of recommendations will then be determined by the aggregate similarity between all items already rated or viewed by the user and the candidate item. The complete process for constructing this ordered list is discussed next.

The R matrix is constructed in two steps.

Step 1: We obtain the top V items based on ratings and views in a particular tag scope ts and Geohash region L, where $V = S * lf$, S is our sample size and lf is a factor > 1 to fine-tune how much long-tail content to retrieve.

Step 2: We draw a random sample of S items from V and aggregate all user spendings on those items in the matrix R.

We note that this technique will result in some loss of very poorly rated items, which may contribute to determining which items not to recommend to a user. However, we in general want to recommend high quality items to users, so cutting off low quality items in this step already ensures a certain minimal level of quality in recommended items, as well as reduces the computation time.

Next we compute the $m \times m$ item model matrix M by pairwise similarity tests of all items in R across all users. As similarity measure we adopt the frequent item adjusted conditional probability-based measure in [2]:

$$sim(i,j) = \frac{\sum_{\forall q:R_q>0} R_{q,j}}{Freq(i) \times Freq(j)} \tag{2}$$

where Freq(X) is the number of user valuations of item X. The main reason for adopting this measure is to avoid always recommending frequently occurring (rated or viewed) items, which would impair our goal of achieving increased diversity. See [2] for more details.

The model M may be cached but note that it is specific to a region L and a tag scope ts, so it has restricted applicability. However, the idea is to compute this model in real-time, and to promote diversity which caching would hamper. The only way to achieve that is to cap m, which is done by our two step semi-random construction of R.

Now to apply the model M to a user u to yield a recommendation we construct the vector U where each element represents the valuations by user u of all items in the matrix M. The recommendation is then simply computed as $x = MU$ where we set all existing item valuations in U to 0 in the resulting vector x. Furthermore, we only retrieve the top N items in x as our final recommendation.

In the case of a user who has not valuated any items in M, i.e. a cold start user, we artificially set a low valuation on a random item to force a non all-zero vector x. Without this modification users would not get any recommendations until they started rating or viewing items.

The valuation of an item is a linear combination of the amount of money spent on that item and the number of views of the item. Both of these metrics may be discounted by the time value of money constant r. We have

$$v = \alpha q * (1-\alpha)c \tag{3}$$

where q represents ratings (in Gloe, money spent on an item), and c aggregate views or clicks. Typically, $0.5 < \alpha < 1$ to bias the valuation towards the more explicit ratings.

The recommendation algorithm is summarized in Algorithm 1. Note that items to include in the model, R, and M may all be cached between subsequent calls from different users within the same Geohash area and tag scope (ts). We calculate similarity for each item, but since we can control the sample size S independently of the number of items and the number of users, a sufficiently small $S < 100$ leads to acceptable real-time performance. Using larger geographic areas (L), as well as more general tag scopes both serve to improve the scalability and performance of the algorithm, as it improves the likelihood of a cache hit. On the other hand if the areas are too large or the tag scopes too wide, then the sample of items may be too small to achieve the desired diversity, i.e. to capture the long-tail items. The cache eviction and timeout strategy also has to be chosen carefully to allow users who frequently revisit an area to obtain new suggestions.

The functions BuildModel and ApplyModel are direct adaptations of the algorithms in [2] to calculate pairwise item similarities and select the top N most similar items given a set of items that the user has already seen or rated, respectively.

The GetTopItems function, called to obtain items to include in the model, is sketched in Algorithm 2. It queries the database of views and ratings and selects the items that match the location and tag scope ordered by the time decaying linear combination of views and ratings defined in Equations 1 and 3.

