

Data Stream Analysis for Location-Aware Collaborative Information Retrieval

Andreas Behrend¹, Frank Reichartz², Christian Dorau¹, and Rainer Manthey¹

¹ University of Bonn, Institute of CS III, D-53117 Bonn

² Fraunhofer Institute for IAIS, Schloss Birlinghoven D-53754, Germany
{behrend,dorau,manthey}@cs.uni-bonn.de
frank.reichartz@iais.fraunhofer.de

Abstract. We propose a new approach for enhancing collaborative information retrieval by means of incorporating positional data for a location-aware personalized retrieval process. In our framework, the collaboration between users will be established by building communities based on matching user attributes in a uniform user model. This allows for incorporating automated intra-community collaboration into the retrieval process. In addition, continuously changing location-based similarity measures are employed with respect to queries posed using mobile devices in order to enhance the quality of the community dependent answer rankings. The consideration of continuously arriving user positions, however, leads to a high-frequency stream of data. For the efficient processing and analysis of this stream, incremental data stream processing techniques are employed. Our interdisciplinary approach incorporates both techniques from information retrieval and data stream processing to achieve an extended retrieval process in a collaborative environment.

1 Introduction

Early retrieval systems were much like databases in the sense that only an exact match between a query and a document made a document relevant with respect to a query. Drawbacks of this approach are the missing relevance ranking and the non-consideration of information deducible from the queried documents into the retrieval process. Due to the rapidly increasing number of documents, the need for a more sophisticated form of retrieval emerged. Nowadays, information retrieval (IR) systems not only have to deal with a huge number of documents but also with a large number of users. Today's websearch engines have shown how fast it is possible to even search web-scale document collections. However, they have been only optimized for an efficient query answering for the common user. Recent trends go towards personalization [13, 10, 3, 2] of search results and systems, aiming at improving the retrieval process by taking the particular interests of individual users into account. These systems, however, lack inherent support of user collaboration which allows for enhancing the retrieval process by considering gathered community knowledge. To this end, communities of similar users are built based on matching user attributes given in a uniform user model and query-related feedback is continuously accumulated for future query refinements. The

identification of user communities with shared short- or long-term interests has been already proposed in [15, 3]. Such interests are not only deducible from the context from which the interactions with the systems take place but also from implicit and explicit user feedback. It is even possible to deduce some of the interests of users by analyzing their queries [13].

Mobile computing devices like laptops, PDAs, and smart as well as cell phones, however, are ubiquitous and may provide additional context information which could improve the support of user collaboration and enhance IR. While context-sensitive mobile applications have been recognized in the software development research community, there is only low attention on research for mobile IR. In our approach, we propose to use location-based data to build location-specific communities such that not only similar interests but also the spatial proximity is considered for user collaboration. This is reasonable since studies have shown that spatial proximity could be used as a measure for the likelihood of a successful collaboration between users, which share a spatial region of interest or were located in spatial proximity [8, 20]. The consideration of positional data for location-specific community building and query filtering leads to a huge amount of data which makes permanent storage impossible and instant processing necessary. Consequently, a main challenge is the efficient processing of this data stream caused by the continuously changing positions and the high number of users interacting with our mobile retrieval system. The analysis of data streams has received a lot of interest from the database research community within recent years [5]. Based on our experiences with the TInTo system [7], we propose to employ incremental update propagation techniques for analyzing the streams of positional user data because of the feasibility of this approach.

This paper is organized as follows: In Section 2 our general approach is presented including the employed user model for capturing the characteristics of each user and the system architecture. Afterwards, the efficient analysis of data streams for updating the position dependent attribute values within each user model instance is discussed. Section 4 describes our techniques for user model-based community discovery and collaborative IR through intra-community interaction. Before concluding this paper related work is discussed in Section 5.

2 Our Approach

In this section we describe our general approach for a location-aware collaborative IR system. At first, we specify how personalization of retrieval results is achieved by building location-specific communities and by applying query filtering. Afterwards, the architecture of our system is presented and the main functionalities of its components are briefly discussed.

2.1 Personalization of Retrieval Results

Techniques for personalization of retrieval results have recently received great interest from the IR research community. A possible way is the incorporation of

the local environment of a user into the retrieval process by deducing user-related information from locally stored documents. Experiences with such approaches suggest that the retrieval quality can be enhanced by the usage of information extracted from user-specific data. For our system we opted for the usage of a centrally stored explicit *user model* containing attributes for describing the characteristics of users. We prefer this way over an ontology-based one because it allows for representing different users in a uniform way in a relational database. Additionally, the relations between different attributes are indirectly represented by our similarity functions for community discovery (cf. Section 4.2) and thus, do not have to be explicitly given in form of an ontology.

