Modeling and Exploiting Goal and Action Associations for Recommendations

Dimitra Papadimitriou
University Of Trento
Trento, Italy
papadimitriou@disi.unitn.it

Yannis Velegrakis
University Of Trento
Trento, Italy
velgias@disi.unitn.eu

Georgia Koutrika
Athena Research Center
Athens, Greece
georgia@imis.athena-innovation.gr

ABSTRACT

Recommender systems are used to identify those items in a large collection that are more likely to be of interest to a user. A common principle of most recommenders is that whatever happened in the past is a good indicator of the future. We offer a different perspective. Considering the fact that in real life users do their selections with certain goals in mind, we recommend items (or actions) that help users fulfilling their intended goals using their past only as a way of identifying goals of interest. We introduce a model that connects goals and actions through action sets implementing the respective goals. Such a model captures latent associations among goals and actions and allows the ranking of actions considering different user strategies such as to complete at least one goal with the minimum effort (i.e., minimum number of actions), or to open up more paths for fulfillment of more goals in the future. For each strategy we recommend an algorithm that exploits the user action and goal spaces to rank the actions in a different way. We have performed extensive experimental studies to understand how these techniques are related and compare the results against traditional recommendation methods. The experiments illustrate that it is not possible to replicate the results of our approach using existing techniques.

1 INTRODUCTION

People are daily facing situations in which they have to make choices from large collections of items. Selecting the best answer to a search engine query among those satisfying the query conditions, selecting a movie to watch, an item to purchase, or friend activities to read about in social media, are only some of the most characteristic examples. Recommender systems [3, 7, 12, 16, 20] give advice to users on items that are likely of interest to them. There are two main categories of recommender systems. The first is the collaborative filtering, which is based on the idea that similar users have similar preferences, thus, the analysis of the choices of similar users can result in successful recommendations of items that have not been selected yet. The second category is the content-based which is based on the idea that users would like items that have similar features with items they have liked in the past. The principle behind both approaches is that whatever the past indicated as preference, it is likely to be preferred also in the future.

In this work, we approach the problem based on a different principle. There have been studies in psychology and social sciences [4] that have shown that human actions are not random and unrelated events. They may be of course affected by preferences but they are mainly results of rational selections performed with the purpose of achieving some specific goal that a person has set and aims to fulfill [1].

Based on these studies, we advocate that by recognizing the goals for which actions of the past have been performed, it is possible to identify the driving forces of the users' future actions and make recommendations that better fit these needs. Since the fulfillment of a specific goal may require actions that are highly different in nature, this form of recommendation may recommend actions that are highly different from those of the past, or from those that similar users have done in the past. Note that we may use the term "actions" and not "items" as typically done in recommender systems; with this option we are being more generic since the selection of an item, the purchase of a product, or the watching of a movie are practically all actions.

Existing studies in recommender systems have already recognized that methods taking into account similarity with what has happened in the past are not always matching user expectations and have tried different techniques that focus on other aspects such as serendipity, novelty and diversity to improve the quality of recommendations [9]. However, these solutions are not principled and are not driven by some specific, user-selected, well-defined target while in many recommendation scenarios there exist targets that users are willing to reach. For instance, in online learning platforms, users may target at specializations or/and degrees. In employment-oriented social networking services such as LinkedIn users are encouraged to take actions that will lead them into their next position. In addition, they can see how some actions can lead to the same target following different career paths. Moreover, users may perform actions that will lead them to the fulfillment of commercial goals such as to get discount coupons, or everyday goals such as to become fit or to cook.

Consider, for instance, the case of a customer in a supermarket that has placed in the cart a kilo of potatoes and carrots. A content-based recommendation will try to propose products that are close to what is already in the cart, i.e., similar to potatoes and carrots which means it may propose other kinds of vegetables, or even suggest other types of potatoes. On the other hand, a collaborative filtering system may suggest light beer or red peppers, because these items have been bought in the past by customers with similar preferences. Both methods, through clearly different routes, recommend items based on the customer's past. Instead, by taking into account that the items in the customer's cart can be combined with other items to produce one or more food recipes, the system can open up new options to the customer. For instance, considering a recipe to make an olivier (russian) salad that includes: potatoes, carrots and pickles, an item to be recommended would be pickles. Another useful ingredient would be nutmeg that is a spice used for mashed potatoes and pan-fried carrots, two recipes that require products some of which are already in the customer's cart. Such a recipe-based recommendation of products may not be justified by similarity to products already in the cart, neither by other product combinations found frequently in the

carts of other customers. This means that neither association rules nor techniques that detect correlations among items can be employed to make such recommendations since they highly depend on the popularity of these item sets. So, unless we consider the product combinations found in the recipes, these products will not be recommended by other techniques. Furthermore, given the recipes, the recommendations can be optimized for an overall benefit. For example, recommended products may give the ability to the customers to maximize the number of recipes that they can materialize.

Considering goals in the recommendation problem is challenging. The challenge comes from the fact that, in real life, there are typically multiple goals that one needs to fulfill at any given time. Each of these goals may require fewer or more actions in order to be fulfilled, and there may exist alternative ways for the fulfillment of a specific goal. Users have to reason on the priorities between the goals they try to achieve and the benefit they will have by the execution of each action towards the fulfillment of these goals. For instance, some users may prefer actions that help them fulfill a goal as soon as possible, while others may prefer actions that help the advancement of as many goals as possible. A goal-oriented recommender will have to leverage the goals by first recognizing the intended user goals, decide the priorities among them, and quantify the benefit of each action in relationship to the intended goals and in conjunction with the other possible actions.

We introduce a new family of recommendation strategies, i.e., *goal-based recommendations*, that deal with the above challenges. The goal-based strategies identify the goals for which exists evidence that the user is aiming at achieving. The evidence originates from the previous user activity, i.e., the actions that the user has already performed. Given this goal space, the strategies explore the sets of actions that lead to the fulfillment of these goals and contain actions that the user has already performed to find actions which the user has not performed and may be willing to complete. The sets of actions together with the goals they fulfill constitute the user's *goal implementation space*. The likelihood that the users will like an action from the candidate set of actions in this space depends on their approach towards the goals they would like to fulfill. We have identified three different strategies for exploring and exploiting the user's spaces in order to select the actions to be recommended. The three strategies correspond to three different policies based on which users often make their selections.

The first strategy is the *Focus* that examines each of the action sets in the user's goal implementation set to find which of them lead to the fulfillment of the goal that is closest to completion, either because most of the required actions have been already performed ($Focus_{cmp}$), or because they require only a few more actions ($Focus_{cl}$). Then, it forms the recommendation lists from the actions in these action sets. It is the policy preferred by users that need to fulfill at least one goal through the actions in the current recommendation list. The second strategy, *Breadth*, is not examining each action set in the user's goal implementation space separately. It considers more than one set of actions at the same time. Specifically, it evaluates and ranks the actions in the user's action space based on all the sets this action participates and selects those actions that belong in as many sets as possible together with as many as possible actions from the user activity. This strategy is for users that would like to fulfill as many goals as possible, if possible, through this recommendation list, but in order to maximize the number of fulfilled goals, they are willing to complete some or all of them in the future, i.e., not only through the actions in the current recommendation list. This way it keeps some "paths" open for the future (i.e., unfulfilled goals) but those paths contain the minimum number of additional actions. We also suggest a third strategy, the *Best Match*, that similarly to *Breadth* is not trying to fulfill at least one goal through the current recommendation list. It recommends actions that contribute to the goals of the user's goal space. However, in contrast to *Breadth*, *Best Match* evaluates an action considering the whole goal space, not only the goals to which this specific action contributes. It generates a profile for the user and estimates a similarity between this profile and the actions to be recommended. The action representation shows how much that action contributes to the fulfillment of the various goals and the user profile how many of the user actions contribute to the various goals. It is a policy that may end up in the fulfillment of many goals in the future. However, it is a strategy for users that are interested in actions that are more useful (contribute more) to the goals to which the user has has put more effort in the past (and respectively less to goals to which the user has put less effort).
Our contributions can be summarized as follows:

- We introduce and formally define the notion of goal-oriented recommendation, which evaluates every action considering the goals which the current user may be willing to fulfill and how that action contributes to the fulfillment of one or more of these goals together with other actions of the user (Section 3).
- We explain how it differs from existing techniques and why the latter cannot be used to offer this type of recommendation (Section 2).
- We present different strategies for ranking the candidate actions, with each strategy implementing a different policy in prioritizing the goals and selecting the actions to be recommended (Section 5).
- We describe efficient ways of implementing the above strategies and materializing the goal oriented recommendation paradigm (Section 4).
- We study the effectiveness of our methods and compare them to the state-of-the-art recommendation approaches. We show that goal-based approaches can recommend actions that bring the user closer to the fulfillment of goals that are related to her/him, are highly different from each other and at the same time from actions performed by other users in the past (Section 6).

