Cross-Domain Activity Recognition

Vincent Wenchen Zheng, Derek Hao Hu, Qiang Yang
Department of Computer Science and Engineering,
Hong Kong University of Science and Technology,
Clear Water Bay, Kowloon, Hong Kong
{vincentz,derekhh,qyang}@cse.ust.hk

ABSTRACT
In activity recognition, one major challenge is huge manual effort in labeling when a new domain of activities is to be tested. In this paper, we ask an interesting question: can we transfer the available labeled data from a set of existing activities in one domain to help recognize the activities in another different but related domain? Our answer is "yes", provided that the sensor data from the two domains are related in some way. We develop a bridge between the activities in two domains by learning a similarity function via Web search, under the condition that the sensor data are from the same feature space. Based on the learned similarity measures, our algorithm interprets the data from the source domain as the data in the domain with different confidence levels, thus accomplishing the *cross-domain knowledge transfer* task. Our algorithm is evaluated on several real-world datasets to demonstrate its effectiveness.

Author Keywords
Activity Recognition, Cross Domain, Transfer Learning, Web Search

ACM Classification Keywords
I.2.6 Learning: Knowledge acquisition

General Terms
Algorithms

INTRODUCTION
With the proliferation of sensor technologies, the task of recognizing human's activities from a series of low-level sensor readings has drawn many interest in AI, user modeling and ubiquitous computing areas [24, 7]. Early activity recognition algorithms are based on logic, in which conclusions are deducted from the observations and "closed world" assumptions [12]. As the sensor data became available, recent activity recognition research focuses more on reasoning under uncertainty. For example, Bui introduced abstract Hidden Markov Model to represent the user's activity hierarchy [2]. Liao et al. applied a hierarchical Markov model to estimate a user's locations and transportation modes [14]. Yin et al. applied a dynamic Bayesian network to infer a user's actions from raw WiFi signals, and an N-gram model to infer the users' high-level goals from actions [30]. Similarly, Patterson et al. also applied a dynamic Bayesian network to recognize fine-grained activities by aggregating the abstract object usage [17]. Hu and Yang used a skip-chain Conditional Random Field and an activity correlation graph to model the concurrent and interleaving activities [8, 9]. In a recent work [10], Hidden Markov Models with an infinite state space was also used to detect abnormal activities from sensor readings.

Although much work has been done in activity recognition, a major challenge still remains: given a new domain of activities, it usually requires a lot of human efforts to label the sensor data for training a recognition model. Data labels are usually meaningful activity terms associated with the data vectors, such as the sensor readings. If the labeled data for the target activities are few, then the trained activity recognizer may not perform well. Such an issue has become a critical challenge for apply activity recognition systems to practice. In the real world, users may only have a small amount of time and effort to set up an activity recognition system; otherwise, they are quite likely to quit using such a recognition system. Furthermore, users usually do not have the expertise in activity recognition research. Therefore when we try to design some activity recognition algorithm, we should make the algorithm as simple as possible. In terms of data labeling effort, we wish to ensure that the users need not label all activities; if they have done some labeling, it is best for our system to automatically *transfer* the labeled knowledge to help recognize other different, but related activities. In this paper, we show how to transfer the available labeled data from a set of activities to help train a recognizer for another set of different, but related activities.

Consider an example in Figure 1. The taxonomy is an activity taxonomy extracted from the MIT PLIA1 dataset [11, 8], representing the common daily activities. Suppose that some user wants to set up an activity recognition system at his/her home to recognize the activities in the taxonomy. However, the user only wants to spend very little amount of time and effort to label the sensor data, where labels corresponds to activity names such as 'washing dishes'. A user may not be able to label the sensor data associated with all activities described in the taxonomy. For example, the user may only label the sensor data from the activities in the "Cleaning Indoor" category, and leave unlabeled the sensor data from the

