CaKuTa: Combining Information Value and Temporal Context in a Graph Based Recommender

Mehmet Gençer
Istanbul Bilgi University, Department of Computer Science
Kurtuluş Deresi Caddesi No: 47 Dolapdere, 34440 Beyoğlu
Istanbul, Turkey
mgencer@cs.bilgi.edu.tr

ABSTRACT

We present a recommender system which uses a graph based representation to relate items in a system with heterogeneous attributes. In assessing strength of relations between items we combine rarity of common properties with collaborative data on item popularity, with optional inclusion of a random term to introduce diversity in recommendations. For incorporating user preferences into recommendation, our system allows tuning of persistence level while maintaining temporal user context, hence providing a way to prefer or combine ephemeral or persistent preferences of the user.

Keywords

Contextual Recommendation, Information Value

1. INTRODUCTION

In this paper we describe an experimental recommender system, named CaKuTa, which is targeted for systems where descriptive data about item properties is available. Its recommendation algorithm is based on a graph based similarity representation.

CaKuTa is a hybrid system which uses both collaborative and content based information, but has several distinguishing design goals. First, the recommender system is intended to be sensitive to ephemeral context of users' navigation in a system, in addition to using their long term and persistent interests. Most collaborative recommender systems based on user profile similarity are severely limited when user is looking for something which is not hinted by their past activity. This is particularly relevant for new users which perhaps should matter most to e-commerce sites. For new users past activity is limited and incomplete, hence can be misleading. Methods weighing towards collaborative recommendation algorithms have the risk of over classifying users, which may result in poor recommendation quality for new users or in cases where users are exploring a new interest previously unknown to the system. Instead, our design targets to allow maintaining both ephemeral and persistent user interests.

RecSys 2010 Barcelona, Spain

The second design goal of CaKuTa is to minimize both the requirements for its integration into and information required from the underlying system for it to do its work. For this reason, we let items in a system be of heterogeneous type and hence have incompatible attribute sets. In assessing similarity of items and maintaining user interests, we rely on commonality of item properties, and also use their rarity as a proxy of information value of properties in terms of such assessment.

The third design goal is to allow the recommender system to be tuned for different deployment contexts since features of the information in the system, such as scarcity and skew of ratings, amount of attributes, etc., may differ greatly [5]. For this reason, our recommendation algorithm is formulated in a parametric manner, an we discuss how different combinations are possible to let the recommendation system lean towards collaborative or per-user data, or change persistence level in user context maintenance, in order to suit different deployment contexts or recommendation strategies.

Finally, the recommender system is expected to be exploratory, in addition to being consistent, and for this reason introduce some random diversity in its search. It should lead its user to discovery of new items, while at the same time constraining its search with respect to user preferences.

In the following sections we first describe the model for representing item similarities and user preferences, then describe our recommendation algorithm which aims to fulfill the goals described above.

2. DATA MODEL AND ALGORITHM

In order to meet its goals, our design relies on a graph based representation of similarities between items and user preferences, which was partially introduced in a previous work [7]. Given the set of items, $\mathbf{I}$, and users, $\mathbf{U}$, CaKuTa recommendation algorithm makes an assessment of the utility of an item, $j \in \mathbf{U}$, for a user $u \in \mathbf{U}$, given that the user is currently viewing some item $i \in \mathbf{U}$. For modeling the data required by the algorithm, we use the definition of *objects* which are associated with one or more items as item properties. Using this data, similarity between items are assessed using commonality of their associations with objects. User preferences are also modeled as level of interest in various objects, deduced from user's past interest in items which are in turn associated with objects. Finally, item similarity and user preferences are combined in the recommendation algorithm.

