Active Learning to Support the Generation of Meta-examples

Ricardo Prudêncio[1] and Teresa Ludermir[2]

[1] Departament of Information Science, Federal University of Pernambuco, Av. dos Reitores, s/n - CEP 50670-901 - Recife (PE) - Brazil
prudencio.ricardo@gmail.com
[2] Center of Informatics, Federal University of Pernambuco, Pobox 7851 - CEP 50732-970 - Recife (PE) - Brazil
tbl@cin.ufpe.br

Abstract. Meta-Learning has been used to select algorithms based on the features of the problems being tackled. Each training example in this context, i.e. each meta-example, stores the features of a given problem and the performance information obtained by the candidate algorithms in the problem. The construction of a set of meta-examples may be costly, since the algorithms performance is usually defined through an empirical evaluation on the problem at hand. In this context, we proposed the use of Active Learning to select only the relevant problems for meta-example generation. Hence, the need for empirical evaluations of the candidate algorithms is reduced. Experiments were performed using the classification uncertainty of the k-NN algorithm as the criteria for active selection of problems. A significant gain in performance was yielded by using the Active Learning method.

1 Introduction

In several domains of application, there are different algorithms that can be applied to a single problem. In Machine Learning, for instance, different learning algorithms have been proposed to solve learning problems. An important question that arises in such domains is the appropriate selection of algorithms for each problem at hand [1]. The most traditional strategies to algorithm selection involve, in general, expert knowledge, or costly procedures of empirical evaluation [2]. Meta-Learning [3] arises in this context as an alternative solution, capable of automatically acquire knowledge to be used in algorithm selection.

The knowledge in Meta-Learning is acquired from a set of *meta-examples* that store the experience obtained in the application of a number of candidate algorithms in problems investigated in the past. More specifically, each meta-example is related to a given problem and stores: (1) the features that describe the problem; and (2) information about the performance obtained by the candidate algorithms when applied to the problem (e.g. the best algorithm, error rates, execution times,...). A meta-learner is a learning model that receives as input a set of such meta-examples, and generates knowledge relating features of the problems and the performance of the candidate algorithms.

A limitation of Meta-Learning is related to the process of generating meta-examples. In order to generate a meta-example from a given problem, it is necessary to perform an empirical evaluation (e.g. cross-validation) to collect the performance information of the candidate algorithms. Although the proposal of Meta-Learning is to perform this empirical evaluation only in a limited number of problems, the cost of generating a whole set of meta-examples may be high, depending, for instance, on the number and complexity of the candidate algorithms, the used methodology of empirical evaluation and the amount of data available in the problems.

Considering the above context, we present here an original proposal in which Active Learning [4] is used to select problems for meta-example generation. Active Learning is a paradigm of Machine Learning, used in domains in which it is hard or costly to acquire training examples [5]. The main motivation of Active Learning is to reduce the number of training examples, at same time maintaining the performance of the learning algorithms. In our proposal, it corresponds to reduce the set of meta-examples by selecting only the more relevant problems, consequently, reducing the number of empirical evaluations performed on the candidate algorithms.

In order to evaluate the viability of our proposal, we implemented a prototype in which the k-NN (k-Nearest Neighbors) algorithm is used as meta-learner, and a method based on uncertainty of classification [5] is used to select problems for meta-example generation. The prototype was evaluated in a case study which corresponds to the task of selecting two specific algorithms for time series forecasting problems: the Time Delay Neural Network (TDNN) [6] and the Simple Exponential Smoothing (SES) [7]. Experiments performed on a set of 99 problems revealed a gain in the meta-learner performance by using the implemented Active Learning method, compared to a random procedure for selecting problems.

The remaining of this paper is organized as follows. Section 2 brings a brief presentation of Meta-Learning, followed by section 3 which describes, in more details, the proposed solution and the implemented prototype. Section 4 presents the performed experiments and obtained results. Finally, section 5 concludes the paper by presenting some future work.

2 Meta-learning

In different knowledge areas, it is observed the existence of several algorithms that compete to be applied to specific classes of problems. In such situations, both empirical and theoretical results have shown that there is no algorithm uniformly superior, independently on the problem being tackled [8,9,10].