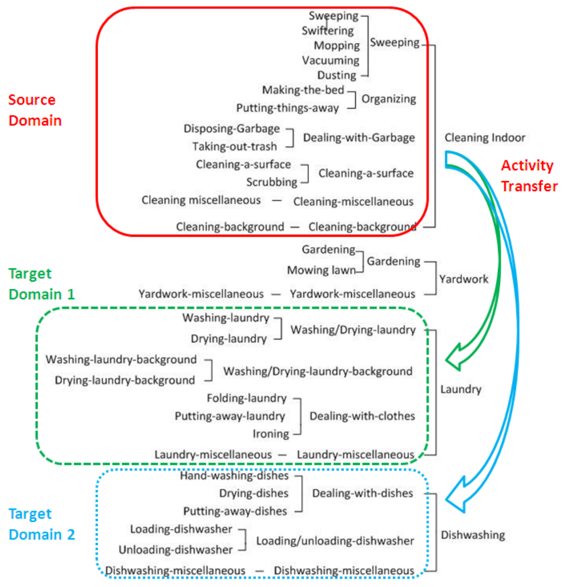


Figure 1. An example of cross-domain activity recognition.

other categories' activities (*e.g.* in "Laundry", "Dishwashing"). So in our problem, we have a *source domain* of activities that has the labeled sensor data from activities in the "Cleaning Indoor" category. We also have some *target domain* that has the unlabeled sensor data from the activities in some other category such as "Laundry" (denoted as "Target Domain 1") or "Dishwashing" (denoted as "Target Domain 2"). Then, we ask the following fundamental questions:

1. *Is it possible for us to use the labeled data in the source domain to help train an activity recognizer in the target domain?* For example, can we use the sensor data from the "Cleaning Indoor" category in training an activity recognizer for the "Laundry" category?

2. *Under what conditions can domain transfer work for activity recognition?*

In this paper, we answer the above questions by presenting a novel algorithm for cross-domain activity recognition (CDAR), which can transfer the labeled data from a source domain to a target domain under the condition that (1) the activities in the source and target domains are related by some Web pages and thus we can build a mapping between them using the Web, and (2) the underlying feature spaces are identical between the source and target domains (they use the same set of sensors, although activity labels can be different). We observe that although the activities in the source domain and the target domain are different, some of them are similar in semantics as well as the corresponding sensor data. For example, in the above Figure 1, one may transfer useful information from activity "Washing-laundry" to "Hand-washing dishes", considering that, although these two activities are different, the underlying physical actions one performs for these two activities are similar, i.e. hand-washing. Therefore, if we have sensors attached to the body arms or hands, the accelerometer values or other motion values detected should be similar for us to transfer the knowledge from the source domain to the target domain. Intuitively, we use similar activities in the source domain to help enrich the labeled data for the target domain. Specifically, we first learn a similarity function between activities in both domains by exploiting Web search and applying information retrieval techniques. We then train a (multi-class) weighted Support Vector Machine (SVM) model with different probabilistic confidence weights learned from the similarity function. Experiments on real world data sets show that the transferred activity recognizer can indeed improve performance by using the auxiliary data and outperform some other state-of-the-art algorithms.

RELATED WORK

Activity recognition aims to infer a user's behaviors from the observations such as sensor data, and has various applications including medical care [19], logistics service [13], robot soccer [25], plan recognition [12], *etc*. However, most of the proposed activity recognition algorithms are focused on only data from one domain, and usually require a lot of labeled data in order to train the recognition model.

Our work exploits the Web to connect two domains. In the past, some previous research works had considered learning common sense knowledge from the Web (such as ehow and KnowItAll [5, 23]) to assist model training. For example, Perkowitz et al. proposed to mine the natural language descriptions of activities (*e.g.* "making-tea") from ehow.com as labeled data, and translated them into probabilistic collections of object terms (*e.g.* "teacup", "teabag", etc.) [18]. Then, they use these probabilistic collections as input data to train a dynamic Bayesian network model for prediction. Wyatt et al. also proposed to mine recipes of the activities from the Web as the labeled data, but they only use this knowledge as a prior and trained a Hidden Markov Model from the unlabeled RFID sensor data [29]. Wang et al. further improved this work by utilizing personal activity data from wearable sensors [28]. They first extracted the actions from the wearable sensors, and then incorporated the actions with the object usage to finally predict the activities. Most previous approaches either exploited only object-usage information or required explicit action modeling. Few of them exploited auxiliary source domains for activity recognition.

