

# Transfer Learning by Reusing Structured Knowledge

Qiang Yang, Vincent W. Zheng, Bin Li,  
and Hankz Hankui Zhuo

■ *Transfer learning aims to solve new learning problems by extracting and making use of the common knowledge found in related domains. A key element of transfer learning is to identify structured knowledge to enable the knowledge transfer. Structured knowledge comes in different forms, depending on the nature of the learning problem and characteristics of the domains. In this article, we describe three of our recent works on transfer learning in a progressively more sophisticated order of the structured knowledge being transferred. We show that optimization methods and techniques inspired by the concerns of data reuse can be applied to extract and transfer deep structural knowledge between a variety of source and target problems. In our examples, this knowledge spans explicit data labels, model parameters, relations between data clusters, and relational action descriptions.*

In machine learning, we often find ourselves in new situations where we have only a few annotated data to build a reliable learning model. This often happens when we meet with new domains and encounter new tasks. In such cases, instead of manually labeling more data, which incurs great costs, an alternative method is to look to related auxiliary data for help. These data may have a different probability distribution or may even be represented in a different feature space. However, there may still be much useful knowledge that we can extract from these auxiliary data to improve the learning performance in the new domain. In our view, transfer learning aims at solving exactly these types of problems, by learning in one task domain and applying the knowledge to another. In the machine-learning and data-mining fields, researchers have considered a variety of related approaches, including multitask learning for classification (Thrun and Mitchell 1995; Schmidhuber 1994; Caruana 1997), learning theory-based transfer learning (Ben-David and Schuller 2003) and text and multimedia classification (Daumé and Marcu 2006; Dai et al. 2007a, 2007b; Blitzer, McDonald, and Pereira 2006; Blitzer, Dredze, and Pereira 2007; Raina et al. 2007).

A major challenge in transfer learning is to identify the common knowledge between auxiliary or source tasks and apply such knowledge to new or target tasks of interest. Such common knowledge can be explicit or implicit and can be represented in more or less sophisticated ways depending on the nature of the problem. Much transfer learning work has been done in the past for classification and regression tasks as reviewed in a recent survey article by Pan and Yang (2010). In these learning tasks, the knowledge to be transferred is mostly declarative in nature, and can be as simple as the data instances or features themselves. By identifying parts of the source-domain data that we can “reuse” in the target domain, these methods typically find parts of a domain that are shared with the target domain. Then, some data instances along with their associated labels are added to the target domain to enable an expansion of the target domain training data. Variations of these methods have been explored in the literature, such as semisupervised learning or active learning. For other tasks, such as those that go beyond classification

and clustering, we may have more sophisticated forms of knowledge to transfer, where the transferred knowledge can take the form of procedural or problem-solving knowledge such as that used in an automated planning task.

In this article, we select three of our recent projects to illustrate three different levels of complexity in the structured knowledge to be transferred. Starting with a data-level reuse view, we first consider transfer learning in an indoor location-estimation task based on WiFi signals, which we model as a classification task in the work of Zheng and colleagues (Zheng et al. 2008). Here we build a “bridge” through a hidden Markov model structure and associated parameters. We use the source-domain data to tune some key model parameters in order to “reconstruct” the target domain training data for classification in a new time period. We will see that the knowledge being transferred is relatively simple, and the objective is to improve the classification accuracy. In the second example, which gives an overview of the work of Li, Yang, and Xue (2009), we move on to consider a more complex form of knowledge for reuse, where we consider how to solve the data-sparsity problem in collaborative filtering, which is a problem that occurs widely in many areas such as social network analysis and product recommendation. While it is difficult to make accurate recommendation when data are very sparse, we might be able to find some related auxiliary domains where the data are relatively dense. We can then identify some common relationship between parts of the data to help learning in the target domain. In a final example, through the work of Zhuo and colleagues (Zhuo et al. 2008), we illustrate how to transfer the procedural knowledge between different domains in acquiring action models for automated planning, where transfer learning helps reduce the manual model-construction effort in a target domain. In this work, the knowledge to be transferred relates the preconditions to postconditions of actions, which is a key ingredient in a problem-solving task. By incorporating the structured knowledge on the relations in different domains, we can learn new actions in the target domain with less effort.

We note that in all three of our examples, the structure of the knowledge to be transferred ranges from simple to complex. Despite the apparent differences between the learning tasks, we show that optimization methods can be applied to extract and transfer structural knowledge that ranges from explicit data labels and model parameters to relations between data clusters and relational action descriptions. In our examples, the classification tasks are less structured than the procedural tasks, although in general, classification or declarative knowledge reuse is not inherently less structured than procedural or problem-solving tasks.

## Transfer Learning in WiFi Localization

Structured knowledge can be found in many machine-learning tasks. In WiFi-based human and object tracking and location estimation problems, domain knowledge is inherently structural in the sense that knowledge depends on the temporal, spatial, topological relationship between different data instances as well as attributes. To illustrate, we first explain what WiFi localization is.

Accurately locating a mobile device in an indoor or outdoor environment is an important task in many AI and ubiquitous computing applications. While global positioning systems (GPSs) are widely used outdoors, GPS signals are found to be easily blocked by a building’s walls in indoor environments. In such cases, alternative solutions for indoor localization need to be found to support the growing location-based services, such as navigation, mobile social network services, and home-based healthcare such as is discussed in Pollack (2007). WiFi localization can be modeled as a classification task, as shown in figure 1. When a mobile device is closer to a wireless access point (AP) in the WiFi environment, it can detect a higher received signal strength (RSS). At different locations, a mobile device can detect different signal strength values from various APs. We can then build a mapping from the detected signal strengths to the locations. Such a mapping function can be modeled as classification or regression learning tasks.

In this work, we exploit a hidden Markov model (HMM) to capture the temporal and topological structures of the problem domain, in which we take into account both the user dynamics and the environmental factors. Once trained, an HMM is able to predict the user locations based on the currently received RSS signals. However, a difficulty is compounded by the fact that the signal distribution changes as a function of time (see figure 2). This is a problem, since to obtain a high level of accuracy, more data have to be manually labeled again in a new time period, which is both expensive and impractical. What we hope to do is to transfer as much previously obtained knowledge as possible. To solve the problem, we assume that the structure of the HMM and some key parameters stay constant while some parameters must change across time periods. While this is a relatively simple form of structural knowledge transfer application for classification prediction, we will see that for solving this problem it is very effective. In the following, we give an overview of our solution in Zheng et al. (2008).

Our high-level transfer learning idea is shown in figure 3. First, we model the prediction problem as a classification problem on a set of discrete loca-