2.1 Item Properties and Item-to-Item Relations

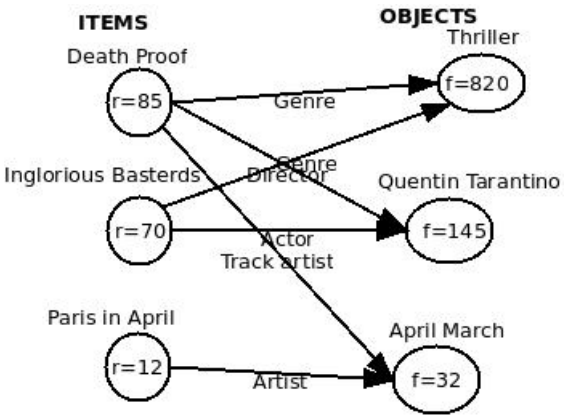

Figure 1: An example of item properties

Properties of items are modeled as a bipartite binary graph which represents the two mode network of relations between items in **I** (e.g. films) and objects (e.g. people) in a set denoted with **O**. With these definitions, properties of an item, i, will be expressed as a set of associations with a subset of objects, denoted as A_i:

$$A_i = \{(o, r_{i \to o}) : o \in \mathbf{O}, r_{i \to o} \text{is a relation label}\}$$

The graph is a binary (i.e. unweighted) one since the associations are considered to be equal value. Attachment of different importance to properties would be problematic to do so on an objective basis.

As an example consider the movie Death Proof. The properties of this item would be the set:

$$(QuentinTarantino, Director)$$
$$(Thriller, Genre)$$
$$(AprilMarch, SoundTrackArtist)$$
$$\ldots$$

While the relation labels are useful for search or presentation purposes, they are of little importance for our recommendation algorithm. For this reason, from this point on we will let A_i denote simply the set of objects with which the item i is associated with:

$$A_i = \{o : o \in \mathbf{O}, i \text{ is associated with } o\}$$

A visualization of such relations in a system is shown in Figure 1. The figure also indicates some fictional ratings of items, and frequencies of objects. The frequency, f_o, of an object, o, is simply the number of items associated with the object; or more formally, in-degree of o in the bipartite graph (or equivalently, the number of sets A_i which contain o). Concerning the item ratings, the fact that we use a simple scalar rating needs explanation. Recommender systems often rely on elaborate user ratings. However such ratings may be scarce [5], and when available may unevenly represent user feedback if a relatively small proportion of users provide most of the ratings. For this reason, we envision using the count of visits of an item as its single dimensional and scalar implicit rating value, denoted as r_i for item i; although replacing or combining such rating with explicit ratings by users is possible. However, this choice of rating assessment simplifies integration of a recommender system with the host system it is deployed on.

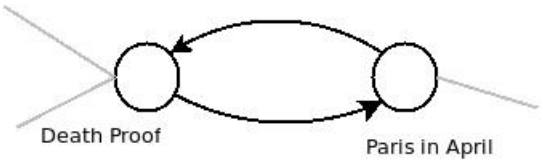

Figure 2: Example portion of item-to-item graph

Please note that there is a path in the example graph in Figure 1 connecting an album (Paris in April) to a movie via the artist who recorded both the album and also a soundtrack which is used in the movie. For this reason, it is possible to establish a relationship across items from different cultural domains (music and film), and in turn to make cross recommendations. This prospect was the reason that we have chosen the name CaKuTa (an abbreviation of cross cultural recommendation, in Turkish).

The bipartite graph representing relationships between items and objects allow us to make a transformation to generate a graph of similarity relations between items, as shown in Figure 2. While the item-to-properties graph was a binary one, item-to-item graph is weighted. In item-to-item graph, an item has directed edges to items with which it has at least one common object association.

