# Location-aware online learning for top-k recommendation

Róbert Pálovics[a], Péter Szalai[a], Júlia Pap[a], Erzsébet Frigó[a], Levente Kocsis[a], András A. Benczúr[a]

[a] *Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI)*

**Abstract**

We address the problem of recommending highly volatile items for users, both with potentially ambiguous location that may change in time. The three main ingredients of our method include (1) using online machine learning for the highly volatile items; (2) learning the personalized importance of hierarchical geolocation (for example, town, region, country, continent); finally (3) modeling temporal relevance by counting recent items with an exponential decay in recency.

For (1), we consider a time-aware setting, where evaluation is cumbersome by traditional measures since we have different top recommendations at different times. We describe a time-aware framework based on individual item discounted gain.

For (2), we observe that trends and geolocation turns out to be more important than personalized user preferences: user-item and content-item matrix factorization improves in combination with our geo-trend learning methods, but in itself, they are greatly inferior to our location based models. In fact, since our best performing methods are based on spatiotemporal data, they are applicable in the user cold start setting as well and perform even better than content based cold start methods.

Finally for (3), we estimate the probability that the item will be viewed by its previous views to obtain a powerful model that combines item popularity and recency.

To generate realistic data for measuring our new methods, we rely on Twitter messages with known GPS location and consider hashtags as items that we recommend the users to be included in their next message.

*Keywords:* Online learning; geolocation information; geographic hierarchy; cold start; ranking prediction.

*Email addresses:* rpalovics@ilab.sztaki.hu (Róbert Pálovics), pszalai@ilab.sztaki.hu (Péter Szalai), papjuli@ilab.sztaki.hu (Júlia Pap), fbobee@ilab.sztaki.hu (Erzsébet Frigó), kocsis@ilab.sztaki.hu (Levente Kocsis), benczur@ilab.sztaki.hu (András A. Benczúr)

## 1. Introduction

Geospatial and temporal context information plays an important role in content recommendation. While context in general can be incorporated in recommenders as user and item metadata, non-stationary context information can be highly relevant and require new online learning models.

Ideally, a geographical information based content recommendation system relies on the knowledge of both user and item geolocation. Indeed, we may assume that the user mobile device sends its GPS position along with a recommendation query. On the other hand, the geolocation of an item may often be ambiguous, change in time, or spread to a whole metropolis, state or region. In addition, some content may have obvious connection to certain locations but others can have global interest on different levels such as native speakers of a language, within a continent, or even worldwide.

In this paper we design position based recommender methods that, in addition to user preferences, also learn item locality by relying on collaborative filtering by online machine learning [1]. Recommender systems often have to serve in online environments that can be highly non-stationary. Content and in particular locality context is non-stationary and may change in time both for users and items, primarily due to changes in trends and geographic spread of information. Traditional recommender algorithms may periodically rebuild their models, but they cannot adjust to quick changes in trends caused for example by timely information. In our recent experiments [32, 33], we observed that even a simple, but online trained recommender model can perform significantly better than its batch version. Our new recommender methods are intended to be beneficial in information systems with temporal geographical information:

- Content and in particular locality context is non-stationary and may change in time both for users and items, primarily due to changes in trends and geographic spread of information, which ask for *online machine learning* [1];
- The user feedback is *implicit* [18, 36]: typically, items are viewed, consumed but probably no feedback of like or dislike is given. The fact that an item does not appear for a user may mean lack of interest but also lack of knowledge on the existence of the item;
- Evaluation is *top-k* [13, 12], i.e. we measure if the next item consumed by the user appears in the recommended items with highest predicted score.

In this paper, we consider distance, region and location as side information, with the novel element that neither users nor items are bound to one single location. Items may in fact relate to certain locations as well as being popular worldwide. Earlier results on recommendations in location-based social networks surveyed in e.g. [5, 40] combine spatial ratings for non-spatial items, nonspatial ratings for spatial items, and spatial ratings for spatial items [28]. Compared to other geographic recommenders that we describe in Section 1.1, we face the problem of the fuzzy relation of users and items with locations.

Some content may have obvious connection to certain locations but others can have more widespread interest on different levels such as language, continent, or even worldwide. Dealing with this, our models rely on the hierarchy of regions from a global or continent-wide level down to a village or city district to attribute the momentary popularity of an item to levels of locations. The use of hierarchical structure of locations for recommender systems is surveyed in [5]. We use the open hierarchical database of Global Administrative Areas (GADM, `http://gadm.org`). We also remark that spatial context may not follow geopolitical boundaries; in this case we may use any tree based hierarchy, e.g. KD-trees [37] in our procedure. The nodes of the tree may also overlap as we do not rely on disjoint nodes in our modeling phase.

Our models have two main components. The first is the hierarchical structure of geographical areas with corresponding node wights that reflect the relevance of the given area. Second, we introduce functions to model the likelihood of the consumption of a given item at a given area at a given time. Since items may have a very strong time dependence at a location, we consider methods of recommendation by online machine learning. Compared to standard collaborative filtering methods, we process events only once and in the order they have appeared. Immediately after a user consumes an item, the corresponding model weights are adjusted with a high learning rate. The final prediction arises as the weighted combination of the item estimated probabilities along the path of the GADM tree from the leaf location of the user up to the root. In other words, we build up our models above the location hierarchy. Whenever we recommend for a user at a given location, we consider the entire path of the location to the root of the GADM tree.

For our experiments, we construct data based on Twitter, a service that can be considered as a mix of a social network and news media [24], and in addition, an information system with geographical information. We investigate the problem of recommending Twitter hashtags for users, based on the temporal geolocation information of both the users and the hashtags. Twitter data is appropriate to demonstrate the novel element in our task in that neither users nor items (hashtags) are bound to one single location. Hashtags may in fact relate to certain locations as well as be popular worldwide.

In a system like Twitter both the actual topics, and the interest of the users rapidly change over time. The main goal of a user on Twitter is to adopt relevant and *new* topics. As we have no information on which tweets are read by the users but we know the new hashtags they tweeted, we use the hashtag publishing information to measure user topic adoption. Therefore our aim is to recommend new hashtags, i.e. hashtags that the user has not used before. Recommending new hashtags can result a more homogenized set of hashtags in the Twittershpere. This is practically useful both for the users and for the service provider. The recommendations are obtained by learning online from the stream of geo-tagged tweets.

Since hashtag usage is highly volatile, the problem calls for an online method. Whenever a user sends a geo-tagged tweet with a hashtag she has not used earlier, we consider the event as a trigger for recommendation. We measure the

accuracy of our methods in the online evaluation framework of [32] based on discounted cumulative gain (DCG) computed individually for each event and averaged over time.

Our baseline methods include online matrix factorization [32] and content based methods [17]. Surprisingly, it turns out that these baselines perform much weaker than the distance based methods and contribute relatively little to the final prediction. This observation justifies the importance of the temporal and geographic item context. The locality of Twitter hashtag adoption in both spatial and temporal sense was observed among others by Kamath et al. [20]. They state that "hashtags are a global phenomenon [...] but distance between locations is a strong constraint on the adoption [...] and follow a spray-and-diffuse pattern".

Our methods are applicable in the *user cold start* situation as well, when only the location but no prior actions of the user is known. Observe that the new methods use item and location information and user-item matrix factorization adds little to the overall accuracy. Hence our method is capable of recommending items to new users, given that their location is known. To justify, we will measure recommendation quality as the function of the items consumed by the users.

