Recommendation by Examples

Rubi Boim and Tova Milo
School of Computer Science,
Tel Aviv University
boim@post.tau.ac.il, milo@cs.tau.ac.il

ABSTRACT

Recommender systems usually rely on user profiles to generate personalized recommendations. We argue here that such profiles are often too coarse to capture the current user's state of mind/desire. For example, a serious user that usually prefers documentary features may, at the end of a long and tiring conference, be in the mood for a lighter entertaining movie, not captured by her usual profile. As communicating one's state of mind to a system in (key)words may be difficult, we propose in this work an alternative method which allows users to describe their current desire/mood through examples. Our algorithms utilizes the user's examples to refine the recommendations generated by a given system, considering several, possibly competing, desired properties of the recommended items set (rating, similarity, diversity, coverage). The algorithms are based on a simple geometric representation of the example items, which allows for efficient processing and the generation of suitable recommendations even in the absence of semantic information.

1. INTRODUCTION

Recommender systems aim to provide *personalized* recommendations by capturing the user's *taste*. Solutions range from ones using semantic properties of users and items (e.g. age, genre, etc.) to semantic-less ones such as Collaborative Filtering that are based only on users scores for items (e.g. "people who liked this set of items also liked...")[16]. While many recent works focus on improving the accuracy of such systems, we argue that the granularity of "user's taste" that they capture is too coarse. Indeed, none of these methods, sophisticated as they might be, captures the user's *current "state of mind"*, or her *current "mood"*. For instance, a user might usually prefer documentary features, but before a date with her boyfriend she might be in a lighter romantic mood and prefer recommendations for a romantic comedy. Alternatively, the same user may be sharing her laptop with her daughter and thus get recommendations for the new "Harry Potter" book, rather than ones that interest her, because her daughter often uses her laptop and thus the profile is more affected by her choices.

A key difficulty in providing suitable recommendations to users in such a scenario is that it is not always easy for a user to describe her current mood/desire to the system in (key)words. Indeed, recent works suggested the use of an *example*, instead of verbal description [11]. For instance, *Pandora* [11] asks users for an example of a song they would want to hear, then attempts to generate a playlist of similar songs. (We will discuss what "similar" means later).

But is a single example indeed enough to describe the user's current state of mind/mood? We argue that the answer is *No*. A user, for instance, might have had a long day at work and is interested in watching a "light" movie (that is, an enjoyable movie which does not require its full attention). Capturing such a desire with a single movie example is hard, as there are different kinds of light movies, e.g. of different genres. If she gives, for instance "American Pie" - a comedy - as a (single) example, or alternatively "The Rock" - an action movie, the two recommendation lists would be very different: In the first case it is likely to consist only of comedy-like movies, whereas in the second case of only action-like features. While each individual list indeed contains relevant items, it clearly does not cover the relevant spectrum. The overspecialization may further yield improper recommendations, e.g. "heavy" action movies like "Predator". Indeed, what is desirable here is to use jointly the two examples above to capture the user desire more accurately, recommending action-comedy features like "Lethal Weapon" or "Bad Boys". (We refer below to such items as *joint representatives*).

This paper aims to provide precisely such methods for capturing the user's current desire/mood through multiple examples. Our proposed algorithm utilizes the user's examples to refine the recommendations generated by a given recommender system, by considering several, possibly competing, properties of the proposed items (to be further discussed in the sequel); (a) the *similarity* to the given individual examples, (b) the *joint similarity* to subsets of the examples, as illustrated above, (c) the (possibly personalized) items scores - often called *rating* - given by the recommender system (items with higher rating are more relevant), (d) the *diversity* of the recommended items, and (e) their *coverage* of the examples.

The problem that we address here is a particular type of personal, context-aware, recommendations. Prior work on Context-Aware Recommender Systems (CARS) (e.g. [10, 7, 15]) typically assumes the presence of some (auxiliary) context information, such as location, time, domain, semantics, etc., which plays a key role in their algorithms. Our work targets common situations where such data does not suffice to indicate the current user mood, or is not available. (If such information is present, it can still be leveraged by the underling recommender system, as explained below).

Our work integrates several existing technologies in a new algorithm to produce a user mood-specific recommendations list. Given the set of user examples, we first take a Collaborative Filtering (CF) approach [16] to evaluate the similarity between the given examples and the items in the database, without requiring any semantic information. (We explain briefly in the sequel how CF works).