Other related approaches for personalization are mostly limited to term-based attributes in order to utilize existing techniques for the incorporation of feedback into the relevance ranking. These approaches solely utilize user-specific feedbacks to generate personalized rankings and cannot handle other (non-textual) types of attributes such as positional ones. The consideration of different types of attributes in our approach, however, requires some kind of measurement for the semantic similarity of attribute values. It is easy to see that those similarity measures are directly related to the attribute types and are specifically given for each type (cf. Section 4.2). An instance of the user model is stored and continuously updated for each user on the server side of our system whereas each user has to manage an *user profile* on the client side in which he stores selected attribute values for the usage by our system. Note that the user model differs from the user profile in additional attributes that are not provided by the user but derived from the system using implicit feedback data.

The employed user profiles contain one attribute for storing the current location of a user. The location could be derived by a GPS component or by using access point signals of mobile devices. Updates of the *user profile* are submitted to the system either continuously within a fixed interval or ad-hoc together with a posed query. In our centrally stored user model, however, various location-related attributes are managed which basically represent a summary of a user movement with respect to three predefined time granularities (cf. Section 3.1). The computation of these derived attribute as well as the determination of location-specific communities is performed using SQL views which are incrementally maintained using update propagation. This allows for the fast computation of matching communities based on the maximal amount of data and the efficient determination of new community characteristics. The discovered communities will be used to foster the collaboration of users. We establish an intra-community information sharing for allowing implicit user collaboration, that is, the automated recommendation of documents based on personal information on community members which is used to enhance relevance ranking.

It is obvious that location-specific communities with respect to similar long and medium time granularities are quite useful for enhancing the quality of the retrieval process. As users who share a spatial proximity are likely to have acknowledged some kind of local experiences, their feedback may be considered more valuable with respect to a location-specific query (e.g., concerning hotels

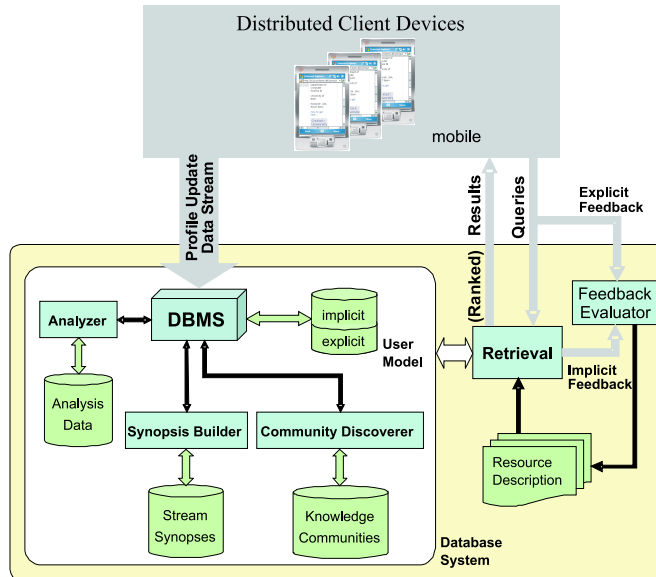


Fig. 1. Location-Aware Collaborative IR System

or sightseeing points). A tourist may even consider a location-specific community with short time granularity useful, e.g. in order to meet other community members with a similar information need.

2.2 System Overview

Figure 1 summarizes the components of our general framework. From mobile client devices profile updates are continuously sent to the database component of our location-aware IR system. These comprise all values of explicit dynamic user model attributes, e.g., a user's current position, which are then employed to update the explicit part of the user model. In order to maintain the implicit part of the user model, various other components are utilized:

1. The *Synopses Builder* is employed to create summaries from various data streams which may not be saved in their entirety.
2. The *Analyzer* is used to (incrementally) derive analysis data by employing (delta) SQL views.
3. The *Community Discoverer* uses the values of explicit user attributes together with derived data from the analysis phase to determine knowledge communities.

All data derived by the above components are managed by the underlying DBMS and are accessed by the retrieval components of our system via the JDBC interface. The retrieval component receives client-side queries and returns as results

ranked lists of relevant documents or resources. For the ranking the current user-specific data given in the user model is employed together with resource descriptions. Finally, the feedback evaluator extracts implicit feedback from user queries and processes explicit user feedback in order to keep the resource descriptions up to date.

3 Positional Data Streams

The processing of data streams has attracted a lot of attention within the database community recently and led to various extension proposals for conventional databases [5]. In a data stream context, some or all of the data are not available for random access from disk or memory, but rather arrive as one or more continuous data streams. In general, data streams are considered to be different from conventional stored relations in several ways. The elements in a stream usually arrive online and are potentially unbounded in size. Additionally, each stream element is timestamped and the arrival frequency is usually so high that the stream data cannot be completely stored within the underlying database. Consequently, various techniques have been proposed to compute summary structures from a given stream (so-called *synopses* [1, 17]) which are stored instead of the stream itself.