Algorithm 1 GetRec(**latitude**,**longitude**,$\mathbf{ts}$,$\mathbf{N}$, **user**)

$L \leftarrow$ GeoHash(**latitude**,**longitude**)
items $\leftarrow$ GetTopItems(L,**ts**,$S * lf$)
$k \leftarrow \min(S, ||items||)$
items $\leftarrow$ RandomSample(items,k)
$R \leftarrow$ CreateItemUserMatrix(items)
$M \leftarrow$ BuildModel(R,k)
$U \leftarrow$ CreateUserItemVector(**user**, items)
$x \leftarrow$ ApplyModel(M,U,$\mathbf{N}$)
return NonZero(x)

Algorithm 2 GetTopItems($\mathbf{L}$,$\mathbf{ts}$,$\mathbf{V}$)

items $\leftarrow$ ...
SELECT URL, ...
($\alpha \sum \mathbf{L}.q$ + (1-α) $\sum \mathbf{L}.c$))/$(1+r)^T$ as v
FROM DB.$\mathbf{L}$
WHERE $\mathbf{L}$.tag LIKE '$\mathbf{ts}$%'
GROUP BY URL
ORDER BY v LIMIT $\mathbf{V}$
return items

EVALUATION

To evaluate our implementation, we studied all the geographic areas identified by a 3-character Geohash (corresponding to an approximate 100x100 mile geographic area) with at least two user-contributed ratings or clicks in Gloe. This amounted to 365 locations (4k users, 17k ratings + clicks). For each location we took the average latitude, and average longitude for all recommendations as the epicenter for our search. We then picked the top-5 contributors in terms of ratings and clicks in each location. For each of these users we then constructed two recommendation lists, one based on a radius search 30 miles from the search epicenter picking the top-N recommended items (the standard query in our current mobile clients), and the other based on our top-N item and location-based collaborative filtering algorithm described in the previous section. N was set to 5 here. All model parameters chosen are summarized in Table 1.

Table 1. Model parameters used in experiment

Parameter	value	Description
N	5	number of items to recommend
ts	*root*	tag scope (full tree)
S	50	number of items in CF model
lf	5	factor of additional items sampled
α	0.5	no bias towards clicks or ratings
r	0.01	1% daily time decay of ratings

The main goal of the experiment is to see if our CF algorithm can reduce popularity bias compared to the aggregate rating based algorithm, with minimal negative affect on performance and popularity. Therefore we measure the response time to compute and retrieve the recommendations, the diversity among the items in the recommendations, and the popularity of items in the recommendations. We also ran the CF algorithm with and without cached models. All metrics were computed across all users for each location.

The response time is computed on the server side so it does not include any network overhead, but it does include calls to the SQL database. Diversity is calculated using average pairwise Levenshtein edit distance among attributes of items in the recommendations. The attributes included were, location (Geohash strings), tags, titles, and URLs. To normalize the diversity metric to be in the interval $[0, 1]$ we calculate:

$$div(s,t) = \frac{lev(s,t) - |l(s) - l(t)|}{max(l(s), l(t)) - |l(s) - l(t)|} \qquad (4)$$

where $lev(x, y)$ is the Levenshtein edit distance between strings x and y, and $l(x)$ is the number of characters in string x. Finally the popularity is computed as the number of clicks on items in the recommended list of items.

We performed a Welch two-sample t-test to determine whether the metrics changed significantly between the pure rank versus CF cases. On a 5% significance level we cannot reject the hypothesis that the mean response time and popularity metrics are the same for the (non-caching) CF case as for the rating case. However, in terms of diversity we see a significant difference, with a p-value less than 10^{-15}. In general, with (non-caching) CF, the average response time decreased slightly, from .75 to .66 (mainly thanks to the Geohash search being faster than the radius search), and the average popularity increased slightly, from 12 to 14 clicks. On

the other hand the diversity increased significantly from .36 to .48. This behavior can be seen visually by studying the separation of the empirical cumulative distribution function (CDF) curves of the three metrics in Figure 2, which also includes the values when caching models. The way to read the graphs is to look at the y-axis values of the curves to obtain the likelihood of the metric being less than the corresponding value on the x-axis across all the sampled locations. Hence, for the response time metrics the curves should be as high as possible and for the diversity and popularity metrics the curves should be as low as possible. To make the differences clearer the mean values for each metric are also marked and shown in the graphs. The conclusions that can be drawn from these graphs are the same as for the statistical tests; the CF algorithm performs slightly better, and has significantly higher diversity, whereas the aggregate popularity is largely unchanged compared to the simple rating based ranking.

