Transparent Approaches to Recommender Systems for C3I

James Schaffer
US Army Research Laboratory (West)
Los Angeles, California
james.a.schaffer20.civ@mail.mil

ABSTRACT

The capability to make quick, effective decisions under uncertainty is critical for Army mission success. One of the most promising ways to deliver and enhance this capability is through recommender systems, which predict needs in advance to prioritize networked information delivery. While many promising recommender methods exist that might fulfill this need, it is unclear how recommendations should be presented to C3I analysts. This is because recommender systems are typically studied in low-risk, high bandwidth domains, which has contributed to an increased focus on maximizing subjective user satisfaction (rather than objective success) while relying on copious amounts of global data. This is in contrast to C3I, which is often practiced under high-risk, low bandwidth (tactical networking) conditions. Since each decision made under such conditions can affect mission success and survival rate, analysts accordingly require a higher level of transparency and provenance for information systems. Moreover, since global uplinks are not always available, C3I analysts require the ability to process information locally in both time and space. Here, we describe two highly transparent, domain-oriented recommendation methodologies to support the needs of C3I analysts, depending on whether or not recommendations need to be calculated locally. Each method demonstrates its theoretical effectiveness on a popular recommender systems benchmark, but further evaluation is needed for the C3I domain.

KEYWORDS

Human-computer Interaction, Recommender Systems, Explanation, Transparency, User Modeling, Personalization, Multi-Attribute Utility, Decision Support Systems

1 INTRODUCTION

The amount of information available to C3I operators will continue to increase due to advances in pervasive sensing, which creates a concern about how to leverage this rich information without overloading the cognitive abilities of operators [12]. A potential solution for this problem lies in intelligent information systems, such as recommender systems, that automatically filter and deliver relevant information. Despite this promise, university and commercial research on recommender systems typically violate the assumptions of tactical network conditions - sparse data, low bandwidth, intermittent connectivity, adversarial intervention, and high consequences for poor decisions. To meet the needs of C3I, recommender systems need to be adapted to be less complex, more transparent, more robust, and more local.

High performance recommender systems are comprised of complex algorithms, which are difficult or impossible to understand for most users. Collaborative filtering (CF) approaches, which have demonstrated superior predictive power, have several issues that make them unsuitable for tactical networking: cold-start limitations, reliance on dense global preference data, and low inherent transparency. While these systems would be quite effective in low-risk or commercial applications, decision makers in high-risk military applications may struggle with placing their confidence in these types of systems, which can be unpredictable in their behavior. This is for three reasons: 1) the underlying models of CF algorithms are ignorant of domain constraints and are limited in their ability to keep the human in-the-loop, 2) humans, who frequently overestimate their own abilities while downplaying the abilities of others [22], may ignore CF recommendations when flaws are perceived, and 3) automation bias [9] may cause poor decisions if over-trust develops. This creates a catch-22 for AI design, and has contributed to the rising popularity of the notion of explainable AI [14, 30].

To address trust concerns [28], recommender systems researchers have proposed various ways to improve the transparency of their systems, such as providing justifications or explanations to users. "Explanation research" has been furthered in both expert and recommender systems [27], and approaches usually detail a "trace of logic" of that system's operation. For instance, case-based reasoning systems may present a multi-point argument for why a particular course of action is recommended [1]. Collaborative filtering recommender systems may reveal part of the social network structure that was used to generate the recommendation [16]. Typically, these explanations have been designed to maximize system-specific trust [24], with little regard for over-trusting [17] or automation bias. Simultaneously, many explanations from recommender systems sometimes do not provide users with a mechanism to provide direct feedback to system.

In a further complication, many explanations are dependent on the properties of the algorithms they explain. For instance, a collaborative filtering explanation (e.g., the recommended item is liked by friends of friends), would not work for a decision tree. One consequence of this is that empirical evaluations of different explanation styles must make compromises when comparing across algorithms. This makes it difficult to pin down a "best" explanation method - since such a method may not even be compatible with future algorithms. This difficulty was identified by the Nunes et al. [27] survey, which calls for more reproducibility in explanation evaluation via divorcing algorithmic reasoning from its explanation. In this research, we refer to these as *universal* explanations, since they are compatible across algorithms.

The above issues highlight that intelligent information systems are too complex to be suitable for C3I, that their explanations do little to remedy problems caused by their complexity, and resorting to simple information systems would compromise their accuracy. Is this a catch-22 situation? How can we design simple, transparent interfaces for recommender systems that do not compromise accuracy or universality? We hypothesized that the answer may

lie in research on human decision making. Before the rise in popularity of recommender systems, decision technology culminated in the use of multi-attribute utility (MAU) [10]. MAU's influence can be seen on almost every e-commerce website and has inspired many content-based recommender systems [2, 18] MAU has also gained traction in tactical networking research - interviews with C3I operators yielded a preference for the interface [32]. This may be because MAU is simple in concept, locally computed, and easy to explain.