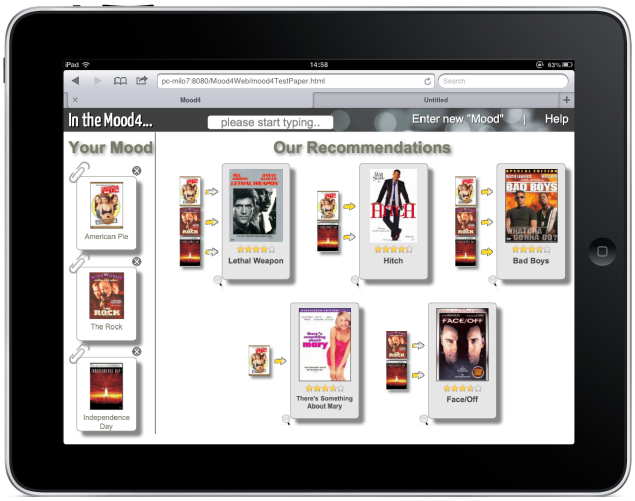


Figure 1: Mood4 's main screen (iPad version)

Our algorithm then creates a geometric representation of the items space, in order to find the joint representatives mentioned above (also to be discussed in the sequel) which play a key role in the algorithm. The geometric representation is then updated and blended with the (personalized) rating of each item, as given by the underlying recommender system, in preparation for the final weighting mechanism. Finally, we use a novel technique from [5], based on priority-medoids, to diversify the final recommendations.

Mood4. To demonstrate the effectiveness of our approach, we implemented the above solution in the Mood4 prototype system. Mood4 is designed as a plug-in the can be deployed on CF-based recommender system. Figure 1 depicts a screenshot of Mood4 when used in the context of movie recommendations, following the example of the "light-viewing" mood discussed above. "American Pie" (comedy), "The Rock" (action) and "Independence Day" (action) are the examples chosen by the user to describe her current desire/state of mind (on the left part of the screen). The recommendation generated by Mood4 are presented on the main screen, and indeed include joint representatives such as "Bad boys" and "Lethal Weapon" (both are action-comedy) . Interestingly, Mood4 also captures a different kind of relationship - one that is not only based on the genre, but also on the casting: "Hitch" is a comedy movie, and its leading actor is Will Smith, which stars in "Independence Day". This is a especially interesting as it is achieved in spite of the fact that no semantic information on movies is used by Mood4 , and is due to the power of CF. Finally, Mood4 also provides two useful features to its users: a visual explanation for each recommendation, and a "zoom-in" facility attached to each recommended item, allowing users to further explore similar recommended items.

Mood4 was demonstrated in [4] where only a high-level description of the system was given. The current paper presents the underlaying model, the algorithms, and their experimental evaluation. Section 2 briefly describes CF and presents the geometric representation that we use. Section 3 presents the joint representatives, the enhanced geometric representation, and the items selection process used to generate the recommendations. The experimental results and the related work are finally presented in Sections 4 and 5.

2. GEOMETRIC REPRESENTATION

In this section we present the model on which we base our recommendations. We start with notations and a short description of CF, then we describe the geometric representation that we employ.

Let I be a finite domain of items and let $S \subseteq I$ be the set of example provided by the user. We refer the items in S as *seeds*. Our goal is to find a subset $I_k \subset I$, of size k, of items to recommend the user, matching best her mood as reflected by the provided seeds.

Collaborative Filtering. Collaborative Filtering (CF) is a method of making automatic predictions, and thereby recommendations, for how much a user would like a given item, based on the ratings that other (similar) users gave to (similar) item. Unlike semantic-based systems, where recommendations are made by analyzing the (semantic) properties of each item (such as color, genre, size, etc.), CF utilizes only raw user ratings (such as 1 to 5 stars). Intuitively, it is based on the assumption that users who agreed in the past on items ratings are likely to agree again in the future. Recommendations are made first by the estimation of unknown ratings, that is the items the user have yet rated, and then by the selection of the items with the highest estimated rating. (Intuitively, these items are the ones the system believes the user would like best).

The main component in CF system is the similarity estimation between two items (users). Intuitively, each item is viewed as a vector of ratings in a multi-dimensional space, where each dimension represents the rating of the user corresponding to that dimension. (The similarity between users is analogously defined). Similarity between items is then evaluated by measuring the distance between different vectors, by some distance measure such as cosine or Pearson's correlation coefficient [12]. Rating predictions are traditionally computed in two steps: First, we search for a neighborhood of items, similar to the given one, that have already been rated. Then the predicted rating is computed by aggregating (e.g. averaging) the known ratings of the neighborhood members. In the reminder of this paper we denote by $rate(u, i)$ the (predicted) rating of an item i by a user u. For simplicity, we assume the current user is known from the context and simply use $rate(i)$. More generally, $rate(i)$ can be used to denote the value of a more sophisticated personalized algorithm, if such is available. We denote the similarity between two items i and j by $sim(i, j)$. W.l.o.g. we assume below that distance values are in the range of $[0, 1]$ (a larger value yield a higher similarity). When this is not the case one may naturally map the values to this range.