Queries over data streams vary in two important aspects from queries in a traditional database management system. The first aspect is covered by the distinction between *one-time queries* and *continuous queries* [31]. One-time queries are evaluated once over a point-in-time snapshot of the data set whereas continuous queries must be evaluated continuously as new data arrive within an underlying stream. The second aspect is covered by the distinction between *predefined queries* and *ad hoc queries*. A predefined query is supplied to a data stream management system before any relevant data has arrived. Predefined queries are usually continuous queries and simplify query optimization and memory management in contrast to arbitrary unscheduled ad hoc queries.

Query processing in a data stream context faces the problem of limited time and memory resources which lead to various proposals for approximate query answering. One technique for this is to evaluate the query not over the entire past history of stream data but rather over sliding windows of most recent data from the streams, only. In our approach, the efficient evaluation of predefined queries using sliding windows of fixed sizes is considered.

3.1 Analysis of Positional Data Streams

Particularly for the purpose of location-aware community building it is important to analyze where a user is primarily located. For this purpose, we suppose every mobile device to be equipped with a GPS module permanently updating the corresponding parts of a user profile with an accuracy up to 15 feet. As part of the earlier described automated user model update process, the accordingly changed user profile (or relevant components of it) is then continuously sent to

the provider. This is the basis for any location-based application such as routing, environmental IR and especially community building and ought to be done in a timely fashion, e.g. every 30 seconds. The resulting stream consists of triples of the form $\langle UID, Pos, Time \rangle$ where UID denotes a user's ID and Pos his position at time point $Time$. In our system, the generic user model contains three types of attributes whose current values directly depend on this stream of positional data and are described in more detail in the following.

A *location history* represents a user's movement by recording his location over a certain period in time. In order to distinguish a user's short term behavior from his medium and long term behavior, three corresponding location histories of different granularity are maintained by our system. While the short term location history comprises the most recent 60 minutes of GPS data, the medium and long term histories cover the last month and the last year, respectively. Due to the high volume of positional data transmitted it is not feasible to record the complete content of the corresponding stream. Instead a summary of the user movement - the *short/medium/long term location history synopsis* - is built and explicitly stored in the user model.

To this end, the incoming positional data is clustered according to predefined rectangular regions. To guarantee similar memory requirements for any kind of location history synopsis, the region size increases with the length of the underlying time period. Thus, a location history synopsis consists of a chronologically ordered sequence of regions together with the information how long the user stayed within each particular one. Note that continuously updating the synopses from the stream of GPS data is - even for a large numbers of users - computationally undemanding and leads to a considerable data reduction. For any further computation depending on the location history these synopses are used instead of the underlying GPS data stream.

UID	RegionID	RegionEntry	RegionStay
...
14122	44315	2007-05-04 07:34:26	37
14123	50456	2007-05-04 07:30:21	60
14123	50457	2007-05-04 07:31:21	330
14123	50456	2007-05-04 07:36:51	120
...

Fig. 2. Short Term Synopses View V_1

For each of the three location history synopses, a *location history center* is maintained pinpointing the center of a user's movement. Given the visited regions, the value of this attribute is calculated as the mean of region centers weighted by the duration of region stay. Note that the meaningfulness of this value strongly depends on the observed variability of positional data. For each granularity we assume the location history synopses of all users to be stored in form of a materialized SQL view. For computing the users' location history centers, we use an SQL view defined over the location history synopses extracted

from the stream of GPS data. The sample materialized view V_1 in Figure 2 represents a snapshot of the short term location history synopsis in which each user is associated with the regions he stayed within the last 60 minutes.

The attribute RegionEntry records the entry time for the corresponding region while the duration of stay in seconds for each region is stored using the RegionStay attribute. A new region entry tuple signals the exit from the most recent visited region at the same time and may even be a reentering into an already visited region as in the example above. As soon as the exit time of the last visited region falls outside the 60 minutes interval, the corresponding region entry tuples will be deleted. The time of region stay for the current region is updated every 30 seconds while new GPS data arrives until a new region is entered. For a given user, the location history center is defined as

$$\frac{1}{RS_{\Sigma} \cdot RE_{\#}} \cdot \sum_{i=1}^{RE_{\#}} RS_i \cdot [x_i, y_i]$$

where $RE_{\#}$ denotes the total number of region entry tuples of a given user, RS_i is the time this user stayed in a region whose center coordinates are given by $[x_i, y_i]$ and RS_{Σ} the total number of seconds he stayed in regions. Note that RS_{Σ} is ideally 3600 seconds (or 60 minutes) but is usually a little bit higher. This is due to the fact that only the smallest number of region entries of a user is considered which cover together at least 60 minutes. The result of the above equation is a pair of coordinates $[x_c, y_c]$ representing the location history center for a single user. Its computation is realized by several SQL views. As an example, for determining the sub-expression $\sum_{i=1}^{RE_{\#}} RS_i \cdot [x_i, y_i]$ the following view V_1 is employed

```
CREATE VIEW V2 AS
SELECT UID, sum(RegionStay*REGION.X) AS SumX,
           sum(RegionStay*REGION.Y) AS SumY
FROM V1,REGION
WHERE V1.RegionID=REGION.ID
GROUP BY UID
```