To answer our research question, this paper details two methods for building MAU interfaces for recommender systems while accomodating tactical networking constraints. The core inspiration of each approach is to use complex learning algorithms to configure simple models of users that make domain-oriented predictions. In this case, the simple model is the MAU form, which generates an accordingly simple interface. The interface is thus "initially configured" by the learning method, which can be overridden by the user. This removes the need for the C3I analyst to perfectly understand the learning algorithm, which may be impossible in some cases (e.g., neural networks [25] or matrix factorization [19]). The next section details the first method, which details heuristics for building accurate MAU models based on a limited amount of user preferences. In the following section we detail the second method, which uses a CF algorithm as a starting point to boost the user model, sacrificing some locality and speed for improved accuracy. We also formalize the idea of "advice translation," a way of thinking about how to automatically generate universal explanations for recommender systems, and more broadly, intelligent information systems.

2 METHOD 1: MAUSVR

This section presents the first method, MAUSVR, which learns user preferences directly from previously consumed or liked items. An key observation necessary for this approach is that support-vector regression (SVR) with a linear kernel produces a model that is a weighted linear sum of products of domain features – essentially a MAU model (MAUM). However, building MAUM using naive SVR has fairly poor performance on the ranking problem when compared with CF techniques and even does worse than ranking by the mean average rating. We present a few heuristic modifications to SVR that can produce more effective MAUM. The resulting technique, which we call MAUSVR, can instantaneously learn a user's preferences for attributes and thereby provide recommendations. Although preference/rating data in the C3I domain is not yet available, this initial study uses a sample of the MovieLens 20M dataset consisting of 5 million ratings was used to assess ranking quality, robustness to sparsity, and speed. The evaluation presented here indicates that MAUSVR would be suitable for tactical networks. Moreover, the resulting method is competitive with CF in terms of accuracy, and as long as domain features are provided, remains as domain-independent as SVR itself. MAUSVR was able to be built instantaneously ($< 100ms$) for more than 75% of the evaluated users with an off-the shelf Java implementation of SMOreg. These findings indicate promise for the use of MAUSVR in real-time decision support systems operating in sparse data conditions.

2.1 Description of the Method

MAUSVR aims to learn the MAUM of an individual user's preferences through support-vector regression (SVR). A MAUM can be described as:

$$f(x) = \overline{\omega} \cdot \phi(x) + b \tag{1}$$

where $\phi(x)$ returns an array that are the attribute values of item x. This results in simply a linear sum of weighted attributes. Here, an SMOreg implementation is used [31] to build MAUM, however, some modifications are needed to achieve high quality of item ranking. The modifications are based on two observations. First, any number of MAUM can be linearly combined into a single model that still matches the form of an MAUM, such that the weights ($\overline{w}$) are the weighted sums of the ensemble weights and the bias (b) is the weighted sum of the ensemble biases. Second, the mean item rating approach, which is relatively stable at all sample sizes, is surprisingly good at the ranking problem. In this section we describe the three modifications, which each have associated parameters. In each case, a greedy linear sampling strategy was used on the MovieLens dataset to determine the ideal value for the parameter.

First, a bagging approach [5] was utilized to improve both the ranking quality and speed of SMOreg. The time complexity of support vector regression is also fairly bad: $O(max(n,d)min(n,d)^2)$ [8], so limiting the maximum size of any bag greatly improves the scalability of the approach. The number of learners in the ensemble I was determined with the following function of a user's profile size, $|P|$:

$$I = 10 + min\left(\frac{|P|^2}{B_{max}^2}, 200\right)\text{,} \tag{2}$$

To minimize the worst case build time of the ensemble, our evaluation suggests that the maximum number of iterations should be limited to 210 and the maximum bag size, B_{max}, should be set to 100.

Although tuning experiments indicated that accuracy slightly increases as I increases past 210, MAUSVR will also become less suited to being used as an interactive system, as update times will become longer. In our tuning of MAUSVR, capping the value of I at 210 sacrifices very little ranking quality while still having an average/median build time that is instantaneous ($< 100ms$) and a worst case build time that can still keep the user's attention (about 3 seconds) [26].