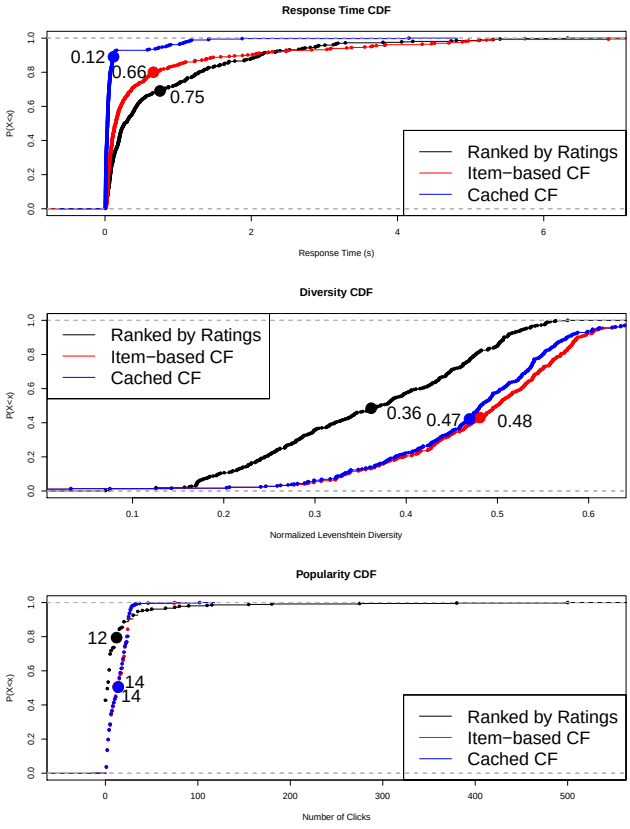

Figure 2. Response time, Diversity and Popularity CDFs. For response time a higher curve is better and for the other two metrics a lower curve is better. All values are across all users within the same location. A point in the graph corresponds to a sampled location.

CONCLUSIONS

We have shown in some preliminary experiments that location-aware collaborative filtering is a promising technique for improving diversity in search results to address the problem of popularity bias without compromising performance or popularity significantly.

We note that the fact that the aggregate popularity did not change much when applying our CF algorithm is not a sign that the popularity bias, or as we define it lack of diversity, did not change, but rather a desirable feature we achieved by sampling from a larger pool of popular items.

Surprisingly, our experimental diversity results were not very sensitive to changes in the lf parameter, controlling the factor of additional items to sample from, or the r parameter, representing time decay. We, however, attribute this to the fact that the amount of usage is not high enough in the live system across all sampled locations, as opposed to some intrinsic feature in our algorithm.

Future work includes more explicit use of geography in the CF model, making trade-offs between novelty and popularity and optimizing click-through rates using economic models of attention. As we get more usage of these new features we can also analyze them more directly in the live system using techniques such as A/B testing.

REFERENCES

1. E. Amitay, N. Har'El, R. Sivan, and A. Soffer. Web-a-where: geotagging web content. In *SIGIR'04*, pages 273–280, Sheffield, United Kingdom, 2004. ACM.

2. M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. *ACM Trans. Inf. Syst.*, 22(1):143–177, 2004.

3. Z. Gantner, S. Rendle, and L. Schmidt-Thieme. Factorization models for context-/time-aware movie recommendations. In *CAMRa '10: Proceedings of the Workshop on Context-Aware Movie Recommendation*, pages 14–19, New York, NY, USA, 2010. ACM.

4. G. Go, J. Yang, H. Park, and S. Han. Using online media sharing behavior as implicit feedback for collaborative filtering. *Social Computing / IEEE International Conference on Privacy, Security, Risk and Trust*, 0:439–445, 2010.

5. D. G. Luenberger. *Investment Science*. Oxford University Press, 1998.

6. T. Sandholm, H. Ung, C. Aperjis, and B. A. Huberman. Global budgets for local recommendations. In *RecSys '10*, pages 13–20, New York, NY, USA, 2010. ACM.