The paper is organized as follows. After the related results, in Section 2 we describe our online evaluation framework for our highly time sensitive implicit top-k recommendation task. We describe our methods in Section 3, including the notion of the Global Administrative Areas hierarchy, the models for item temporal behavior, and the combination of recommender methods by online learning. In Section 4 we describe the matrix factorization, nearest neighbor, and content based baseline methods. Finally we give the Twitter based data generation procedure and the results of our experiments in Section 6.

### *1.1. Related work*

In our results, the key factor for recommendation quality is location. Surveys on *recommendations in location-based social networks* [5, 40] combine spatial ratings for non-spatial items, nonspatial ratings for spatial items, and spatial ratings for spatial items [28]. In [22], an architecture of a location based recommender system is described with dynamic as well as static attributes both for users and for items. They illustrate their architecture with hotel recommendation, a task that would fit into our models by replacing the current location of a user by a potentially ambiguous tentative trip location. By using the same architecture as [22], we give location based models in this paper.

In our geographic recommender method, neither users nor items have direct well-defined geolocation. We are aware of no other results that use *external data to define the hierarchy of locations* for recommendation tasks. Regions-of-Interest partitioning is examined in [26]. Instead of administrative districts, the authors establish natural regions by applying $k$-means clustering on the dataset. Other results propose Optics [46] and grid-based [45] clustering to compile a structure of locations for GPS trajectory mining. Similar to our

result, in [15], GADM is used over the same Twitter data set, but only for visualization purposes.

Most of previous publications on geographic recommender systems work with *check-in data*, where each of the items has a predefined static location. Bao et al. [4] build a hierarchy tree based on the text content to define user similarity for collaborative filtering. They combine local and distant event comparison with finding local experts. Probabilistic Matrix Factorization [9] relies on the geographic information of the items and the typical location of the users. They observe that "users tend to check in around several centers, where the check-in locations follow a Gaussian distribution at each center [... and] the probability of visiting a place is inversely proportional to the distance from its nearest center; if a place is too far away from the location a user lives, although he/she may like that place, he/she would probably not go there." As seen by the above quote, check-in physical locations, unlike the geographic aspects of tweets, primarily determine user behavior and enable very different methods that cannot be applied in our task.

Other check-in recommendation results [7, 16] use matrix factorization, the latter also combines it with geographic regularization. Note that factor model recommenders [14] perform much weaker than our geographic models, hence we concluded that location information has to be used in a different way in our hashtag recommendation task. Several results consider similarity based recommenders [47]; for example, they exploit that closer locations have much higher probability of being visited [5]. Note that our recommender task is very different from check-in data, since physical constraints of distance are not necessarily affecting Twitter content.

In [23], Flickr geotags are used for travel route recommendation, concentrating on routes and not individual places. User similarity based methods may combine friendship information with the distance of the user home locations [42, 43].

While we use Twitter mainly to simulate the task, we survey Twitter analysis and in particular, hashtag recommendation results next to compare with our methods. Twitter is a mixture of a social network and news media [24], where users follow each other [19] and use hashtags [20] to organize their messages. The complexity of Twitter data gives rise to a number of prediction tasks with a wide variety of possible techniques and solutions. Lerman and Gosh give an empirical comparison of information contagion on Digg vs. Twitter [27]. Several results *predict retweet count*: Cheng et al. [10] based on network features, Bakshy et al. [2, 3] on `bit.ly` URLs, and [34] on time sensitive modeling.

*Hashtag recommendations* are addressed in several recent papers: Chen et al. [8] give methods for efficiently maintaining a sliding window for time aware recommendation, and Diaz et al. [14] introduce methods to compute matrix factorization online. These results are orthogonal to our exploitation of the location information, and these approaches can be combined to yield a scalable recommender architecture.

Hashtag recommendation is proposed to help the users in allocating terms to their posts and increase the homogeneity of hashtag usage in [44, 17, 25].

These methods recommend after the user finished writing a new target tweet. All these results use user or item based similarity search over the tf-idf by using cosine distance as one method. In [44], several other distance metrics are tested, but apparently cosine of tf-idf performed best. In [25], combinations of user and item content similarity are tested with roughly 20-30% improvement in hit rate by using user and not just item content.

We use the method of [17] as a content based baseline in our experiments. They also use terms in the past tweets of the user. Time is considered by separating hashtags used over long periods of time vs. only for a short lifespan; however for both sets of hashtags, they observe an exponential decay of relevance in time.

In a more complex content based recommender, [29, 30] solve a different task: they predict the future popularity of hashtags by using feature engineering, Latent Dirichlet Allocation and also combining with information on the mention graph. No experiments are known whether more complex content feature engineering would significantly improve hashtag recommendation performance.

In the hashtag recommendation literature [14, 8, 44, 17, 25, 29, 30], we observe that the differences in recommendation quality are much smaller than the difference between our location models compared to the baselines. Consequently, we concentrate on exploring the use of location in recommenders and their combination with simple baselines.

As additional relevant results addressing hashtag use, *Spatial statistics of hashtag adoption* are analyzed by Kamath et al. [20]. Cheng et al. [11] give methods to geolocalize tweets based on content. Mocanu et al. [31] use a data set similar to ours to analyze geographical properties like homogeneity and seasonal patterns of language usage at scales ranging from country-level to city neighborhoods. Similar to our use of the Global Administrative Areas, *regions-of-interests partitioning* is examined in [26] by applying $k$-means clustering to establish natural regions over Twitter data. None of these papers exploit the results in recommender systems.

Our method is capable of handling the cold start recommendation problem as we rely only on the user location and not necessarily the past set of user items. The cold start problem was introduced in [38] as the task of recommending items not yet rated by the users. A very similar task is to recommending for users with no past activity. Cold start recommendation is known to be difficult: in [35] it is observed that metadata based recommendation has much lower quality if ratings exist. We separately measure the performance of our location based methods for users with no past events, ratings even implicit, and show the strength of geolocation for cold start recommendation.

We consider the task of top-k recommendation first introduced in [39]. The difficulty in top-k recommendation compared to individual item rating prediction is that the ratings of potentially all candidate items have to be computed to find those best fitting to the preference of the user. In [13], item similarity based recommenders are given where the list of similar items are maintained to reduce the candidates. In [12], popular items are used as candidates to limit the number of ratings to be computed to obtain the top-k recommendation. In our

location based methods, candidates are maintained at the nodes of the location hierarchy tree. The methods of [12, 13] are orthogonal to our procedure and could be used in combination, especially with the baselines, to speed up the recommendation phase.

## 2. Online recommendation and evaluation

We use the online recommendation framework described in [32], in which model training and evaluation happen simultaneously, iterating over the dataset only once, in chronological order. Whenever we see a new user-item pair, we assume that the user becomes active and reveals her location to the recommender system. In this case, we recommend items of potential interest for the user that we match against the actual item consumed. The recommendation is online, hence it depends on the context at the exact time instance of the item. If a user $u$ views item $i$ at time $t$ in location $\ell$, our models give a score $\hat{r}(u, i', \ell, t)$ for each item $i'$ seen so far, and recommend to $u$ the $k$ items with the largest values from those that $u$ has not seen before.

### *2.1. Online top-k recommendation*

We address the top-$k$ recommendation task [13, 12], where the goal is not to rate some of the individual items but to provide the best candidates. In a time sensitive or online recommender that potentially recomputes prediction after each and every new item, we have to generate a new top-$k$ recommendation list for every single event in the test period. The online top-$k$ task is hence different from the standard recommender evaluation settings, since there is always a single item only in the ground truth and the goal is to aggregate the rank of these single items over the entire testing period.