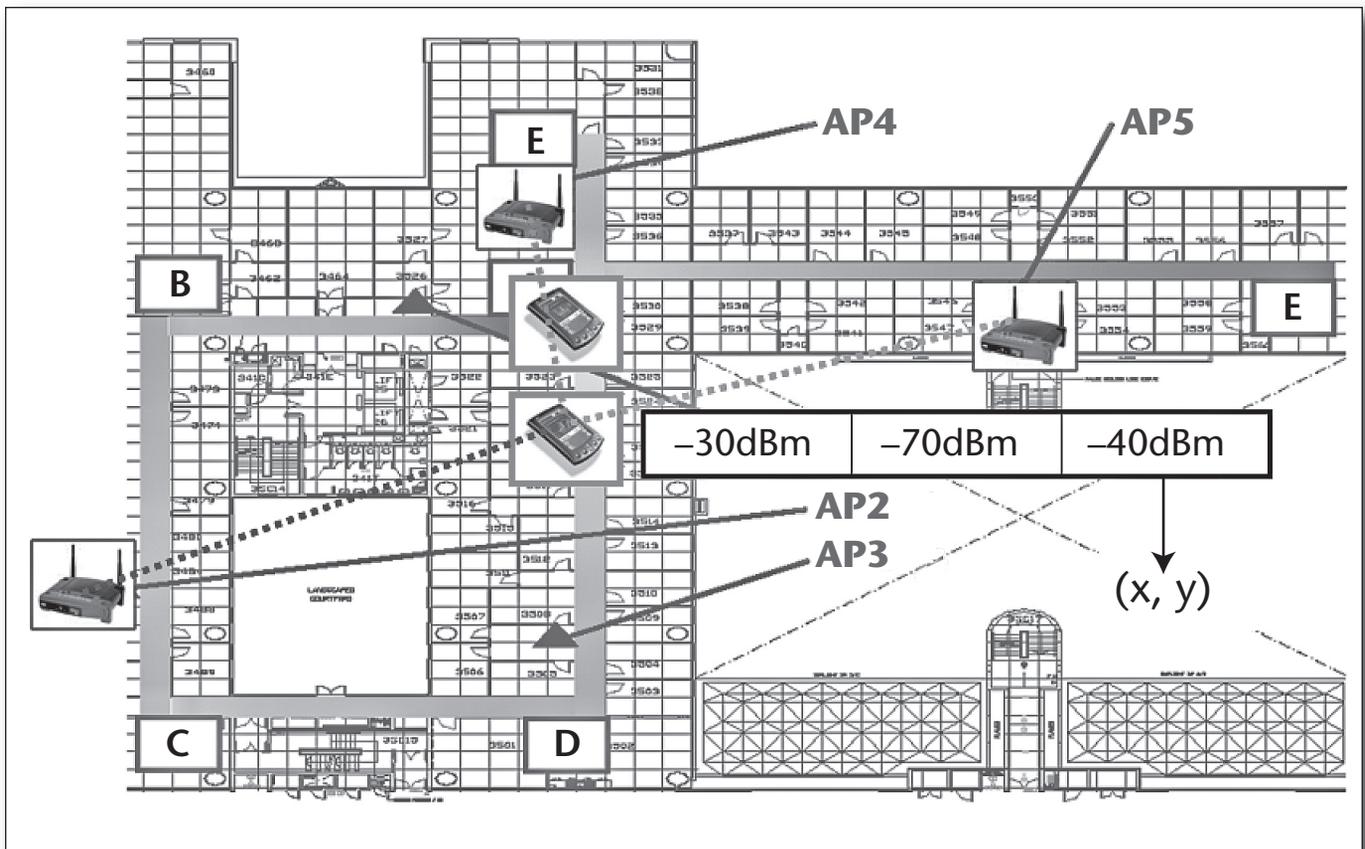


Figure 1: WiFi Localization as Classification into Location Grids.

tion grids. At time 0, we collect RSS data with location labels over the whole area. This step is time consuming, but is done only once. Based on this data, we train an HMM model with parameters  $\theta_0 = (\lambda_0, A_0, \pi_0)$  for localization at time period 0. In an HMM model,  $\lambda_0$  is a collection of signal values and location label pairs, which is called a radio map.  $A_0$  is the transition matrix that describes the way the user moves; it is a probability transition matrix mapping from one location state to another.  $\pi_0$  is a probability distribution showing where a user is most likely located; it is the prior knowledge on the likelihood of user locations. In a WiFi localization problem,  $\lambda_0$  often changes over time because the signal strength varies depending on a large number of environmental factors, such as the number of people around, the temperature and humidity, and so on.  $A_0$  can also change over time, because at different time periods, people may conduct different activities. For example, at noon, people are more likely to have lunch at a canteen, while during working hours, people are more likely to move within an office area. Therefore, both  $\lambda_0$  and  $A_0$  need to be adapted to their new values as  $\lambda_t$  and  $A_t$  for a new time period  $t$ . Among the parameters,  $\pi_0$  can be seen as relatively constant over time,<sup>2</sup> since in many situations, basic human

behavior does not change dramatically in an indoor environment. For example, a professor usually stays at his office longer than he walks in corridors during the day.

Thus, at a new time period  $t$ , we will transfer the HMM model and parameters  $(\lambda_0, A_0, \pi_0)$  as follows:

At time 0, we select some locations to put some sniffing WiFi readers, which we call “reference points.” These reference points are used to collect up-to-date RSS values with known location labels. The intuition is that, if we know how the signals change in some reference-point locations, we may use this knowledge to roughly predict the RSS values of other locations. Our solution here is to use a multiple linear regression model to capture the temporal predictive correlations between the RSS values at the reference points and those at other locations. We first learn a set of regression coefficients  $\alpha^k = \{\alpha_{ij}^k\}$  that encode the signal correlation between  $n$  reference-point locations and one non-reference-point location  $k$ . Based on the regression formula, when we know the RSS values at the reference locations, we exploit a linear interpolation to rebuild the radio map  $\lambda$  at time  $t$ . The regression weights  $\alpha$  can be learned with time 0’s data, which are assumed to be constant over time.

Given an initial HMM  $\theta_0 = (\lambda_0, A_0, \pi_0)$  as the base

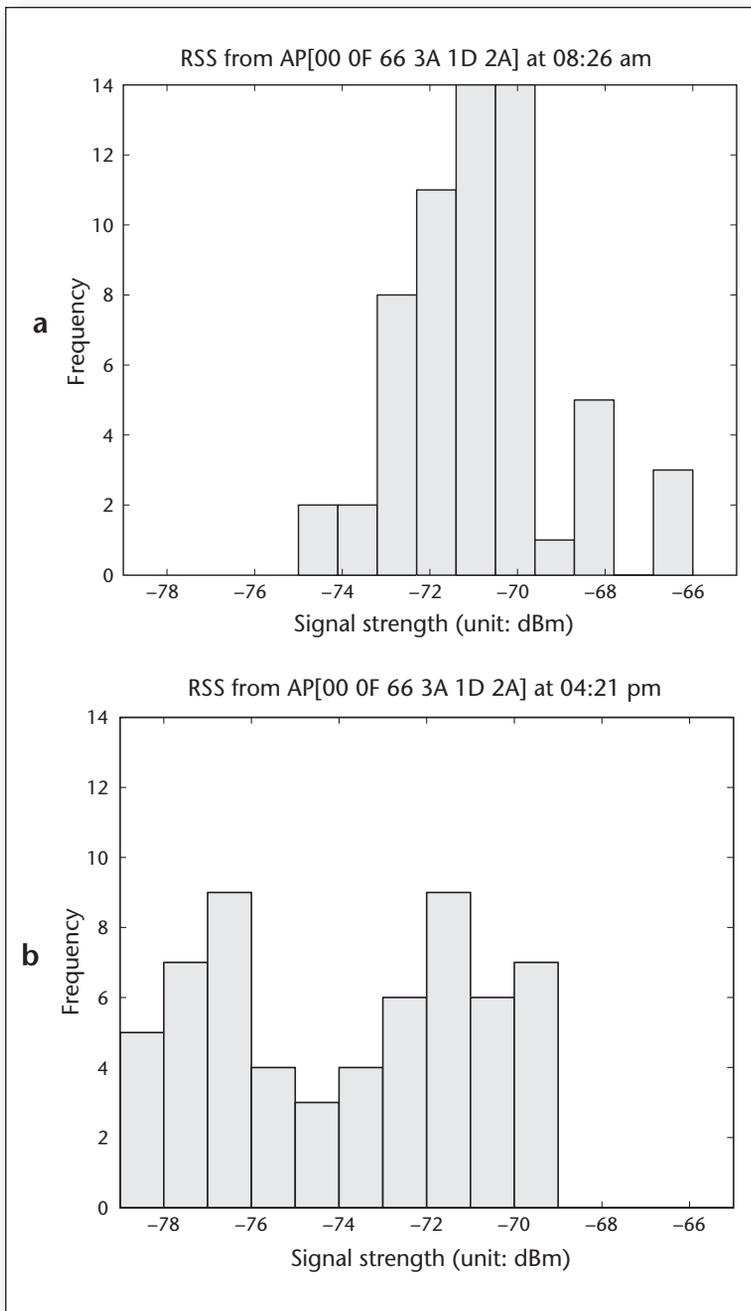


Figure 2: RSS Variations over Time at a Fixed Location.