Geometric Representation. Our algorithm starts by computing, for each seed item $s \in S$, a neighborhood $N(s)$ consisting of the K items most similar to it, for some constant value K (we later show that $K = 50$ provides us the best results). We then look at the geometrical representation of the seeds and their neighborhoods. Note that, in general, sim is not always a metric function. But to simplify the presentation we will assume below that it is, and display the geometric representation on a simple 2-d plan, where the geometric distance between items represents the sim function (items with a higher similarity measure will appear closer). Figure 2 (a) depict the geometric representation of the example discussed in the Introduction. The rating of each item (as evaluated by the underlying recommender system) is given next to the item's name. The neighborhood of each seed item defines a *circle* where the corresponding seed is the center and the radius is defined by the least similar item in its corresponding neighborhood.

But how can we decide which items should we choose to present the user? (That is, to construct the set I_k). As mentioned in the Introduction, our algorithm considers several (possibly conflicting) properties. We recall below these properties and describe their geometrical equivalent requirement: For (a), the *similarity* to given individual seeds, the geometric requirement is to choose items closest to the center of the circles. For (b), the *joint similarity* to subsets of the seeds, the requirement is to choose items in the intersections of

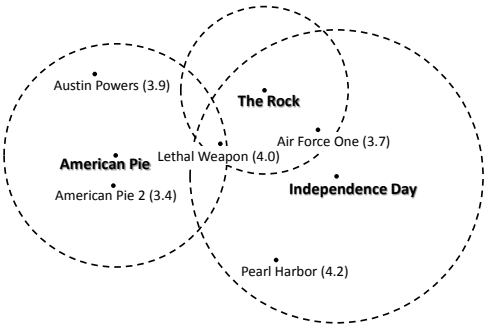
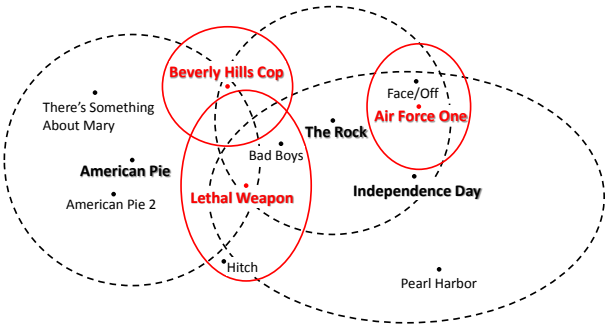


Figure 2: (a) Geometric Representation (partial)



(b) Geometric Representation with Joint Representatives (partial)

the circles (more of this to be followed). For (c), the *ratings* of the items, there is not (yet) direct relation to the geometric representation, and in general, the requirement is to choose items with the highest ratings. For (d), the *diversity* of the recommended items, the requirement is *not* to choose items too similar to each other. Finally, for (e), the *coverage* I_k of the seeds, the requirements is, broadly speaking, to choose items throughout the entire geometric representation, instead of focusing (or neglecting) a specific area.

3. GENERATING RECOMMENDATIONS

Our algorithm naturally blends all the above properties by interpreting the geometrical representation of the items. A key role in the algorithm are the *join representatives*, which we next explain.

3.1 Joint Representatives

Joint representatives are the items that captures best the mood reflected by multiple seeds. Intuitively, these are the items which resembles most to *all* the seeds of a specific subset. In the geometric representation, these are the items located in the intersection of multiple circles. Each such area corresponds to a subset of seeds that are the center of the overlapping circles. Intuitively, the items in each such area are similar (only) to all seeds of the corresponding subset. For example, in Figure 2 (a) American Pie 2 is indeed similar only to the seed American Pie, but Air Force One is similar to both The Rock and Independence Day.

For each subset of seed items, corresponding to the overlapping areas discussed above, we wish to find the item that "represents" it the most. To decide which item to choose between (possibly) several candidates (that is, ones that all are similar to all the subset of seeds), we consider the rating of the items as well. Formally, for a subset of seeds $S' \subseteq S$ we define the joint representative item to be the one that maximizes the value of the following equation:

$$JointRep(S') = \underset{i \in N(s) | s \in S'}{\arg\max} [\min_{s \in S'} [\frac{sim(i,s) * rate(i)}{avg_{j \in N(s)}(rate(j))}]]$$

The formula captures the following intuition: Intuitively, the joint representative is the item which resides in the center of the geometric area, with respect to the subset of seeds. To find this item, we search the candidate items (the ones in the corresponding neighborhoods) for the item whose minimal similarity among the subset of seeds is the maximal. To consider item ratings as well, we combine the similarity measure with the rating of each item and divide it by the mean rating of the neighborhood of the corresponding seeds.