while the table REGION stores the center coordinates $[X, Y]$ for each region ID. To quantify the meaningfulness of a location history center, a third attribute is used within our system: the *central location history region*. Basically, the value of this attribute is the radius of the smallest circle around the location history center covering 80 percent of all/the positions contained in the corresponding location history synopsis. A small radius represents a low positional variability and thus a high significance of the location history center while a large radius correspondingly indicates a low significance. Again, the values of this attribute are computed for each user employing SQL views.

Summarizing the discussion above, SQL views are employed for processing a continuous GPS data stream generating induced streams of updated location

history synopses. The values of location-dependent attributes result from continuous queries over these induced data streams. Their computation is again realized via SQL views and is based on sliding windows whose size depends on the respective time granularity (short/medium/long term).

3.2 Incremental Stream Analysis

In the previous section, we distinguished the primary stream of GPS data from the induced streams of location history updates. For computing location-dependent user attribute values, continuous queries over these induced streams are posed as SQL views. Since these pre-defined continuous queries are based on sliding windows of considerable sizes, update propagation techniques can be used for focussing on the changes resulting from a new user's position.

Update propagation has been intensively studied for many years mainly in the context of integrity checking and materialized views maintenance [16]. The key idea is to transform each SQL view already at schema design time to a so-called *delta view*, a specialized version referring to changes in the underlying tables, only. The original view definitions are solely employed once for materializing their initial answers while the specialized versions are employed afterwards for continuously updating the materialized results. Under the assumption that a great portion of the materialized view content remains unchanged, the application of delta views considerably enhances the efficiency of view maintenance.

To illustrate our approach, let us consider again the view definition V_2 which was employed as part of the computation for users' location history centers. V_2 was based on the region entries for each users given in V_1 (cf. Figure 2) and a region table, both used to compute total time of a user's region stay within the last 60 minutes. The changes with respect to V_1 resulting from users movements are given by table `Ins_V_1` comprising newly inserted region entries or updated times of region stays and by table `Del_V_1` storing deleted region entries, respectively. The additional time of region stay for each user resulting from tuples within `Ins_V_1` can be determined incrementally by applying the following specialized view `Ins_V2`:

```
CREATE VIEW Ins_V2 AS
  SELECT UID, sum(Ins_V1.RegionStay*REGION.X) AS DX,
             sum(Ins_V1.RegionStay*REGION.Y) AS DY
  FROM   Ins_V1,Region
  WHERE  Ins_V1.UID=Ins_V1.UID
  GROUP BY Ins_V1.UID
```

Instead of recalculating the total time of region stay, solely the new time periods to be added for each user are determined. The accumulated time periods which fall outside the 60 minutes interval and thus, have to be deleted for each user can be calculated analogously. Having these accumulated changes, the materialized version of V_2 can be updated using the following statement:

```

Update V2_Mat SET
SumX=SumX+(SELECT Ins_V2.DX - Del_V2.DX FROM Ins_V2
            OUTER JOIN Del_V2 ON Ins_V2.UID=Del_V2.UID
            WHERE (V2_Mat.UID=Ins_V2.UID OR V2_Mat.UID=Del_V2.UID)),
SumY=SumY+(SELECT Ins_V2.DY - Del_V2.DY FROM Ins_V2
            OUTER JOIN Del_V2 ON Ins_V2.UID=Del_V2.UID
            WHERE (V2_Mat.UID=Ins_V2.UID OR V2_Mat.UID=Del_V2.UID))
WHERE EXISTS
(SELECT * FROM Ins_V2 OUTER JOIN Del_V2 ON Ins_V2.UID=Del_V2.UID
 WHERE V2_Delta.UID=V2_Mat.UID)

```

The update statement modifies for each user the accumulated coordinate values `SumX` and `SumY` if a time period to be added or to be deleted is recorded for him by corresponding tuples in `Ins_V2` or `Del_V2`, respectively. As a user ID might occur in one of these tables only, an OUTER JOIN operation is employed for generating corresponding Null values interpreted as zeros within the subsequent summations. The update statement is continuously evaluated in regular intervals. The resulting changes with respect to table V_2 are in turn applied to incrementally updating the location history centers of each user.

In the same way, all other views employed in our system are systematically transformed into a specialized delta view. For lack of a view compiler, however, all views are transformed manually into delta views which is unproblematic since all continuous queries considered in our system are pre-defined. The utilization of delta views imposes only a small computational overhead resulting from more complex view definitions and the maintenance of materialized data in comparison to the reduced number of intermediate derivations. The performance gains scales with the sliding window size as the ratio of changed and unchanged tuples becomes smaller. That is, the performance gain considerably increases when applying the above mentioned delta views to medium and long term location history synopses.