Meta-Learning is a framework that defines techniques to assist algorithm selection for learning problems (usually classification and regression problems) [3]. In a strict formulation of Meta-Learning, each training example (or *meta-example*) is related to an individual learning problem investigated in the past and stores: (1) a set of features (called *meta-attributes*) that describes the problem; and (2) a class attribute which indicates the best algorithm for the problem, among a set of candidate algorithms. The *meta-learner* in the strict formulation is simply a classifier algorithm which predicts the best algorithm for a given problem based on its descriptive meta-attributes [8].

The meta-attributes usually are statistical and information theory measures of the problem's dataset, such as number of training examples and attributes, correlation between attributes, class entropy, presence of outliers in the dataset, among others [11]. The class label stored in a meta-example is usually defined by empirically evaluating each candidate algorithm on the problem. This can be performed, for instance, via a cross-validation experiment using the available problem's dataset. The accuracy of each classifier is estimated by cross-validation, and the class label is assigned to a problem according to the algorithm with highest estimated accuracy.

Although the strict Meta-Learning has been investigated by different authors (see for instance [1,8,12,13,14,15]), other Meta-Learning techniques have been proposed to provide more informative solutions to algorithm selection. In [16], the authors proposed a meta-learner not only to predict the best algorithm but also to predict the applicability of each candidate algorithm to the new problems being tackled. In [9], the NOEMON system combined different strict meta-learners in order to provide rankings of the candidate algorithms. In [10], the authors applied instance-based learning to provide rankings of algorithms, taking into account the predicted accuracy and execution time of the algorithms. In [17], the authors used a regression model as meta-learner in order to predict the numerical value of the accuracy for each candidate algorithm.

The concepts and techniques of Meta-Learning were originally evaluated to select algorithms for classification and regression problems. However, in recent years, Meta-Learning has been extrapolated to other domains of application [12,14,18]. In [12,14], for instance, the authors proposed the use of Meta-Learning to select algorithms for time series forecasting. In [18], the authors applied Meta-Learning to support the construction of planning systems. Considering these applications, Meta-Learning can be viewed as a more general framework to algorithm selection, and it is expected to be useful in problems related to a large range of domains.

3 Active Learning for Meta-example Generation

An important step in the generation of meta-examples is to estimate the performance of the candidate algorithms on a set of problems. The evaluation of performance is accomplished by applying a pre-defined methodology of experiments (e.g. hold-out, cross-validation,...) to the available datasets. The information resulting from the performance evaluation is used to define the target attribute in the Meta-Learning task (e.g. the class corresponding to the best algorithm).

The generation of meta-examples may be a costly process depending on a number of aspects, such as the methodology of experiments adopted to performance evaluation, the number of problems available to generate meta-examples, and the number and complexity of the candidate algorithms.

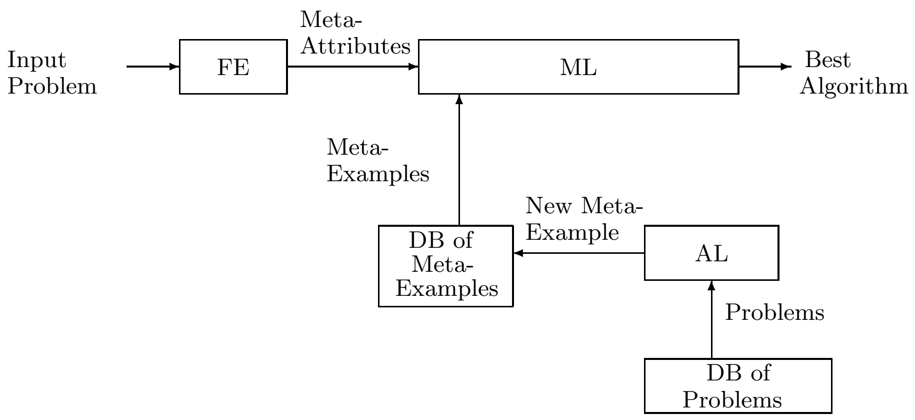

Fig. 1. System Architecture

In the above context, we proposed here the use of Active Learning to support the generation of training examples for Meta-Learning. Active Leaning is a paradigm of Machine Learning in which the learning algorithm has some control over the inputs on which it trains [4]. The main objective of this paradigm is to reduce the number of training examples, at same time maintaining the performance of the learning algorithm. Active Learning is ideal for learning domains in which the acquisition of labeled examples is a costly process, such as image recognition [5], text classification [19] and information filtering [20].