As we aim to transfer the labeled data across domains, our work is also related to *transfer learning*, which is a state-of-art learning paradigm in machine learning [3]. Transfer learning aims at transferring knowledge from some source domains to a target domain. In general, the data from the both domains may follow different distributions or be represented in different feature spaces. There are several main approaches to transfer learning in the past. The first approach can be referred to as the instance-transfer [22], where the training examples in a source domain are weighted for better learning in a target domain. The second approach can be referred to as the feature-representation-transfer [21], which finds a "good" feature representation that reduces difference between both domains for training. The third approach is parameter-transfer [6], which discovers shared parameters or priors between the models in both domains. The fourth approach is the relational-knowledge-transfer, which builds the mapping of relational knowledge between both domains [16]. Our work can be seen as an instance-transfer approach. However, our work is different from the most previous works, because they usually assume that the domains can have different feature spaces but share the same label space. In our work, we consider that the domains can share the same feature space (*e.g.* a same set of sensors at a home), but have different label spaces (*i.e.* different activities). We also notice that, Kasteren et al. have tried to apply transfer learning to help train an activity recognition model in a new house [26]. But they also considered the two domains (houses) have the same label space, which is different from our work.

PROBLEM FORMULATION

We consider two domains that have the same set of sensors spanning a feature space but have different activity (label) spaces. Specifically, we have a source domain with a set of activities $\mathcal{A}_{src} = \{a_1, ..., a_m\}$, and a target domain with another set of activities $\mathcal{A}_{tar} = \{a_{m+1}, ..., a_n\}$. $\mathcal{A}_{src}$ and $\mathcal{A}_{tar}$ do not overlap, *i.e.* $\mathcal{A}_{src} \bigcap \mathcal{A}_{tar} = \emptyset$. In other words, we have two sets of activities, one of which called source domain and the other of which is called target domain. In the source domain, the sensor readings are all labeled with activity names, and they will be used as training data. In the target domain, we don't have any labeled data for training, but we know how many activities are in the target domain and what their names are. Our aim is that, given some sensor readings (*i.e.* test data) from the target domain, we can use the labeled training data from the source domain to learn a recognizer and thus recognize their activity labels.

Notice that we are constraining our training data in the source domain whereas the test data is in the target domain. We will not test the sensor readings drawn from the source domain in the testing phase. Such a setting is reasonable due to two reasons. First, as the source domain has plentiful labeled data, using traditional learning approaches such as Naive Bayes, decision tree or support vector machines would be enough to recognize the source domain's activities. So in this paper, we are more focused on recognizing those activities from the target domain without any labeled data. Second, it is not hard for us to identify whether a sensor reading is drawn from the source domain or the target domain, by using some external sensor information. For example, we can have a location sensor in the house to identify where the user is, so we can know a sensor reading related to the user belongs to the source domain's activities (say, "making-the-bed" in the bedroom) or the target domain's activities (say, "laundry" in the laundry closet).

We also make an underlying assumption that the source domain's activities and the target domain's activities do have some kind of relationship. For example, "laundry" and "cleaning Indoor" are related because they both involve some kind of "cleaning". However, "laundry" and "watching TV/movies" may only be weakly related, so we may not be able to transfer that much useful knowledge from "laundry" to "'watching TV/movies" and hence the algorithm may not perform well. We studied the impact of such domain differences in the experiment section.

To be more precise, let $x \in \mathbb{R}^k$ be a k-dimensional sensor reading (*i.e.* feature) vector at some time slice, and y be a random variable whose value represents an activity (*i.e.* label). In the source domain, we have plentiful labeled training data $\mathcal{D}^{trn}_{src} = \{(x^{(i)}_{src}, y^{(i)}_{src})\}^{T_1}_{i=1}$, where $y^{(i)}_{src} \in \mathcal{A}_{src}$. In the target domain, we do not have any training labeled data; instead, we only have some test data $\mathcal{D}^{tst}_{tar} = \{x^{(j)}_{tar}, y^{(j)}_{tar}\}^{T_2}_{j=1}$, where $y^{(j)}_{tar} \in \mathcal{A}_{tar}$ are used as ground truth for testing only. We would note that the source domain's sensor data and the target domain's sensor data share the same feature space in $\mathbb{R}^k$, but the two domains have different label spaces.

In this paper, we make a simplification; that is, we break the sequences into time slices each, and omit the possible sequential information we could take advantage of in dealing with the problem of cross-domain activity recognition. There are several reasons for this. The first reason is that, when we constrain ourselves to using training data from only one subset of activities from the original training data (source domain), we are already "choosing" the activities in the sequences and have damaged the sequential information contained in the original data. The second reason is that, our paper, being the first paper to tackle the problem of cross-domain activity recognition formally, would put more emphasis on how to transfer useful knowledge between different activity sets; or, more loosely speaking, how to calculate similarities between different activities and demonstrate that such a simple method is indeed effective in cross-domain activity recognition tasks. Therefore, in this paper, we omit the sequential information and treat each time sequence of sensor readings with length T as T instances in the training or testing datasets.