model, at a new time period  $t$ , we improve  $\lambda_0$  by applying the regression analysis as mentioned above and obtain a new HMM  $\theta_t = (\lambda_t, A_0, \pi_0)$ . At time  $t$ , if we collect some additional unlabeled user traces by simply walking around the environment, we can further improve the new model. Since these trace data encode the knowledge of user transition behaviors and the current time period's signal distributions, they can be used to transfer the localization model to  $\theta_t^{new} = (\lambda_t^{new}, A_t, \pi_0)$  by an expectation-maximization (EM) algorithm.

To test our model, we set up an experimental

environment in an academic building equipped with 802.11g wireless networks at Hong Kong University of Science and Technology. The area is 64 meters  $\times$  50 meters, including five hallways. It is discretized into a space of 118 grids, each measuring 1.5 meters  $\times$  1.5 meters. Our evaluation metric is based on classification accuracy, which is calculated as the percentage of correct predictions over all predictions. In our problem, a random guess would result in only 1 percent accuracy. We collected labeled WiFi data at three time periods: 08:26 am, 04:21 pm and 07:10 pm. We used the 08:26 am data to build the base model and carry out adaptation at other time periods. At each location grid, 60 samples were collected. We randomly split 2/3 of the data as training and the other 1/3 as testing. Additional unlabeled traces are collected for building the HMM at each time period.

We first test the localization accuracy over different data distributions without adaptation. The results are shown in figure 4. We use the 08:26 am data to build the base model  $\theta_0$ , and then apply  $\theta_0$  to predict the labels for test data traces of the three time periods. As shown in figure 4a, the localization accuracy of 08:26 am data is the highest, at 92 percent.<sup>3</sup> This high accuracy is due to the fact that the test data follow the same distribution with the training data. As time goes by, the signals become more noisy and changing, and the performance drops. At 04:21 pm, the busiest time in the work area, the noise level reaches the highest because of many people walking around at that time. During this period, the accuracy thus drops to the lowest point of about 68 percent, which is unsatisfactory. This observation implies a need for transferring the localization model over different data distributions.

We compared our model, denoted as TrHMM, to a number of state-of-the-art baseline models. The most notable of these models is the Radar system in Bahl and Padmanabhan (2000), which is based on a  $k$ -nearest-neighbor algorithm. We also compared our model to two other systems, Landmarc in Ni et al. (2003) and LeManCoR in Pan et al. (2007), which are weighted  $k$ -nearest-neighbor and semisupervised learning systems, respectively. We used 10 percent of the locations as reference points and 5 unlabeled traces for adaptation in TrHMM and LeManCoR. We ran the experiments 5 times and reported the confidence intervals. The results are shown in figures 4b and 4c. As we can see, the TrHMM system consistently outperforms the other methods over time, especially when the number of reference points is small. We have varied many other parameters, and the relative performance of various systems stay the same: TrHMM greatly outperforms others in all cases (Zheng et al. 2008).

To encourage more research in this area, we made the data set available.<sup>4</sup> In Yang, Pan, and

Zheng (2008) we give a detailed description of the learning tasks.

To summarize, in this section, we have considered transfer learning for a classification task in a WiFi localization problem. The knowledge to be transferred here corresponds to explicit structured knowledge (that is, HMM model structure) and some associated parameters. Even though the system performs relatively well in the tested domains, it still relies on the availability of several reference points located in various locations for the purpose of collecting up-to-date data. When the new ground-truth data are very sparse, such as when we try to measure a large geographical area, the system will encounter problems in its adaptation. A possible solution for this problem is to identify deeper knowledge that can be transferred from other time periods or geospatial locations.

### Transfer Learning for Collaborative Filtering

In the previous section, we discussed an approach to transferring the model structure and the associated parameters from one setting to another, as the data distribution changes over time. In this section, we consider a more sophisticated problem, when the relations between two sets of entities at a group level can be transferred from one domain to another. We aim to explore such structural knowledge for solving a collaborative filtering problem.

Recommender systems make product suggestions to users based on their past ratings and selections. A major technique in making recommendations is collaborative filtering, or CF (Resnick et al. 1994; Sarwar et al. 2001), which aims to find similar users and items to help make the recommendation. A typical representation employed is a user-item rating matrix, where each entry represents how a user has rated an item. The accuracy of CF methods depends on the density of a rating matrix. In many real-world recommender systems, users often rate only a very limited number of items. Thus, the rating matrix is often extremely sparse, resulting in poor recommendation performance. This sparsity problem has been a major bottleneck for many current CF methods.

In a transfer learning setting, we may borrow useful knowledge from another rating matrix from a different but related CF domain. Consider the scenario of launching a new book-rating service. Due to a lack of visitors in the beginning, the recommendations based on CF may be very inaccurate. Now, suppose that we already have a dense movie-rating matrix available on a popular movie-rating service. Is it possible to establish a bridge between the two rating matrices and transfer useful rating patterns from the movie-rating matrix,

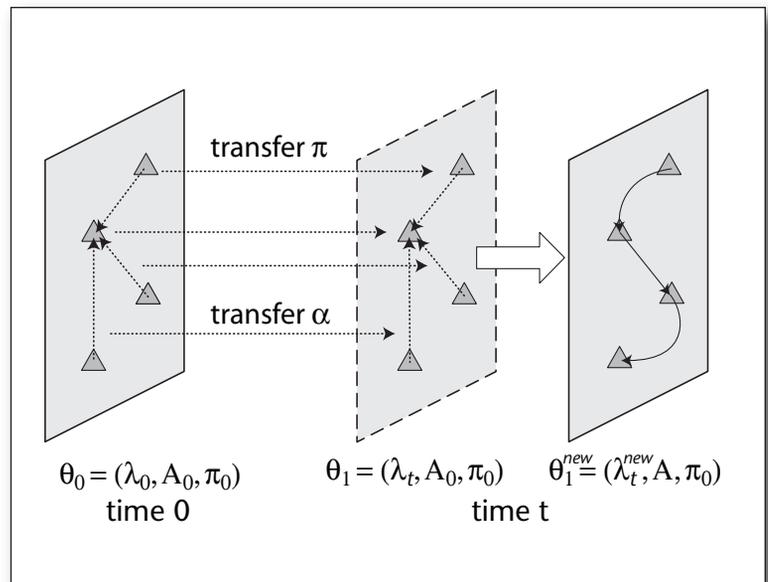


Figure 3. Model Transfer from Time 0 to Time  $t$ .

The triangles denote reference points in the area.

so that we may obtain higher performance in recommending books?