### *2.2. Average DCG for online evaluation*

We use the quality metric of [32]. If $i$ is the next item for the user, DCG@$k$ is defined as the following function of the rank of $i$ returned by the recommender system,

$$\mathrm{DCG@}k(i) = \begin{cases} 0 & \text{if } \mathrm{rank}(i) > k; \\ \dfrac{1}{\log_2(\mathrm{rank}(i)+1)} & \text{otherwise.} \end{cases} \tag{1}$$

The overall evaluation of a model is the average of the DCG@$k$ values over all items of the testing period.

Since DCG is a slow decreasing function of the rank, our DCG evaluation may consider a large number of tags of potential interest to each user. Note that in our unusual setting of DCG evaluation, there is a single relevant item and hence for example no normalization is needed as in case of the DCG measure. Also note that the DCG values will be small since the nDCG of a relative short sequence of actual messages will roughly be equal to the sum of the individual DCG values. Furthermore, we never recommend the same item used in the training set, and for this reason, the best DCG average could reach the value

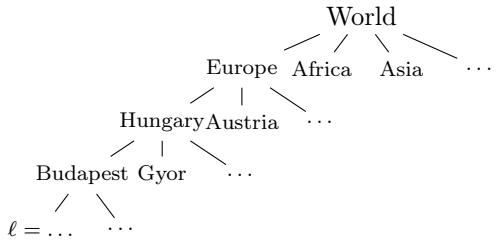


Figure 1: First three levels of the tree and the user location $\ell$.

of 1 even if the only modification to the top-$k$ recommendation would be the removal of the previous relevant results.

### *2.3. Ramifications for the Twitter data*

The online evaluation framework described in this section applies to general user-item setting. Since we use Twitter hashtag recommendation to generate a realistic task for our experiments, next we describe some specific elements of interpreting the online evaluation metric.

First, we have no access to the list of tweets read by the user. For this reason, we approximate the evaluation by assuming that the user includes the relevant new hashtags in the next message.

Furthermore, we never recommend hashtags already appearing in some earlier tweet of the user. We could easily modify our method by periodically revisiting hashtags not used for a given time. However actively used hashtags depend more on external information beyond the scope of a recommender system.

Also, we give no recommendation when a user tweets but includes no new hashtag. In this case, DCG would be zero, which would modify all our results by the fraction of the messages with no new hashtags but leave the relative power of the methods the same.

Finally note that in a single event, the user may include more than one items, i.e. add more than one new hashtag to the same tweet. In this case, for uniformity, we add the DCG values without normalization, unlike in nDCG.

## 3. Modeling

### *3.1. Recommendation by location hierarchy*

Our recommendation model assumes the existence of a hierarchical structure over the geographical locations, for example the GADM tree that will be shown in Section 6.2. We denote the leaf of the tree that is closest to the current GPS location of the user by $\ell$, and the path in the tree from the root node to location $\ell$ by $\mathrm{Path}(\ell)$. For illustration, an example subtree is given in Figure 1.

We further assume a function $s(i, n, t)$ that scores the likelihood of an item $i$ being used in node $n$ of the tree at time $t$. In Sections 3.2–3.4, we propose three variants of such scoring functions that depend on the history of the given item

in the particular node. The individual scores along the path corresponding to the users location are combined linearly:

$$\hat{r}(u, i, \ell, t) = \sum_{n \in \mathrm{Path}(\ell)} w_{n,t} \cdot s(i, n, t), \tag{2}$$

where $w_{n,t}$ are node specific weights that aim to capture the relevant level of granularity for a given location or region. The weights $w_{n,t}$ are independent of the items and characterize the area $n$ only. The weights along the current path are adapted by online gradient descent, optimizing for RMSE:

$$w_{n,t} = w_{n,t-1} + \eta \left( r(u, i, \ell, t) - \hat{r}(u, i, \ell, t) \right) s(i, n, t),$$

where $\eta$ is a learning rate and $r(u, i, \ell, t)$ is the target value, that is set to 1, if the item was consumed at time $t$, and 0 if not. The gradient of $w_{n,t}$ in the RMSE is $2 \cdot s_n \cdot$ error, where error is the difference of the predicted and actual score.

We have to take special consideration for the zero target values $r(u, i, \ell, t) = 0$ for implicit feedback data. In our tasks, the events imply only user interest but no feedback of like and dislike. In most of our models, we need negative instances as well for training. In order not to saturate the model with an overwhelming majority of zero score items, the typical method that we apply in our models is that we generate negative training instances by selecting $N$ random items uniformly corresponding to the time of the positive instance. Negative instances correspond to items that the user did not see at the given time and location. Only the positive items and negative sample elements $i$ are given to (2).

In our experiments we also investigate models where we set all $w_{n,t}$ values constant, i.e. we do not learn the weights. Several variants are investigated for setting the weights: (1) *world*: only the root of the three has non-zero weight, (2) *continent*: only nodes corresponding to continents have non-zero weights, (3) *country*: same for countries, (4) *leaves*: only the leaves have non-zero weights, and (5) *tree*: all weights are set equal.

### *3.2. Temporal popularity*

To determine popularity, for each location in the tree, we compute the number of occurrences of the item $i$ at the given location. For defining time intervals, we use a predefined time discretization that we test between a minute and a day. As it follows power-law distribution, we use the logarithm of the temporal popularity values as node scores: $s(i, n, t) = \log(\mathrm{pop}(i, n, t))$, where $\mathrm{pop}(i, n, t)$ denotes the number of occurrences of item $i$ in node $n$ in the time interval ending at time $t$.

### *3.3. Item recency*

Our next method estimates the chance of the appearance of an item by considering its most recent usage. The advantage of this method is that it is more sensitive to changes in trends. While it may more aggressively overfit to

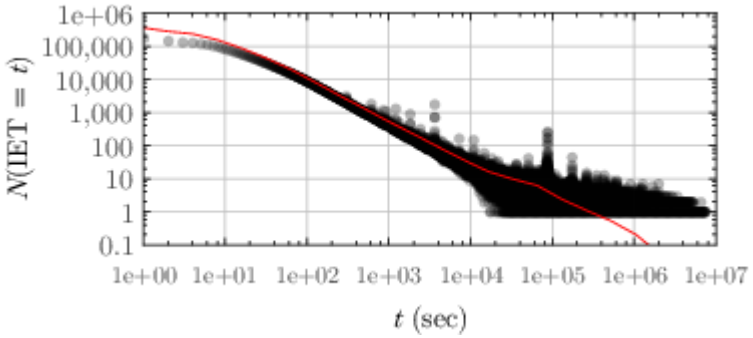


Figure 2: Inter-event distribution.

single events, overall it performs similar to and combines very well with the popularity based method. As we will observe in our experiments, the inter-event time distribution follows power law, in accordance with several earlier observations [6, 41, 21]

$$P(\tau = t) = (\alpha - 1) \cdot t^{-\alpha} \text{ and } P(1 \leq \tau \leq t) = 1 - t^{(1-\alpha)}. \tag{3}$$

Given the distribution of the time between consecutive appearances, we may estimate the chance that the item appears in the next $\Delta t$ time if we have not observed it for time frame $t$ by

$$P(t < \tau \leq t + \Delta t | \tau > t) = \frac{P(\tau \leq t + \Delta t) - P(\tau \leq t)}{1 - P(\tau \leq t)} \tag{4}$$

$$= \frac{(1 - (t + \Delta t)^{(1-\alpha)}) - (1 - t^{(1-\alpha)})}{t^{(1-\alpha)}} = 1 - \left(1 + \frac{\Delta t}{t}\right)^{(1-\alpha)}.$$

For location sensitive prediction we maintain the last appearance of each item for every node in the geolocation tree. We compute the estimate of (4) in each node by using a global $\alpha$ value.