PROPOSED APPROACH

Algorithm Overview

Our work belongs to instance-transfer category in transfer learning framework. In general, the instance-transfer algorithms are motivated by data instance importance sampling [22]. That is, the training data from a source domain are weighted to train a model for the target domain, and the weights can be generally seen as the similarities between the source domain's data and the target domain's data. The more similar some source domain's data are to the target domain's data, the higher weights the source domain's data will have in learning. Different from the previous work on instance-transfer which measures the similarities from the data (features), we show that in our cross-domain activity recognition problem, we need to measure the similarities from the label information.

We first present an overview of our cross-domain activity recognition (referred to as CDAR below) algorithm to provide the readers with a high-level sense of our algorithm. Our CDAR algorithm can be generalized into three steps.

In the first step, we aim to learn a similarity function between different activities by mining knowledge from the Web. In particular, we will use Web search to extract related Web pages for the activities, and then apply information retrieval techniques to further process the extracted Web pages. After that, we will use some similarity measure, such as Maximum Mean Discrepancy in Eq. (2), to calculate the similarities between any pair of activities from source domain and target domain. Such similarities will be used later to propagate labels for domain transfer.

In the second step, given the assumption that we only have labeled training data in the source domain but no labeled training data in the target domain, it is impossible to follow supervised learning methods to train a recognizer for the target domain's activities. Therefore, by using the similarity values we have learned in the first step, we aim to generate some pseudo training data for the target domain with some confidence values. Here "pseudo training data" are the training data with the same feature values as in the source domain, but relabeled with the activity labels in the target domain. Such data relabelings will be assigned with some confidences, whose values equal to the similarities we calculated in the first step; and these confidences will measure how "strong" a particular training data instance in the source domain can be explained as the data instances in the target domain.

In the third step, by using the pseudo training data, we can apply a weighted Support Vector Machine method [4] to train a classifier, so that we can use it to recognize the activities in the target domain.

In the following sections, we would describe each step of our algorithm in detail.

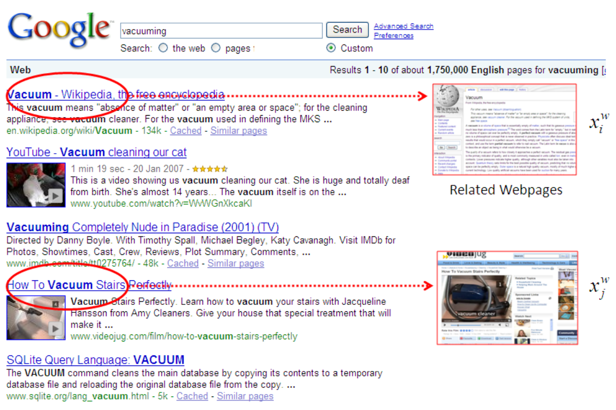
Learning the Similarity Function

In this section, we will show how to learn a similarity function for any pair of activities from the source domain and the target domain. To achieve this, we will novelly exploit the Web data.

Calculate Similarity from Web Data

With the proliferation of the Web services, there are emerging Web pages that describe the daily activities. For example, we can easily find many web pages introducing how to make coffee. Such web pages encode the human understanding to the activity semantics, such as what kind of activity it is, what kind of objects it uses, *etc*. This semantics can greatly help in measuring the similarities between the activities.

In practice, as the activity names are known, we can employ Web search to extract web pages related to the activities. For example, for an activity "Vacuuming" defined in the taxonomy of Figure 1, we can search on Google with the query "Vacuuming" as shown in Figure 2. Then we can get a list of search results on the page. By clicking all search results, we can get a set of Web pages. Although the Web pages contain a lot of information, only a small amount of it is related



Results **1** - **10** of about **1,750,000 English** pages for **vacuuming**

Vacuum - Wikipedia, the free encyclopedia
This **vacuum** means "absence of matter" or "an empty area or space"; for the cleaning appliance, see **vacuum** cleaner. For the **vacuum** used in defining the MKS ...
en.wikipedia.org/wiki/**Vacuum** - 134k - Cached - Similar pages

