SemTree: Ontology-Based Decision Tree Algorithm for Recommender Systems

Amancio Bouza, Gerald Reif, Abraham Bernstein, Harald Gall
Department of Informatics University of Zurich
{bouza, reif, bernstein, gall}@ifi.uzh.ch

ABSTRACT

Recommender systems play an important role in supporting people when choosing items from an overwhelming huge number of choices. So far, no recommender system makes use of domain knowledge. We are modeling user preferences with a machine learning approach to recommend people items by predicting the item ratings. Specifically, we propose SemTree, an ontology-based decision tree learner, that uses a reasoner and an ontology to semantically generalize item features to improve the effectiveness of the decision tree built. We show that SemTree outperforms comparable approaches in recommending more accurate recommendations considering domain knowledge.

Keywords

Recommender System, Ontology-Based Decision Tree, User Model, Feature Creation, Semantic Web, Ontology

1. INTRODUCTION

People are overwhelmed with the amount of items offered by online stores or webguides. Thus, recommender systems play an increasingly important role in supporting people getting items they like (e.g. music, movies, locations). In general, different people do not share the same taste and interests. Therefore, using average product rating over all users is not expressing a single user opinion adequately. A good example showing the benefit of a recommender system is the Amazon online store. Amazon supports people finding interesting items by providing recommendations like: "People that bought product X, also bought product Y".

However, the existing recommender systems do not consider domain knowledge to improve the accuracy and effectiveness of recommender systems. An item can be expressed simply by a feature vector where each dimension represents an item feature. The values in the feature vector are boolean and denote if the item provides those features or not. With ontologies item features and relations among them can be expressed machine readable and therefore can be used to compute recommendations.

We assume that item features are not independend from each other from a user perspective and therefore doing reasoning on item features can gain more information about the user's preferences. We therefore propose an algorithm that builds user model based on the overall rating of the item. To build the user model the algorithm considers a domain ontology to semantically interprete the features. The features are instances of concepts in the ontology. The main idea is that making an assertion about a superclass of a feature can gain more information then the assertion about the single feature.

2. RELATED WORK

Different approaches exists that predict an item rating by summing the existent ratings and weight each rating with the associated user similarity [4]. Other recommender systems use item co-occurence in purchases [6] and rank them by the frequency they appear together.

In Quickstep, an ontology is used for user profiling [5] for research papers. Quickstep learns by observing user behaviour in which research domain a user is interested and provides other papers of that research domain as recommendations. In [3] domain ontologies are used to extract feature to describe items. The extracted features are then applied to machine learning algorithms. In our approach we already have the features but we are generalizing features during the learning process to improve the result.

3. APPROACH

Our recommender system basically consists of two parts: the user model builder and the recommendation generator. In order to provide the user with recommendations, the user has to rate a couple of items first. The user model builder uses the feature vectors of the rated items in combination with the user's ratings to learn what combination of features leads to which rating. The result is a user model that expresses the user preferences. To provide the user with recommendations the recommendation generator takes all feature vectors of not yet rated items and predicts on the basis of the learned user model the user's rating for them. The computed ratings are used to rank the items.

3.1 User Model Builder

The user model builder learns a decision tree using the feature vectors of each item and ratings to classify them. In the following the pseudo code in Listing 1 shows the recursively process of selecting the feature with the highest information gain to build the decision tree node. In a first step, the algorithm calculates for every feature its information gain by splitting the (item) instances into two sets. The first set contains all item instances that provide the feature and the second set contains all item instances that do not. In a second step, the algorithm gets a list of superclasses for every feature and analogously calculates the associated information gain. In depth, the algorithm splits the item instances into two sets again. The first set contains item instances that provide at least one feature that is an instance of the

superclass. The second set contains all item instances that do not provide any feature that are instances of the superclass.
Next, the feature or feature-superclass with the highest information gain is used as decison tree node and to split the item instances into the two sets. For both sets of item instances the algorithm continues to select the feature or superclass with the highest information gain to build a new subtree recursively until no splitting of the item instances are possible.