2 RELATED WORK

Goal modeling. Goal modeling has attracted a lot of research interest for decades. However, the focus of the different fields has been in goal and next action inference [13] such as prediction of the next action in a sequence, e.g., the next web page to click or the next location to be [11, 18]. The purpose of such systems is for instance to promote the inferred actions or act in anticipation of the user's actions [2]. To infer the next action(s) they employ models such as probabilistic (state transition) models, e.g., Bayesian Networks [15], or Markov models [18] or other variations [2].

Recommender Systems. Our method retrieves actions to be recommended but does not consider neither the user's neighborhood activity nor the activity of the current user as in state-of-the-art recommendation approaches but the actions in the implementations of the various goals. In contrast to *Collaborative Filtering* [7, 8] that exploits previous item selections or interactions that similar users have performed, it selects implementations that contain subsets of the user activity and can be adequately extended to lead to

the fulfillment of one or more goals. It also differs from *Content-based Filtering* [3] that recommends items similar to what the user has used in the past with a high degree of satisfaction.

Association rule mining. Association rule mining analyzes the user's histories to identify groups of items appearing together and use this as the basis for making recommendation [19]. The approach is based on popularity, while our technique is not affected by popularity fluctuations. Furthermore, different actions may often appear together but for different goals, which means not only that recommendations different than ours will be made, but these recommendations will also be incomplete, i.e., they will manage to fulfill none of the goals, since the system is confused and unable to distinguish the different intended goals of the actions that appear together.

3 GOAL-BASED RECOMMENDATIONS

Actions, Goals and Goal Implementations. We assume the existence of a set $\mathcal{U}$ of *users*. Users perform actions such as the purchase of an item, the visit of a web page, the watching of a movie, or any other recordable task. We consider the existence of a set $\mathcal{A}$ of *actions*.

People set the goals that they need to achieve and then they decide to perform those actions that they believe will help them fulfill their goals. We denote $\mathcal{G}$ the set of goals. A set of actions constitutes an *activity*, which means that there are $2^{\mathcal{A}}$ different possible activities. The activities that are intended for a goal $g \in \mathcal{G}$, alongside the respective goal, are referred to as *goal implementations*.

Definition 3.1. A *goal implementation*, or simply implementation, is a pair $\langle g, A\rangle$ with $A \in 2^{\mathcal{A}}$ and $g \in \mathcal{G}$.

Goal Implementation Data sources. Goal implementations can be found in sources related to almost every aspect of human activity. Recipes, for instance, are implementations of specific goals (the food that the recipe is about). Online learning platforms have specializations and degree that are implemented through courses. Each specialization is associated with one or more set of courses indicating the actions required to achieve the goal, i.e., the specialization. Goals can also be found in online stores. Many online clothing stores, for instance, give users the ability to form *outfits* and annotate them with labels such as 'for friend meetings", "to be warm" and so forth. Those outfits constitute implementations of the goal, with the goal being the label. With this knowledge, when the system recommends some item to a user apart from considering the user preferences on characteristics such as color or material (content based filtering) or considering what clothes others have bought in the past (collaborative filtering), it can employ a goal-based recommendation technique and suggest items that can be combined with clothes the user has bought in the past to form complete outfits.

Another rich source of goal implementations are social networks or specialized web sites where users record and share success stories of things that they do in life. Examples include 43Things (*https://43things.com*) and wikihow (*www.wikihow.com*), where users describe actions to achieve real-life goals. There are many works on transforming such textual descriptions into a structured form, like an ontology [10, 14, 17], or a taxonomy [6, 21]. They typically employ structural information such as HTML tags or enumeration. A different way to create such datasets from web pages is by posing queries of the form "in order to + *a goal description*" on search engines [21].

Figure 1: Combinations and the goals they serve

One of the datasets that we are using in the experimental evaluation of this work contains 18k goal implementations that we have extracted by performing *action identification* on user-generated descriptions about everyday goals such as *learn english*, *travel to Italy* and so forth from the 43Things website. We did this action extraction with a module that we have developed for this purpose, that works on a simpler model and for plain text (ref. Section 4). We do not elaborate further on the extraction task since it is orthogonal to the focus of the current paper and a different line of work of ours.

Example 3.2. Figure 1 depicts a set of goal implementations from an online clothing store. We denote a goal implementation set as L. The columns indicate outfit purposes (the goals) while the rows are the items (the actions). If we depict by a_k the action of buying the item i_k, then the implementation set is:

Implementation	⟨Goal,Activity⟩	
$p1$	$\langle g_1, A_1\rangle$	where $A_1=\{a_1, a_2, a_3\}$
$p2$	$\langle g_2, A_2\rangle$	where $A_2=\{a_1, a_2, a_4\}$
$p3$	$\langle g_3, A_3\rangle$	where $A_3=\{a_1, a_4, a_5\}$
$p4$	$\langle g_3, A_4\rangle$	where $A_4=\{a_3, a_5, a_6\}$
$p5$	$\langle g_3, A_5\rangle$	where $A_5=\{a_1, a_3, a_5, a_6\}$

Recommendation Setting We assume an implementation set L. The set may have been constructed through one of the many methods mentioned earlier that are already available in the literature, or through the text-based module we developed for the experimental evaluation.

The actions that the user has already performed is referred to as the *user activity* H. We do not know why these actions have been performed but given the goal implementation set. there is a number of possible goals that the user may have had in mind when performing each of these actions. These goals constitute the *goal space* of the activity H.

The goal space makes all the actions that contribute to one or more goals in the goal space to be likely of interest to the user. Our aim is to recommend to the user actions that are not in H, and which the user would be happy to perform. However, not all the actions offer the same benefit. What action the user would be more willing to perform depends on what priorities the user puts on the goals. Some actions may help towards the fulfillment of many goals, while others towards the fulfillment of goals almost completed. Thus, we need to create a ranked list of the actions to recommend according to some criterion. Depending on the criterion/policy we use, a different recommendation strategy is materialized. These policies comprise the topic of the next section.

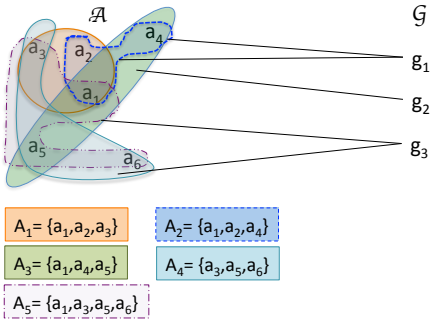

Figure 2: Illustration of our model.

4 GOAL MODEL

Our aim is to recommend to the users a set of actions considering the goals that they can fulfill given their activity. Those goals are associated with at least one action from the user activity, i.e., at least one action contributes to one or more of their implementations. An action $a \in A$ is said to *contribute* to a goal g through a goal implementation p, when there exists a goal implementation $p=\langle g, A\rangle$. and we denote it as $a \rightsquigarrow^p g$.

The set of goals that are associated with an action forms its *goal space*.

Definition 4.1. Given a goal implementation set L, the *goal space* of an action a is the set $\mathcal{GS}(a)=\{g \mid g \in \mathcal{G} \wedge p \in L \wedge a \rightsquigarrow^p g\}$.

The goal space extends naturally to the case where we have a set of actions A instead of one, to be the union of the goal spaces of the individual actions in A, i.e., $\mathcal{GS}(A)=\cup_{a \in A}\mathcal{GS}(a)$. Considering the individual goal spaces of each action $a \in H=\{a_1, .. , a_n\}$ there are two extreme cases: (i) $\mathcal{GS}(a_1) \cap \ldots \cap \mathcal{GS}(a_n) = \emptyset$ (i.e., there are no common goals) and (ii) $\mathcal{GS}(a_1) \cap \ldots \cap \mathcal{GS}(a_n) = \mathcal{GS}(H)$ (i.e., the goal spaces of all actions are the same), where we have no evidence whether one or more goals constitute a priority for the user. In all the other cases the goal spaces are a valuable source of information and should be exploited for retrieving the actions.

Another important factor that should be considered is that actions are not independent from each other since subsets of actions co-contribute to one or more goals. Therefore, given an action, there exist other actions that should be also performed in order for a goal in the goal space to be fulfilled. The set of these actions forms the action space of an action.

Definition 4.2. Given a goal implementation set L, the *action space* of an action a is the set $\mathcal{AS}(a)=\{a' \mid g \in \mathcal{G} \wedge p \in L \wedge a \rightsquigarrow^p g \wedge a' \rightsquigarrow^p g \wedge a \neq a'\}$.