CaKuTa design uses the insight that properties that are less frequent are more informative of a user's intentions. For this reason the weight of a relation in the item-to-item graph, from an item i to another item j should be reversely proportional to frequencies of their common properties, $A_i \cap A_j$. On the other hand, following the common sense thinking in collaborative recommendation, an edge in the item-to-item graph should have a weight which is proportional to the rating of its destination vertex. For this reason we calculate the weight of directed relation, $w(i, j)$, as:

$$w(i,j) = c \cdot {r_j}^x \left(\sum_{o \in (A_i \cap A_j)} \frac{1}{{f_o}^y} \right) + d \cdot \varepsilon \qquad (1)$$

Where x, y, c, and d are some non-negative constants, and ε is a uniformly distributed random variable, $0 \leq \varepsilon \leq 1$. These constants control the type of relational attachment provided by the weight calculation. For example, the choice of $d = 0$ and positive c, x and y, will result in a relational assessment which adheres to the description above since the result is proportional to rating of the destination item but reversely proportional to frequency of objects, i.e. respects information value. Considering the example shown in Figure 1, it is possible, depending on the choice of x and y, that the music album Paris in April is assigned as the largest weight edge from the source item, the movie Death Proof, in the item-to-item graph shown in Figure 2, although other items have more than one property in common with the source. This is simply because the artist April March is a rare property compared to others (its frequency value is 32, compared to 145 and 820 of other properties in the example), hence having a greater multiplier $\frac{1}{f_o^y}$, although the music album has a lower rating.

The readers will notice that the first term in Equation 1 is not normalized, hence making the choice for constants difficult. For this reason it may be useful to break down the constant c so that to explicitly include some sort of a

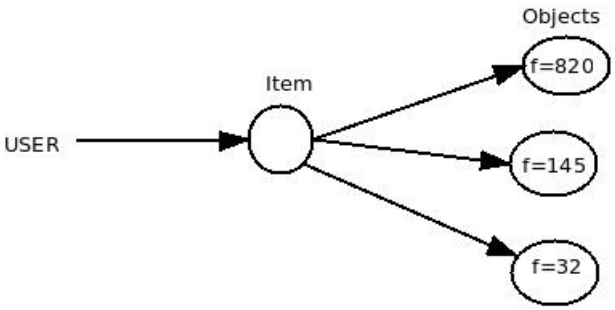

Figure 3: User to property relations

normalization term[1].

Different choice of the constants will result in various types of establishing item-to-item relations. For example using $x < y$ in the above example will mean that similarity of properties are more valued than item ratings. Using a positive d value in Equation 1, on the other hand, will introduce a certain level of randomness in item-to-item relation strength. The intention here is using a relatively small random component which improves exploratory capabilities while still respecting item similarity, hence improving diversity of recommendations.

Normalization with respect to frequency of keywords is a common technique in recommender systems [2]. However, employment of such normalization in a graph based method as ours results in a natural combination of item attributes and their rarity with ratings to determine local relevance of items.

2.2 User Preferences

Here we describe how a user's level of interest in objects is induced from user's purposeful browsing of items (but can be combined or replaced with explicit ratings or profile information). An example is visualized in Figure 3. For a user, u, the level of interest of user in an object, o, will be denoted as $p(u, o, t)$, where t denotes the time in terms of sequence index of user's visits to items. In updating user interests, given that the item browsed by the user at time t is i_t, we compute interest levels as an exponential moving average:

$$p(u,o,t) = \begin{cases} (1-\alpha)\cdot p(u,o,t-1) + \alpha\frac{\overline{f}}{f_o} & \text{if } o \in A_{i_t}, \\ (1-\alpha)\cdot p(u,o,t-1) & \text{otherwise.} \end{cases} \quad (2)$$

where α is the smoothing factor, $0 \leq \alpha \leq 1$, and $\overline{f}$ is the mean of object frequencies:

$$\overline{f} = \frac{\sum_{o\in\mathbf{O}} f_o}{|\mathbf{O}|}$$

This normalization with $\overline{f}$ in Equation 2 together with having f_o as denominator serves to incorporate the information value of properties where rare properties are assigned higher interest levels, similar to the case for Equation 1. Please note that the definition of time variable, t, in the above description simply refers to sequence number of actions of the particular user, and not intended to be used as an absolute time information for any computation across multiple users.