In the context of stream analysis, update propagation methods for computing exact answers have been neglected in favor of studying approximative query answering techniques. In particular, sliding windows are often thought of as an approximation techniques reluctantly imposed due to the infeasibility of computing over all historical data. In our scenario, however, sliding windows are part of the desired query semantics. Due to their considerable sizes, update propagation becomes a quite natural choice for computing the small number of deletions and insertions while usually a great portion of the derived tuples remains unchanged.

3.3 Performance Issues

Obviously, the incorporation of positional data and community specific interests can only enhance the quality of the entire retrieval process. But with an increased amount of fast changing data the question arises whether this information can be timely processed by our system in view of a large number of possible clients. At the current stage of implementation a complete performance evaluation of our

system is not possible yet. However, the feasibility of our approach is illustrated by the technical indicator tool TInTo [7] in which delta techniques for analyzing high frequent data streams have been successfully employed.

TinTO is an experimental system aiming at demonstrating the usefulness and feasibility of applying conventional SQL queries for analyzing a wide spectrum of data streams. As application area serves the analysis of streams of stock market data, exhibiting sufficiently many of those characteristics for which relational query technology can be reasonably considered in a stream context. TinTO is a technical investor tool for computing so-called technical indicators - numerical values calculated from complex functions over stock market data - using SQL views. The underlying data stream consists of timestamped stock prices which may change several times within second. An indicator value is computed with respect to a user-defined and indicator-specific part of a chosen price history which corresponds to a sliding window continuously shifting as new price data arrives. In order to perform a synchronized update of indicator values, TInTo employs delta-views over the high frequent stream of stock market prices. Compared with the complete re-evaluation of indicator views, this incremental approach led to a remarkable performance gain that fully scales with the sliding windows size allowing to recompute multi-level aggregates within microseconds.

In our location-aware IR system, values of location-dependant attributes result from continuous queries over an induced data stream of updated history synopses. Their computation is also based on delta views utilizing sliding windows whose size depends on the chosen time granularity (short/medium/long). As our system considers delta view complexities and window sizes similar to those of the TInTo system, it is reasonable to assume that a timely processing of the fast changing positional data streams by our proposed system is indeed possible.

4 Collaborative Retrieval

In this section, we consider community discovery and automated user interaction for a location-aware IR system in collaborative environments. Location-aware IR means to incorporate position-related attribute values into the relevance ranking process. In [4] it has been proposed to automatically associate positional data with documents and rank those documents higher which have a relatedness to these positional data. This will increase the retrieval quality at least for positional related queries.

4.1 User Model

The user model is applied to capture the characteristics of a user using attributes of an arbitrary type. The employed attributes can be classified according to various dimensions: dynamic vs. static, public vs. private, and implicit vs. explicit. Values of dynamic attributes are frequently changing whereas values of static attributes remains constant. For example, most position-related attributes are

dynamic ones while biography-related attributes are usually static. Additionally, the user may chose certain attributes to be private in order to make them not visible to other users. Values of explicit attributes are directly taken from user profiles whereas values of implicit attributes are generated by using explicit attributes and arbitrary other data in our system. User preferences – for instance, determined by his interaction history with our system – and community memberships are examples of implicit attributes. In order to have a flexible and accurate user model design we have chosen a function-based approach to model the characteristics of the users which abstracts from the technical realization and storage details.

Let $\mathcal{U} := \{u_1, \dots, u_n\}$ be the set of all current users and $\mathcal{A} := \{a_1, \dots, a_m\}$ be the set of all possible attributes within the user model. Each attribute a_i has its own domain \mathcal{D}_i . For each attribute a_i we define a function $v_i : \mathcal{U} \rightarrow \mathcal{D}_i$ which returns the current attribute value for a given user. Note that v_i are partial functions because attribute values may be undefined. For example, a user could decide not to specify his gender in his profile. Let $\mathcal{P} := \{p_1, \dots, p_k\} \subset \mathcal{A}$ be the set of attributes which are position-related, we define the content function $c : \mathcal{U} \rightarrow P(\mathcal{A}) ; u \mapsto \{a_i \in \mathcal{A} \mid v_i(u) \text{ isdefined}\}$ which maps each user to the set of attributes which are defined for him. Further let $\hat{c} : \mathcal{A} \rightarrow P(\mathcal{U}) ; a_i \mapsto \{u_j \in \mathcal{U} \mid a_i \in c(u_j)\}$ be the function which maps an attribute to the set of users having a value for this attribute defined. This formalization of the underlying user model abstracts from a technical realization and forms the formal basis for the community discovery in our approach.