The action space, similarly to the goal space, extends naturally to the case where we have a set of actions A instead of one, to be the union of the action spaces of the individual actions in A, i.e., $\mathcal{AS}(A)=\cup_{a \in A}\mathcal{AS}(a)$.

Example 4.3. In the implementation set of Example 3.2, since action a_1 participates in the activities A_1, A_2, A_3 and A_5, its implementation space is the $\mathcal{IS}(a_1)=\{p_1, p_2, p_3, p_5\}$, and its goal space the $\mathcal{GS}(a_1)= \{g_1, g_2, g_3, g_5\}$. Its action space is the set of all the other actions in A_1, A_2, A_3 and A_5, i.e., $\mathcal{AS}(a_1)=\{a_2, a_3, a_4, a_5, a_6\}$.

Thus, above we have determined two very important association types that correspond to two basic "operations" given an activity, i.e., a set of actions A: to form the *goal space* $\mathcal{GS}(A)$, and the *action space* $\mathcal{AS}(A)$. Moreover, we need a matching function connecting the goals with the action set(s) that implement them.

We suggest a model that sees each activity A in the L as a *hyperedge* that connects the actions that participate in it. Moreover, it labels each activity A with the goal that fulfills given a goal implementation $\langle g, A\rangle$. Figure 2 graphically illustrates our model that we call *association-based goal model*.

Given a small goal implementation set, we can, for instance, form the user goal space by visiting one by one all the implementations and check whether there exist any common actions in the user activity and their activity (i.e., the set intersection). However, when moving to hundreds or millions of implementations, the cost gets prohibitive. Therefore, we should implement our model in a way that apart from capturing all the associations, *allows us to form efficiently the goal and action spaces considering the interconnections among actions and goals through the goal implementations*.

In order to retrieve the information we need in real time, we employ a set of indexes. We first build an index *A-idx* for the action set and an index *G-idx* for the goal set. Keeping the information derived from the goal implementation set L needs a more complex structure. We refer to each goal implementation using a unique identifier id. We split the information of the goal implementation pairs in two indexes: *Goal Implementation ActiVity index* (*GI-AV-idx*) and *GI-G-idx* (*Goal Implementation Goal Index* (*GI-G-idx*). The first one matches the activity of a goal implementation to the id of the goal implementation where it belongs. We store a set with the ids of the actions. The second index matches each goal id to all the implementation ids that exist for the specific goal. Now we need to connect the goal implementations with the actions they contain. For this, we use *A-GI-idx* (*Action to Goal Implementation Index*) that retrieves all the goal implementation ids where an action contributes, i.e., the implementation ids (pIds) s.t., $a \rightsquigarrow^p g$.

Equations 1 and 2 describe how we exploit the above index structures to implement the basic operations that we described earlier, i.e., to form the goal and action space given an activity.

$$\mathcal{GS}(A)=\{GI\text{-}G\text{-}idx[pId] \mid a \in A \wedge aId=A\text{-}idx[a] \wedge pid=A\text{-}GI\text{-}idx[aId]\} \quad (1)$$

$$\mathcal{AS}(A)=A-\{GI\text{-}AV\text{-}idx[pId] \mid a \in A \wedge aId=A\text{-}idx[a] \wedge pid=A\text{-}GI\text{-}idx[aId]\} \quad (2)$$

5 STRATEGIES

Having built the *association-based goal model* described above, we can retrieve the actions to be recommended exploiting their associations with the actions in the user activity and the goals in the user goal space. In practice, we perform set operations to evaluate how strong the associations are. We suggest different strategies considering options with which we believe users prioritize actions. The first strategy examines the associations of the user activity and each of the sets of actions that contribute to goals from the user goal space. Examining each set of actions separately and not in conjuction with other action sets as well helps users to stay *focused* on one goal at a time (*Focus* strategy). On the other hand, the second strategy examines more than one action sets at the same time. For this reason we call it *Breadth*. Breadth gives priority to actions that are strongly associated with each other and the user activity; such actions can be exploited in the fulfillment of a subset of the user goal space at the same time. There is

also a third strategy that considers at the same time all the action sets that are associated with the user activity. In this case, the associated goals are an evidence of the user preferences on goals (goal-oriented profile). The strategy is referred to as *Best Match*. The three strategies that are called respectively *Focus*, *Breadth* and *Best Match* are described in the Subsections 5.1 and 5.2, 5.3 respectively.

5.1 Focus

Strategy *Focus* gives the user the option to have access to actions that lead to the *completion* of one of the goals in the user goal space. For an action set A in a goal implementation $\langle g, A\rangle$ where $g \in \mathcal{GS}(H)$, the intersection $\mid A \cap H \mid$ gives the number of actions in the implementation that have been already performed. Focus does consider the association of the user activity and the actions sets in the implementation set. But that is not enough to retrieve the actions that together with a subset of the user activity H form the activity of a goal implementation in the library L, i.e., they comprise the actions that are required for the goal to be completed. For this purpose we introduce two measures, goal implementation *completeness* and *closeness*, that evaluate and rank the candidate action sets and by extension the respective goal implementations. Completeness considers the proportion of the actions that are common in the user activity (set intersection) and the actions in the examined action set while closeness considers the common actions in comparison with the remaining actions.

$$completeness(\langle g, A\rangle, H) = |A \cap H|/|A|\rangle \tag{3}$$

$$closeness(\langle g, A\rangle, H) = 1/|A - H| \tag{4}$$

Goal implementation completeness is inspired by plan inference in plan libraries for intelligent agents [5]. However, in intelligent agents the aim is to predict which sequence of actions the agent is following (i.e., the agent has already selected a plan) while in our problem the recommendation mechanism aims to guide the user to options that s/he may have not considered without the recommendation system.

Algorithm 1 Focus Ranking

Ranks actions based on completeness (step 3) or alternatively closeness (step 3) of the corresponding goal implementations*
Input Set H, Set CA, int k
Output List R
1 $CI \leftarrow []$,$R \leftarrow \{\}$
//get the goal implementations that connect the goals in $\mathcal{GS}$(H) with the actions in H
2 *for all* $aId \in H$, *for each* pid ∈ A-GI-idx[aId]
3 $\mathcal{IS}$= { A-GI-idx[aId] | $a \in A \wedge$ aId=A-idx[a] }
4 *end for all*
5 *for each pId* in $\mathcal{IS}$
6 $\langle p, sc\rangle \leftarrow \langle\ pId,\ \frac{|A \cap H|}{|A|}\ \rangle$
6* $\langle p, sc\rangle \leftarrow \langle\ pId,\ \frac{1}{|A-H|}\ \rangle$
7 CI.add($\langle g, A\rangle$,$sc\rangle$)
8 *end for all*
9 rank CI based on sc
10 return the top k actions from the action sets of the top implementations (set CI)

We rank the goal implementations in the set of the implementations that are associated with the user activity in descending order of completeness. Given this ranking, the list of action recommendations is formed as follows. We pull from the first goal implementation all the actions that have not been performed yet, i.e., that are not in the user activity, and we add them to the recommendation list. If more actions are needed for the top-k list, we pull the next goal implementation and so forth, until the list gets full. Note that it may be the case that the remaining slots in the top-k list are fewer than the actions of the next goal implementation in the ranked list of implementations. In this case, we can decide to leave the list with fewer recommendations or expand k to include the required actions for this implementation.

The completeness ranking function promotes the actions in the activities of the goal implementations with the largest completeness (see Algorithm *Focus Ranking*, line 3). This way the recommendation mechanism guides the user to actions that will lead to the fulfillment of the goal for which the user has already done most of the work, i.e., she has performed most of the actions needed for its fulfillment.

5.2 Breadth

With the strategy Focus, a single goal drives the recommendation process. This can be very restrictive if the user is not that determined to fulfill the specific goal. Therefore, we give the user another option: *Breadth* that evaluates every candidate action a considering a subset of goals in the goal space. This subset consists of the goals that are connected to the candidate action a through one or more goal implementations, i.e., the goal space $\mathcal{GS}(a)$. The reasoning behind this is that since every action in the action set $\mathcal{A}$ can participate at the same time in more than one goal implementations in the set L and possibly contribute to a number of goals, its benefit should be estimated based on all these goals.