Listing 1: Pseudo code

```
public SplitModel buildClassifier(instances)
{
  For each feature in featurevector
    SplitModel s = new SplitModel(feature);
    s.calculateInfoGain(instances);
    list.add(s);

  For each feature get superclasses
    For each superclass in superclasses
      SplitModel s = new SplitModel(superclass);
      s.calculateInfoGain(instances);
      list.add(s);

  SplitModel bestSplit = selectBestModel(list);
  Instances[] subsets = bestSplit.split(instances);

  For each instances in subsets
    SplitModel node = buildClassifier(instances);
    this.addChild(node);

  return bestSplit;
}
```

3.2 Recommendation Generator

The recommendations are made by classifying the items by the learned user model. To provide the user with a recommendation list all items are classified by the learned user model and ranked on the basis of the predicted ratings.

4. EVALUATION

We evaluated our approach quantitatively and calculated the mean average error (MAE) and the root mean square error (RMSE), two common metrics to evaluate the accuracy of recommender systems [2]. The MAE sums the difference between the predicted rating and the real rating and normalizes it. On the other site, the RMSE squares the error before summing and normalizing it. The RMSE weights the size of an error made higher and therefore is a better indicator of the error sizes done.
We used the Netflix Prize dataset [1] that originally consists of 17700 movies, 480189 users and 100480507 ratings from 1 to 5 on an integer scale. To enrich this dataset with movie informations we used the movie genre information from the IMDb (The Internet Movie Database) to build the feature vectors. Since movie titles tend to be used by several movies we made the restriction as necessary and sufficent that movie titles are exactly the same and the difference of the movie years are minimal because the year information were sometimes wrong in one of the two datasets. With this restriction we identified 10210 netflix movies in the IMDb. Therefore our evaluation dataset for the evaluation consists of 10210 movies, 479453 users and 83412500 ratings. Since we are evaluating a machine learning algorithm we used the probe set, that is provided by Netflix, as test set and the evaluation dataset without the test set as training set.

Table 1: Evaluation of Recommender Systems

Algorithm	RMSE	MAE
SemTree	1.1223011	0.8175
J48	1.3171503	0.9109841
AverageRating	1.0943271	0.8194
RandomGenerator	1.9295140	1.5443373

As we can see in Table 1 a simple randomly choosen rating performs worst as expected. The predicted ratings by the J48 algorithm from the WEKA project [7] perform better then the random generator. SemTree outperforms the J48 algorithm. However, we expect better recommendations by enriching the ontology with more movie informations. Therefore we can conclude that using domain knowledge learning a user model improves the accuracy of predictions.

5. CONCLUSIONS

We have proposed an ontology-based decision tree algorithm that uses a domain ontology and a reasoner to split instances with more generalized features (superclasses of features) then the features in cases where generalized features in form of superclasses perform better. The evaluation has shown that our approach outperforms other approaches. For future work we intend to use more movie information and compare our approach with more recommender systems.

6. ACKNOWLEDGMENTS

This work was partially supported by the Innovation Promotion Agency CTI Switzerland.

7. REFERENCES

[1] J. Bennett and S. Lanning. The netflix prize. *KDD Cup and Workshop*, 2007.

[2] J. L. Herlocker, J. A. Konstan, L. G. Reveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 2001.

[3] D. Kudenko. Ontology-based constructive induction (extended abstract).

[4] D. Lemire and A. Maclachlan. Slope one predictors for online rating-based collaborative filtering. In *Proceedings of SIAM Data Mining (SDM'05)*, 2005.

[5] S. E. Middleton, N. R. Shadbolt, and D. C. de Roure. Ontological user profiling in recommender systems. In *ACM Transactions on Information Systems*, 2004.

[6] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, 2001.

[7] I. H. Witten and E. Frank. *Data Mining - Practical Machine Learning Tools and Techniques.* 2005.