The main motivation of the use of Active Learning in our context is to select only the more relevant problems for Meta-Learning, and hence, to reduce the number of empirical evaluations on the candidate algorithms. Figure 1 presents the general architecture of system following our proposal. The system has three different phases: meta-example generation, training and use, described as follows.

In the meta-example generation, the Active Learning (AL) module selects from a base of problems, those ones considered the most relevant for the Meta-Learning task. The selection of problems is performed based on a pre-defined criteria implemented in the module. The candidate algorithms are then empirically evaluated on the selected problems, in order to collect the performance information related to the algorithms. Each generated meta-example (composed by meta-attributes and performance information) is then stored in an appropriate database.

In the training phase, the Meta-Learner (ML) acquires knowledge from the database of meta-examples generated by the AL module. This knowledge associates meta-attributes to the performance of the candidate algorithms. The acquired knowledge may be refined as more meta-examples are provided by the AL module.

In the use phase, given a new input problem, the Feature Extractor (FE) module extracts the values of the meta-attributes. According to these values, the ML module predicts the best candidate algorithm. For that, it uses the knowledge previously acquired as a result of the training phase.

In order to evaluate the proposal, we implemented a prototype to select between two candidate algorithms for time series forecasting problems: the Time-Delay Neural Network (TDNN) [6] and the Simple Exponential Smoothing model (SES) [7]. Both algorithms were used for short-term forecasting of stationary time series (with no trend or seasonality). According to [14], both algorithms are indicated to forecast stationary time series, however the quality of the forecasts provided by the algorithms may be very different depending on the time series at hand.

In the current prototype, the k-Nearest Neighbors (k-NN) algorithm was used in the ML module, and an Active Learning method based on classification uncertainty of the k-NN [5] is used in the AL module. In the next sections, we provide more details of the implemented prototype.

3.1 Feature Extractor

In our proposal, the FE module implements p meta-attributes $X_1, \ldots, X_p$ which correspond to features that describe the input problems. Hence, each problem e is described by a vector $\mathbf{x} = (x^1, \ldots, x^p)$ in which $x^j = X_j(e), (j = 1, \ldots, p)$.

In the implemented prototype, each input problem consists of a time series to be forecasted, and $p = 10$ features were used as meta-attributes:

1. Length of the time series (X_1): number of observations of the series.
2. Mean of the absolute values of the 5 first autocorrelations (X_2): high values of this feature suggests that the value of the series at a time point is very dependent of the values in recent past points.
3. Test of significant autocorrelations (X_3): presence of at least one significant autocorrelation taking into account the first 5 ones.
4. Significance of the first, second and third autocorrelation (X_4, X_5 and X_6): indicates significant dependences in more recent past points.
5. Coefficient of variation (X_7): measures the degree of instability in the series.
6. Absolute value of the skewness and kurtosis coefficient (X_8 and X_9): measure the degree of non-normality in the series.
7. Test of Turning Points for randomness (X_{10}): The presence of a very large or a very low number of turning points in a series suggests that the series is not generated by a purely random process.

All implemented features are directly computed from the avaliable series data, which has the advantage of avoiding subjective analysis, such as visual inspection of plots.

3.2 Meta-learner

The Meta-Learning task in the current prototype corresponds to the strict formulation described in Section 2, i.e. the meta-learner is a conventional classifier that uses the meta-attributes to predict the best candidate algorithm. In the implemented prototype, we used the k-NN algorithm as the strict meta-learner. In

[10], the authors presented some advantages of using instance-based algorithms, such as the k-NN, as meta-learners. Instance-based algorithms are extensible: once a new meta-example becomes available, it can be easily integrated without the need to initiate re-learning. According to [10], this is relevant for algorithm selection since the user typically starts with a small set of meta-data that increases steadily with time.

In this section, we describe more formally the meta-learner used in the prototype. Let $E = \{e_1, \ldots, e_n\}$ be the set of n problems used to generate a set of n meta-examples $ME = \{me_1, \ldots, me_n\}$. Each meta-example is related to a single problem and stores the values of p features $X_1, \ldots, X_p$ for the problem and the value of a class attribute C, which indicates the best algorithm for the problem, among L candidates. In our prototype, we have $p = 10$ meta-attributes describing problems and $L = 2$ candidate algorithms (TDNN and SES).