First, in order to evaluate a candidate action a, we should take into consideration the number of goal implementations in its implementation space, i.e., the $\mathcal{IS}(a)$. We will refer to this quantity as *utility*.

$$u(a) = |\{A - GI - idx[aId] \mid a \in A \wedge aId = A - idx[a]\}| \tag{5}$$

$\forall pid \in$ A-GI-idx[aId]. The larger the utility of an action, the larger the benefit that the user can have by a single action. For instance, in the Example 3.2, the action of buying item i_5 (i.e., a_5) is part of three goal implementations: p_3, p_4, p_5, i.e., it can be used in 3 different outfits. Hence, it can be considered as more beneficial to the user compared to the action of buying i_6 (i.e., a_6) that contributes only through 2 goal implementations: p_4 and p_5. However, considering the user activity H={a_2, a_3}, we remark that the user has not showed interest to goal implementation p_3 ($p_3 \notin \mathcal{IS}(H)$). Consequently, goal implementation p_3 should not have been taken into consideration. Thus, we need a measure that captures the utility of an action considering the user activity as well. Moreover, recommending actions of high utility is not enough. We should also consider how related, or else strongly connected, is a candidate action to the user activity. To do so, we need to consider how many of the actions in the user activity are connected to action sets that fulfill goals in the examined goal subspace.

$$sc(a, H, Breadth) = \sum_{\forall \langle g, A\rangle\ where A \cup H \neq \emptyset,\ and a \in A} \mid A \cup H \mid \tag{6}$$

The above equation captures both the utility of a candidate action and its relatedness to the user activity. Now, we can rank

Algorithm 2 Breadth Ranking

Ranks Candidate actions based on all the Implementations of the user's Implementation space where they participate
Input: Set H (user activity), int k
Output: top k actions

```
1   R← {}
2   for all aId ∈ H, for each pid ∈ A-GI-idx[aId]
3      IS= { A-GI-idx[aId] | a ∈ A∧ aId=A-idx[a] }
4   for each pId in IS:
5       ActionsInP ← GI-AV-idx[pId]
6       comm ←| ActionsInP∩H |
7       for each aId ∈ ActionsInP
8           if aId in R.keys:
9               R.add( ⟨ aId, R[aId] + comm ⟩)
10          else:
11              R.add( ⟨ aId, comm ⟩)
12  rank R on score value and return the top k actions
```

Algorithm 3 Get-Goal-Based-Profile

Creates the user profile that reflects her connections with the Goal Space
Input: Set H (user activity)
Output: $\vec{H}$ vector in $\mathcal{GS}(H)$ that aggregates the contribution of all actions in H

```
1   H⃗←∅
2   GP_map←∅
3   for all aId ∈ H, for each pid ∈ A-GI-idx[aId]
4      IS= { A-GI-idx[aId] | a ∈ A∧ aId=A-idx[a] }
5   for each pId in IS:
6       gIdTmp ← GI-G-idx[pId]
7      if gIdTmp in GP_map.keys:
8          GP_map[gIdTmp]← GP_map[gIdTmp]+1
9      else:
10         GP_map[gIdTmp]←1
11  /*convert map GP_map to a vector in F^GS(H) space*/
12  for each gId in GS(H)
13      H⃗.add(GP_map[gId])
```

the candidate actions and get the recommendation list R. To form the recommendation list, we rank the candidate actions using the function described in Equation 6.

Algorithm *Breadth Ranking* presents in pseudocode the steps of the *Breadth*. The algorithm does not estimate the score (ref. Eq. 6) of each action in the $\mathcal{AS}(\mathcal{H})$ separately. It examines each associated implementation and updates the score of all the actions of the $\mathcal{AS}(\mathcal{H})$ that belong in the current implementation. This way, when all the implementations have been examined the action scores (ref. Eq. 6) are ready and the action ranking takes place.

5.3 Best Match

Best Match policy in contrast to *Breadth* that evaluates each action in the user action space considering only the goals in the goal space to which this specific action contributes, considers the whole goal space. In fact, it generates a user profile that reflects the effort that the user has made towards all the goals and retrieves actions that contribute similarly to those goals. *Best Match* considering the user goal space, represents every action as a vector and aggregates the representations of the individual actions in the user activity into a single vector. The final vector constitutes the user profile. Subsequently, the candidate actions can be ranked based on their similarity to the user profile. Such an approach promotes actions that contribute to most of the goals in the user's goal space.

Goal-based user representation. In recommendation systems, user profiles are described in terms of the features of the items that a user prefers. In our case, a profile captures the user dedication towards a set of goals. We consider that the more actions from the user activity H contribute to a specific goal in the goal space $\mathcal{GS}(H)$ of the user activity, the more the user cares for this goal.

Hence, we consider that an action a can be represented as a vector $\vec{a}$ in the feature space $F^{\mathcal{GS}(H)}$ (as an item is represented by considering features in content-based recommendation methods). One option would be to form a boolean vector, $\forall i \in \{0, |\mathcal{GS}(H)|\}$, where $g \leftarrow F^{\mathcal{GS}(H)}[i]$:

$$\vec{a}[i] = \begin{cases} 1, if \;\; \exists p \leftarrow \langle g, A\rangle \;\; s.t. \;\; a \in A, g \in \mathcal{GS}(H) \\ 0, otherwise \end{cases} \quad (7)$$

The problem with the above representation is that it disregards the fact that an action in the user activity may contribute to a goal through one or more implementations. Therefore, instead of the boolean representation, we adopt a vector representation where $\vec{a}[i]$ is defined to be the number of goal implementations p s.t. $a \rightsquigarrow^p g$ and $g \in \mathcal{GS}(H)$. The value in each position of the vector $\vec{a}$ becomes $\forall i \in \{0, |\mathcal{GS}(H)|\}$, where $g \leftarrow F^{\mathcal{GS}(H)}[i]$:

$$\vec{a}[i] = \sum_{\forall p \leftarrow \langle g, A\rangle \; s.t. \; a \in A, \; g \in \mathcal{GS}(H)} 1, \quad (8)$$

To get the *user profile*, we aggregate all the representations of the actions of the user activity in the feature space $F^{\mathcal{GS}(H)}$ into a single vector. The user profile captures for each goal in $\mathcal{GS}(H)$ how many of the user actions contribute to this goal considering the different goal implementations for the same goal as well. Since the user profile is generated based on the current user activity H, we denote it as $\vec{H}$.

$$\vec{H} = \sum_{\forall a \in H} \vec{a} \quad (9)$$

For example, for the user activity: H={a_2, a_3}, the number of goal implementations where at least one of the actions of the user activity participate is 4. The user profile is $\vec{H}$ ={ 3, 0, 2 }. In the user profile is reflected the fact that the user has performed a_1 and a_3 that contribute to g_1: "*meeting friends*" 3 times and to g_3:"*going to the office*" via one goal implementations each, and that the user has shown her/his preference to the goals g_1 and g_2 over the rest of the goals in the goal space $\mathcal{GS}(H)$.

Goal-based representation of candidate actions. To rank the candidate actions against the user profile, we represent each candidate action in the same goal space, i.e., as goal vectors in the space $F^{\mathcal{GS}(H)}$ in the exact same way the actions from the user activity have been represented (ref. Eq. 8).

Distance-based Ranking. To rank the candidate actions, we can use a standard metric between the user profile and each of the candidate actions, as follows:

$$sc(a, H, Best\ Match) = dist(\vec{H}, \vec{a}) \quad (10)$$

For instance, considering the Example 3.2, action a_1 from the user activity H would be closer (smaller distance) to the user profile than that of a_4 since the first contributes to g_1: "*meeting friends*" via two goal implementations and via another goal implementation to g_3:"*going to the office*" as well; while the latter contributes to g_1 via only one goal implementation and to g_2:"be warm" to which the user has shown no interest.

Algorithm 4 Best Match Ranking

Ranks actions based on their distance to the goal-based user profile

Input user activity H, int k
Output top k actions

$R \leftarrow \{\}$, $CA \leftarrow \mathcal{AS}(A)$-$H$
$\vec{H} \leftarrow$ **Get-GoalBased-Profile**(H)
for each aId in CA:
  $GP_{map} \leftarrow \emptyset$
  $\mathcal{IS} \leftarrow \mathcal{IS}(aId)$
  for each pId in $\mathcal{IS}$:
    gIdTmp ← GI-G-idx[pId]
    if gIdTmp in GP_{map}.keys:
      GP_{map}[gIdTmp]← GP_{map}[gIdTmp]+1
    else:
      GP_{map}[gIdTmp]←1
  /*convert map GP_{map} to a vector in $F^{\mathcal{GS}(H)}$ space*/
  for each gId in $\mathcal{GS}(\mathcal{H})$
    $\vec{a}$.add(GP_{map}[gId])
  $\langle aId, sc \rangle \leftarrow \langle$ aId, $dist(\vec{a},\vec{H})\rangle$
  R.add($\langle$ aId, sc $\rangle$)
rank R on sc and return top k actions

Algorithms *Get-GoalBased-Profile* and *Best Match Ranking* describe the procedure. *Get-GoalBased-Profile* forms the goal-based vector representation of the user (user profile) by considering for each action in the user's activity all the implementations where the examined action belongs (i.e., its implementation space) in order to find to which of the goals of the user's goal space it contributes and add one in the respective position of the vector $\vec{H}$. On the other hand, *Best Match Ranking* compares the user profile with the goal-based vector representation of each action in $\mathcal{CA}$ by considering again the goal implementation space of the actions and the goals to which they contribute. and ranks them according to their distance with the user profile to get the top k.