Second, mean ratings should be used to boost the performance of MAUSVR. Fortunately, mean ratings do not require large amounts of global data to get good estimates: assuming a standard deviation of 1.0 stars, it only takes 50 samples to estimate a mean rating to within about a quarter of a star (this also means that mean ratings do not have to be updated frequently). In MAUSVR, this "baseline" MAUM (M_b) can be blended with the trained ensemble model (M_e) to produce the rating prediction of item r_i by quantifying the confidence α of the latter, as follows:

$$r_i = \alpha M_e(i) + (1 - \alpha) M_b(i) \tag{3}$$

Since the predictive power of the ensemble model increases only as the user's profile size increases, quantifying the confidence is

relatively straightforward. The following function, wherein β is a tuning parameter, of the user's profile size $|P|$ was used:

$$\alpha = \frac{log(|P|)}{\beta} \tag{4}$$

Third, SVR needs a way to deal with discrete attributes. One way to handle discrete-valued attributes $\mathcal{A} \in \{a_1, a_2, ...a_n\}$ is by creating n new numeric attributes $a_i \in \{0, 1\}$, however, this can quickly lead to an overwhelming number of attributes for a human decision maker to consider. This problem is exacerbated when the attribute can take multiple values (e.g. movie genre and cast). Having excessive attributes with little mutual information can also hurt the performance of SMOreg. For instance, a single movie can have hundreds of cast members, perhaps few of which could be found elsewhere in a user's profile. For this reason the total number of discrete attributes was pruned down to a reasonable subset. In the implementation of MAUSVR, a cutoff was used to remove attribute columns that had fewer than ϵ instances. ϵ was determined with the following function, which has one tuning parameter γ affecting the cutoff:

$$\epsilon = \gamma \frac{|P|}{n} \tag{5}$$

Based on our experimentation, we recommend MAUSVR's β and γ parameters to be fixed to 6.0 and 75.0 respectively. Additionally, the "slack" variable C in each SVR has to be set. We found that the ensemble method was not particularly sensitive to C, however, smaller values of C result in faster build times. Thus, we recommend setting C to 0.1.

2.2 Evaluation

MAUSVR was evaluated based on its performance on the ranking problem in comparison with CF (similar to [20]). The MovieLens 20M dataset was used for evaluation. As of this writing, CF techniques (including matrix factorization) still achieve the highest accuracy scores on this dataset. The first 5 million of the 20 million ratings were selected (the ordering of MovieLens is randomized) and the remaining were discarded to decrease the time needed for evaluation. For the remaining 5 million ratings, some of the rated items had missing attribute values and were discarded. Then, the dataset was pruned one final time by removing users that had fewer than 20 ratings, resulting a dataset with just under 4.8 million ratings, which we will call ML5M. For each user in ML5M, the 20% most recent ratings provided by that user were set aside as the testing set. The remaining 80% of ratings were used as training data. 12 experiments were run, with each experiment randomly discarding up to 99% of the training data for each user in the dataset, resulting in the 12 different densities shown on the x-axis of Figure 1. One additional dataset was created where all but 10 item ratings for each user were removed, which we refer to as the "10-item" dataset. This was intended to mimic situations where ratings are being elicited to "cold-start" a recommendation algorithm.

MAUSVR was compared to CF algorithms in the popular LensKit API [11]: Item-Item CF (IICF), FunkSVD (FUNK), and Slope-One (SLOPE1). Additionally, all algorithms were compared to the mean rating approach (AVG RATING). User-User CF was considered, but was not included in the final experiment due to poor performance and long training times relative to IICF.

The tuning parameters in the CF approaches were maximized using a greedy linear sampling strategy: FUNK was run with 25 hidden features and the deviation damping of SLOPE1 was set to 1.0. It is important to note that the mean item rating approach was set as the baseline for all CF algorithms, that is, the average rating is used when the CF similarity matrix does not connect a user to a specific item. In this evaluation, we focus on the ranking problem, since prediction accuracy is irrelevant to decision-making domains (for instance, it is not useful to predict very accurately that a piece of information is not irrelevant). The median number of items in a profile's test set was 15, so NDCG@5 is used.

We evaluate MAUSVR's build times with objective standards of interactiveness [26] rather than comparatively with CF. This was for three reasons, which relate to accuracy, comparative fairness, and relevance: first, implementation details can severely impact build times; second, cold builds of the CF algorithms in Lenskit can take upwards of fifteen minutes; and third, MAUSVR is targeted for interactive systems and thus instantaneous responses ($< 100ms$) are desired. It might be possible to modify the Lenskit implementations of CF to update interactively (for instance, in user-user CF, a user could theoretically get updated recommendations by only calculating a single row in the matrix), however, it is unknown if instantaneous timings could be achieved and thus it is outside the scope of this work.

Movie content for use in the MAUSVR training process was pulled from the TMDb API[1]. Features analyzed were: runtime, revenue, genre, language, collection (e.g. "Star Wars," "Marvel"), director, lead actor, cast (multiple categorical), and user-generated tags (multiple categorical). For each categorical and multiple categorical feature, SMOreg will add additional columns for each discrete value that the feature takes. "Cast" and "Tags" were pruned according to Equation 5. When the learned ensemble M_e is blended with the baseline M_b, the "mean rating" attribute is added to the final MAUM. The final weight of this attribute is determined solely by Equations 3 and 4.