Let $D = \{c_1, \ldots, c_L\}$ be the domain of the class attribute C where each class label $c_l \in D$ represents a candidate algorithm. In this way, each meta-example $me_i \in ME$ is represented as the pair $(\mathbf{x}_i, C(e_i))$ storing: (1) the description $\mathbf{x}_i$ of the problem e_i, where $\mathbf{x}_i = (x_i^1, \ldots, x_i^p)$ and $x_i^j = X_j(e_i)$; and (2) the class label associated to e_i, i.e. $C(e_i) = c_l$, where $c_l \in D$.

Given a new input problem described by the vector $\mathbf{x} = (x^1, \ldots, x^p)$, the k-NN meta-learner retrieves k meta-examples from ME, according to the distance between meta-attributes. The distance function (*dist*) implemented in the prototype was the unweighted L_1-Norm, defined as:

$$dist(\mathbf{x}, \mathbf{x}_i) = \sum_{j=1}^{p} \frac{|x^j - x_i^j|}{max_i(x_i^j) - min_i(x_i^j)} \tag{1}$$

The prediction of the class label for the new problem (i.e. the prediction of the best algorithm) is performed according to the number of occurrences (votes) of each $c_l \in D$ in the class labels associated to the retrieved meta-examples.

3.3 Active Learning

As seen, the ML module predicts the best algorithms by using a set of meta-examples generated from *labeled* problems, i.e. the problems in which the best candidate algorithm is known. The AL module, described in this section, receives a set of *unlabeled* problems, i.e. the problems in which the candidate algorithms were not yet evaluated and, hence, the best algorithm for each problem is not known. Therefore, the main objective of the AL module is to incrementally select unlabeled problems to be used for generating new meta-examples.

In the prototype, the AL module implements a method for selecting unlabeled problems which is based on the criteria of *classification uncertainty* of the k-NN algorithm [5]. In this criteria, initially, the k-NN algorithm classifies each unlabeled example by using the available labeled examples. A degree of uncertainty of the provided classification is assigned for each unlabeled example. Finally, the unlabeled example with the highest classification uncertainty is then selected.

The classification uncertainty of the k-NN algorithm is defined in [5] as the ratio of: (1) the distance between the unlabeled example and its nearest labeled neighbor; and (2) the sum of the distances between the unlabeled example and its nearest labeled neighbors of different classes. A high value of uncertainty indicates that the unlabeled example has nearest neighbors with similar distances but conflicting labeling. Hence, once the unlabeled example is labeled, it is expected that the uncertainty of classification in its neighborhood should be reduced.

In our context, let E be the set of labeled problems, and let $\widetilde{E}$ be the set of unlabeled problems. Let E_l be the subset of labeled problems associated to the class label c_l, i.e. $E_l = \{e_i \in E | C(e_i) = c_l\}$. Given the set E, the classification uncertainty of k-NN for each $\widetilde{e} \in \widetilde{E}$ is defined as:

$$\mathcal{S}(\widetilde{e}|E) = \frac{\min_{e_i \in E} dist(\widetilde{\mathbf{x}}, \mathbf{x}_i)}{\sum_{l=1}^{L} \min_{e_i \in E_l} dist(\widetilde{\mathbf{x}}, \mathbf{x}_i)} \tag{2}$$

In the above equation, $\widetilde{\mathbf{x}}$ is the description of problem $\widetilde{e}$. The AL module then selects, for generating a new meta-example, the problem $\widetilde{e}^* \in \widetilde{E}$ with highest uncertainty:

$$\widetilde{e}^* = argmax_{\widetilde{e} \in \widetilde{E}} \mathcal{S}(\widetilde{e}|E) \tag{3}$$

Finally, the selected problem is labeled (i.e. the class value $C(\widetilde{e}^*)$ is defined), through the empirical evaluation of the candidate algorithms using the avaliable data of the problem.

In our prototype, the labeling of a time series is performed through the empirical evaluation of TDNN and SES in forecasting the series. For this, a hold-out experiment was performed, as described in [13]. Given a time series, its data was divided into two parts: the fit period and the test period. The test period consists on the last 30 points of the time series and the fit period consists on the remaining data. The fit data was used to calibrate the parameters of both models TDNN and SES. Both calibrated models were used to generate one-step-ahead forecasts for the test data. Finally, the class attribute was assigned as the model which obtained the lowest mean absolute forecasting error on the test data.