5.4 Complexity analysis of strategies

The complexity of the goal-based recommendation mechanisms is mainly determined by the action *connectivity* of the association-based goal model, i.e., the average number of implementations where an action belongs. *Focus* first retrieves all the implementations that are associated with the user activity and estimates the *closeness* or the *completeness* of each of the implementations. Therefore, the cost of the mechanism is estimated as the product of $\mid H \mid$, connectivity and of the cost of set intersection or asymmetric set difference that are the main operations of the two alternatives of the *Focus* algorithm. The estimation of completeness may be more time consuming in practice (ref. Figure 7). For instance, in our implementation, intersection takes more time than set difference for sets of equal size due to the larger number of element removals that are required in the latter. *Best Match* also starts with retrieving the implementations that are associated with the user activity and then transforms them to vectors in the feature space $F^{\mathcal{GS}}(H)$ (O($\mid H \mid$*connectivity)). Subsequently, this transformation is also performed in all the actions in the action space of the user activity $\mathcal{AS}(H)$. For an action a, the complexity of forming its space $\mathcal{AS}(a)$ depends again on the action connectivity and the average implementation length. Since connectivity is significantly higher than the size of the average goal implementation and the user activity, the time cost is mainly determined by connectivity: O(connectivity*$|H|$ + connectivity*average implementation length). On the other hand, *Breadth* retrieves the associated implementations and gets the intersections of the actions in each of the implementations and the user history. The asymptotic time complexity for the first step is O($|H|$*connectivity) and for the second step O(connectivity*set intersection cost).

In practice, if we consider two association-based goal models built on sets of the same connectivity with the second one containing more implementations, the recommendation time would be higher for the set with the larger number of implementations. The reason is that in larger implementation sets the spaces of associated actions, goals and implementations of the individual actions in the user activity are not overlapping and by consequence their union gets larger. Nevertheless, the algorithms scale very well even on sets of millions of implementations (ref. Figure 7).

6 EVALUATION

For our experimental evaluation, we examine two different scenarios: (*a*) a grocery store where clients (users) buy food products, and (*b*) a system where users record actions they perform in their lives such as *read a book* or *eat healthy*. We selected these scenarios to show that goal-based recommendation can be used both in practical scenarios where existing recommendation techniques have already been applied, and to offer innovative services that have not been available so far. In both scenarios, we want to recommend to the users *actions of interest* (i.e., buy + "a product" and everyday actions respectively). The actions are characterized to be of interest based on the goals that they serve: food products can serve *food recipes*, while everyday actions can serve *life goals* such as *lose weight* or *learn english*. Another reason for examining these scenarios is that they cover two different cases: the first one covers the case where the same action participates in a great range of goal implementations (on average 1.2K impl.), while in the second case, most actions are limited to specific "families" of goals (on average an action participates in 3.85 implementations).

Dataset Description. The first dataset is an open source grocery shopping dataset (https://github.com/julianhyde/foodmart-data-mysql) that contains *1560 food products* (items) and records of customer purchases in different time slots, i.e., *carts*. The food products are organized in 128 (sub)classes such as "baking goods", "seafood", "fruit", "spices" and so forth. Clients can utilize these products in various *recipes* to produce *different dishes* (goals). We used a dataset of *56.5k recipes* from a food ontology (http://data.lirmm.fr/ontologies/food#Recipe). The number of implementations in which an action participates on average, i.e., the *connectivity*, is 1.2K. We run our recommendation techniques and the state-of-the-art algorithms using as input, i.e., current user activity, 20.5k client carts.

The second dataset consists of goal implementations from a *goal-setting online social platform* called 43Things where users could publish the goals they set in their lives, "cheer" other users' goals and efforts, and provide descriptions about how they managed to fulfill their goals. We have extracted *18047 goal implementations* that contain *3747 goals* such *pay my depts*, *get a new job*, *lose weight*, and *5456 actions* e.g., *stop eating at restaurants* and *drink more water*. Both goals and actions are identified by unique identifiers. In contrast to the foodmarket dataset, users are focused on a few real-life goals. In total, we have examined 8071 users. The majority of the users (5047 users) are pursuing one goal, 1806 of them pursue 2 goals, 623 pursue 3, and 595

Goal Id	*Actions Performed for Goal Fulfillment*
g_1	{3,4,5,6,7, 8,9,10,11}
g_2	{12,13,14,15,16,17,18,19,20}
g_3	{1,2}

Initial User Activity	{3,4,5,6,7, 8,9,10,11,12, 13,14,15,16,17,18,19,20,1,2}
User activity used as input (unhidden actions)	{11,20,8,12,1,13 }

Table 1: Forming of user activity.

pursue more than 3 goals. Moreover, the action *connectivity* is very low, 3.84. The fact that actions here in contrast to actions that involve food ingredients are useful in a narrow range of goals and by extension goal implementations makes the analysis of the two sets more intriguing. User activities consist of all the actions that a user has performed for the fulfillment of all the goals that s/he has set. Therefore, in order to evaluate the recommenders we should hide a portion of the actions from the real user activities before applying the recommendation techniques. For instance, Table 1 illustrates all the actions that a user has performed to fulfill three goals. To get the input for the recommenders, we first concatenate the actions of the three respective implementations into the vector $\{1, 2 \ldots 20\}$. Subsequently, the vector elements are shuffled and the 30% of the actions is considered to be the known user activity. The rest 70% is kept for evaluation purposes. Specifically, the recommenders, having some evidence that the user is interested in fulfilling one or more goals, should retrieve actions that are associated with those goals. In the example, the activity that remains unhidden consists of 6 actions. Two of the actions regard the first goal, three of them the second goal, and one action regards the last goal. Actions are both about goals that are closer to fulfillment and about goals for which there is no strong evidence. Some goals may also remain hidden.

Comparison with the State-of-the-art. Beyond the goal-based mechanisms, we examine how state-of-the-art recommendation approaches behave under the same context. We consider a nearest-neighbor Collaborative Filtering method (CF KNN) [20] and a matrix factorization method that employs alternating least squares with weighted-lamda-regularization (ALS-WR) to factorize the user-item matrix before performing the recommendations (CF MF) [8]. For the CF KNN since the user feedback is implicit, i.e., user *selection*, *non-selection*, the neighborhoods have been formed by employing jaccard coefficient, or else Tanimoto coefficient. The used implementation of the CF MF is from Mahout framework (https://mahout.apache.org/). We also consider a Content-based method that represents actions and users using domain-specific features, i.e., that builds vector-based representations for profiling the actions and users. For the foodmarket dataset the used domain-specific features are the 128 (sub)categories of the food products (e.g., "baking goods", "seafood"). Based on the description of each food product, we matched each product to an ingredient leaving out products that are not included in any recipe, such as napkins. Therefore, each cart can be seen as the *user activity*, the set of recipes as the *goal implementation set L*, while the *actions* refer to the purchase of certain products/ingredients. On the other hand, for the 43T dataset, there are no widely accepted domain-specific features; therefore, we do not apply the content approach. For the goal-based recommendations, we used the methods described in Subsections 5.1, 5.2, and 5.3), namely $Focus_{cmp}$ and $Focus_{cl}$, *Breadth* and *Best Match*.

Subsection 6.1 compares all methods. Subsection 6.2 focuses on the time efficiency of the goal-based methods.

6.1 Evaluation and Comparisons

Since we are introducing a novel recommendation approach, in Subsection 6.1.1, we first verify that this approach, indeed offers a different perspective to the users. To do so, we perform several *comparisons* on the results (i.e., the recommendation lists) produced by all the goal-based and the standard recommendation mechanisms.