2.2.1 Ranking Quality. A comparison of ranking quality (NDCG@5) gain plotted against data density is shown in Figure 1. NDCG@5 gain is the difference in NDCG@5 between the algorithm and the result obtained from randomly shuffling the test set. Data density is the inversion of sparsity, or the number of ratings available over the size of the user/item matrix. MAUSVR performs best at the ranking problem until data density reaches 0.30%. MAUSVR performs 47% better than FUNK and 37% better than AVG RATING at 0.04% data density. However, MAUSVR performs 16% worse than FUNK at 0.73% density. At no point does MAUSVR drop below the performance of AVG RATING, which makes it unique amongst the algorithms evaluated.

A comparison of NDCG@5 on the 10-item rating dataset is shown in Figure 2. MAUSVR is the only algorithm that can beat the AVG RATING approach, which it does by 22%.

2.2.2 Build Speed. Figure 3 illustrates the relationship between profile size and build time. The average profile size was 147 ratings

[1] https://www.themoviedb.org/

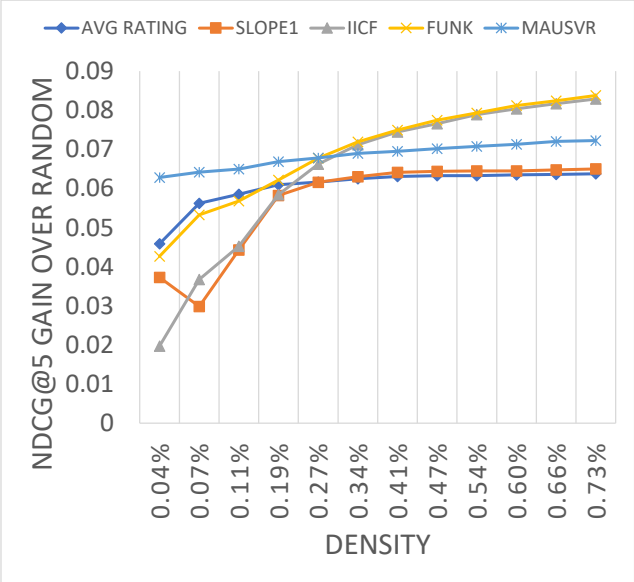

Figure 1: NDCG@5 gain over random shuffling for each algorithm that was evaluated, for the various densities that were sampled. MAUSVR has superior NDCG@5 under sparse conditions, which are common in other recommendation contexts and predicted to affect the C4ISR domain.

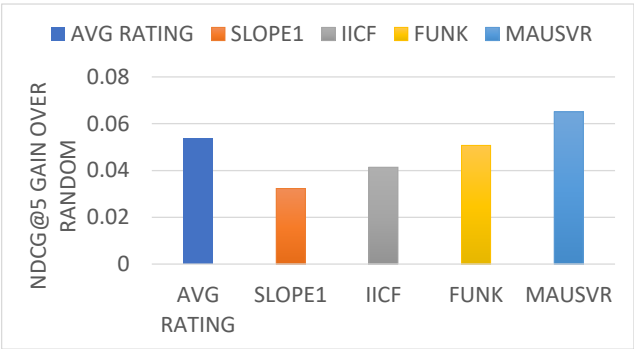

Figure 2: NDCG@5 gain over random shuffling for each algorithm that was evaluated, for the 10-item dataset.

with an average build time of 75 milliseconds. The median profile size was 71 ratings with a build time of 4 milliseconds. The third quartile build time was 74 milliseconds with a profile size of 160. The max build time was 3.2 seconds for a profile size of 2993.

Although the time complexity of an SVR is $O(max(n,d)min(n,d)^2)$ [8], the size of any SVR problem n in MAUSVR was fixed to be at most 100, and the total number of learners in the ensemble is limited to 210. This means that the time complexity of MAUSVR is simply $O(d^3)$.

3 METHOD 2: ADVICE TRANSLATION

Here we introduce the concept of advice translation – an approach to automatically generating universal explanations for recommender

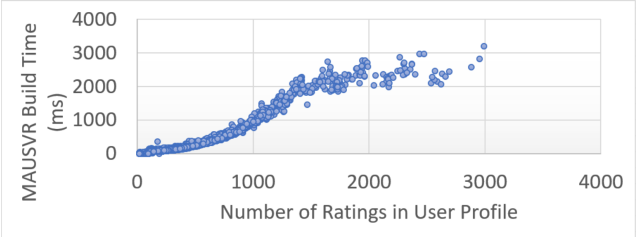

Figure 3: The build time (milliseconds) for each user profile, plotted against the number of ratings. For users in the first three quartiles the build time was instantaneous (< 74*ms*).

systems. The explanations built using this method rationalize the output of algorithms into domain terminology through a proxy model. This is in contrast with existing perspectives and methodologies that seek to explain the way an algorithm works or that merely highlight features of recommended items. Herein we demonstrate one method of advice translation, which uses support-vector regression to generate a MAU interface for matrix factorization provided only a domain ontology and the list of predictions. Similar to the first method, this allows a user to visualize what the matrix factorization model has learned in domain terms and provide interactive feedback, however, this approach can take better advantage of global data relationships by piggybacking on CF.