In practice, we take an additional "filtering" step that avoids choosing joint representatives which are "too close" to any of the seeds. While this verification can be easily checked once a specific threshold is manually set, we take an alternative approach that avoids such arbitrary selection by considering the entire neighborhood of each of the original seeds. Specifically, we consider as "too close" items ones whose similarity with any of the seed items is above the mean similarity of that seed and its neighborhood items.

"Too far" items, on the other side, also needs to be addressed. When the seeds of a specific subsets are too far away from each other, no item can be close enough to provide meaningful representation. For instance, in the movie domain, there is no meaningful representative item for two seeds from totaly different domains such as Documentary and Fantasy. As in the opposite "too close" items scenario, we wish to avoid arbitrary threshold to define which seeds are too far (or alternatively which joint representative are not meaningful). We use instead a simple condition that requires $JointRep(S')$ to be a positive greater than 0. For scenarios were the value equal to 0, or below, the joint representative is undefined.

3.2 Enhanced Geometric Representation

The next step of the algorithm is enhancing the geometric representation. We start by augmenting the seed set with the joint representatives discussed above. Specifically, we go over all possible subsets of seeds (analogous to go over all intersections of circles) and for each subset we evaluate the corresponding joint representative item. Note that the same item may be the joint representatives of different subsets, and in this specific case, we only add it once.

We next update the basic similarity measure used before. We combine (multiply), as for the joint representatives, the similarity and the rating of the items, then normalize the results not only by the mean rating, but also by the mean similarity of the neighborhood of the corresponding seed item. This refined similarity measure between an item i and a seed s is given by:

$$Sim'(i,s) = \frac{sim(i,s) * rate(i)}{avg_{j \in N(s)}(sim(j,s)) * avg_{j \in N(s)}(rate(j))}$$

It will (i) better help our algorithm merge the neighborhoods on the next step, and (ii) will prevent biased towards highly rated items.

Figure 2 (b) depicts the updated plot which includes the joint representatives and the updated similarity measure. The joint representatives added by the algorithm are in red, and their neighborhoods are surrounded by solid circles (as opposed to the dashed ones surrounding the seeds). Note that these neighborhoods may include new items - that is, ones which were not members in any of the original neighborhoods. "Hitch" in this current scenario is such an example. Also note that the circle symmetry shown before in Figure 2 (a) is not preserved as the similarity measure is now blended with the rating. The ratings shown in (a) are thus omitted.

We finally briefly discuss the complexity of this part of the algorithm. Finding all the joint representatives requires traversing all possible subsets of seed sets which clearly impose an exponential overhead on the algorithm. In practice, however, the size of the seed set is small (around $|S| = 3$) and thus does not pose any restriction on the running time. Moreover, as we will later see in the experiments, the algorithm is extremely efficient even for larger sets of seeds. Additionally, we will consider an optimization that caches the neighborhoods of all items which will reduce the running time to (only) a few milliseconds regardless to the size of $|S|$.

3.3 Items Selection

The next step of the algorithm extracts from the enhanced geometric representation a weighted list of the top K most relevant items (based on all the properties discussed above). From within this list we will choose in the next step the set of item I_k to recommend the user. Note that we do not directly choose the set I_k in order to use the diversification mechanism based on the priority cover-trees (to be explained shortly). The current center of the circles, intuitively, describes best the users's current mood. We would thus like to choose the K items closest to all centers. We next define, for each item i in the geometric representation, a weight function $MoodRel(i)$ that reflects its overall relevance to the mood reflected by the input seeds. The weight function is given by:

$$MoodRel(i) = \#circles(i) * mSim + \max_{s \in S^+} Sim'(i,s)$$

where S^+ is the augmented seed set (with the joint representatives), $\#circles(i)$ is the number of circles i resides within the enhanced geometric representation and $mSim$ is a constant bigger than the maximal value of Sim'. Note that for the next stage of the algorithm, we are less interested in the exact values $MoodRel()$ returns, and instead, we utilize the *order* it projects. Thus, the function orders the items by the number of circles they resides within, and to break ties, it sorts the items by their distance to the seed / joint representative closest to them.

We finally note that the normalization made by the $Sim'()$ function is designed to provide a good balance (in terms of coverage) in the selection of the top-K items. Without it, the variation in both the $sim()$ and $rate()$ functions, which $Sim'()$ is built on, can lead to a biased list, where large sections of the geometric representation can remain without any "representatives" in the top-K list.

3.4 Diversification

The last step of the algorithm refines the top-K items, evaluated before, into the final set of recommendations I_k (of size k) that would be presented to the user. To get into perspective, the default value we chose for K was 50 and for k was 5, which is a standard selection that fits most of today's mobile screens. Note that simply choosing the top-k items from the top-K list may lead to an unattractive overly homogenous set of recommendations; for example, in the movie domain, a set all consisting of sequel movies. Clearly the user would prefer to be presented with a wider and more diverse subset of the highly rated items (possibly with an option to view more sequels via a click on a "more of that" button).