- We compare the lists formed by the goal based mechanisms with the two standard recommendation mechanisms (ref. C.1.1. *Result Overlapping*).
- Collaborative filtering is based on the past activities of similar users (to the current user), while our algorithm is intended for discovering useful actions, i.e., actions that will help the user fulfill one or more goals. Thus, we examine whether actions that appear frequently *in the activities of other users* (popular actions) appear frequently *in the recommendation lists* as well. In other words, we study which recommendation mechanisms perpetuate the collective user behavior (ref. C.1.2. *Correlation of appearances in the user activities and the respective recommendation lists*).
- Next, we examine how useful the actions in the top-10 lists of each algorithm are for the user. To measure *usefulness*, we estimate the completeness of the goals in the user's goal space after s/he performs the recommended actions (ref. C.1.3. *Usefulness*).
- We also study how *similar* the recommended actions in each list are presenting their (max, min and avg) pairwise similarity based on their domain-specific characteristics. Retrieving items that are very similar to each other is often considered a drawback of the Content-based filtering. It is important to understand how the rest of the examined approaches work as well (ref. C.1.4. *Pairwise similarity of the recommended actions*).
- Moreover, we examine how many of the actions in the recommendation lists have been indeed performed by the users. These actions are not of course part of the considered user activity but the users "like" them since they have performed them at some point (ref. C.1.5. *Average Percentage Of Recommended Actions that the user has indeed Performed.*)

Subsequently, Subsection 6.1.2 further examines the actions retrieved by the goal-based mechanisms (ref. C.2.1 *Frequency of Retrieved Items*) and presents the percentage of common actions in their top-10 recommendation lists (ref. C.2.2 *Result Overlapping of Goal-based methods*).

6.1.1 Comparison of all Approaches. *C.1.1. Result Overlapping*. Table 2 illustrates a very low overlapping of the top-10 lists formed by the goal-based mechanisms with the lists formed by the two state-of-the-art approaches. This result is expected, since as we have explained in Section 2, these approaches adopt fundamentally different philosophies.

C.1.2. Correlation of the number of appearances in the user activities and the number of appearances in the respective recommendation lists of the *top-20 most popular actions*. Table 3 illustrates the Pearson's correlation between these two numbers. Correlation

	Food Market			*43T*	
Methods	*Overlap.with Content Filt.*	*Overlap.with Collab. Filt. kNN*	*Overlap. with Collab. Filt. Matrix Factorization*	*Overlap. with Collab. Filt. kNN*	*Overlap. with Collab. Filt. Matrix Factorization*
Best Match	2.31%	0.85%	0.34%	0.13%	0.06%
$Focus_{cmp}$	1.49%	0.85%	0.36%	0.11%	0.008%
$Focus_{cl}$	1.85%	0.85%	0.35%	0.08%	0.01%
Breadth	2.32%	0.86%	0.35%	0.26%	0.11%

Table 2: Overlap of the top-10 actions retrieved by the goal-based mechanisms and the standard recommendation approaches.

	Food Market	*43T*
Methods	*Correlation*	*Correlation*
Content	0.115	-
Collaborative KNN	0.45	0.75
Collaborative MF	0.78	0.87
Best Match	-0.13	-0.24
$Focus_{cmp}$	-0.048	-0.26
$Focus_{cl}$	-0.02	-0.27
Breadth	-0.04	-0.15

Table 3: How correlated the recommendation lists with the top-20 popular actions in the user activities are.

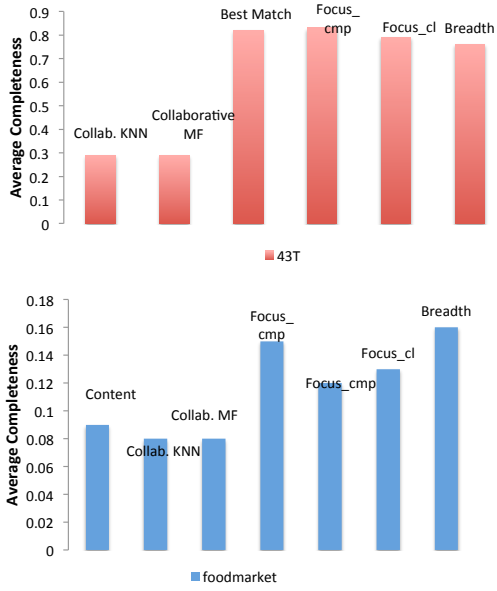

Figure 3: The average goal completeness per list after the user follows the recommended actions in each list.

takes negative values from -1 to 1, with 1 reflecting highly correlated values. Collaborative filtering, which looks into the past actions of similar users for actions that may be of interest to the current user, shows the highest correlation. On the other hand, goal-based methods show negative correlation. They do not promote actions that were popular (frequent) so far. The content-based approach shows a lower correlation than collaborative filtering, which is still high in comparison to the goal-based methods.

C.1.3. Usefulness: the completeness of the goals in the user's goal space after s/he follows the recommended actions. The actions recommended to the user can help her get closer to the fulfillment of (or fulfill) one or more goals. Table 4 shows the average average

Methods	*Food Market Completeness*			*43T Completeness*		
	Avg-Avg	*Avg-Max*	*Avg-Min*	*Avg-Avg*	*Avg-Max*	*Avg-Min*
Content	0.09	0.67	0.05	-	-	-
Collaborative KNN	0.08	0.63	0.05	0.29 0.29	0.32 0.33	0.26 0.26
Collaborative MF	0.08	0.64	0.05	0.29	0.32	0.26
Best Match	0.15	0.79	0.076	0.82	0.87	0.77
$Focus_{cmp}$	0.12	0.73	0.064	0.83	0.88	0.77
$Focus_{cl}$	0.13	0.74	0.062	0.789	0.84	0.73
Breadth	0.16	0.8	0.076	0.76	0.8	0.72

Table 4: How complete become the goals of the user after s/he follows the recommended actions per list.

(AvgAvg), min (MinAvg) and max (MaxAvg) completeness values for all the recommendation lists formed for the two datasets. Figure 3 shows graphically the average values. These values are estimated by finding first the average, minimum and maximum values of completeness of all the goals that are related to the user *considering each list separately*. Subsequently, the average for all the recommendation lists is estimated. The goals that we consider in the estimation of goal completeness in the case of the 43T are those that the user has added in the system, while in the case of the food market we consider the whole user's goal space since we do not have any information about which goals the user is pursuing in reality. In the foodmarket dataset, the goal implementation space can be large and not every goal can be fulfilled by performing only 10 actions (i.e., the actions in the recommendation list) in any case. As a consequence, the AvgAvg values in this dataset are not that informative in comparison to those of the 43T dataset.

We observe that *Breadth* and *Best Match* in the first dataset and $Focus_{cmp}$ in the second dataset manage the largest completeness (considering both the user activity and the recommended actions), while the lowest contribution is met in the state-of-the-art algorithms. The results are explained by the fact that *Best Match* considers the whole user's goal implementation space, *Breadth* creates a well-connected subspace, while $Focus_{cmp}$ selects a single goal (actually a single implementation), if possible, and extends to a few more to complete the recommendation list. *If the user wants to get closer to a wider range of goals, s/he should select Breadth; otherwise (i.e., if s/he is focused on a few goals), s/he should select* $Focus_{cmp}$. *Best Match* and $Focus_{cl}$ follow.

C.1.4. Pairwise similarity of the recommended actions (i.e., the corresponding products) in each list. Table 5 shows the pairwise similarity among the retrieved actions in each recommendation list. Due to the lack of widely-accepted domain-specific characteristics for the actions in the 43T dataset, we study the food market

Methods	*Pairwise Action Similarity*		
	AvgAvg	*AvgMax*	*AvgMin*
Content	0.81	1	0.6
Collaborative KNN	0.16	0.5	0.05
Collaborative MF	0.15	0.77	0.04
Best Match	0.33	0.72	0.22
$Focus_{cmp}$	0.24	0.31	0.21
$Focus_{cl}$	0.24	0.34	0.19
Breadth	0.33	0.73	0.22

Table 5: Pairwise feature-based similarity among the actions within each recommendation list for the foodmarket dataset.

dataset. AvgAvg is estimated in two steps: first *the average pairwise similarity* considering all the action pairs within each list is estimated, and then the average of the derived values is estimated. The same applies for AvgMax and AvgMin. As expected *Content* shows the highest value with an AvgAvg pairwise value 0.8 and AvgMin value 0.6. Collaborative filtering shows the lowest similarity (AvgAvg 0.15), while all goal-based mechanisms are found in the middle (avg-avg: 0.24-0.33). However, looking at the average maximum pairwise values (AvgMax), we see that the goal-based methods *Best Match* and *Breadth* often share a pair of very similar actions in their lists (on average their max pairwise similarity values are 0.72 and 0.73 respectively). The two *Focus* methods are the goal-based methods that retrieve highly dissimilar actions in most of the cases.