This section assesses the feasibility of the proposed advice translation approach. The challenge is to specify a technique for building the proxy model (with a linear internal representation) such that it does not mis-translate what MF has learned about the user. In other words, the proxy model should order the list of all potential items in a way identical to MF. A quantitative analysis of the proposed approach is given showing that it can achieve NDCG@{5,10} [34] within 1% of the original MF model, which we believe makes it a fair representation.

3.1 Description of the Method

In this section we describe the advice translation approach as a high level concept, then we give the details on how this approach was implemented to explain an MF algorithm through an MAU interface.

3.2 Overview of the Approach

Figure 4 shows an overview of the advice translation approach to explain the output of recommender systems. Conceivably, this approach could be used for any body of predictions (classed or numeric), but here we limit the scope to recommender systems, which essentially serve to re-rank databases of items according to a learned user model. The upper (blue) part of Figure 4 shows the typical process of recommendation. In the absence of the advice translation process, the predictions (in orange) would simply be presented to the user. What we propose is to instead build a second, simpler user model that represents the original model in more familiar domain terms. Specifically, the representation is parameterized by real concepts from the domain and the predictions are used to learn what these parameters should be to minimize the difference in item ranking. This can be accomplished by treating the list of

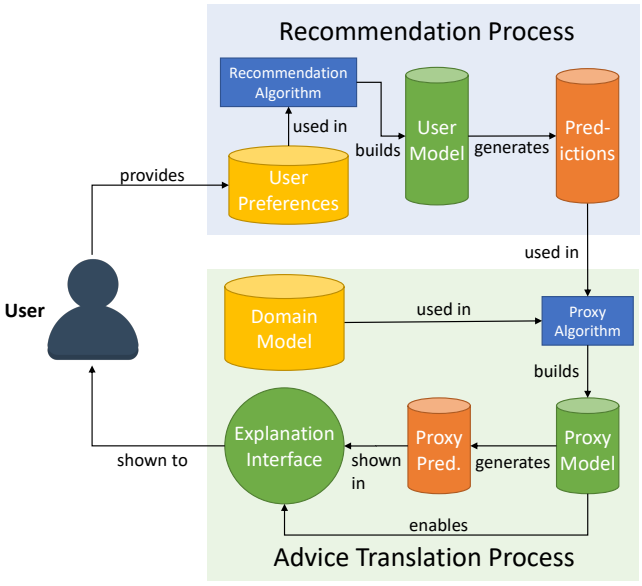

Figure 4: Overview of the advice translation process, showing the typical recommendation process (blue) and the proposed explanation building method (green).

predictions from the first model as a machine learning problem, where the class of each instance is the prediction and the features of each instance are domain attributes. For instance, a decision tree can be built that describes instances of restaurants in terms of the style of food, the price, popularity, and so on. A representation of the proxy model and its parameters are then presented to the user, who can provide feedback and tweak the user-friendly proxy.

The advice translation approach is similar to hybrid recommendation [6], but there are two fundamental differences. The first is that the proxy algorithm is not involved in the prediction process. This is because it should be only used to present the predictions. Unclassified instances or database items can be passed to the original algorithm/user model and subsequently a new proxy should be built for each session. This also implies the proxy model should be re-built when more profile information is obtained from the user. Second, the purpose of hybridizing recommendation strategies is to maximize the predictive accuracy of such algorithms at the cost of simplicity, whereas the purpose of advice translation is to maximize transparency and trust at the potential cost of predictive accuracy, in line with Gunning et al. [14].

The primary challenge of advice translation is to prevent distortion of the ranking generated by the original recommendation algorithm. This can be caused due to differences in the degrees of freedom between the original and proxy models. Advice translation, like any other content-based explanation, also requires a domain model to be known, and for each instance or database item to be annotated.

3.3 An Example of Advice Translation: Using SVR as a Proxy of MF

To demonstrate the feasibility of advice translation, here we create an MAU interface through support vector regression (SVR) to explain a matrix factorization (MF) model. Conceivably, the SVR methodology described here should work for any ranking algorithm and any domain, as long as the prediction values and instance attributes can be provided. This approach works essentially because SVR with a linear kernel has the same form (sum of weighted attributes) as an MAU process for determing the relevance r of an item x in the database of alternatives (Formula 1). Once the weights are learned, these can be used as the default weights in an MAU interface, which can then be inspected and further tweaked by users.

An ideal proxy for an MF model learned for a specific user is the set of weights $\overline{\omega}$ that eliminates differences in rank ordering (here, we use NDCG@k) when the utility function $r(x)$ is computed for all items $\overline{x}$ and the items are sorted in descending order by $r(x)$. When the NDCG is 1.0, the weights of the MAU are the ones that exactly replicate the ordering of the original MF algorithm.