To achieve this, we employ the diversification mechanism from [5]. It is based on the notion of priority-medoids, which naturally clusters items based on both their distances from each other and by their assigned rating ($MoodRel()$ in our context). As it is an NP-Hard problem, an approximation is done by priority cover-trees [5].

4. EXPERIMENTS

We implemented the algorithms discussed above in the Mood4 system [4]. We used Mood4 in a variety of settings to evaluate the described algorithms by (i) empirical evaluation, (ii) a user study, and (iii) performance measures. We next give a summary of all results.

System settings. In all experiments, Mood4 was deployed on top of a CF-based recommender system (C2F [3]). For estimating item similarity, we used Pearson's correlation coefficient [12]. We install Mood4 on an Intel machine (Core i7-870), running on 2.93 GHz with 16GB memory and Win7 64Bit (we used only a single core and 2.5GB of RAM). As data, we used a real-life data set from the cinema domain, provided by Netflix [2], which contains over 100 million distinct raw movie ratings (such as 1 to 5 "stars"), by almost 500,000 users, and no semantic information on the movies.

Experiments settings. The first step of the experiments is the selection of the seed set S. Throughout the experiments we construct a seed set by choosing first a single item, which will act as the base of the set, and then enrich it with more items that will construct the final set. We tested both *random* and *popular* item selection as the base of the set and found similar trends for both approaches (thus, the results presented here are for random). The exact number of items added (to the base), and their distances are variables which will be set in the upcoming experiments (more on that below). Once we have evaluated a seed set S we used it as input to several algorithms (see below) in order to evaluate, for each algorithm, a set of items to recommend the user (I_k).

Algorithms. We denote by *Mood4* the main algorithm described in this paper. We first compared it to two baseline algorithms (i) *MaxRel* (Maximum Relevance) that returns the top k items with the highest $rate()$ values; The candidate items being all the items of the surrounding neighborhoods of the seeds, and (ii) *MaxSim* (Maximum Similarity) that, instead, selects the items with the highest $sim()$ values, as well as to a simpler variant of *Mood4*, denoted *Mood4^*, that avoids the use of the joint representatives. (The geometric representation is considered only with the original seeds).

We have also tested a few variations of the above algorithms, such as *MaxDiversity* [5], that diversifies the selected highest rated items, and variant that uses only a single seed example (as in Pandora) to capture the user's mood. As they were always outperformed by *Mood4^*, which may be viewed as a generalization of both, we omit their results here. We finally note that we did not compare our performance to algorithms from the Context-Aware Recommender Systems field as all algorithms we are aware of require additional auxiliary information (e.g. semantics, context, etc.) which is not available here. See Section 5 for further discussion.

4.1 Empirical Evaluation

We start by presenting the empirical measures used to evaluated the algorithms, and then present the different parameters tested. Then we present a summary of the results.

Measures. Following we briefly describe the different measures used to evaluate the results of the above algorithms. Each measure is the sum of a different function over the recommended items I_k.

Rating. The classical property of recommender system. The higher the (general) relevance of the recommended items the better. Formally, $Rating(I_k) = \sum_{i \in I_k} rate(i)$.

Diversity. Avoids monotonicity recommendations and generates a wider view of the available items. For simplicity, we used the same method as [5] and took the inverse value of the similarity. The higher the value, the more diverse are the items, and better the results. Formally, $Diversity(I_k) = \sum_{i,j \in I_k} (1 - sim(i,j))$.

Similarity (Max). Measures the similarity of the recommended items with the original seeds by choosing, for each item, the maximal similarity among all seeds. A higher value yields a more similar relation to a specific seed, and thus a better result. Formally, $Sim_{(max)}(I_k) = \sum_{i \in I_k} \max_{s \in S} sim(i,s)$.

Similarity (Average). An alternative measure of similarity that reflects the joint similarity of the recommended items to all seeds simultaneously. Specifically, the measure returns the average similarity among all seeds, and thus, a higher value also yields a better result. Formally, $Sim_{(avg)}(I_k) = \sum_{i \in I_k} avg_{s \in S} sim(i,s)$

Variance. Measures "how far" are the recommended items from the original seeds. Unlike previous measures, lower values here means tighter items and thus better results. Formally, $Var(I_k) = \sum_{i \in I_k} \sum_{s \in S} \frac{(sim(i,s) - sim(i,S))^2}{|S|}$ where $sim(i,S)$ is the average similarity between i and all seeds in S.