C.1.5. Average percentage of recommended actions that the user has indeed performed (per recommendation list). In the food market dataset, we consider as the user's current activity a single cart; we have more than one cart for the same user in different time slots though. On the other hand, in the 43T dataset we consider only the 30% of the actions that the users have performed to fulfill their goals. Therefore, we can check whether the different techniques by considering only the actions in the user activity, recommend actions that the user has performed. We should clarify that the average percentage of recommended actions that the user has indeed performed does not reflect the precision of the recommendation tasks since the user has not acted after checking the recommendation lists. In fact, it shows the percentage of the recommended actions for which the user has shown interest at some point. Unlike precision, being able to retrieve actions that the user would anyway perform can be an advantage or a disadvantage for a recommendation technique depending on the view point of the application. If the purpose of the recommendation system is to show to the user unknown actions as well, a very high percentage is not preferable. On the contrary, if the purpose of the system was to provide the user with a discount coupon in order to keep her/him satisfied, a high value would be preferable. Keeping that in mind, we can say that the average percentage represents the Average True Positive Rate. Figure 4 illustrates for each method the *Avg TPR* for top-5 and top-10 lists. In the top-5 lists, first the *Best Match*, then the $Focus_{cmp}$ and *Breadth* show the largest percentage. In the top-10 lists of the foodmarket dataset though, it is the *Content* method that shows the highest percentage. Nevertheless, all the methods show low percentages in the foodmarket dataset. This is explained by the fact that we have no more than 3 carts for each user.

6.1.2 Further Comparison of Goal-based results. Considering the lists derived from the goal-based methods, we have already argued about the fact that the appearance of an action in

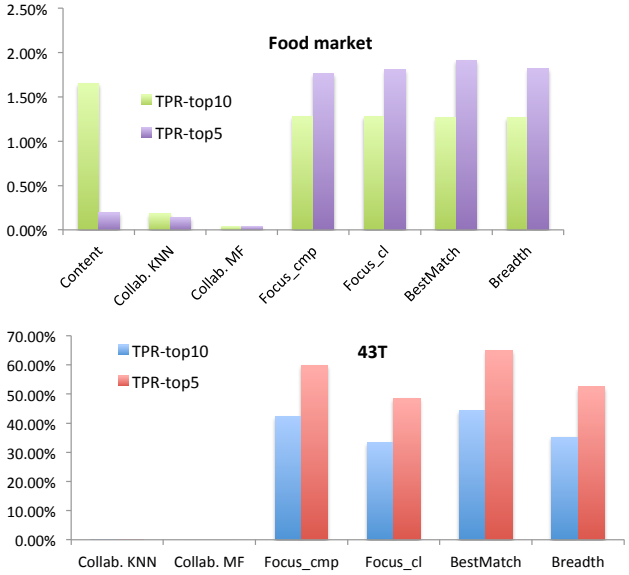

Figure 4: Percentage of recommended actions that the user has indeed performed (True Positive Rate for top-5 and top-10 lists).

the recommendation lists is not correlated to its appearance in the user activities (ref. Table 3). Next we also present whether there exist actions that monopolize the recommendation lists, and how different the recommendation lists formed by the alternative goal-based methods are (*Result Overlapping of Goal-based methods*).

C.2.1. Frequency of Retrieved Items. In recommenders, we do not want certain actions to monopolize the recommendation lists. In the 43T dataset, the frequency of an action in different recommendation lists is very low: *at maximum 0.001*. On the other hand, in the food market dataset, where there are a lot of actions that participate in a great number of implementations (average connectivity 1.2k), the frequency is higher. Figure 5 illustrates that the majority of actions appear with frequency less than 0.2. However, *Best Match* and then *Breadth*, in their effort to serve more than one goal at the same time, repeat the same actions in more recommendation lists (22% and 14% actions respectively with frequency above 0.2). *The actions with high frequency are those that appear frequently in subsets of implementations that share common actions*. Actions that appear in many goal implementations *but together with different actions in each goal implementation* are not selected more frequently. On the contrary, Figure 6 shows that very few actions that appear frequently in the goal implementation sets are in the end selected by any goal-based mechanism. The great majority (more than 92%) of the retrieved actions (by all the goal-based mechanisms) appear in the implementation set with a frequency less than 0.2.

C2.2. Result Overlapping of Goal-based methods. In Paragraph *C1.1*, we have presented how different are the results of the goal-based mechanisms from those of the standard recommendation methods, next we present the result overlapping of the goal-based mechanisms. Table 6 illustrates the percentage of common actions in their top-10 lists considering again as input the 21k real carts and the 8k user activities of the food market and the 43T datasets respectively. First of all, we observe a great overlapping in the results of *Best Match* and *Breadth*: 98% and 79% respectively.

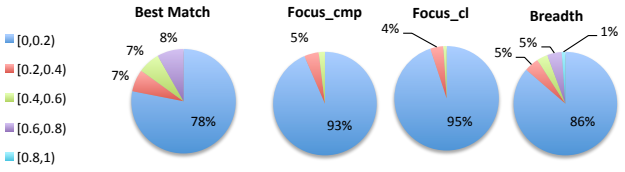

Figure 5: How often the same action appears in the recommendation lists that have been formed for the user activities of the food market dataset. Distribution of actions in frequency ranges.

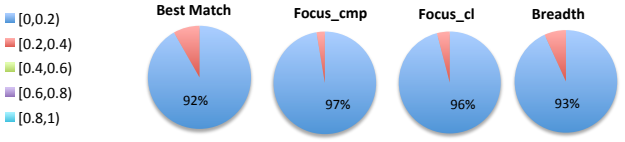

Figure 6: How often the same retrieved action appears in the goal implementation set (herein recipes). Distribution of actions in frequency ranges.

	Food Market	*43T*
Methods	*Overlapping*	*Overlapping*
Best Match-$Focus_{cmp}$	42%	68%
Best Match-Breadth	98%	79%
$Focus_{cmp}$-Breadth	44%	71%
$Focus_{cl}$-$Focus_{cmp}$	35.6%	78%
$Focus_{cl}$-Best Match	49%	72%
$Focus_{cl}$-Breadth	49%	72%

Table 6: Common actions in the top-10 recommendation lists.

The overlapping is higher in the first case because in the food market ingredients participate in a lot of recipes at the same time. Therefore, *Breadth* instead of examining subsets of the user's goal space to evaluate a certain action, it ends up considering (almost) the whole goal space similarly to *Best match*. In general, the user profile that *Best Match* considers reflects more strongly her/his preference towards a subset of goal(s); thus it (almost) neglects the rest of the goals in the user's goal space the same way *Breadth* does. Since the two algorithms show similar behavior, *Breadth* is preferred since, as we will see in Subsection 6.2, *Breadth* is significantly more efficient in terms of time.

Moreover, $Focus_{cmp}$ and $Focus_{cl}$ retrieve the same actions in 35.6% and 78% of the lists respectively. In these cases, there exist goal implementations for which the user has performed most of the actions (completeness) and at the same time these are the implementations with the less remaining actions. Furthermore, $Focus_{cl}$ and $Focus_{cmp}$ show an overlapping of over 40% and 70% (for the respective datasets) with *Breadth* and *Best Match*. This is justified by the fact that the Focus mechanisms after popping out all the actions of the goal implementation on which they have selected to focus, they move on to another goal implementation. Therefore, they select actions from different goal implementations as *Breadth* and *Best Match* do. Another way to see this is that the latter two algorithms select actions that serve more than one goals at the same time; but that means that the selected actions serve each single goal on its own as well.

Another observation is that the overlapping in the lists for the 43T dataset is larger than in the lists for the food market dataset in all the cases because the action space of the users are wider in the latter dataset due to the high action connectivity. Considering a larger set of candidate actions, the algorithms are not forced to select the same actions due to lack of alternatives.

Set	*Connectivity*	*Num Of Distinct Actions*	*Num Of Implementations*
IS	1.2K	380	56K
IS_2	7.6K	380	282K
IS_3	15.6K	380	564K
IS_4	50.3K	380	1.6M
IS_5	8K	10K	120K
IS_6	8K	100K	1.2M
IS_7	8K	1M	12M

Table 7: Goal Implementation Sets.

Alg	IS (56K Impls, Conn 1.2K)	IS_2 (282K Impls, Conn 7.6K)	IS_3 (564K Impls, Conn 15.6K)	IS_4 (1.6M Impls, Conn 50.3K)
Best Match	0.37 s	1.5s	3 s	9.7s
$Focus_{cl}$	0.001s	0.053	0.096s	0.35s
$Focus_{cmp}$	0.091s	0.42	0.86s	2.98s
Breadth	0.006s	0.089s	0.089s	0.34s

Table 8: Average Execution Time in implementation sets with high connectivity.