We used the Weka [35] implementation of SVR as a proxy for the Lenskit [11] implementation of FunkSVD (an efficient implementation of MF). We conducted an offline experiment involving 954 MovieLens [15] users wherein the NDCG of FunkSVD and the proxy model built by SVR was compared. The experiment process was similar to Figure 4, except we also set aside each user's most recent ratings as a test set. We evaluate the NDCG of the proxy to recreate the list of predictions generated by FunkSVD and the NDCG of the proxy to order each user's training data. As further validation, we compare the performance of FunkSVD to the proxy on the test set, although this situation falls outside the practice of advice translation. RMSEA and MAE (see [7]) were considered but ultimately not included, since SVR may distort the scale of each prediction. This would be inconsequential for an MAU interface, since the idea of utility is unit-less [23].

The FunkSVD model was built using the same sample of 5 million random ratings from MovieLens, similar to the evaluation for the first method. 20% of the most recent ratings were set aside as a test set. Then, a list of predictions was generated for each user in the sample. Again, the predictions were annotated with the same set of attributes in the first evaluation (pulled from the TMDb API). Then, each user was treated as an individual machine learning problem by our custom Weka code, which built an SVR on each set of annotated predictions using the TMDb data as features and the FunkSVD prediction as the class value.

A principal challenge in training SVR on the FunkSVD prediction data was poor scalability [31]. In our experimentation, training set sizes of greater than 2K instances quickly become too time-intensive to be practical for interactive interfaces, however, the list of predictions generated for each user by FunkSVD was about 16K. To get around this limitation, we experimented with several sampling strategies and built multiple learners, each with a fraction of the available prediction data. Due to the unique constraints on the advice translation problem, it is possible to select the best learner from the ensemble by testing performance on the training data

Sampling Strategy	Description
RANDOM-N	N learners of training set size 1.6K are created. The list of 16K predictions is broken up into 1.6K lists of 10 predictions and a random instance is chosen from each list. We evaluate RANDOM-N with $N = 1$, $N = 5$, $N = 10$, and $N = 15$.
SPREAD-N	The list of 16K predictions is broken up into N lists by omitting instances that are not a modulo of N. For instance $N = 10$, the first list contains the 1st, 11th, 21st, and so on items. This effectively uses every single prediction in the dataset. We evaluate \textit{SPREAD-N} with $N = 8$, $N = 10$, $N = 12$, and $N = 15$.

Table 1: Sampling strategies used to evaluate the advice translation approach.

Method	*NDCG@5*	*NDCG@10*
FunkSVD	0.9038	0.9230
Proxy	0.8971	0.9138
MAUSVR	0.8943	

Table 2: Performance of FunkSVD and its SVR proxy on the test set. MAUSVR is included for comparison - the proxy method makes a slight gain over MAUSVR (about a 40% improvement when adjusting for the baseline).

after the build step. Note that this step is only valid because overfitting is not an issue. If the relevance of new instances needs to be predicted, FunkSVD can perform the predictions and a new proxy can be built. We tested the sampling strategies shown in Table 1.

3.4 Results

We tested each sampling strategy and measured precision@k and NDCG@k. NDCG@5,10,100 are used since lower listings are less relevant. Moreover, as k grows, NDCG asymptotically approaches 1.0, so this information is less useful in evaluation. For precision, we considered any prediction value greater than 5.0 a hit, and anything less a miss. This is because FunkSVD generates many predictions of > 5.0 before undergoing typical normalization steps. Results are shown in Figure 1. The random strategies performed almost uniformly worse than the spread strategies, with SPREAD-12 performing the best. As can be seen, Precision@k and NDCG@k are strongly correlated. Moreover, the NDCG@100 vs the NDCG@5 results indicate every sampling strategy would become more slightly more confused as the number of predictions shown increases. The Precision@5 results indicate that in the top five recommendations, there is a 50% chance of one prediction being confused.

Next, Table 2 shows the performance of FunkSVD on the test set when compared with its proxy. This test illustrates the performance of the proxy by familiar standards of recommendation research. The difference in NDCG@5 of FunkSVD and its proxy is only 0.74%, which grows to a mere 1% for NDCG@10.