### *3.4. Exponentially weighted decay*

This method estimates the probability of an item by assuming a gradient descent update:

$$\hat{p}(i, n, t + 1) = \hat{p}(i, n, t) - (1 - \gamma) \frac{\partial (\hat{p}(i, n, t) - y(i, n, t))^2}{\partial \hat{p}(i, n, t)},$$

where $y(i, n, t)$ equals 1, if item $h$ was used for node $n$ at time $t$, and 0 otherwise. By simple computation, with $\hat{p}(i, n, 0) = 0$ we get the familiarly looking

$$\hat{p}(i, n, t) = (1 - \gamma) \sum_{\tau : y(i,n,t)=1, \tau < t} \gamma^{t-\tau}.$$

Since the probability of an item also has a power-law distribution we temper the large values by using a logarithm transform. By dropping constant scaling, the score function will be $s(i,n,t) = \log(1 + \sum_{\tau:y(i,n,t)=1,\tau<t} \gamma^{t-\tau})$. To give more insight in the parameter, in the experiment we replace $\gamma$ by the halving time $t_{1/2}$: $\gamma = 0.5^{1/t_{1/2}}$.

While the first method estimates popularity over longer time, the second is expected to react better to items gaining rapidly in popularity. This third method can be seen as a compromise between the two, i.e. it is estimating popularity (in fact, $\hat{p}(i,n,t)$ converges to the expected probability, if the popularity is stationary), but gives exponentially larger weight to the more recent events.

### *3.5. Method combination by online learning*

We blend our individual methods to form a stronger recommendation. We combine some of our models linearly by learning the combination weights by stochastic gradient descent. The learning algorithm is similar to the one we applied for the node weights (see Section 3.1 ).

## 4. Baseline methods

The most popular and successful approaches to recommendation are matrix factorization, and nearest neighbor methods. These will be included as baselines in the experiments, and their implementation is described in the next subsections. Besides matrix factorization, we apply two different nearest neighbor methods in our experiments. First, we define distance based on time and geolocation. Next, we introduce a content based recommender that uses the text of the tweets to compute similarity and distance.

### *4.1. Online matrix factorization*

A matrix factorization model characterizes each user $u$ by a vector $P_u$ and each item $i$ by a vector $Q_i$. The score for a given item is represented by the scalar product $P_u Q_i$.

For batch evaluation, it is standard to iterate several times over the training set until convergence. However, it was shown in [32] that for online scenarios stochastic gradient descent is more efficient (which is consistent with the experience in online learning, in general).

We apply online matrix factorization for implicit feedback problems. As in Section 3.1, after each moment when a user considers an item, that item becomes a positive instance and we sample $N$ items uniformly as negative instances. Then, the user and item vectors are updated by gradient descent on the mean square error between the target and the predicted score. The target of positive instances is 1, and for the negative instances is 0.

*4.2. Nearest neighbor model*

While many neighborhood approaches rely on finding users with similar behavior, on our dataset this is less efficient since too many users have only a few tweets. Alternatively, we rely on the geographical closeness. Thus, we consider the geographically nearest $k$ occurrences of the item. The score of the item arises by aggregating the function of the time elapsed and the geographic distance of past occurrences,

$$\sum_{i=1}^{k} f(t_i)\text{dist}_i^{-1}, \tag{5}$$

where $t_i$ is the time elapsed since tweet $i$ and $\text{dist}_i$ is the distance. For the time function $f$, we give an estimate in Section 3.3, but in fact any decreasing function would suffice.

*4.3. Content based recommendation*

To provide an information rich baseline, we let our content based recommender access, with the exception of the hashtag, the full text of the actual tweet to recommend hashtags. Note that none of our other methods have information beyond user, time and location.

Our method is based on [17], who use the cosine similarity of tf-idf. We do not separate hashtags used over longer periods of time: for simplicity, we use a global exponential decay over all hashtags. Also for simplicity of implementation, we do not consider the past items of the users. Note that in [17, 44], the differences between content based recommender quality are much smaller than the difference between our location models compared to the baselines.

One component of [17] that resulted in significant gains in our experiment is the TemporalTweetMax strategy that we implemented as our baseline. We multiply the similarity of the new tweet with one in the history by $N(t) = e^{-\eta t}$, where $t$ is the difference of the time between the new and the old tweet and $\eta$ is a constant as in [17]. Similar to the exponentially weighted method of Section 3.4, we replace $\eta$ by the halving time.

## 5. Overview of the recommendation architecture

Our recommender system processes each user request by producing a top list of recommendations, and then updates the recommender model by the actual item taken by the user in an online learning fashion. The pseudocode is given in Algorithm 1 and an overview diagram in Fig. 3.

Next we analyze the computational complexity of our algorithms. Note that our algorithms are top-k recommenders, where computational costs depend on the candidate generation procedure [12]. In our analysis, we will use $c$, the number of candidate items, as a parameter. Candidates may potentially be all items. Note that in our baseline experiments, we used all items as candidates to provide as strong as possible baselines.

**Algorithm 1** Overall Pseudocode of the Recommender System

**procedure** RECOMMEND(user $u$, location $\ell$, timestamp $t$, text $T$)
  **for all** location nodes $n$ containing $\ell$ **do**
    **for all** items $i$ in $n$ **do**
      $\text{Pop}(i,n) = \log(\text{pop}(i,n,t))$ as in Section 3.2
      $\text{Rec}(i,n) = 1 - (1+\Delta t/t)^{(1-\alpha)}$ as in (4).
      $\text{Dec}(i,n) = \log(1 + \sum_{\tau:y(i,n,t)=1,\tau<t} \gamma^{t-\tau})$ as in Section 3.4
  $\hat{r}(u,i) \leftarrow$ combine $\text{Pop}(i,n)$, $\text{Rec}(i,n)$, $\text{Dec}(i,n)$ by $w_{n,t}$ as in (2).
  **for all** items $i$ **do**
    $\text{MF}(u,i) = \sum_k P_{uk} Q_{ik}$
    $\text{NN}(u,i) = \sum_{i=1}^{k} f(t_i)\text{dist}_i^{-1}$ as in (5)
  **for all** items $i$ similar to $T$ **do**
    $\text{Cont}(u,i) = e^{-\eta t}\cos(i,T)$ as in Section 4.3
  Return top $i$ of linear combination $\hat{r}(u,i)$, $\text{MF}(u,i)$, $\text{NN}(u,i)$, $\text{Cont}(u,i)$
**procedure** UPDATE(user $u$, location $\ell$, item $i$, timestamp $t$)
  **for all** location nodes $n$ containing $\ell$ **do**
    Store $(i,t)$ at $n$
    Update $\text{pop}(i,n,t)$ in sliding window
    Update $w_{n,t}$ by gradient descent
  Generate negative items
  **for all** positive and negative items $j$ **do**
    **for all** factors $k$ **do**
      Update $P_{uk}$, $Q_{jk}$ by gradient descent
  Update combination weights by gradient descent

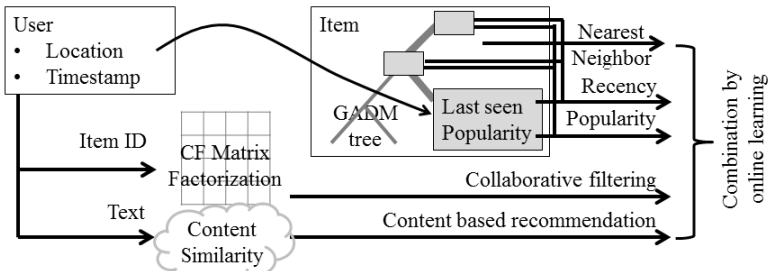


Figure 3: The recommender architecture.