4 Experiments and Results

In the performed experiments, we used 99 time series collected from the Time Series Data Library (TSDL)[1]. This repository contains time series data from several domains, most of them used as benchmark problems in the forecasting field. Both algorithms (TDNN and SES) were empirically evaluated for forecasting each series (as seen in section 3.3), and hence, 99 meta-examples were generated for the experiments with Meta-Learning.

The prototype was evaluated for different configurations of the k-NN meta-learner (with $k = 1, 3, 5, 7, 9$ and 11 nearest neighbors). For each configuration,

[1] TSDL - http://www-personal.buseco.monash.edu.au/~hyndman/TSDL

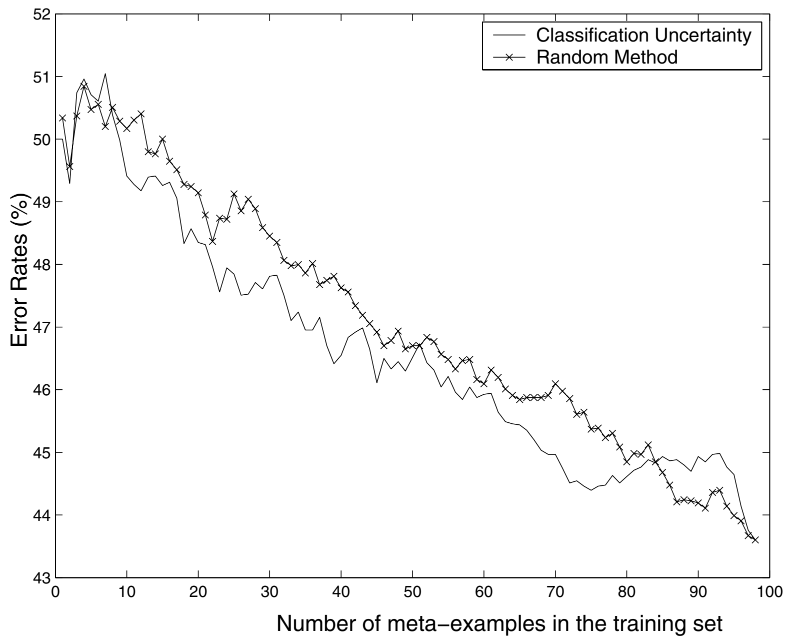

Fig. 2. Average curves of error rates obtained by k-NN meta-learner for both the classification uncertainty and the random active learning methods

a leave-one-out experiment was performed to evaluate the performance of the meta-learner, also varying the number of meta-examples provided by the Active Learning module. This experiment is described just below.

At each step of leave-one-out, one problem is left out for testing the ML module, and the remaining 98 problems are considered as candidates to generate meta-examples. The AL module progressively includes one meta-example in the training set of the ML module, up to the total number of 98 training meta-examples. At each included meta-example, the ML module is judged on the test problem left out, receiving either 1 or 0 for failure or success. Hence, a curve with 98 binary judgments is produced for each test problem. Finally, the curve of error rates obtained by ML can be computed by averaging the curves of judgments over the 99 steps of the leave-one-out experiment.

As a basis of comparison, the same above experiment was applied to each configuration of k-NN, but using in the AL module a random method for selecting unlabeled problems. According to [5], despite its simplicity, the random method has the advantage of performing a uniform exploration of the example space.

Figure 2 presents the curve of error rates obtained by the k-NN meta-learner averaged across the different configurations of the parameter k. The figure presents the average curve obtained when both methods were used: the classification uncertainty (described in section 3.3) and the random method. As it is expected, for both methods, the error rate obtained by the ML module decreased as the number of meta-examples in the training set increased. However, the error rates obtained by deploying the classification uncertainty method were, in general, lower than the error rates obtained by deploying the random method. In fact, from 8 to 84 meta-examples included in the training set, the classification uncertainty method steadily achieved better performance compared to the random method.

5 Conclusion

In this paper, we presented an original work that proposes the use of Active Learning to enhance the generation of examples for Meta-Learning. In order to verify the viability of our proposal, we implemented a prototype in which the classification uncertainty criteria is used to select meta-examples for a k-NN meta-learner. The prototype was evaluated in a task of selecting two candidate algorithms for time series forecasting, and the experiments results were promissing.