4.2 Community Discovery

We propose a community discovery approach based on the presented user model. The consideration of dynamic positional attributes is made possible by an efficient processing of positional data streams. To build meaningful communities of users we classify users according to their similarity which is quantified by the grade of matching characteristics. Consequently, a measurement for similarity of two users is needed which is solely based on their user model attribute values.

Retrieval approaches incorporating communities into the retrieval process have shown a great potential for collaborative IR. However, these approaches usually consider a single attribute, only. For example, in the context of recommender systems and collaborative filtering clustering techniques have been used to improve the quality of recommendations [32, 19] which are limited to a collection comparison to determine user similarity. To this end, user similarity is either computed using item co-occurrences [29] or by comparing term distributions [6]. Other community-based IR systems (e.g. [3]) use queries within a search session to compute a similarity between sessions of different users which are again limited to one attribute type namely the session.

In contrast to this, each attribute of our user model can be associated with a function measuring the *semantic similarity* of two attribute values. There are similarity functions $s_i : \mathcal{D}_i \times \mathcal{D}_i \rightarrow \mathbb{R}$ for each attribute $a_i \in \mathcal{A}$. It is reasonable to restrict the co-domain of these functions to the interval $[0, 1]$ for normalizing

similarity values. For example, simple attributes with numerical values such as age, weight or height are associated with quite simple similarity functions. For all attributes with numerical values where the distance on the number line would represent a semantic similarity measure, the following function is used

$$s_{\text{num}} : x, y \mapsto \frac{\min(x, y)}{\max(x, y)}.$$

For estimating the similarity between local user collections, approaches from the field of collaborative filtering can be employed.

We define similarity functions operating on sets of attributes. Let $A := \{a_{i_1}, \dots, a_{i_m}\} \subset \mathcal{A}$ be the subset of all attributes which are considered for an instance of the community discovery process. With this in mind we can define a similarity function $S_A : (\mathcal{D}_{i_1} \times \dots \times \mathcal{D}_{i_m}) \times (\mathcal{D}_{i_1} \times \dots \times \mathcal{D}_{i_m}) \rightarrow \mathbb{R}$ which measures the semantic similarity of the values for this attribute set.

This increased expressiveness can be illustrated by considering the attributes *location history center* and *central location history region*. Each similarity alone could be easily measured. For the attribute *lhc* one could use the euclidean distance for the definition of a similarity function $s_{\text{lhc}} : x, y \mapsto \frac{1}{1 + \|x - y\|_2}$ and for the attribute *clhr* the function s_{num} .

A possible measure for similarity could be the intersection of the two circles defined by these attribute values which is given by function $S_{\text{loc}} : (r_1, (x_1, y_1)), (r_2, (x_2, y_2)) \mapsto \frac{1}{4\pi \cdot r_1} \sqrt{(-d + r_1 + r_2)} \cdot \sqrt{(d + r_1 - r_2)} \cdot \sqrt{(d - r_1 + r_2)} \cdot \sqrt{(d - r_1 - r_2)}$ with $d := \|x - y\|_2$. S_{loc} determines the amount of positional proximity which may serve as an indicator for common interests. Note that many similarity functions can be expressed in form of SQL views and therefore offer the opportunity to employ update propagation for their incremental maintenance as proposed in Section 3.

The usage of a similarity function allows a community discovery process that is independent of the attribute types. We use graph-based clustering algorithms which identify dense connected subgraphs for community discovery [9]. These algorithms partition the vertices of a graph $G = (V, E)$ into subsets C_1, \dots, C_m , forming clusters. Based on an attribute set A , one can define the graph $\mathcal{G}^A := (\hat{c}(A), E_A)$ with $E_A := \{(u_j, u_k) \in \hat{c}(A) \times \hat{c}(A) \mid S_A(v_i(u_j), v_i(u_k)) \geq \theta_A\}$ for a specific threshold θ_A . The vertices $\hat{c}(A)$ represent the set of users having a value for every attribute of A defined while the edges represent a similarity between users above the threshold θ_A . This transformation allows the employment of graph clustering algorithms for community discovery. The derived clusters correlate with the community of users with matching characteristics. We define $\mathcal{C}^A := (C_1^A, \dots, C_m^A)$ as the discovered communities. The graph represents a specialized form of social network where an edge exists between two users if an interaction is likely to occur.

Note that the partition of the user set depends on the attribute set and the employed similarity function. In our case, the clustering is based on positional attributes P and thus users are grouped into the same community who have positional similarities.

4.3 Intra-Community Interaction

In order to create an environment of collaborative information gathering, we consider two specific user model attributes. A user's *local collection* contains the locally stored documents or previously accessed documents whereas *personalization information* represents user-specific data derived from the local collection or the interaction history with the retrieval system. Both attributes are used to enhance the relevance ranking of a standard retrieval model.