Intuition tells us that this is possible, since movies and books are somewhat related. Transfer learning can be beneficial if we identify the right knowledge to transfer. While individual users may be different between the target and auxiliary domains, groups of such users may include adults and teenagers, and students and office workers. The tastes of these user groups for item groups may be consistent across domains. If we can establish the correspondence between the groups, we can then alleviate the data-sparsity problem by using the auxiliary domain knowledge. The central thesis here is that we can transfer the informative and yet compact *group-level* rating patterns from the auxiliary rating matrix to help fill in some missing values in the target task. In terms of structured knowledge to be transferred, this group-level preference or rating knowledge is more sophisticated than the explicitly expressed HMM structure and parameter knowledge in the last section, as the group-level correspondence must be learned and the relationship being transferred concerns relations between groups of individuals rather than individual instances.

We refer to the target task as a  $p \times q$  book-rating matrix that has a sparse structure  $X_{tgt}$  which has very few observed ratings. Using this matrix to make recommendations, we may only obtain poor prediction results with CF. To solve this problem, we consider an available auxiliary domain that is related to the target learning task, which is a dense  $m \times n$  movie-rating matrix  $X_{aux}$ . We refer to the collection of group-level rating patterns to be trans-

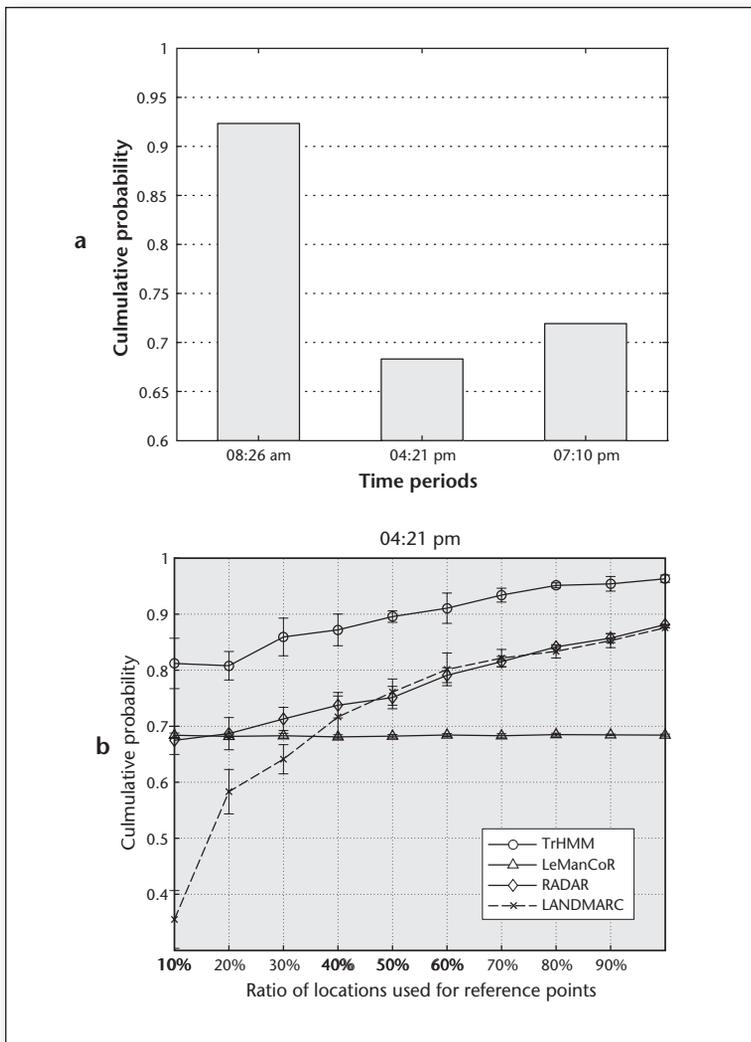


Figure 4. Experimental Results on WiFi Localization.

ferred as a “codebook,” which is a  $k \times l$  ( $k < m$ ,  $k < p$ ,  $l < n$ ,  $l < q$ ) matrix that relates user groups to item groups. By assuming the group-level rating patterns in  $X_{tgt}$  to be similar to  $X_{aux}$ , we can reconstruct the target rating matrix by expanding the codebook. An example of a codebook can be found in figures 5 and 6. The user groups I, II, and III can be students, professors, and workers, respectively. Likewise, the item groups A, B, and C can be comedies, dramas, and documentaries, respectively. The codebook then states facts such as “professors generally prefer to watch documentaries.”

To learn the codebook, our first step is to compute the user and item groups in the auxiliary rating matrix  $X_{aux}$ . To this end, we need to simultaneously cluster the rows (users) and columns (items) of  $X_{aux}$ . This can be done through various coclustering algorithms, with which we can compute the codebook by averaging all the ratings in

each user-item cocluster as an entry (see figure 5). Through the codebook, we can then transfer the user-item rating patterns from  $X_{aux}$  to  $X_{tgt}$  by expanding the codebook.

By assuming that there exists an implicit correspondence between the user/item groups of the auxiliary task and those of the target task, we can reconstruct  $X_{tgt}$  by expanding the codebook through duplicating certain rows and columns in the codebook. The duplication of the  $i$ -th row/column in the codebook means that there is a set of users/items in  $X_{tgt}$  that behave in a similar way to their corresponding group prototypes in  $X_{aux}$ . The reconstruction process of  $X_{tgt}$  expands the codebook as it reduces the differences between the observed ratings in  $X_{tgt}$  and the corresponding entries in the reconstructed rating matrix, based on some loss function. An example of the target rating matrix reconstruction is illustrated in figure 6. We use a user-cluster membership matrix and an item-cluster membership matrix to expand the codebook learned from  $X_{aux}$ . We denote the cluster index by a single “1” in each row of the user-cluster membership matrix and each column of the item-cluster membership matrix. The two membership matrices in figure 6 are optimal since the reconstructed ratings have no difference than the observed ratings in  $X_{tgt}$ . We can then fill the missing ratings in  $X_{tgt}$  with the corresponding entries in the reconstructed target rating matrix. We provide more algorithmic details in Li, Yang, and Xue (2009).

To test the idea, we have conducted a series of experiments to test the codebook transfer (CBT) algorithm. The auxiliary data include EachMovie,<sup>5</sup> which is a movie-rating data set comprising 2.8 million ratings (scales 1–6) by 72,916 users on 1628 movies. Another domain is the MovieLens<sup>6</sup> data, which is used as the target CF task. The MovieLens domain is a movie-rating data set comprising 100,000 ratings (scales 1–5) by 943 users on 1682 movies. We randomly select 500 users with more than 40 ratings and 1000 movies (rating ratio 11.6 percent). Another target task we have tested on is the Book-Crossing<sup>7</sup> dataset, which is a book-rating data set comprising more than 1.1 million ratings (scales 1–10) by 278,858 users on 271,379 books. We randomly select 500 users and 1000 books with the most ratings (rating ratio 3.02 percent). We also normalize the rating scales from 1 to 5.