Larger values of α in Equation 2, close to 1, results in fast forgetting a user's shown interest in an object, and provides an ephemeral assessment of user preferences. Small values, on the other hand, will result in a smooth average, hence will provide persistent user preferences. The choice of smoothing factor is a matter of strategy, rather than that of necessity. Indeed, it is possible, and perhaps advisable, to maintain both types of user preferences, and producing recommendations as a mix of the results or applying a recommendation algorithm to each.

An interesting question concerning Equation 2 is what to use as the seed value of user interests, i.e. $p(u, o, 0)$? This is once again a matter of recommendation strategy. While a zero value seems to be the sensible choice for computing ephemeral preferences, a unit value for persistent preferences may well serve incremental refinement of user preferences discovery; i.e. it initially assumes that the user is interested in all objects equally, rather than being interested in none.

2.3 Combining Item Similarity with User Preferences in Recommendation

With the definitions so far, we can proceed for assessment of the utility score of an item, j, for a user, u, at time t, given that user is currently viewing an item i_t. In order to incorporate temporal user preferences (whether ephemeral or persistent) we must modify the definition in in Equation 1 as follows:

$$u(i_t, j, u, t) = c \cdot {r_j}^x \left(\sum_{o\in(A_{i_t}\cap A_j)} \frac{p(u,o,t)}{{f_o}^y} \right) + d\cdot\varepsilon \quad (3)$$

Intuitively, the equation above assigns non-zero scores to all items that have at least one common attribute with the item currently viewed by the user. In doing so, it assigns higher scores to more popular items, to those that are linked through rare attributes, and in the mean time using a multiplier for each attribute that reflects the user's past interest in the attribute. In this score calculation, the smoothing factor used for maintaining user interest (the constant α in Equation 2) determines how ephemeral or persistent is the temporal context used in this scoring. Furthermore the final term is used to introduce randomness in exploration of the regions of the item-to-item graph that are around the current item. The combined choice of constants c and d determine the balance of consistency and exploratory elements in scoring, whereas the combined choice of constants x and y determine the relative impacts of item popularity (collaborative rating) and attribute rarity (information value).

With this definition in hand, a single best recommendation for the user u at time t, $r(u, t)$, can be obtained by choosing the item which has the maximum score:

$$r(u,t) = \max_{j\in\mathbf{I}} u(i_t, j, u, t)$$

On the other hand, most recommender systems are expected to select multiple recommendations. A trivial option in this case would be to choose elements with largest scores; e.g. if n recommendations are requested, such a set, $R(u, t, n)$, could be selected as:

$$R(u,t,n) = \max_{j\in\mathbf{I}}{}^n u(i_t, j, u, t)$$

[1]For example as $c = c_1 \cdot \frac{\overline{|A_i\cap A_j|}\cdot\overline{f}^y}{\overline{r}^x}$, where $\overline{|A_i \cap A_j|}$ is the mean size of the sets of common properties, and $\overline{f}$ and $\overline{r}$ are the mean rating and frequency in the dataset.

However, a more pragmatic option, in our view, would be to combine results of various scorings where some of these scorings use ephemeral user context while others use persistent user context, some introduce exploration while others stick to consistency, etc. Assuming that n recommendations are requested and there are m available utility scorings, labeled as $S_u = u_1, u_2, \ldots, u_m$, the set $R(u, t, n)$ can be constructed as:

$$R(u,t,n) = \bigcup_{u_k \in S_u} \max_{j \in \mathbf{I}}^{n/m} u_k(i_t, j, u, t) \qquad (4)$$

There is an apparent problem in this method when n is not divisible by m, and the problem gets worse when the variety of scorings, m, becomes larger. We do not address this problem here, instead we note that it is possible to make posterior assessment of different scorings on a per user basis, and assign different relevance values to each scoring so that the set of recommendations are constructed as a non-uniform combination with respect to these relevance values instead of the simply put uniform combination of maxima in Equation 4. Nevertheless, the method of combination largely depends on the content of the system and usage patterns, hence it is a strategic matter as well as being a technical one.