YouTube - **Vacuum** cleaning our cat
1 min 19 sec - 20 Jan 2007 - ★★★★★
This is a video showing us **vacuum** cleaning our cat. She is huge and totally deaf from birth. She's almost 14 years... The **vacuum** itself is on the ...
www.youtube.com/watch?v=WvWGnXkcaKI

Vacuuming Completely Nude in Paradise (2001) (TV)
Directed by Danny Boyle. With Timothy Spall, Michael Begley, Katy Cavanagh. Visit IMDb for Photos, Showtimes, Cast, Crew, Reviews, Plot Summary, Comments, ...
www.imdb.com/title/tt0275764/ - 48k - Cached - Similar pages

How To **Vacuum** Stairs Perfectly
Vacuum Stairs Perfectly. Learn how to **vacuum** your stairs with Jacqueline Hansson from Amy Cleaners. Give your house that special treatment that will make it ...
www.videojug.com/film/how-to-**vacuum**-stairs-perfectly

SQLite Query Language: **VACUUM**
The **VACUUM** command cleans the main database by copying its contents to a temporary database file and reloading the original database file from the copy. ...
www.sqlite.org/lang_**vacuum**.html - 5k - Cached - Similar pages

Figure 2. Extract Web data for activities.

to the semantics of the searched activity. So we apply the information retrieval to retrieve the useful information for each Web page.

In particular, for each Web page, we first extract all the text in it, and then treat the extracted text as a document d_i with a bag of words. Such a document d_i can further processed as a vector x_i^w, each dimension of which is the term frequency-inverse document frequency (tf-idf) [15] of a word t:

$$tf\text{-}idf_{i,t} = \frac{n_{i,t}}{\sum_l n_{i,l}} \cdot \log \frac{|\{d_i\}|}{|\{d_i : t \in d_i\}|},$$

where $n_{i,t}$ is the number of occurrences of word t in document d_i. Besides, $|\{d_i\}|$ is the total number of collected documents, and $|\{d_i : t \in d_i\}|$ is number of documents where word t appears. The terms in the tf-idf equation are explained as follows:

- The first term $\frac{n_{i,t}}{\sum_l n_{i,l}}$ denotes the frequency of the word t that appears in the document d_i. If the word t appears more frequently in the document d_i, then $|\{d_i : t \in d_i\}|$ is larger, and thus the whole term is larger. For example, in the returned Web page of "Vacuuming", the word "clean" may appear many times, so its term frequency is high. It means that such a word encodes some semantics of "Vacuuming", and it has higher weights in the Web data's feature vector.

- The second term $\log \frac{|\{d_i\}|}{|\{d_i : t \in d_i\}|}$ denotes the inverse document frequency for the word t. If the word t appears in more documents of the corpus, then $|\{d_i : t \in d_i\}|$ is larger, and thus the whole term is smaller. For example, for the word "the", it is used in almost all the documents, but it is a stop word without any meaning. Hence, its inverse document frequency will vanish to zero, thus the whole tf-idf value of the word is zero. It means that such a word does not encode any semantics of the searched activity, so it can be removed from the Web data's feature vector.

Therefore, for an activity u (*e.g.* "Vacuuming"), we can get a set of documents $\mathfrak{D}_u^w = \{x_i^w | i = 1, ..., m_u\}$, with each x_i^w as a tf-idf vector. Similarly, for another activity v (*e.g.*

"Washing-laundry"), we can also Google it and get another set of documents $\mathfrak{D}_v^w = \{z_i^w | i = 1, ..., m_v\}$, with each z_i^w as a tf-idf vector.

After having the extracted Web data $\mathfrak{D}_u^w$ and $\mathfrak{D}_v^w$, now we will show how to measure the similarity between the activity u and the activity v. Note that a possible choice to calculate the similarity between two (Web) data distributions is using the Kullback-Leibler (KL) divergence as [29] did. However, generally the Web text data are high-dimensional and it is hard to model the distributions over the two different data sets. Hence, we propose to use the Maximum Mean Discrepancy (MMD) [1], which can directly measure the distribution distance without density estimation, to calculate the similarity.