4 DISCUSSION

The first method, MAUSVR, shows promise for tactical networking conditions. Under sparse conditions of rating availability, MAUSVR had an NDCG@5 gain that was up to 47% higher than FunkSVD. Like other content-based approaches, performance at high sparsity is due to effective leveraging of domain information. MAUSVR also performs strictly above the average rating approach, due to the uncertainty quantification modification. In CF techniques, uncertainty cannot be captured as easily, since accuracy depends on the sparsity of the item-item or user-user matrix. For this reason, CF techniques are often hybridized with other approaches to boost performance on sparse data, however, this comes at the cost of increased system complexity and opaqueness - whereas MAUSVR's model is unaffected by blending with the average rating approach. Additionally, MAUSVR was able to be built instantaneously $< 100ms$ for more than 75% of profiles, using un-optimized Java code and off-the-shelf implementations of SMOreg. Some users would experience delays of greater than $100ms$, but no more than $3000ms$. Nielsen's research [26] suggests that at this limit the system would need to show indications of operation, but that user flow would not be adversely affected. This implies a responsive, MAUM-style user interface could be immediately built on top of MAUSVR, similar to [3] and [4].

The second evaluation presented here has demonstrated the feasibility of the SVR advice translation approach for arbitrary algorithms. The advice translation approach can work for discrete class based classifiers as well, conceivably even very opaque learners such as convolutional neural networks (e.g., [21]). However, SVM would need to be used instead of SVR. New strategies for building proxy models would need to be used, since we cannot say with confidence that the sampling strategies described here would generalize to other proxy methods. This is perhaps the biggest hurdle to future advice translation approaches, especially as database sizes increase. The number of items needing to be predicted in this example was fairly modest at 16K. As the number of items needing to be ranked increases, the sampling strategy must become more and more sparse to accommodate the costly build times of SVR.

Both of the approaches described in this paper have the limitation of requiring information to be modeled in domain terms. However, we believe that this should not be seem as a limitation specific to the advice translation approach, but as a cost that must be paid to create transparent systems. The cost may well be worth it: content-based explanations have been shown to be preferable by users [13], so the pursuit of rich content information should be seen as worthwhile. Moreover, in Edwards et al's [10] original survey of decision technology, it was noted that the main challenge to MAU interfaces is the collection of content data. However, this is becoming easier in recent years: the internet of things [36], crowd-sourcing [33], and pervasive computing [29] have partially automated the process of creating rich databases of metadata.

Figure 2 shows the accuracy difference between MAUSVR and the advice translation (Proxy) approach. The difference appears to be small, however, when accounting for the baseline NDCG (0.82), it becomes just under a 40% improvement. This implies a hybrid approach might be best - when the uplink is available, recommendations could be generated via CF and presented via advice translation. When the uplink becomes unavailable or fails, MAUSVR could be used as a backup. However, this schema ignores other human usability issues - human factors research is needed to determine if the small NDCG gain is worth the added system complexity.

5 CONCLUSION

In summary, this research has presented two recommendation methods that target the high risk C3I domain. The recommendation techniques draw their inspiration from multi-attribute utility, which may enable improved human comprehension of recommendations. While both methods have demonstrated their promise on a popular recommender systems benchmark, more offline and online evaluation is needed to determine their impact on key Army capabilities.

REFERENCES

[1] Agnar Aamodt. 1993. Explanation-driven case-based reasoning. In *European Workshop on Case-Based Reasoning*. Springer, 274–288.

[2] Svetlin Bostandjiev, John O'Donovan, and Tobias Höllerer. 2012. TasteWeights: a visual interactive hybrid recommender system. In *RecSys*, Padraig Cunningham, Neil J. Hurley, Ido Guy, and Sarabjot Singh Anand (Eds.). ACM, 35–42. http://dl.acm.org/citation.cfm?id=2365952

[3] Svetlin Bostandjiev, John O'Donovan, and Tobias Höllerer. 2012. Tasteweights: a visual interactive hybrid recommender system. In *Proceedings of the sixth ACM conference on Recommender systems*. ACM, 35–42.

[4] Svetlin Bostandjiev, John O'Donovan, and Tobias Höllerer. 2013. LinkedVis: exploring social and semantic career recommendations. In *Proceedings of the 2013 international conference on Intelligent user interfaces*. ACM, 107–116.

[5] Leo Breiman. 1996. Bagging predictors. *Machine learning* 24, 2 (1996), 123–140.

[6] Robin Burke. 2002. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction* 12, 4 (2002), 331–370.

[7] Tianfeng Chai and Roland R Draxler. 2014. Root mean square error (RMSE) or mean absolute error (MAE)?–Arguments against avoiding RMSE in the literature. *Geoscientific model development* 7, 3 (2014), 1247–1250.

[8] Olivier Chapelle. 2007. Training a support vector machine in the primal. *Neural computation* 19, 5 (2007), 1155–1178.

[9] Mary Cummings. 2004. Automation bias in intelligent time critical decision support systems. In *AIAA 1st Intelligent Systems Technical Conference*. 6313.

[10] Ward Edwards and Barbara Fasolo. 2001. Decision technology. *Annual review of psychology* 52, 1 (2001), 581–606.

[11] Michael D Ekstrand, Michael Ludwig, Joseph A Konstan, and John T Riedl. 2011. Rethinking the recommender research ecosystem: reproducibility, openness, and LensKit. In *Proceedings of the fifth ACM conference on Recommender systems*. ACM, 133–140.