Alg	IS_5 (10K Actions, 120K Impls	IS_6 (100K Actions, 1.2 Impls	IS_7 (1M Actions Actions, 12M Impls
Best Match	0.713s	3.37s	5.38s
$Focus_{cl}$	0.0017 s	0.0026s	0.0034s
$Focus_{cmp}$	0.0024s	0.0035s	0.0055s
Breadth	0.0029s	0.0052 s	0.008s

Table 9: Average Execution Time in implementation sets with a large number of actions and implementations.

6.2 Scalability

We ran the 4 goal-based strategies (i.e., the 3 strategies plus the extra option for *Focus*) on goal-based association models that have been built based on 7 implementation sets of different characteristics: (*a*) implementation set size, (*b*) action set size, and (*c*) number of implementations in which an action participates on average (*connectivity*). Table 7 illustrates the implementation sets. In sets IS_2, IS_3 and IS_4 the connectivity is stretched up to 50.3K in a set of 1.2M implementations, while in IS_5, IS_6 and IS_7 the size of the action set and the number of implementations increases by 10 times (IS_7 consists of 1M actions and 12M implementations).

Results. Figure 7 illustrates the average time per information need (i.e., per user activity) in secs considering each of the different implementation sets. We observe that the *Best Match* shows the highest execution times in all the cases. The reason is that in goal-based profiles the feature space is not fixed, and thus the representation of the actions is formed on the fly. The rest of the mechanisms show low recommendation time even in the extreme cases of the sets IS_4 and IS_8 (connectivity: 50315, average participation: 19M, and connectivity: 12110, average participation: 137M respectively). On the other hand, $Focus_{cl}$ shows the lowest

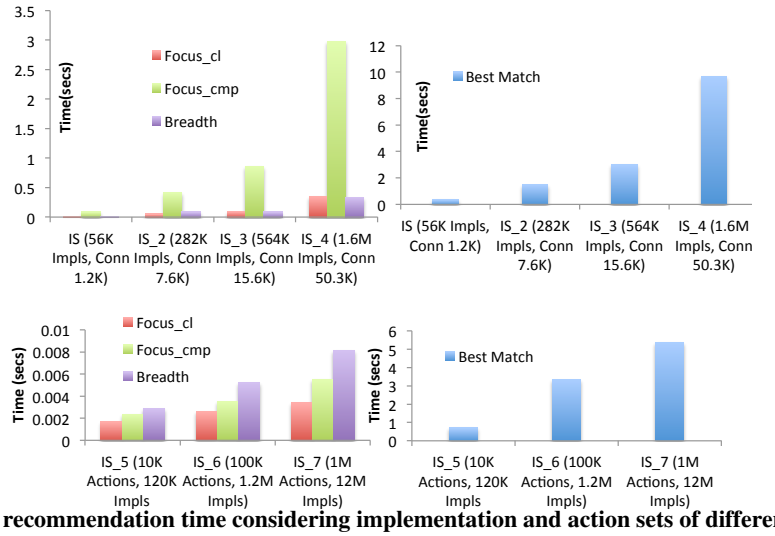

Figure 7: Average recommendation time considering implementation and action sets of different characteristics.

execution time. The difference between $Focus_{cl}$ and $Focus_{cmp}$ results from the two set operations that the mechanisms use, i.e., asymmetric difference and intersection respectively.

In conclusion, the goal-based mechanisms scale well even in sets of millions actions and implementations. Moreover, *the number of actions and implementations alone do not affect much the execution time, it is the higher connectivity that results in higher execution times*.

7 CONCLUSION

Based on the theory that goals rationalize and by consequence trigger user actions, we introduce a family of recommendation approaches that recommend actions seeing them in respect with a number of goals that the users may fulfill through different action sets. We have presented 3 strategies, each one incorporating goals into the scoring of actions in a different way. The action selections of the goal-based mechanisms are not affected by their domain-based similarity with the actions in the user's activity, nor by the activities of other users. However, they are affected by the benefit of the actions to be recommended to the goals in the user's goal space. The strategies *Breadth* and *Best Match* focus on more than one goal at a time. In fact, the latter considers all the goals in the goal space independently from the examined action. On the other hand, the *Focus* mechanisms focus on the fulfillment of one goal at a time. Nevertheless, they all increase the average goal completeness in the user's goal space without retrieving actions that monopolize the goal implementations. Moreover, all the mechanisms create different recommendation lists for different inputs (i.e., user activities). As part of our future work, we have been examining methodologies that enhance the goal-based mechanisms by considering the user preferences on certain domain-specific characteristics, i.e., hybrid goal-based and content-based approaches.

REFERENCES

[1] Henk Aarts and Ran R. Hassin. 2004. Automatic goal inference and contagion: On pursuing goals one perceives in other people's behavior. (01 2004), 153–167.

[2] Marcelo G. Armentano and Analía Amandi. 2009. Goal Recognition with Variable-order Markov Models. In *Proceedings of the 21st International Jont Conference on Artifical Intelligence (IJCAI'09)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1635–1640.

[3] Suhrid Balakrishnan. 2010. On-demand Set-based Recommendations *(ACM RecSys '10)*. ACM, New York, NY, USA, 313–316.

[4] Richard W. Byrne and Anne E. Russon. 1998. Learning by imitation: A hierarchical approach. *Behavioral and Brain Sciences* 21, 5 (1998), 667–684.

[5] Sandra Carberry. 2001. Techniques for plan recognition. *User Modeling and User-Adapted Interaction* 11, 1-2 (2001), 31–48.

[6] Ada S. Chulef, Stephen J. Read, and David A. Walsh. 2001. A Hierarchical Taxonomy of Human Goals. *Motivation&Emotion* 25, 3 (2001), 191–232.

[7] Mukund Deshpande and George Karypis. 2004. Item-based top-N Recommendation Algorithms. *ACM Trans. Inf. Syst.* 22, 1 (2004), 143–177.

[8] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *In IEEE International Conference on Data Mining (ICDM 2008*. 263–272.

[9] Neil Hurley and Mi Zhang. 2011. Novelty and Diversity in Top-N Recommendation – Analysis and Evaluation. *ACM Trans. Internet Technol.* 10, 4, Article 14 (March 2011), 30 pages.

[10] Yuchul Jung, Jihee Ryu, Kyung-min Kim, and Sung-Hyon Myaeng. 2010. Automatic Construction of a Large-scale Situation Ontology by Mining How-to Instructions from the Web. *Web Semant.* 8, 2-3 (July 2010), 110–124.

[11] S. Keren, A. Gal, and E. Karpas. 2015. Goal Recognition Design for Non-Optimal Agents. In *AAAI*. 3298–3304.

[12] Jiye Li, Bin Tang, and Nick Cercone. 2004. Applying association rules for interesting recommendations using rule templates. In *Advances in Knowledge Discovery and Data Mining*. Springer, 166–170.

[13] Dimitra Papadimitriou, Georgia Koutrika, John Mylopoulos, and Yannis Velegrakis. 2016. The Goal Behind the Action: Toward Goal-Aware Systems and Applications. *ACM Trans. Database Syst.* 41, 4, Article 23 (Nov. 2016), 43 pages.

[14] Paolo Pareti, Ewan Klein, and Adam Barker. 2014. A Semantic Web of Know-how: Linked Data for Community-centric Tasks *(WWW '14 Companion)*. ACM, New York, NY, USA, 1011–1016.

[15] Donald Patterson, Lin Liao, Dieter Fox, and Henry Kautz. 2003. Inferring High-Level Behavior from Low-Level Sensors. In *Lecture Notes in Computer Science*, Vol. 2864. 73–89.

[16] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing Personalized Markov Chains for Next-basket Recommendation *(WWW '10)*. ACM, New York, NY, USA, 811–820.

[17] Jihee Ryu, Yuchul Jung, Kyung-min Kim, and Sung H. Myaeng. 2010. Automatic Extraction of Human Activity Knowledge from Method-Describing Web Articles. *Proceedings of the 1st Workshop on Automated Knowledge Base Construction* (2010).

[18] Eldar Sadikov, Jayant Madhavan, Lu Wang, and Alon Halevy. 2010. Clustering Query Refinements by User Intent. In *In (WWW '10)*. ACM, New York, NY, USA, 841–850.

[19] J. J. Sandvig, Bamshad Mobasher, and Robin Burke. 2007. Robustness of Collaborative Recommendation Based on Association Rule Mining *(RecSys '07)*. ACM, New York, NY, USA, 105–112.

[20] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based Collaborative Filtering Recommendation Algorithms *(WWW '01)*. ACM, USA, 285–295.

[21] Markus Strohmaier, Mark Kröll, and Christian Körner. 2009. Automatically annotating textual resources with human intentions. In *Proceedings of the Hypertext 2009*. 355–356.