In the experiments, we compare CBT to several baseline state-of-the-art methods, including PCC, which is the Pearson correlation coefficient-based method; CBS, which is a scalable cluster-based smoothing (Xue et al. 2005); and WLR, which is a weighted low-rank approximation (Srebro and Jaakkola 2003) method. The last two methods use clusters and matrix factorizations, respectively, to fill in missing values. The evaluation metric we

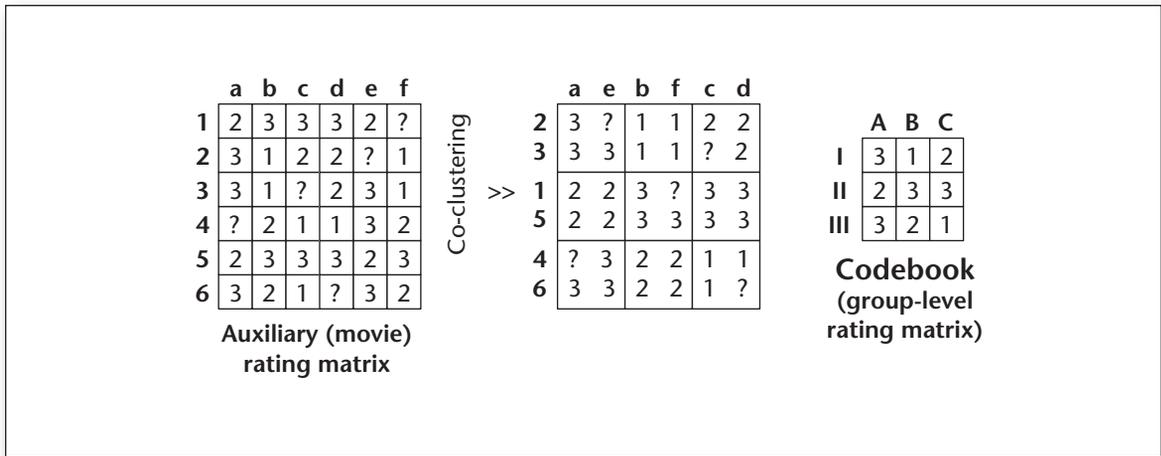


Figure 5. Codebook Construction by Coclustering the Auxiliary Rating Matrix.

An entry in the codebook denotes a rating provided by a user group on an item group.

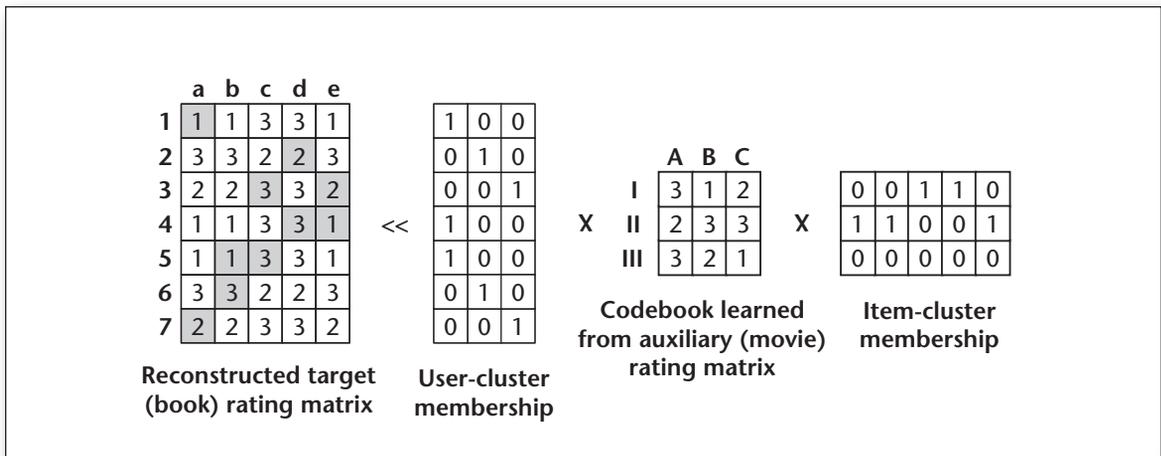


Figure 6: Target Rating Matrix Reconstruction by Expanding the Codebook.

The shaded entries are the filled-in missing values. The unshaded ones are observed ratings, whose values are identical to the reconstructed entries (loss is 0).

adopt is the often-used mean absolute error (MAE):  $\sum_{i \in T} |r_i - \tilde{r}_i| / |T|$ , where  $T$  denotes the set of test ratings,  $r_i$  is ground truth, and  $\tilde{r}_i$  is predicted rating. A smaller value of MAE means a better performance.

We show the comparison results in table 1, where we can see that our codebook-based method outperforms all other baseline methods on both target datasets. In these domains, CBT is more effective in alleviating the sparsity problem by transferring useful knowledge from the dense auxiliary rating matrix. More details on the experimental results can be found in Li, Yang, and Xue (2009).

To summarize, in the codebook-based knowledge transfer (CBT) approach for CF, we transfer group-level rating patterns as an implicit form of structured knowledge. The knowledge is transferred in the form of a codebook, which relates user groups to item groups. We can then strength-

en the recommendation performance by expanding the codebook. Experimental results show that codebook transfer can clearly outperform several state-of-the-art baseline methods.

## Transferring Structured Knowledge for Planning

In the previous two sections, we introduced two forms of structured knowledge transfer, that is, knowledge transfer in terms of HMM structure and parameter knowledge and user-item ratings at a group level. In this section, we will introduce a higher level of structured knowledge transfer, which is described in terms of the relations between state conditions and actions. The problem is centered on knowledge acquisition in automat-

| Target Data   | Method     | MAE          |
|---------------|------------|--------------|
| MovieLens     | PCC        | 0.930        |
|               | CBS        | 0.874        |
|               | WLR        | 0.915        |
| Book-Crossing | <b>CBT</b> | <b>0.840</b> |
|               | PCC        | 0.677        |
|               | CBS        | 0.664        |
|               | WLR        | 1.170        |
|               | <b>CBT</b> | <b>0.614</b> |

Table 1: MAE on the Two Target Datasets.  
(Average over 10 splits.)

ed planning, where traditionally people have resorted to manual labor to encode action models in each task domain in order to enable a planning system to generate plans to achieve goals. In its simplest form, these action models describe the conditions of when an action applies and the consequences of applying the action in a planning domain, in terms of preconditions and postconditions using a set of literals. Our goal is to learn a good collection of action models from a small number of observed plan traces in a target domain by making use of the action model structures in a related source domain through transfer learning.

Table 2 describes the actions in two domains, one for block-stacking tasks and another for describing the actions of a truck driver. In the second action, the action board-truck has preconditions such as (at ?truck ?loc), and a postcondition (driving ?driver ?truck). Traditionally, these pre- and postconditions are specified by human experts, so that plans such as the ones shown in table 2 can be generated. However, it is often expensive or even infeasible to rely on human experts to generate action models in a new planning domain. Thus, it is desirable to automatically or semiautomatically acquire action models based on observations of plans in these domains.

We now introduce our action-model learning algorithm *t*-LAMP (Zhuo et al. 2008), which stands for transfer learning action models from plan traces and is based on the observation that many planning domains share some common knowledge on action models that can be reused. *t*-LAMP accomplishes transfer learning by exploiting a source planning domain where some action models may have similar functions as those in the target domain. For example, suppose that we wish to learn a logical model for the action “board-truck,” which should describe the state changes when a truck driver moves into the driver seat. This action is one of many in a transportation planning

domain known as the driverlog domain, where the objective is to plan sequences of moves for a number of items to be transported from one location to another under a variety of constraints.

Taking a closer look at the action “board-truck,” we note that it is intended to mean that the driver “?driver” enters a truck “?truck” in a location “?loc,” where “?x” means x is a variable. If this driver “?driver” wants to enter “?truck,” then the driver seat of “?truck” is required to be empty. This is represented by the precondition “(empty ?truck).” We note that this relationship is similar to the action of “stack” from the block-stacking domain, which involves a robot arm moving blocks from one stack of blocks to another. The action “stack” in this domain requires that the block “?y” be clear beforehand (that is, the precondition “(clear ?y)” is satisfied) for the block “?x” to be stacked on block “?y.” Intuitively, the action “board-truck” is similar to “stack,” because the driver and the driver seat in the driverlog domain can be considered as similar to a block. Thus, knowing how to represent the “stack” action in the block-stacking domain is helpful for the task of learning a model of the action “board-truck.” By observing this analogy between the domains, we can reduce the number of training examples significantly.