Parameters. We consider next the parameters tested. In each experiment, we highlight the effect of each one by fixing all others to a given default value while varying the parameter of interest.
Number of Seeds ($|S|$) - the number of seeds entered by the user.
Neighborhood Size (K) - the size of both the neighborhoods in the geometric representation, and of the final weighted list.
Seed distances - the distance (similarity) between the different seeds, and like the number of seeds, it depends on the user's input (e.g., in the movie domain, the user can enterer two comedies and an action feature, or instead, three comedies).

We note that we also tested a varying range of values for k (the number of recommended items). As it had no affect on all measures, we chose a common value $k = 5$ for all experiments.

Summary of the Results. Due to space constraints, we present the results only for (1), and give a brief summary for (2) and (3). The results are averages of 1000 different seed sets.

Number of Seeds (1). We start the experiment by evaluating the affect of the number of seeds ($|S|$) used. Note that in a real system setting this parameter is not fixed and depends solely on the user's input. We nevertheless tested a varying (reasonable) range of possible values, staring from $|S| = 2$ and up to $|S| = 6$.

We start with the rating measure depicted in Figure 3 (a). *MaxRel* is ranked the highest, not surprisingly, as it focuses solely on the rating property, and is naturally increases with the size of $|S|$. *MaxSim* however, which is focused solely on the similarity, is clearly ranked the lowest. Both *Mood4* and *Mood4ˆ* are ranked between the two baselines as they do not focus on a single property. Between them, we can see that the use of the joint representatives has a small (negative) affect on the rating measure (we will later see the positive affect on the average similarity). This is also the case for the diversity measure depicted in Figure 3 (b); Algorithm *Mood4ˆ* achieves a higher value than *Mood4* as $|S|$ increases. Essentially, this means that the items in the set I_k evaluated by *Mood4* are closer to each other than the items generated by *Mood4ˆ*. It is expected as the joint representatives used by *Mood4* are designed to help the algorithm to discover items that are similar to several seeds together, and thus, the final recommended items will naturally be more related to each other than the items who are similar only to a specific seed. This is also the reason why the diversity measure slightly decrease for *Mood4* with the increase of $|S|$. Nevertheless, *Mood4* still achieves a higher diversity measure compared to *MaxRel* and *MaxSim*.

We continue with the similarity (max) measure depicted in Figure 3 (c). Broadly speaking, the results were not too affected by the size of $|S|$, as expected, as we focus here on the *single* seed similarity. In terms of ranking, *MaxSim* is clearly at the top, followed by *Mood4ˆ* which focuses, as *MaxSim*, on the similarity of a single seed. *Mood4*, which focuses on the joint similarity, is ranked next before the last *MaxRel* which does not consider the similarity at all. For the similarity (average) measure, depicted in Figure 3 (d), the results were entirely different: except for low values of $|S|$, *Mood4* achieves the best results among all algorithms. (The order among the rest of the algorithms remains the same). This is surprising as we would expect *Mood4* to outperform *MaxSim* even for small values of $|S|$. We found that this discrepancy is due to the *variance* of the items. Intuitively, a large similarity (average) value can be evaluated when (only) a single item holds a high value of similarity. For example, the similarity (average) value of the similarities $\{0.8, 0.2, 0.2\}$ is higher than of $\{0.4, 0.3, 0.3\}$ ($0.4 > 0.333$). It comes on the expense of a much higher variance ($0.08 > 0.018$ in this example). Figure 3 (e) depicts the variance of the similarity for the varying $|S|$. Unlike all previous figures, lower measures yield here better results. We can see that the variance is indeed much lower for *Mood4* than for *MaxSim*, even for low values of $|S|$. We thus conclude that *Mood4* achieves the best joint seed similarity.
Neighborhood size (2). We tested a range of neighborhood sizes from $K = 5$ up to $K = 200$. Note that this is the only parameter that needs to be fixed by our algorithm, and thus, the goal of the experiments was to find the optimal value for it. Intuitively, a value too low would restrict the algorithm to use items too similar to the original seeds, while a value too high may consider irrelevant items (that is, ones that are not related to the seeds). We observed that $K = 50$ provided a good balance between all measures. Interestingly, we manually (subjectively) tuned the algorithm prior to these empirical experiments and chose the same value.
Seed distances (3). Like the number of seeds, this parameter is defined in a real world setting solely by the user (that is, it is derives by the items she chooses as seeds). For the experiments, once the based seed was selected, we ordered the all items by their similarity to it, and then selected the i'th and the $i * 2$'th elements as the second and third seeds for a varying i (from $i = 1$ to $i = 100$). Interestingly, we found that *all* algorithms were not too affected by the different distances, except for the average similarity (slightly decreased with the increase of the distances) which was expected. We conclude that this parameter has no crucial affect on the algorithms, and thus, chose a balanced value ($i = 50$) as default value.