*Recency, popularity and decay.* In our new methods, each node $n$ of the GADM tree stores a candidate set $C_n$ of most popular items at the location, much smaller than the set of all items. The recommendation procedure is linear in $|\bigcup_n C_n|$ while update is $O(1)$.

Table 1: Properties of the original dataset.

| | | |
|---|---|---|
| number of tweets | 1,423,278,863 | (100%) |
| tweets with coordinates | 1,266,004,930 | (88.95%) |
| tweets with hashtags | 173,493,860 | (12.19%) |
| tweets with mentions | 655,340,289 | (46.04%) |

*Matrix factorization.* We have to maintain $k$ factors of all candidates, with complexity $O(c \cdot k)$ for recommendation and $O(1)$ for update.

*Nearest neighbor.* Recommendation complexity is linear in the number of candidates. The set $\bigcup_n C_n$ suffices. There is no update step.

*Content based.* We compute the similarity of each candidate to the text $T$ of the new tweet. Recommendation complexity is $O(c \cdot |T|)$. Candidate generation may be improved by using text index to include only tweets with at least one common word. Also, old tweets may be discarded. There is no update step.

## 6. Experiments

### *6.1. Data set*

We use a four-month collection of 400 million geo-tagged Twitter messages detailed in [15]. We mention that the metadata of tweets may contain not only GPS coordinates but also a *place* attribute that can contain the name and type of the place. However, we found the place attribute often ambiguous and less reliable. We summarized the properties of the full data in Table 1. Due to the irregularities of their collection procedure, we used the data between February 1 and May 30, 2012, hence the online learning period lasts three months (see Fig. 4).

Since hashtag count follows power-law distribution, most of the hashtags are quite rare and we use only the hashtags that appear more than 5 times. This way we exclude about 90% of the hashtags, but most of the hashtag timeline remains. We also exclude the hashtags that appear in the first month of the collection to recommend newly spreading hashtags for the users. The properties of the final cleansed dataset are summarized in Table 2 and we show the final number of posted hashtags for each day in Figure 5.

### *6.2. Global Administrative Areas*

We collected all 214,230 nodes from the GADM database, from which 190,315 are leaves. The depth of the tree is 6, and includes 5 levels from the GADM tree plus continent-country relations. The first three levels are visualized in Figure 1. The hashtag time series data covered 30,450 leaves from the tree.

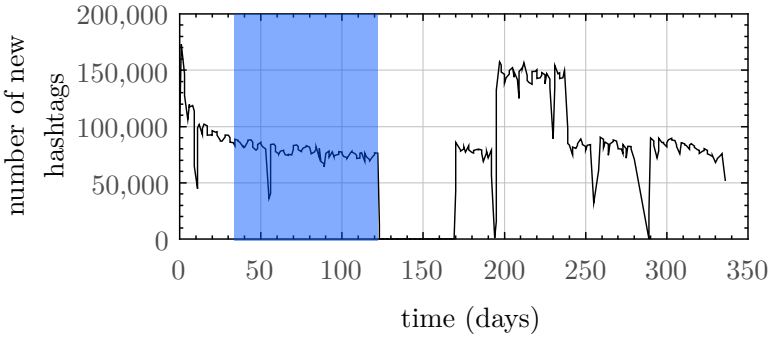


Figure 4: Daily count of *new* hashtags.

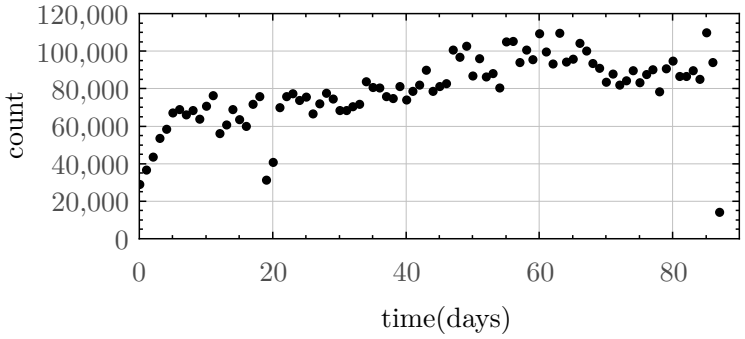


Figure 5: Daily number of posted hashtags in the final cleansed dataset.

Table 2: Properties of the cleansed dataset.

| | |
|---|---|
| number of records | 6,978,478 |
| number of unique user-hashtag pairs | 2,993,183 |
| number of users | 792,860 |
| number of hashtags | 268,489 |
| number of countries | 49 |

### *6.3. Parameter analysis*

In this subsection, we analyze the effect of the model parameters specific to the methods such as learning rate, negative rate, dimension, or time frame. In this set of graphs the performance in measured as the average DCG@100 over the testing period.

#### *6.3.1. Matrix factorization*

The factor model have four parameters: the dimension of the latent vectors $d$, the learning rate $\eta$, the regularization rate $\lambda$, and the number of negative

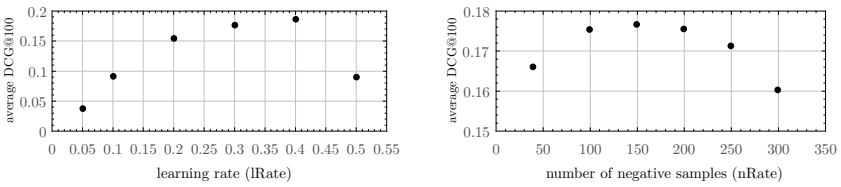


Figure 6: **Left**: average DCG@100 for SGD based MF with different learning rates. **Right**: average DCG@100 as the function of the number of negative samples generated for all positive instances in the test set.

samples $N$. We show the performance of the factor model by varying the the learning rate and the negative rate in Figure 6. Increasing the learning rate improves the performance up to a certain point, since it helps to adapt to the non-stationarity of the data. If the learning rate takes values higher than 0.4, the algorithm becomes unstable with many latent vectors diverging to infinity. The optimal number of negative samples seems to be consistent to previous studies that suggest an optimal range from 60 to 100. We do not show graphs for the remaining two parameters as did not appeared of interest. In the following experiments, the parameters will be set as follows: $d = 10$, $\eta = 0.4$, $\lambda = 0.001$ and $N = 99$.

*6.3.2. Nearest neighbor*

Nearest neighbor has as main parameter the number of neighbors $k$, and the parameters of the recency function. The former is set to 1000, while the parameters of the recency function is discussed in Section 6.3.5. This model has the weakest performance of all models, and no other parameter setting seemed to improve it.

*6.3.3. Content based recommendation*

Content based method has the halving time as main parameter (see 4.3). In Figure 7 we show the overall performance of the model measured in DCG@100 as the function of the halving time. Note that if we do not include the time-decaying term in the model, the average performance is roughly 0.08. As a result, including the time decaying term is important, but the performance is then less sensitive to the halving time.

*6.3.4. Temporal popularity*

The tree-based temporal popularity has the size of the time frame $\Delta t$ as leading parameter, and additionally the parameters of the learning algorithm, such as the learning rate, and the negative rate. The performance of the algorithm while varying the time frame is shown in Figure 8. Computing the popularity on time frame less then 1 hour leads to poor performance, since it gives a too noisy evaluation, while increasing the time frame for longer than a few hours also harms the performance giving too much influence to past events.