[12] Mica R Endsley. 1995. Toward a theory of situation awareness in dynamic systems. *Human Factors: The Journal of the Human Factors and Ergonomics Society* 37, 1 (1995), 32–64.

[13] Fatih Gedikli, Dietmar Jannach, and Mouzhi Ge. 2014. How should I explain? A comparison of different explanation types for recommender systems. *International Journal of Human-Computer Studies* 72, 4 (2014), 367–382.

[14] David Gunning. 2017. Explainable artificial intelligence (xai). *Defense Advanced Research Projects Agency (DARPA), nd Web* (2017).

[15] F Maxwell Harper and Joseph A Konstan. 2016. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS)* 5, 4 (2016), 19.

[16] Jonathan L. Herlocker, Joseph A. Konstan, and John Riedl. 2000. Explaining Collaborative Filtering Recommendations. In *Proceedings of ACM CSCW'00 Conference on Computer-Supported Cooperative Work*. 241–250. http://www.acm.org/pubs/articles/proceedings/cscw/358916/p241-herlocker/p241-herlocker.pdf

[17] David B Kaber and Mica R Endsley. 1997. Out-of-the-loop performance problems and the use of intermediate levels of automation for improved control system functioning and safety. *Process Safety Progress* 16, 3 (1997), 126–131.

[18] Bart P Knijnenburg, Svetlin Bostandjiev, John O'Donovan, and Alfred Kobsa. 2012. Inspectability and control in social recommenders. In *Proceedings of the sixth ACM conference on Recommender systems*. ACM, 43–50.

[19] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009).

[20] Pigi Kouki, Shobeir Fakhraei, James Foulds, Magdalini Eirinaki, and Lise Getoor. 2015. Hyper: A flexible and extensible probabilistic framework for hybrid recommender systems. In *Proceedings of the 9th ACM Conference on Recommender Systems*. ACM, 99–106.

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.

[22] Justin Kruger and David Dunning. 1999. Unskilled and unaware of it: how difficulties in recognizing one's own incompetence lead to inflated self-assessments. *Journal of personality and social psychology* 77, 6 (1999), 1121.

[23] José Ramón San Cristóbal Mateo. 2012. Multi-attribute utility theory. In *Multi Criteria Analysis in the Renewable Energy Industry*. Springer, 63–72.

[24] D Harrison Mcknight, Michelle Carter, Jason Bennett Thatcher, and Paul F Clay. 2011. Trust in a specific technology: An investigation of its components and measures. *ACM Transactions on Management Information Systems (TMIS)* 2, 2 (2011), 12.

[25] LR Medsker and LC Jain. 2001. Recurrent neural networks. *Design and Applications* 5 (2001).

[26] Jakob Nielsen. 1994. *Usability engineering*. Elsevier.

[27] Ingrid Nunes and Dietmar Jannach. 2017. A systematic review and taxonomy of explanations in decision support and recommender systems. *User Modeling and User-Adapted Interaction* 27, 3-5 (2017), 393–444.

[28] John O'Donovan and Barry Smyth. 2005. Trust in recommender systems. In *Proceedings of the 10th international conference on Intelligent user interfaces*. ACM, 167–174.

[29] Mahadev Satyanarayanan. 2001. Pervasive computing: Vision and challenges. *IEEE Personal communications* 8, 4 (2001), 10–17.

[30] Rashmi Sinha and Kirsten Swearingen. 2002. The role of transparency in recommender systems. In *CHI'02 extended abstracts on Human factors in computing systems*. ACM, 830–831.

[31] Alex J Smola and Bernhard Schölkopf. 2004. A tutorial on support vector regression. *Statistics and computing* 14, 3 (2004), 199–222.

[32] Niranjan Suri, Giacomo Benincasa, Rita Lenzi, Mauro Tortonesi, Cesare Stefanelli, and Laurel Sadler. 2015. Exploring value-of-information-based approaches to support effective communications in tactical networks. *IEEE Communications Magazine* 53, 10 (2015), 39–45.

[33] Amazon Mechanical Turk. 2012. Amazon mechanical turk. *Retrieved August* 17 (2012), 2012.

[34] Yining Wang, Liwei Wang, Yuanzhi Li, Di He, Wei Chen, and Tie-Yan Liu. 2013. A theoretical analysis of NDCG ranking measures. In *Proceedings of the 26th Annual Conference on Learning Theory (COLT 2013)*.

[35] Ian H. Witten and Eibe Frank. 2005. *Data Mining: Practical machine learning tools and techniques, 2nd Edition*. Morgan Kaufmann, San Francisco.

[36] Feng Xia, Laurence T Yang, Lizhe Wang, and Alexey Vinel. 2012. Internet of things. *International Journal of Communication Systems* 25, 9 (2012), 1101.