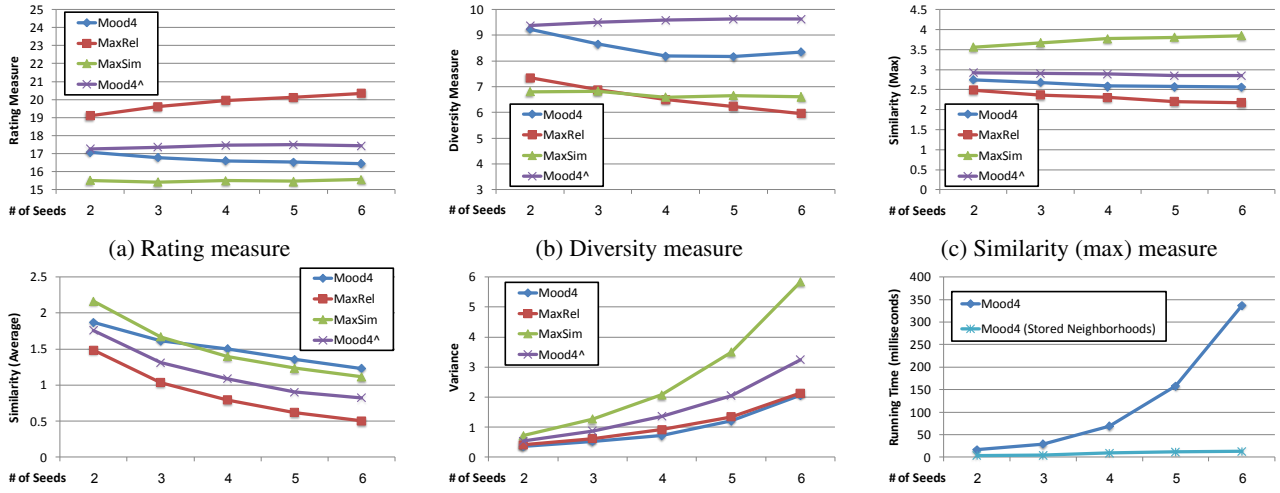
4.2 User Study

The ideal way to assess the real-world quality of the algorithms is by measuring the *click-through rate* (CTR) it derives in an actual deployed system. As we do not possess such a luxury setting, we took a alternative approach and asked volunteers to compare and assess themselves the outputs of the tested algorithms. Note that a subjective assessment of recommendations is not a trivial task, as the volunteer needs to be familiar with all the items the algorithms output in order to estimate how well they match her current desire. (In our context, to watch the suggested movies).

Our initial survey included 40 volunteers. Specifically, each one was asked to choose three items (movies), indicating their current mood, and then compare three sets of recommendations generated in response by the *Mood4*, *Mood4ˆ* and *MaxSim* algorithms. (The volunteers were able slip the output and repeat the process if recommended unfamiliar movies whose relevance they could not asses). Then, in the course of the Mood4 demo in [4], 30 more users where asked to to assess the algorithms quality, in a similar setting. The trends in the two experiment were similar: The majority of users (70%) preferred the recommendations of *Mood4*. Algorithm *Mood4ˆ* came second (20%) while *MaxSim* came last (10%).

4.3 Performance

To conclude we show that our algorithm operates fast enough to support real life scenarios of a working system. Figure 3 (f) depicts the running time of Mood4 with the defaults settings for a varying number of seeds $|S|$. The results are an average of 1000 different runs, and are presented in milliseconds. For the common value of $|S| = 3$ we can see that the running time is around 25 milliseconds. When $|S|$ increases, the running time naturally increases but not as dramatically as expected (evaluating the joint representatives requires exponential behavior). We further examined the specific running times of each part of the algorithm and found that the majority of the time was spent on the evaluation of the item neighborhoods. We thus implemented an additional module to Mood4 that caches the neighborhoods, once evaluated, in memory for fast future use. We refer the tweaked implementation by *Mood4 (Stored Neighborhood)* and its results are also depicted in Figure 3 (f). As we can see, the running time drops dramatically from the non optimized version of *Mood4* and does not exceed 15



(a) Rating measure (b) Diversity measure (c) Similarity (max) measure

(d) Similarity (avg) measure (e) Variance measure (f) Running time

Figure 3: Results for varying number of seeds ($|S|$)

millisecond even for the $|S| = 6$. Note that for the Netflix data set used in here we only needed less than 3.5 MB additional memory.