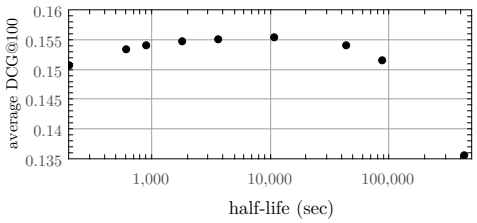


Figure 7: Average DCG@100 as the function of the half life parameter for the content based recommender.

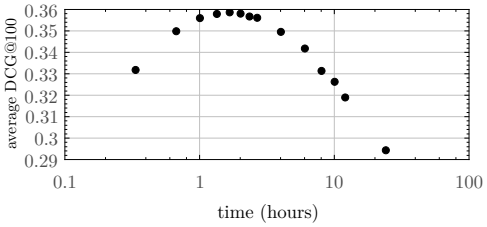


Figure 8: Average DCG@100 for the geolocation based popularity model as the function of the length of the time frame.

In the following we set $\Delta t = 2h$. For the online SGD that adapts the node weights, we set $\eta = 0.0001, \lambda = 0.0001, N = 4$.

*6.3.5. Item recency*

The parameters of the hashtag recency are the time frame $\Delta t$, and the exponent $\alpha$. For the later, one could look at the inter-event distribution of the data set, and find the exponent with the best fit. The distribution is shown in Figure 2 is indeed power-law as suggested in Section 3.3, and $\alpha = 1.2$ results in the best interpolation. The ranking performance for the parameters are shown is Figure 9. The time frame does not appear to have a strong influence on the performance, except for shorter ones. The optimal value for the exponent seems to be around 2, and the corresponding performance is considerably better than for the value of 1.2 suggested from interpolation of the data. In what follows, we set $\Delta t = 2h$, and $\alpha = 2$. For the online SGD that learns the node weights, we set $\eta = 0.0001, \lambda = 0.0001, N = 4$.

*6.3.6. Exponentially weighted decay*

This method has the halving time $t_{1/2}$ as main parameter. The optimal value for the halving time appears to be around 15 minutes (see Fig. 10), decaying for values larger than half hour. The performance with respect to the negative rate is interesting, because this is the only method for which increasing the negative rate to large values improves the performance. The chosen set of parameters are: $t_{1/2} = 900, \eta = 0.0001, \lambda = 0.0001, N = 10,000$.

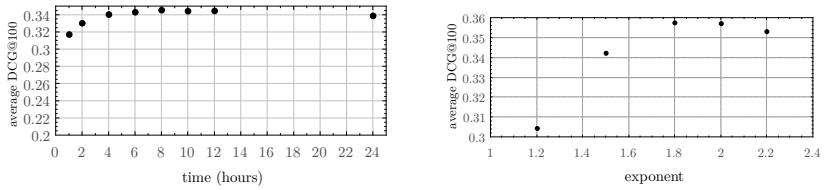


Figure 9: Average DCG@100 as the function of parameter $\Delta t$ (**left**), and $\alpha$ (**right**) for the recency and tree based recommender.

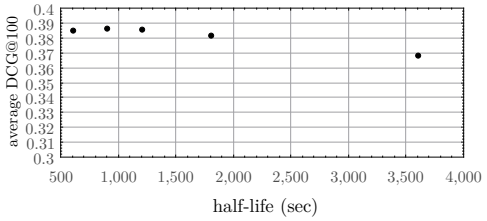


Figure 10: Average DCG@100 as the function of the half-life parameter of the exponentially weighted tree based recommender.

### *6.4. Performance comparison*

In this section, we compare the performance of the various models. In the graphs we show the average cumulative DCG, that is the average DCG until a certain number of days.

For the tree based methods we show the variant with learned weight and the variants with fixed weight (see Section 3.1): world, continent, country, leaves, and tree. The graphs for the three scoring functions (temporal popularity, hashtag recency, and exponentially weighted decay) are shown in Figures 11, 12, and 13. For the popularity and the exponentially weighted functions the relative performance of the variants are rather similar: *leaves* performs the poorest, then *world* and *continents*, with the tree and country variants being close to each-other as best performers. It seems that country is the right level of granularity here, going higher or lower in the tree impairing the performance. The similarity between the two functions is not surprising since both estimate popularity.

For hashtag recency, the relative performance is different. The country still seems as the best level of granularity, but the tree variant with learned weights is considerably better. Overall, we can conclude that the learning algorithm is able to find the right level of granularity (in terms of performance at least), and is able to outperform it in some cases.

The performance of the tree based methods with learned weights, and the three baselines, nearest neighbor and matrix factorization, are compared in Figure 14. The three baselines perform much worse than the tree based methods, with the factor model being the better of the three over longer period. Of the tree based methods, the exponentially weighted decay outperforms the other

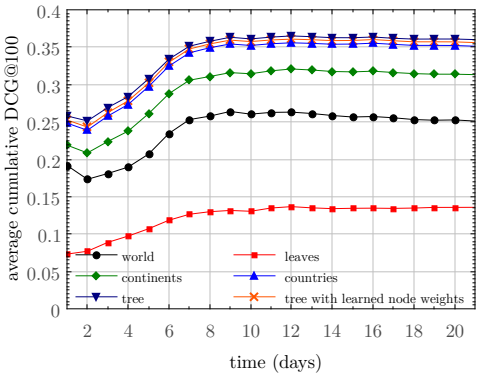


Figure 11: Average cumulative DCG@100 for the tree based popularity model and its baseline variants.

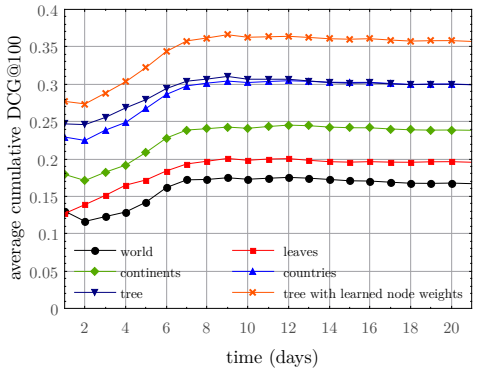


Figure 12: Average cumulative DCG@100 curves of the recency based methods.

two, which have similar performance. In the beginning temporal popularity has similar performance to the exponentially weighted decay, while later its performance decreases relative to the latter. This can be explained by the fact that the exponentially weighted functions can remember past events for a longer time, while the popularity forgets the events that are out of its time window. In this sense, it can be seen that the exponential scoring function takes advantage of the ideas from the other two scoring functions.

### *6.5. Cold start users*

In this section we consider how the number of tweets in the dataset for a user influences the recommendation performance. The average DCG for the hashtag recency, the content, and the factor model is shown if Figure 15. The same dataset is used for training as before, but the DCG is averaged for users that have a certain number of posted hashtags during the measurement. It can be seen that the factor model performs poorly for users that have only a few tweets.

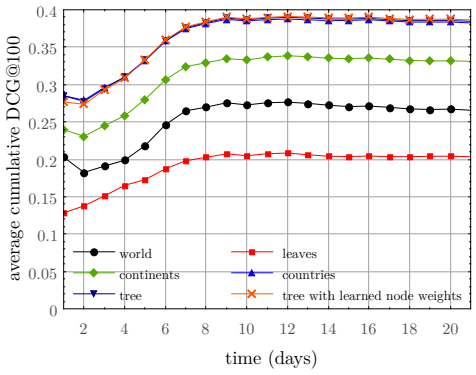


Figure 13: Average cumulative DCG@100 curves of the exponential decay methods.