We highlight here that, despite the originality of the proposal, it still has some limitations that will be dealt with in future work. Although the classification uncertainty method obtained good results in the investigated Meta-Learning task, other Active Learning methods for k-NN have presented very competitive results in tasks related to other contexts [5].

Another question to investigate is the use of Active Learning not only for strict k-NN meta-learners, but also for other Meta-Learning techniques (as those cited in section 2). The choice of the strict k-NN meta-learner in our prototype was due to the fact that different authors in literature have focused their efforts on this kind of meta-learner or on similar techniques. However, other Meta-Learning techniques have been achieved good results in algorithm selection and hence, they also have to be investigated in the context of the current proposal.

References

1. Kalousis, A., Gama, J., Hilario, M.: On data and algorithms - understanding inductive performance. Machine Learning 54(3), 275–312 (2004)
2. Kalousis, A., Hilario, M.: Representational issues in meta-learning. In: Proceed. of the 20th International Conference on Machine Learning, pp. 313–320 (2003)
3. Giraud-Carrier, C., Vilalta, R., Brazdil, P.: Introduction to the special issue on meta-learning. Machine Learning 54(3), 187–193 (2004)
4. Cohn, D., Atlas, L., Ladner, R.: Improving Generalization with Active Learning. Machine Learning 15, 201–221 (1994)
5. Lindenbaum, M., Markovitch, S., Rusakov, D.: Selective sampling for nearest neighbor classifiers. Machine Learning 54, 125–152 (2004)
6. Lang, K., Hinton, G.: Time-delay neural network architecture for speech recognition. In CMU Technical Report CS-88-152, Carnegie-Mellon University, Pittsburgh, PA (1988)
7. Brown, R.G.: Smoothing, Forecasting and Prediction. Prentice-Hall, Englewood Cliffs (1963)
8. Aha, D.: Generalizing from case studies: a case study. In: Proceedings of the 9th International Workshop on Machine Learning, pp. 1–10 (1992)

9. Kalousis, A., Theoharis, T.: Noemon: Design, implementation and performance results of an intelligent assistant for classifier selection. Intelligent Data Analysis 3(5), 319–337 (1999)
10. Brazdil, P., Soares, C., da Costa, J.: Ranking learning algorithms: Using IBL and meta-learning on accuracy and time results. Machine Learning 50(3), 251–277 (2003)
11. Engels, R., Theusinger, C.: Using a data metric for preprocessing advice for data mining applications. In: ECAI-98. Proceedings of the 13th European Conference on Artificial Intelligence, pp. 430–434 (1998)
12. Prudêncio, R., Ludermir, T.: Selection of models for time series prediction via meta-learning. In: Proceedings of the 2nd International Conference on Hybrid Systems, pp. 74–83 (2002)
13. Prudêncio, R., Ludermir, T., de Carvalho, F.: A modal symbolic classifier to select time series models. Pattern Recognition Letters 25(8), 911–921 (2004)
14. Prudêncio, R., Ludermir, T.: Meta-learning approaches to selecting time series models. Neurocomputing 61, 121–137 (2004)
15. Leite, R., Brazdil, P.: Predicting relative performance of classifiers from samples. In: Proceedings of the 22nd International Conference on Machine Learning (2005)
16. Michie, D., D. J. S., Taylor, C.C., (eds.): Machine Learning, Neural and Statistical Classification. Ellis Horwood, New York (1994)
17. Bensusan, H., Alexandros, K.: Estimating the predictive accuracy of a classifier. In: Proceedings of the 12th European Conference on Machine Learning, pp. 25–36 (2001)
18. Tsoumakas, G., Vrakas, D., Bassiliades, N., Vlahavas, I.: Lazy adaptive multicriteria planning. In: Proceed. of the 16th European Conference on Artificial Intelligence, pp. 693–697 (2004)
19. Tong, S., Koller, D.: Support vector machine active learning with applications to text classification. Journal of Machine Learning Research 2, 45–66 (2002)
20. Sampaio, I., Ramalho, G., Corruble, V., Prudêncio, R.: Acquiring the preferences of new users in recommender systems - The role of item controversy. In: Proceedings of the ECAI 2006 Workshop on Recommender Systems, pp. 107–110 (2006)