5. RELATED WORK AND CONCLUSION

A key objective of recent research on recommender systems is to improve the ratings estimation and thereby the overall recommendations [16, 9]. Recently, there has been extensive research in the context of social networks. Examples include [6] that allows users to tweak the underlying algorithms to better balance the information derived from different networks, [17] that generates multi-modal graphs to describe multi-network information, and [14] that uses a latent factor model to process such data. All these works leverage (external) social information. They are complimentary to ours and can be used to enhance our basic CF model of ratings.

Context awareness has been recently addressed by a subclass of recommender systems called CARS (Context-Aware Recommender Systems) [10, 7, 15]. Here, recommendations are generated not only based on the user-item rating matrix, but also using additional dimensions which represents contextual information such as location, domain, semantics, etc. For example, [10] exploits spatial, social and temporal (context) information in order to better capture the user profiles in the context of photo recommendations, [7] used POI (points of interests) in order to understand the user's intentions, and [15] devised a mood specific movie similarity measure in order to provide, like us, recommendations that fits your current mood. All these works leverage the context of the user by exploiting some (additional) external information (location - [10, 7], social - [10] and semantics [15]). In contrast, our work extract the user's context without requiring any information but the local items data set. If such additional information is present, we can incorporate the above methods into our underlying rating prediction mechanism to improve our mood capturing recommendations.

Most relevant to our work is the problem of group-recommendations [1, 13]. While one may view each group member as an "example", the algorithms from [1, 13] are not suitable here as they (i) focus on reducing the group disagreement (e.g. maximizing the similarity to all the seeds), not incorporating the other properties discussed above [1], and (ii) they require the use of semantics (genres) [13]. Bootstrapping is also related, and its often addressed by methods for choosing the items needed to be rated by new users which maximize the prediction gain [8]. A difficulty is that a user may be asked to rate unfamiliar items. In contrast, our algorithm generates recommendations instantly upon entering familiar items.

From the industry, Pandora [11] is a good example for a system that generates recommendations by asking the user to enter a sample song (or an artist) she likes, generating in response a playlist of similar tunes. Unlike Mood4 , Pandora only focuses on a single example at a time (even with its new "variety" option), and does not try to find overlapping features as Mood4 .

In summary, we presented in this paper a novel method for generating recommendations based on the user's current mood. The recommendations are generated based on a geometric representation model, formed using the user's input examples, extended with the use of joint representatives. We showed that our effective algorithm operates extremely quickly in practice and is feasible for a real time large scale system. Finally, we note that while our solution does not require semantic data on the items, leveraging such data (when available) may be an intriguing direction for future work.

6. REFERENCES

[1] S. Amer-Yahia, S. B. Roy, A. Chawla, G. Das, and C. Yu. Group recommendation: Semantics and efficiency. *PVLDB*, 2009.

[2] J. Bennet and S. Lanning. The netflix prize. *KDD Cup*, 2007.

[3] R. Boim, H. Kaplan, T. Milo, and R. Rubinfeld. Improved recommendations via (more) collaboration. *WebDB*, 2010.

[4] R. Boim and T. Milo. In the mood4: Recommendation by examples. *EDBT*, 2013.

[5] R. Boim, T. Milo, and S. Novgorodov. Diversification and refinement in collaborative filtering recommender. *CIKM*, 2011.

[6] S. Bostandjiev, J. O'Donovan, and T. Hollerer. Tasteweights: A visual interactive hybrid recommender system. *RECSYS*, 2012.

[7] H. Costa, B. Furtado, D. Pires, L. Macedo, and A. Cardoso. Context and intention-awareness in pois recommender systems. *CARS*, 2012.

[8] N. Golbandi, Y. Koren, and R. Lempel. Adaptive bootstrapping of recommender systems using decision trees. *WSDM*, 2011.

[9] Y. Koren and J. Sill. Ordrec: an ordinal model for predicting personalized item rating distributions. *RECSYS*, 2011.

[10] F. D. A. Lemos, R. A. F. Carmo, W. Viana, and R. M. C. Andrade. Towards a context-aware photo recommender system. *CARS*, 2012.

[11] Pandora. Pandora internet radio.

[12] J. L. Rodgers and W. A. Nicewander. Thirteen ways to look at the correlation coefficient. *The American Statistician*, 1988.

[13] S. Seko, T. Yagi, M. Motegi, and S. Muto. Group recommendation using feature space representing behavioral tendency and power balance among members. *RECSYS*, 2011.

[14] Y. Shen and R. Jin. Learning personal+social latent factor model for social recommendation. *KDD*, 2012.

[15] Y. Shi, M. Larson, and A. Hanjalic. Mining mood-specific movie similarity with matrix factorization for context-aware recommendation. In *CAMRa*, 2010.

[16] X. Su and T. Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, 2009.

[17] P. Symeonidis, E. Tiakas, and Y. Manolopoulos. Product recommendation and rating prediction based on multi-modal social networks. *RECSYS*, 2011.