The document collections of other community members can be incorporated in the retrieval process by assuming that documents occurring in these collections are particularly relevant. Consequently, the relevance ranking of a certain document depends on its community collections frequency which is the amount of local collections including this document. This approach is adopted from the field of user-based collaborative filtering, where intra-community recommendations of items (documents) are widely used to improve the filtering quality. In [13], Dou et. al. show that the quality of personalized IR can be improved by incorporating recommendations of similar users. While they solely consider predefined topics for computing user similarity, our approach allows for the employment of much more sophisticated user similarity measures.

Personalization information may be used in several ways to personalize the retrieval results. The approaches in [30, 10] use personalized term weights to rerank retrieval results. These weights are stored as part of the personalization information and can be averaged over all community members in order to gain community specific term weights ("*community weights*"). The relevance ranking modified by these community weights yields a collaboration enhanced ranking which emphasizes common interests of the community over individual ones.

The query independent relevance evidences obtained by using the local collection attribute and the personalization information attribute are employed according to the general guidelines in Craswell et. al. [12] for estimating the relevance $r(q, d, C)$ of a document d to a query q with respect to a community C . If a user u is member of the communities $C_u = \{C_{i_1}^{A_1}, \dots, C_{i_n}^{A_n}\}$ related to the attribute sets A_1, \dots, A_n the relevance $r(q, d, u)$ is determined by

$$r(q, d, u) = \sum_{C_{i_j}^{A_j} \in C_u} w_{u, A_j} \cdot \Lambda_{u, C_{i_j}^{A_j}} \cdot r(q, d, C_{i_j}^{A_j}).$$

$\Lambda_{u, C_{i_j}^{A_j}}$ is the average similarity of user u to all other members of community $C_{i_j}^{A_j}$ and w_{u, A_j} is user u 's importance weight for 'attribute set A_j '-based communities. This document ranking formula incorporates all available information to establish a collaborative retrieval process.

5 Related Work

The most frequently taken approach for text retrieval is the well-established BM-25 formula [24, 25, 28, 26] which can be quite easily modified in order to incorporate additional factors and relevance evidences. In [12], it is shown that

retrieval quality is directly related to the origin of factors and the query independence of sources. Additionally, the BM-25 formula inherently supports the integration of feedback information by modified Robertson-Sparck-Jones term weights which could be used for personalization [30, 10].

The modelling of user interests by user models is commonly employed for the personalization of rankings and in information filtering systems. In the context of IR, user models have been used in centralized systems [2] as well as in distributed environments [22] where the requirements on the model directly determine which techniques or forms of representation can be used. User models are also employed in other research fields to render possible a generalized personalization of applications [23].

Location-awareness has been already studied in geographic IR systems where the association of a document's location (or positional scope) with its content provide new ways to extract information about documents and users. The location of a document is either determined by analyzing its content [4] or - in case of web documents - by using the server-location [11]. The extraction quality of a toponym or a geospatial location for a document is, however, limited by the ambiguity problem [21]. The usefulness of ontologies for the retrieval of geospatial data has been shown in [18].

Context-sensitive IR approaches additionally consider several aspects of a retrieval request context [27]. In [13] it has been shown how user interactions with the system may provide relevant context information. These interactions allow for aggregating users into focussed communities [15]. The knowledge sharing in these communities allows the extraction of additional retrieval information exceeding pure document content. The application of community discovery to social networks has been studied in [14].

6 Conclusion and Future Work

In this paper, we have described a new approach to collaborative IR by integrating positional data. The consideration of these data enriches the community building process by the aspect of location-awareness. The location-specific communities in turn may be employed to enhance the quality of the entire retrieval process. In order to achieve a timely processing of high frequently changing positional data, update propagation techniques have been proposed for an incremental data stream analysis. Aside from the extensively discussed stream of positional data, various other streams can be identified in our general framework. These include further dynamic user profile data as well as user queries and user feedback. Update propagation techniques are generic, that is, they can be employed for the incremental analysis of those data streams as well. To which extent update propagation allows for enhancing their timely processing and analysis will be an interesting future research subject.