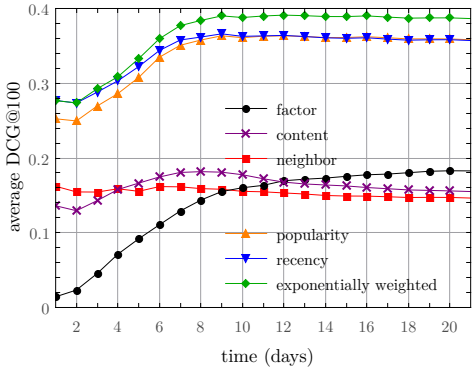


Figure 14: Best performances of the models.

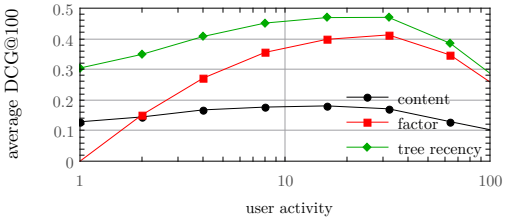


Figure 15: Performance of the recency, the content, and the matrix factorization model as the function of user activity.

This is natural since it is personalized model, and it has little information about these users. The factor model performs worse than the tree based model even for users with higher activity, but the difference between the two is becoming smaller. The content based model is better than the factor model for new users, but in general it performs significantly worse compared to the tree based model for all types of users.s

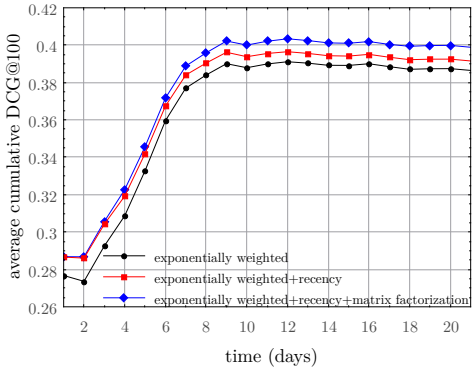


Figure 16: Combination of the best three different models.

Table 3: Best performance methods and their combination.

| | DCG@100 | DCG@10 |
|---|---|---|
| content based | 0.1555 | 0.1490 |
| nearest neighbor | 0.1464 | 0.0970 |
| matrix factorization (mf) | 0.1827 | 0.1638 |
| popularity (pop) | 0.3601 | 0.3370 |
| recency (rec) | 0.3571 | 0.3233 |
| exponentially weighted (exp) | 0.3865 | 0.3577 |
| exp + rec | 0.3916 | 0.3619 |
| exp + rec + mf | 0.3989 | 0.3693 |

### *6.6. Online combination*

In our final experiments we compared and combined our strongest methods. In Figure 16 we plotted the average cumulative DCG@100, for the standalone exponential weighted method, in combination with the recency method, and in combination with the recency method and the factor method. The popularity method did not improved the exponentially weighted method and it is not included. It is easy to see that the hashtag recency method (where the tree and learning is crucial) improves in combination with the base method. Similarly, personalization with the factor model improves as well in combination with the tree based methods. Table 3 summarizes the performances of the standalone methods and their best combinations.

### *6.7. Recommending items already seen*

While our aim is to recommend new hashtags for the user, for reference we include results when we recommend hashtags that the given user have already seen. In these experiments, we evaluate our recommendation on all events in the time series. The performance of the six different models with best parameters are shown in Fig. 17. In general, DCG@100 is higher for all models as this

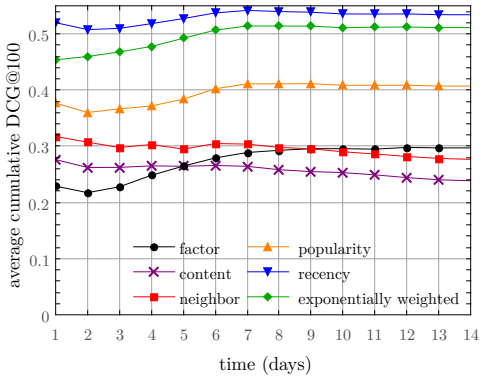


Figure 17: Best methods for predicting new as well as already used hashtags.

is task is easier compared to the new hashtag recommendation. The relative performance of the models are roughly the same as for the new hashtag recommendation. The only exception is that the recency model that performs better than the exponentially weighted model.

## 7. Conclusions

We described new general online, location based learning methods that are beneficial for recommending items with ambiguous geolocation that may change in time. While we believe that our methods apply for most geographic information systems, e.g. hotels, restaurants, etc., for illustration, we measured our methods for Twitter hashtags by, at a given time instance, recommending a top list of hashtags to adopt for users with known GPS position. Since hashtags are not directly bound to a location, may be geographically spread, and vary in popularity at different times, we designed methods that exploit the time and location context.

In our main findings, it has turned out that traditional methods such as matrix factorization, nearest neighbor and event content based hashtag recommendation perform weak for this task. Our best methods are based on measuring popularity and recency over a geographical hierarchy and learn the importance of the locations. Surprisingly, user personalization has little contribution to recommendation quality, hence our best methods apply in the user cold start setting as well. When restricting our findings to the task of hashtag recommendation, we emphasize that the location based recommenders outperform content even *without having access to the text of the actual candidate tweet.*

An important finding is that recency, the information on the very last occurrence of an item, has an element orthogonal to temporal popularity that combine well in recommenders. The exponential decay method gives a good compromise between mid-range more stable popularity and very recent but noisy activity, but even combining all three methods gives additional gains. Also, recency captures very well the particular events happening at the GADM tree location

nodes and for recency, learning the relevance of a node for a user gives strong improvement. On the other hand, popularity and exponential decay already saturates with information near the resolution of countries and little improvement is reached beyond.

In future work, we would like to use our method for contextualized geographic recommendation in other tasks such as hotel or restaurant recommendation. We stated our method in a general way independent of Twitter characteristics. In fact, we only used Twitter as an easy-to-access data set for illustrating our methods.

Some particular ideas that can be easily added to our experimentation include (1) Comparing cluster and KD-tree based methods to the predefined GADM tree to test whether items obey geopolitical boundaries or follow different patterns; (2) Include content based similarity in the location tree nodes, by combining the nearest neighbor formula (5) with content similarity, or even use popularity and recency for the terms and not just the hashtags.

## 8. Acknowledgments

The publication was supported in part by the Momentum Grant of the Hungarian Academy of Sciences, by OTKA NK 105645, and the PIAC_13-1-2013-0197 projects. The projects are supported by Hungarian Government, managed by the National Development Agency, and financed by the Research and Technology Innovation Fund.

[1] J. Abernethy, K. Canini, J. Langford, and A. Simma. Online collaborative filtering. *University of California at Berkeley, Tech. Rep*, 2007.

[2] E. Bakshy, J. M. H., W. A. Mason, and D. J. Watts. Everyone's an influencer: quantifying influence on twitter. In *WSDM*, pages 65–74. ACM, 2011.

[3] E. Bakshy, J. M. Hofman, W. A. Mason, and D. J. Watts. Identifying influencers on twitter. In *WSDM*, 2011.

[4] J. Bao, Y. Zheng, and M. F. Mokbel. Location-based and preference-aware recommendation using sparse geo-social networking data. In *SIGSPATIAL*, pages 199–208. 2012.

[5] J. Bao, Y. Zheng, D. Wilkie, and M. F. Mokbel. A survey on recommendations in location-based social networks. *ACM Transaction on Intelligent Systems and Technology (to be published)*, 2013.