The graph based representation in CaKuTa is also suitable for suggesting recommendation algorithms that use a search in the graph, rather than considering direct links from the source item as we have done throughout our articulation so far. We do not offer a treatment of such methods here, however, since combining edge weights in indirect paths in the item-to-item graph presents interpretation problems and requires a different methodology.

3. STRENGTHS AND LIMITATIONS

While early studies in recommender systems focused on consistency of recommendations with the user preferences [2, 3], recently the emphasis is on the need for making recommender systems think out of the box and provide novel, rather than over-specialized, recommendations[1], hence balancing accuracy with diversity of recommendations [8]. This is in fact a return to the original vision of the field that unlike search engines, recommender systems should help users discover new items [4, 6]. Incorporation of information value by using frequency of item properties in our method provides a way to introduce novelty in recommendations. Furthermore, our parametric formulation and introduction of a random term allows one to tune the accuracy-diversity balance of the recommender system.

On the other hand, explicit treatment of ephemeral and persistent user interests is rarely addressed in recommender systems. Our parametric formulation of user interests allow one to tune the recommendation method to use either type of temporal user contexts, or combine them as desired. For the sake of wider applicability, we have preferred to formulate user interest maintenance as based on counting purposeful browsing of items rather than explicit ratings; however, the latter can easily be incorporated into CaKuTa recommender.

An important advantage of the similarity formulation in CaKuTa is that it does not require a standard set of attributes to assess item similarity. While this makes treatment of novelty in recommendations difficult, for example as opposed to the work of Abbassi et al [1], it renders our system to be suitable for deployment in a wider range of systems, including those that contain more than one type of item such as a mix of music, books, films, etc. Graph based representation of both item similarity and user preferences is amenable to application on heterogeneous item collections that are otherwise resistant to item or profile similarity assessment methods (e.g. clustering or Euclidean distance based similarity metrics) which require availability of standard attribute sets. Furthermore, this flexibility is accompanied by the fact that CaKuTa can present the reasons for its recommendations to users easily, due to incorporation of item-to-property relation labels.

Since implementation of CaKuTa recommender is yet incomplete, we have not conducted extensive experiments. Furthermore, our articulation of the recommender algorithm is short of suggesting rigorous methods for choosing parameters that effect how it behaves. Different systems require different features depending on the properties of their data sets [5]. Hence different experiment sets needs to be designed to provide a tuning method for applying CaKuTa recommender. On the other hand our preliminary performance tests indicate that the recommender method have reasonable computational performance for medium scale book and music collections.

4. REFERENCES

[1] Z. Abbassi, S. A. Yahia, L. V. S. Lakshmanan, S. Vassilvitskii, and C. Yu. Getting recommender systems to think outside the box. In *RecSys '09: Proceedings of the third ACM conference on Recommender systems*, pages 285–288, New York, NY, USA, 2009. ACM.

[2] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6):734–749, 2005.

[3] A. Ansari, S. Essegaier, and R. Kohli. Internet recommendation systems. *Journal of Marketing Research*, 37(3):363–375, August 2000.

[4] R. Burke. Hybrid recommender systems: Survey and experiments. pages 331–370, November 2002.

[5] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, January 2004.

[6] J. B. Schafer, J. Konstan, and J. Riedi. Recommender systems in e-commerce. In *EC '99: Proceedings of the 1st ACM conference on Electronic commerce*, pages 158–166, New York, NY, USA, 1999. ACM.

[7] E. Sevinç, R. E. Başar, and B. Puhaloğlu. Common lisp for a common cultural recommendation system. International Lisp Conference, 2009.

[8] T. Zhou, Z. Kuscsik, J.-G. Liu, M. Medo, J. R. Wakeling, and Y.-C. Zhang. Solving the apparent diversity-accuracy dilemma of recommender systems. *Proceedings of the National Academy of Sciences*, 107(10):4511–4515, March 2010.