In the past, many researchers have developed approaches to acquire various knowledge for automated planning. For example, in Benson (1995), and Lorenzo and Otero (2000), inductive logic programming (ILP) approaches are developed to learn action models given the positive and negative examples of states just before an action occurs. These so-called preimages of actions are critical input to the ILP algorithms. In contrast, *t*-LAMP does not require all states before an action be known, nor does it require negative examples. Instead, it relies on the observed sequences of actions to be available at input time. Moreover, unlike these ILP approaches, *t*-LAMP can transfer the action-model knowledge from other domains. Another related work is by Shahaf, Chang, and Amir (2006), who developed an algorithm for simultaneously learning and filtering, or SLAF, of action models. This work is mostly based on logical encoding of the action models and plan traces in the same domain. Other systems such as GIPO (Simpson, Kitchin, and McCluskey 2007) rely on humans to author action models and planning domain knowledge, in which our automated solution can provide some initial solutions for human experts to edit.

Our transfer learning problem is described as follows. In the target domain, we are given a set of action names and predicates, together with their associated parameter lists. We are also given the parameter types of the predicates and actions. In order to help learning, we also have a set of plan

|   |   |
|---|---|
| <b>Input: predicates, action schemas and plan traces from <i>driverlog</i>:</b> |   |
| predicates:   | (at ?obj - locatable ?loc - location) (empty ?t - truck) ...  |
| action schemas:   | (board-truck ?driver - driver ?truck - truck ?loc - location) ...   |
| plan trace 1:   | (at t1 l1) (at d1 l1) (empty t1)(link l1 l2),<br>(board-truck d1 t1 l1) (drive-truck t1 l1 l2 d1), (at t1 l2) (driving d1 t1) |
| plan trace 2: ...   |   |
| ...   |   |
| <b>Input: action models from the source domain <i>blocks</i></b>                |   |
| action model:   | stack(?x - block ?y - block)  |
| preconditions:  | (and (holding ?x) (clear ?y))   |
| effects:  | (and (not (holding ?x)) (not (clear ?y)) (clear ?x) (handempty) (on ?x ?y)))  |

Table 2. Example of the Learning Problem (Input/Output).

traces  $\mathcal{T}$  that are observed through various sensors (see the input part of table 2). Some state information is also observed along the plan traces. *t*-LAMP is built based on our previous system, ARMS (Yang, Wu, and Jiang 2007), which acquires action models from a large number of observed plan traces. ARMS converts these plan traces, as well as a collection of background knowledge, into weighted satisfiability formulas. It then uses a MAXSAT algorithm to satisfy as many constraints as possible, which results in a plausible action model. A drawback of the ARMS system is that, when the number of traces is small and the intermediate state information is sparse, ARMS will suffer from too much noise, unless we provide many plan traces and intermediate states. In *t*-LAMP, this problem is alleviated by introducing a different but similar planning domain, where the action models bear some similarity to that in the target domain. In the *t*-LAMP system, we consider as part of our input a source domain where some action models are known; these are known as the source action models. Let the set of all action schemas (that is, action names together with parameters) that occur in  $\mathcal{T}$  be  $\mathcal{A}$ . *t*-LAMP outputs the preconditions and effects of each action schema in  $\mathcal{A}$  that it learns. An example of the input and output is shown in table 2. We wish that the inclusion of the source domain can help us improve the quality of learning with less collected state or plan trace information in the target domain. In the transfer learning scheme, the knowledge structure that is being transferred corresponds to the action models, including the pre- and postconditions.

*t*-LAMP learns in the following steps. It first encodes the plan traces as conjunctions of states and state transitions. It then generates all possible candidate formulas that satisfy the planning domain constraints listed in formulas F1–F3 in the target domain. Other formulas can be added to make the solution space tighter and the learned

actions respect the domain constraints. To enable transfer learning, it also encodes the action models of a source domain as a set of formulas and builds mappings between the formulas from the source domain to the candidate formulas from the target domain. Once all these formulas are ready, it finally optimizes the weights of the candidate formulas and generates the final action models in the target domain. Below, we give more details of each step.

In the first step, *t*-LAMP encodes the plan traces as propositional formulas. States and state transitions will be encoded. To do this, we introduce a new parameter in predicates in a way similar to situation calculus (Levesque, Pirri, and Reiter 1998), so that the transition from the state  $s_1$  to the state  $s_2$  can be represented by  $(at\ d1\ l1\ s_1) \wedge \neg(driving\ d1\ t1\ s_1) \wedge \neg(at\ d1\ l1\ s_2) \wedge (driving\ d1\ t1\ s_2)$ . In addition, the fact that the action (board-truck  $d1\ t1\ l1$ ) causes the transition can be represented by a propositional variable ( $board-truck\ d1\ t1\ l1\ s_1$ ). Thus, the transition function  $\gamma(s_1, (board-truck\ d1\ t1\ l1))$  can be represented as  $(board-truck\ d1\ t1\ l1\ s_1) \wedge (at\ d1\ l1\ s_1) \wedge \neg(driving\ d1\ t1\ s1) \wedge \neg(at\ d1\ l1\ s_2) \wedge (driving\ d1\ t1\ s2)$ . In this way, we encode a plan trace as a collection of propositions.

In the next step, *t*-LAMP generates candidate formulas that describe all target-domain action models. If there are  $M$  grounded predicates in the domain, then the space of potential action models will be exponential in  $M$ , since every subset of  $M$  can be a potential set of preconditions and postconditions of an action model  $A$ . While this space can be very large, we can effectively impose additional constraints to limit the search. These domain-knowledge constraints are described as formulas F1 to F3, which are designed to ensure the correctness of the learned action models. For example, the second requirement, F2, shows that the negation of a literal, as part of the effect of an action, must have its corresponding positive literal satisfied before the action is performed.

## Formula F1

(The effect of an action must hold after the execution of this action.) If a literal  $p$  is an effect of some action  $a$ , then an instance of  $p$  is added after  $a$  is executed.

## Formula F2

(The negative effect of an action must have its corresponding positive counterpart hold before the execution of this action.) Similar to F1, a literal  $p$ 's negation is an effect of some action  $a$ , which means an instance of  $p$  is deleted after  $a$  is executed, but it exists before the execution of  $a$ .

## Formula F3

(The precondition of an action will be the subset of the state before the execution of this action.) A formula  $f$  (can be a single literal, or with quantifiers) is a precondition of  $a$ , which means that the instance of  $f$  holds before  $a$  is executed.

The most important step of  $t$ -LAMP is to encode the possible mappings between the two domains to enable knowledge transfer. In this step, we ask the question "is an action  $A$  in the source domain similar to action  $B$  in the target domain in such a way that it might ease our learning of  $B$ ?" While there are potentially many possible mappings between  $A$  and  $B$  in the two domains, in reality such mappings are limited due to a domain characteristic: the mapped actions are required to have similar correspondences of preconditions and postconditions, and the parameters of these pre- and postconditions must correspond to each other as well. For example, when mapping "(clear ?y)" of the domain *blocks* and "(empty ?truck)" of the domain *driverlog*, the *types* of "?y" and "?truck," *block* and *truck*, respectively, are mapped, and this mapping will be used to constrain the mappings of other predicates. In general, given  $N$  predicates in the source domain and  $M$  predicates in the target domain, the number of mappings between predicates from the two domains can be estimated by  $L \times N \times M$ , where  $L$  is the number of mappings between two sets of parameters of predicates.