References

1. S. Acharya, P. B. Gibbons, and V. Poosala. Congressional samples for approximate answering of group-by queries. In *SIGMOD Conference*, pages 487–498, 2000.
2. E. Agichtein, E. Brill, S. Dumais, and R. Ragno. Learning user interaction models for predicting web search result preferences. In *SIGIR conference*, pages 3–10, New York, NY, USA, 2006. ACM Press.
3. R. B. Almeida and V. A. F. Almeida. A community-aware search engine. In *WWW conference*, pages 413–421. ACM Press, 2004.
4. E. Amitay, N. Har’El, R. Sivan, and A. Soffer. Web-a-where: geotagging web content. In *SIGIR conference*, pages 273–280, 2004.
5. B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In *PODS*, pages 1–16, 2002.
6. M. Balabanovic and Y. Shoham. Content-based, collaborative recommendation. *Commun. ACM*, 40(3):66–72, 1997.
7. A. Behrend, C. Dorau, and R. Manthey. TinTO: A tool for the view-based analysis of streams of stock market data. In *DASFAA*, 2007.
8. E. Bradner and G. Mark. Why distance matters: effects on cooperation, persuasion and deception. In *CSCW conference*, pages 226–235, 2002.
9. U. Brandes, M. Gaertler, and D. Wagner. Experiments on graph clustering algorithms. In G. D. Battista and U. Zwick, editors, *ESA*, volume 2832 of *Lecture Notes in Computer Science*, pages 568–579. Springer, 2003.
10. P.-A. Chirita, C. S. Firan, and W. Nejdl. Summarizing local context to personalize global web search. In *CIKM conference*, pages 287–296, 2006.
11. P. Clough. Extracting metadata for spatially-aware information retrieval on the internet. In *GIR conference*, pages 25–30, New York, NY, USA, 2005. ACM Press.
12. N. Craswell, S. Robertson, H. Zaragoza, and M. Taylor. Relevance weighting for query independent evidence. In *SIGIR conference*, pages 416–423, New York, NY, USA, 2005. ACM Press.
13. Z. Dou, R. Song, and J.-R. Wen. A large-scale evaluation and analysis of personalized search strategies. In *WWW ’07 conference*, pages 581–590, 2007.
14. M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99:7821, 2002.
15. M. Gnasa, S. Alda, J. Grigull, and A. B. Cremers. Towards virtual knowledge communities in peer-to-peer networks. In J. Callan, F. Crestani, and M. Sanderson, editors, *Distributed Multimedia Information Retrieval*, volume 2924 of *Lecture Notes in Computer Science*, pages 143–155. Springer, 2003.
16. A. Gupta and I. S. Mumick. Maintenance of materialized views: Problems, techniques, and applications. *IEEE Data Eng. Bull.*, 18(2):3–18, 1995.
17. Y. E. Ioannidis and V. Poosala. Histogram-based approximation of set-valued query-answers. In *VLDB*, pages 174–185, 1999.
18. C. Jones, R. Purves, A. Ruas, M. Sanderson, M. Sester, M. van Kreveld, and R. Weibel. Spatial information retrieval and geographical ontologies – an overview of the spirit project, 2002.
19. A. Kohrs and B. Merialdo. Clustering for collaborative filtering applications.
20. R. E. Kraut, C. Egidio, and J. Galegher. Patterns of contact and communication in scientific research collaboration. In *CSCW conference*, pages 1–12, 1988.
21. J. L. Leidner. Toponym resolution in text : ”which sheffield is it?”. In *SIGIR conference*, pages 602–602, New York, NY, USA, 2004. ACM Press.

22. J. Lu and J. Callan. User modeling for full-text federated search in peer-to-peer networks. In *SIGIR conference*, pages 332–339, New York, ACM Press, 2006.
23. C. Niedere, A. Stewart, B. Mehta, and M. Hemmje. A multi-dimensional, unified user model for cross-system personalization. In *Proceedings of the AVI 2004 Workshop On Environments For Personalized Information Access*, 2004.
24. S. E. Robertson. Understanding inverse document frequency: on theoretical arguments for idf . *Journal of Documentation*, 60:503–520, 2004.
25. S. E. Robertson, S. Walker, M. Hancock-Beaulieu, A. Gull, and M. Lau. Okapi at TREC-3. In *Text REtrieval Conference*, pages 21–30, 1992.
26. S. E. Robertson, H. Zaragoza, and M. Taylor. Microsoft cambridge at trec-13: Web and hard track. In *TREC*, 2003.
27. X. Shen, B. Tan, and C. Zhai. Context-sensitive information retrieval using implicit feedback. In *SIGIR conference*, pages 43–50, New York, ACM Press, 2005.
28. K. Sparck-Jones, S. Walker, and S. E. Robertson. A probabilistic model of information retrieval: development and comparative experiments - part 2. *Information Processing and Management*, 36(6):809–840, 2000.
29. E. Spertus, M. Sahami, and O. Buyukkokten. Evaluating similarity measures: a large-scale study in the orkut social network. In *KDD conference*, pages 678–684. ACM Press, 2005.
30. J. Teevan, S. T. Dumais, and E. Horvitz. Personalizing search via automated analysis of interests and activities. In *SIGIR conference*, pages 449–456, New York, NY, USA, 2005. ACM Press.
31. D. B. Terry, D. Goldberg, D. Nichols, and B. M. Oki. Continuous queries over append-only databases. In *SIGMOD Conference*, pages 321–330, 1992.
32. G.-R. Xue, C. Lin, Q. Yang, W. Xi, H.-J. Zeng, Y. Yu, and Z. Chen. Scalable collaborative filtering using cluster-based smoothing. In *SIGIR conference*, pages 449–456, 2005.