[6] A.-L. Barabasi. The origin of bursts and heavy tails in human dynamics. *Nature*, 435(7039):207–211, 2005.

[7] B. Berjani and T. Strufe. A recommendation system for spots in location-based online social networks. In *Proceedings of the 4th Workshop on Social Network Systems*, page 4. ACM, 2011.

[8] C. Chen, H. Yin, J. Yao, and B. Cui. Terec: A temporal recommender system over tweet stream. *Proceedings of the VLDB Endowment*, 6(12):1254–1257, 2013.

[9] C. Cheng, H. Yang, I. King, and M. R. Lyu. Fused matrix factorization with geographical and social influence in location-based social networks. In *AAAI*, volume 12, page 1, 2012.

[10] J. Cheng, L. Adamic, P. A. Dow, J. M. Kleinberg, and J. Leskovec. Can cascades be predicted? In *WWW*, pages 925–936. 2014.

[11] Z. Cheng, J. Caverlee, and K. Lee. You are where you tweet: a content-based approach to geo-locating twitter users. In *CIKM*, pages 759–768. ACM, 2010.

[12] P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *RecSys*, pages 39–46. ACM, 2010.

[13] M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS)*, 22(1):143–177, 2004.

[14] E. Diaz-Aviles, L. Drumond, L. Schmidt-Thieme, and W. Nejdl. Real-time top-n recommendation in social streams. In *RecSys*, pages 59–66. ACM, 2012.

[15] L. Dobos, J. Szule, T. Bodnár, T. Hanyecz, T. Sebok, D. Kondor, Z. Kallus, J. Stéger, I. Csabai, and G. Vattay. A multi-terabyte relational database for geo-tagged social network data. In *CogInfoCom*, pages 289–294. IEEE, 2013.

[16] H. Gao, J. Tang, X. Hu, and H. Liu. Exploring temporal effects for location recommendation on location-based social networks. In *RecSys*, pages 93–100. ACM, 2013.

[17] M. Harvey and F. Crestani. Long time, no tweets! time-aware personalised hashtag suggestion. In *Advances in Information Retrieval*, pages 581–592. Springer, 2015.

[18] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *ICDM*, *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 263–272. IEEE, 2008.

[19] B. Huberman, D. Romero, and F. Wu. Social networks that matter: Twitter under the microscope. *Available at SSRN 1313405*, 2008.

[20] K. Y. Kamath, J. Caverlee, K. Lee, and Z. Cheng. Spatio-temporal dynamics of online memes: a study of geo-tagged tweets. In *WWW*, pages 667–678. 2013.

[21] M. Kivelä and M. A. Porter. Estimating inter-event time distributions from finite observation periods in communication networks. *arXiv preprint arXiv:1412.8388*, 2014.

[22] M.-H. Kuo, L.-C. Chen, and C.-W. Liang. Building and evaluating a location-based service recommendation system with a preference adjustment mechanism. *Expert Systems with Applications*, 36(2):3543–3554, 2009.

[23] T. Kurashima, T. Iwata, G. Irie, and K. Fujimura. Travel route recommendation using geotags in photo sharing sites. In *CIKM*, pages 579–588. ACM, 2010.

[24] H. Kwak, C. Lee, H. Park, and S. Moon. What is twitter, a social network or a news media? In *WWW*, pages 591–600. ACM, 2010.

[25] S. M. Kywe, T.-A. Hoang, E.-P. Lim, and F. Zhu. On recommending hashtags in twitter networks. In *Social Informatics*, pages 337–350. Springer, 2012.

[26] R. Lee and K. Sumiya. Measuring geographical regularities of crowd behaviors for twitter-based geo-social event detection. In *SIGSPATIAL*, pages 1–10. ACM, 2010.

[27] K. Lerman and R. Ghosh. Information contagion: An empirical study of the spread of news on digg and twitter social networks. In *ICWSM*, 2010.

[28] J. J. Levandoski, M. Sarwat, A. Eldawy, and M. F. Mokbel. Lars: A location-aware recommender system. In *ICDE*, pages 450–461. IEEE, 2012.

[29] Z. Ma, A. Sun, and G. Cong. Will this# hashtag be popular tomorrow? In *SIGIR*, pages 1173–1174. ACM, 2012.

[30] Z. Ma, A. Sun, and G. Cong. On predicting the popularity of newly emerging hashtags in twitter. *Journal of the American Society for Information Science and Technology*, 64(7):1399–1410, 2013.

[31] D. Mocanu, A. Baronchelli, N. Perra, B. Gonçalves, Q. Zhang, and A. Vespignani. The twitter of babel: Mapping world languages through microblogging platforms. *PloS one*, 8(4):e61981, 2013.

[32] R. Pálovics and A. A. Benczúr. Temporal influence over the last.fm social network. *Social Netw. Analys. Mining*, 5(1):4, 2015.

[33] R. Pálovics, A. A. Benczúr, L. Kocsis, T. Kiss, and E. Frigó. Exploiting temporal influence in online recommendation. In *RecSys*, pages 273–280. ACM, 2014.

[34] S. Petrovic, M. Osborne, and V. Lavrenko. Rt to win! predicting message propagation in twitter. In *ICWSM*, 2011.

[35] I. Pilászy and D. Tikk. Recommending new movies: even a few ratings are more valuable than metadata. In *RecSys*, pages 93–100. ACM, 2009.

[36] I. Pilászy, D. Zibriczky, and D. Tikk. Fast als-based matrix factorization for explicit and implicit feedback datasets. In *RecSys*, pages 71–78. ACM, 2010.

[37] F. Preparata and M. Shamos. Computational geometry: an introduction. *Texts and monographs in computer science Show all parts in this series*, 1988.

[38] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock. Methods and metrics for cold-start recommendations. In *SIGIR*, pages 253–260. ACM, 2002.

[39] U. Shardanand and P. Maes. Social information filtering: algorithms for automating "word of mouth". In *SIGCHI*, pages 210–217. ACM, 1995.

[40] P. Symeonidis, D. Ntempos, and Y. Manolopoulos. Location-based social networks. In *Recommender Systems for Location-based Social Networks*, pages 35–48. Springer, 2014.

[41] A. Vazquez, B. Racz, A. Lukacs, and A.-L. Barabasi. Impact of non-poissonian activity patterns on spreading processes. *Physical review letters*, 98(15):158702, 2007.

[42] M. Ye, P. Yin, and W.-C. Lee. Location recommendation for location-based social networks. In *SIGSPATIAL*, pages 458–461. ACM, 2010.

[43] M. Ye, P. Yin, W.-C. Lee, and D.-L. Lee. Exploiting geographical influence for collaborative point-of-interest recommendation. In *SIGIR*, pages 325–334. ACM, 2011.

[44] E. Zangerle, W. Gassler, and G. Specht. On the impact of text similarity functions on hashtag recommendations in microblogging environments. *Social Network Analysis and Mining*, 3(4):889–898, 2013.

[45] V. W. Zheng, Y. Zheng, X. Xie, and Q. Yang. Collaborative location and activity recommendations with gps history data. In *WWW*, pages 1029–1038. ACM, 2010.

[46] Y. Zheng, L. Zhang, X. Xie, and W.-Y. Ma. Mining interesting locations and travel sequences from gps trajectories. In *WWW*, pages 791–800. ACM, 2009.

[47] D. Zhou, B. Wang, S. M. Rahimi, and X. Wang. A study of recommending locations on location-based social network by collaborative filtering. In *Advances in Artificial Intelligence*, pages 255–266. Springer, 2012.