The mapping building process can be considered as a way to make inferences on analogy between the domains. Reasoning by analogy has a long tradition in artificial intelligence (Falkenhainer, Forbus, and Gentner 1986), where  $t$ -LAMP can be seen as going through a scalable matching process to draw analogy between two planning domains to maximize the knowledge to be transferred through action encodings. By building the mappings, we bridge the source domain and the target domain to transfer the structured knowledge in the form of action models. The remaining task for us is to choose the best mapping from the final set of propositional formulas. We can attach weights to these formulas to describe their criticality in final solution. This set of formulas can be considered as an input to a global satisfiability problem, which can be solved using a weighted MAXSAT problem.

Once a MAXSAT problem is set up, a variety of optimization packages can be applied (Borchers and Furman 1997; Richardson and Domingos 2006). MAXSAT attempts to use local search to find a plausible solution to a global constraint-satisfaction problem, where the constraints are taken as input. A limitation of this approach is that no new predicates are invented in the process of search, which is future work to be considered.

To evaluate  $t$ -LAMP, we collected some plan traces, which are observed action sequences, from a number of planning domains: *blocks*<sup>8</sup> and *driverlog*.<sup>9</sup> These traces are generated by creating plans from some given initial and goals states using given action models and the FF planning algorithm.<sup>10</sup> As an evaluation metric, we define error rates of our learning algorithm as the difference between our learned action models and the handwritten action models, where the latter are considered as the "ground truth" in a domain, such as IPC-2. If a precondition appears in our learned action models' preconditions but not in the hand-coded action models' preconditions, the error count of preconditions, denoted by  $E(pre)$ , increases by one. If a precondition appears in some human-generated action models' preconditions but not in our learned action models' preconditions, then the count of  $E(pre)$  is incremented by one. Likewise, the error counts of effects are denoted by  $E(eff)$ . Furthermore, we denote the total number of all the possible preconditions and effects of action models as  $T(pre)$  and  $T(eff)$ , respectively. The error rate of a learned action model is defined as  $R(a) = 1/2 (E(pre) + E(eff)/T(pre) + T(eff))$  where we assume that the error rates of preconditions and effects are equally important, and the error rate  $R(a)$  is within the  $[0, 1]$  range.

Table 3 shows the reduction in error rates of  $t$ -LAMP in transferring from a block-stacking domain to the truck-driving domain, where the percentage is calculated by comparing the error rates of action models learned by  $t$ -LAMP to that obtained from models learned by ARMS (Yang, Wu, and Jiang 2007) (see column two of the table), which does not use transfer learning. In the experiments we did not compare to other systems such as ILP algorithms, because they typically require a different set of inputs. From the table, we can see that the error rates are generally reduced when we apply transfer learning to acquire the action models. More experimental results are shown in Zhuo et al. (2008), which also support our conclusion here.

In summary, our experiments show that in most cases, the more plan traces are available, the lower the error rates will be. This observation is consistent with our intuition. The learned action models form a rough outline of the action's structure, which can be directly used as input to an automated planning system or as initial action models for a human knowledge engineer to adapt from. In

## Notes

1. For example, see <http://foursquare.com>.
2. However, a changing  $\pi_0$  can be seen as an extension.
3. The error distance is 3 meters, which means that predictions within 3 meters of the true location are counted as correct ones.
4. See [www.cse.ust.hk/~qyang/ICD-MDMC07](http://www.cse.ust.hk/~qyang/ICD-MDMC07).
5. See [www.cs.cmu.edu/~lebanon/IR-lab.htm](http://www.cs.cmu.edu/~lebanon/IR-lab.htm).
6. See [www.grouplens.org/node/73](http://www.grouplens.org/node/73).
7. See [www.informatik.uni-freiburg.de/~chiegler/BX](http://www.informatik.uni-freiburg.de/~chiegler/BX).
8. See [www.cs.toronto.edu/aips2000](http://www.cs.toronto.edu/aips2000).
9. See [planning.cis.strath.ac.uk/competition](http://planning.cis.strath.ac.uk/competition).
10. See [members.deri.at/~joergh/ff.html](http://members.deri.at/~joergh/ff.html).

## References

- Bahl, P., and Padmanabhan, V. N. 2000. Radar: An In-Building RF-Based User Location and Tracking System. In *Proceedings of the Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies Infocom (2)*, 775–784. Piscataway, NJ: Institute of Electrical and Electronics Engineers.
- Ben-David, S., and Schuller, R. 2003. Exploiting Task Relatedness for Multiple Task Learning. In *Proceedings of the Sixteenth Annual Conference on Learning Theory*, 825–830. San Francisco: Morgan Kaufmann Publishers.
- Benson, S. 1995. Inductive Learning of Reactive Action Models. In *Proceedings of the Twelfth International Conference on Machine Learning*, 47–54. San Francisco: Morgan Kaufmann Publishers.
- Blitzer, J.; Dredze, M.; and Pereira, F. 2007. Biographies, Bollywood, Boom-Boxes, and Blenders: Domain Adaptation for Sentiment Classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, 432–439. Stroudsburg, PA: Association of Computational Linguistics.
- Blitzer, J.; McDonald, R.; and Pereira, F. 2006. Domain Adaptation with Structural Correspondence Learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 120–128. Stroudsburg, PA: Association of Computational Linguistics.
- Borchers, B., and Furman, J. 1997. A Two-Phase Exact Algorithm for Max-SAT and Weighted Max-Sat Problems. *Journal of Combinatorial Optimization* 2(2): 299–306.
- Caruana, R. 1997. Multitask Learning. *Machine Learning* 28(1): 41–75.
- Dai, W.; Xue, G.; Yang, Q.; and Yu, Y. 2007a. Co-Clustering Based Classification for Out-

| Number of Plan Traces Used in Training | Error Rates of ARMS | Error Reduction by <i>t</i> -LAMP |
|--|---------------------|-----------------------------------|
| 20                                     | 0.346               | 10.1 percent                      |
| 40                                     | 0.302               | 8.9 percent                       |
| 60                                     | 0.203               | 24.6 percent                      |
| 80                                     | 0.139               | 21.6 percent                      |
| 100                                    | 0.078               | 20.5 percent                      |

Table 3: Reduction in Error Rates in the Learned Action Models in the driverlog Domain Using Knowledge Transfer from the block-stacking Domain.

the latter case, we have been collecting evidence on using the result of *t*-LAMP in helping with knowledge engineering and showing the reduction in manual labors. More work is needed in this area, but our initial tests have been encouraging.

## Conclusion and Future Work

In this article, we give an overview on three research works that transferred progressively more sophisticated structured knowledge from a source domain to a target domain to facilitate learning. The simplest of these is a classification problem, which is demonstrated through a WiFi localization problem. In this problem, the structured knowledge to be transferred includes an HMM structure and some key parameters. The benefit is derived from higher classification accuracy in a new time period for localization without incurring too much relabeling of the data. However, the transfer learning is confined to reconstructing the target domain labeled data by means of the source domain data, and to learning the HMM model parameters. This transfer learning task is relatively simple to accomplish in the scale of intelligent endeavor. The next example involves learning clustering-level relations to transfer between recommendation domains, which involves finding and aligning the clusters for transfer. The structured knowledge to be transferred here involves relations among groups of people and items. The last example showed how transferring action model knowledge from domain to domain can help learn bet-

ter action models. Structured knowledge represents action models that relate actions to pre- and postconditions. This is the most sophisticated form of knowledge transfer among the three examples, and the learning task involves reusing procedural knowledge rather than declarative knowledge. Our conclusion is that structured knowledge transfer can be useful in a wide range of tasks ranging from data and model-level transfer to procedural knowledge transfer, and that optimization methods can be applied to extract and transfer deep structural knowledge between a variety of source and target problems.

There is much to be done in transfer learning of structured knowledge. Even through we discussed three successful cases of structured knowledge transfer, an important open problem is how to identify good source domains to transfer from for given target domains. One solution is to study how best to characterize the nature of knowledge transfer between two domains from the view of data and knowledge compression when we consider the source and target domains together. When two domains can be compressed well, they typically share some structural components. Another direction is to come up with a good quantitative measure of similarity between different domains at declarative and procedural levels of learning. For some learning problems, we may be able to identify some intermediate domains where the shared knowledge structures may overlap.

## Acknowledgment

We thank Hong Kong RGC/NSFC Project N HKUST624/09.

- of-Domain Documents. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York: Association for Computing Machinery.
- Dai, W.; Xue, G.; Yang, Q.; and Yu, Y. 2007b. Transferring Naive Bayes Classifiers for Text Classification. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence*. Menlo Park, CA: AAAI Press.
- Daumé, III, H., and Marcu, D. 2006. Domain Adaptation for Statistical Classifiers. *Journal of Artificial Intelligence Research* 26: 101–126.
- Falkenhainer, B.; Forbus, K. D.; and Gentner, D. 1986. The Structure-Mapping Engine. In *Proceedings of the 5th National Conference on Artificial Intelligence*, 272–277. Menlo Park, CA: AAAI Press.
- Levesque, H. J.; Pirri, F.; and Reiter, R. 1998. Foundations for the Situation Calculus. *Electronic Transactions on Artificial Intelligence* 2(2): 159–178.
- Li, B.; Yang, Q.; and Xue, X. 2009. Can Movies and Books Collaborate? Cross-Domain Collaborative Filtering for Sparsity Reduction. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2052–2057. Menlo Park, CA: AAAI Press.
- Lorenzo, D., and Otero, R. P. 2000. Learning to Reason about Actions. In *Proceedings of the 14th European Conference on Artificial Intelligence*, 435–439. Amsterdam: IOS Press.
- Ni, L. M.; Liu, Y.; Lau, Y. C.; and Patil, A. P. 2003. Landmarc: Indoor Location Sensing Using Active RFID. In *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications*, 407–415. Piscataway, NJ: Institute of Electrical and Electronics Engineers.
- Pan, S. J., and Yang, Q. 2010. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*. 22(10): 1345–1359.
- Pan, S. J.; Kwok, J. T.; Yang, Q.; and Pan, J. J. 2007. Adaptive Localization in a Dynamic WiFi Environment through Multi-View Learning. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence*, 1108–1113. Menlo Park, CA: AAAI Press.
- Pollack, M. E. 2007. Intelligent Assistive Technology: The Present and the Future. In *Proceedings of the 11th International Conference on User Modeling*, Lecture Notes in Computer Science 4511, 5–6. Berlin: Springer.
- Raina, R.; Battle, A.; Lee, H.; Packer, B.; and Ng, A. Y. 2007. Self-Taught Learning: Transfer Learning from Unlabeled Data. In *Proceedings of the 24th International Conference on Machine Learning*, 759–766. New York: Association for Computing Machinery.
- Resnick, P.; Iacovou, N.; Suchak, M.; Bergstrom, P.; and Riedl, J. 1994. Grouplens: An Open Architecture for Collaborative Filtering of Netnews. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, 175–186. New York: Association for Computing Machinery.
- Richardson, M., and Domingos, P. 2006. Markov Logic Networks. *Machine Learning* 1-2(62).
- Sarwar, B.; Karypis, G.; Konstan, J.; and Riedl, J. 2001. Item-Based Collaborative Filtering Recommendation Algorithms. In *Proceedings of the 10th International World Wide Web Conference*, 285–295. New York: Association for Computing Machinery.
- Schmidhuber, J. 1994. On Learning How to Learn Learning Strategies. Technical Report FKI-198-94, Fakultät für Informatik, Palo Alto, CA.
- Shahaf, D.; Chang, A.; and Amir, E. 2006. Learning Partially Observable Action Models: Efficient Algorithms. In *Proceedings of the 21st AAAI Conference on Artificial Intelligence*. Menlo Park, CA: AAAI Press.
- Simpson, R. M.; Kitchin, D. E.; and McCluskey, T. L. 2007. Planning Domain Definition Using Gipo. *Knowledge Engineering Review* 22(2): 117–134.
- Srebro, N., and Jaakkola, T. 2003. Weighted Low-Rank Approximations. In *Proceedings of the 20th International Conference on Machine Learning*, 720–727. Menlo Park, CA: AAAI Press.
- Thrun, S., and Mitchell, T. M. 1995. Learning One More Thing. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, 825–830. San Francisco: Morgan Kaufmann Publishers.
- Xue, G.-R.; Lin, C.; Yang, Q.; Xi, W.; Zeng, H.-J.; Yu, Y.; and Chen, Z. 2005. Scalable Collaborative Filtering Using Cluster-Based Smoothing. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 114–121. New York: Association for Computing Machinery.
- Yang, Q.; Pan, S. J.; and Zheng, V. W. 2008. Estimating Location Using Wi-Fi. *IEEE Intelligent Systems* 23(1): 8–13.
- Yang, Q.; Wu, K.; and Jiang, Y. 2007. Learning Action Models from Plan Examples Using Weighted Max-Sat. *Artificial Intelligence* 171(2-3): 107–143.
- Zheng, V. W.; Yang, Q.; Xiang, W.; and Shen, D. 2008. Transferring Localization Models Over Time. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, 1421–1426. Menlo Park, CA: AAAI Press.
- Zhuo, H.; Yang, Q.; Hu, D. H.; and Li, L. 2008. Transfer Learning Action Models with Conditional Effects and Quantifiers. In *Proceedings of the 10th Pacific Rim International Conference on Artificial Intelligence (Prcai'08)*. Lecture Notes in Computer Science 5351. Berlin: Springer.
- Qiang Yang** is a professor in the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, and an IEEE Fellow. His research interests are artificial intelligence (including automated planning, machine learning, and activity recognition) and data mining. He received his PhD from the Computer Science Department of the University of Maryland, College Park in 1989. He was an assistant and then associate professor at the University of Waterloo from 1989 to 1995 and an associate and then full professor and NSERC Industrial Research Chair at Simon Fraser University in Canada from 1995 to 2001. His research teams won the 2004 and 2005 ACM KDDCUP international competitions on data mining. He is the founding editor in chief of the *ACM Transactions on Intelligent Systems and Technology* (ACM TIST) and is on the editorial board of *IEEE Intelligent Systems* and several other international journals. He has been a program committee cochair or general cochair for a number of international conferences, such as ACM KDD 2010, and serves on the IJCAI 2011 advisory committee.
- Vincent W. Zheng** is a PhD candidate in the Department of Computer Science and Engineering, Hong Kong University of Science and Technology. His research interests focus on data mining and machine learning, with applications to human behavior modeling. He is an AAAI member and has publications in data mining and ubiquitous computing conferences such as AAAI, IJCAI, WWW, KDD, Ubicomp, and so on. He also serves as a reviewer for many distinguished computer science conferences and journals.
- Bin Li** received a BEng degree in software engineering from Southeast University, Nanjing, China, in 2004 and a PhD degree in computer science from Fudan University, Shanghai, China, in 2009. He is a postdoctoral research fellow in Institut Télécom SudParis. His research interests include machine-learning and data-mining methods as well as their applications to social network mining, multimedia retrieval, and ubiquitous computing.
- Hankz Hankui Zhuo** is a lecturer in the Department of Computer Science, Sun Yat-sen University in China. He received his PhD from the Computer Science Department of Sun Yat-sen University in 2009. His research interests are automated planning, machine learning, and game theory.