DEFINITION 1. *Let $\mathcal{F}$ be a class of functions $f : \mathcal{X} \to \mathbb{R}$. Let p and q be Borel probability distributions, and let $\mathcal{X} = (x_1, ..., x_m)$ and $\mathcal{Z} = (z_1, ..., z_n)$ be* i.i.d. *samples drawn from distributions p and q, respectively. Then, the Maximum Mean Discrepancy (empirical estimation) is*

$$MMD[\mathcal{F}, \mathcal{X}, \mathcal{Z}] = \sup_{f \in \mathcal{F}} \left(\frac{1}{m}\sum_{i=1}^{m} f(x_i) - \frac{1}{n}\sum_{i=1}^{n} f(z_i)\right).$$

Considering the universal reproducing kernel Hilbert spaces (RKHS), we can interpret the function f as the feature mapping function $\phi(\cdot)$ of a Gaussian kernel [1].

Given the Web data $\mathfrak{D}_u^w = \{x_i^w | i = 1, ..., m_u\}$ for activity u and the Web data $\mathfrak{D}_v^w = \{z_i^w | i = 1, ..., m_v\}$ for activity v, we can finally have the similarity between u and v as

$$sim(u, v) = MMD^2[\mathfrak{D}_u^w, \mathfrak{D}_v^w], \tag{1}$$

where $MMD^2[\mathfrak{D}_u^w, \mathfrak{D}_v^w]$ is the maximum mean discrepancy defined as:

$$\begin{aligned} MMD^2[\mathfrak{D}_u^w, \mathfrak{D}_v^w] &= \left\| \frac{1}{m_u}\sum_{i=1}^{m_u} \phi(x_i^w) - \frac{1}{m_v}\sum_{i=1}^{m_n} \phi(z_i^w) \right\|_{\mathcal{H}}^2 \\ &= \frac{1}{m_u^2}\|K_{uu}^w\|_1 - \frac{2}{m_u m_v}\|K_{uv}^w\|_1 + \frac{1}{m_v^2}\|K_{vv}^w\|_1, \end{aligned} \tag{2}$$

where K_{uv}^w is the Gaussian kernel defined over the data $\mathfrak{D}_u^w$ and $\mathfrak{D}_v^w$. Specifically, K_{uv}^w is a $m_u \times m_v$ matrix, with its entry at row i and column j defined as

$$K_{uv}^w(x_i^w, z_j^w) = \exp\left(-\frac{\|x_i^w - z_j^w\|^2}{2\sigma^2}\right),$$

where σ is the kernel width for the Gaussian kernel function. In Equation (2), $\|\cdot\|_1$ is an entry-wise norm which sums up all the entries in the matrix.

Generating Pseudo Training Data

Now we have the similarity value $\mathrm{sim}(u, v)$ for each pair of activities $u \in \mathcal{A}_{src}$ and $v \in \mathcal{A}_{tar}$. How can we generate a new training data set defined over the label space of the target domain? Recall that, in the source domain, we have the training labeled data $\mathcal{D}_{src}^{trn} = \{(x_{src}^{(i)}, y_{src}^{(i)})\}_{i=1}^{T_1}$, where $y_{src}^{(i)} \in \mathcal{A}_{src}$. For each training instance $(x_{src}^{(i)}, y_{src}^{(i)})$ with $y_{src}^{(i)} = u$ where $u \in \mathcal{A}_{src}$, we will relabel it to get a set of pseudo training data as $\{(x_{src}^{(i)}, v_j, sim(u, v_j)) | v_j \in \mathcal{A}_{tar}\}_{j=1}^{|\mathcal{A}_{tar}|}$. Here, the similarity $sim(u, v_j)$ between activity u and v_j is used as the confidence of such a relabeling. In other words, we duplicate each training instance $|\mathcal{A}_{tar}|$ times; and each duplication will be relabeled using one activity category in the target domain with some confidence. Finally, these relabeled training data duplications, which we call "pseudo" training data, are then used for training classifiers to classify activities in the target domain.

Weighted SVM Method

Now we have a pseudo training data set on the target domain where each data instances in the dataset contains not only a category label but also a confidence value. The confidence value is defined as the similarity value we calculate between two activities. Therefore, the larger the value is, the more similar the two activities are, and the more confident we are when interpreting such a training data instance to this activity in the target domain.

However, training support vector machines with confidence values attached to training instances is a non-trivial task and we apply the method proposed in [4] to accomplish our goal. Interested readers can follow the original paper for technical details. Here we will briefly introduce the weighted SVM model for multi-class classification.

In [4], a "one-against-one" approach is employed for multi-class classification. Given the N classes (in our case, each activity in the target domain is a class, so $N = |\mathcal{A}_{tar}|$), this approach constructs $N(N-1)/2$ classifiers, each of which trains the data from two different classes. For training data from the i^{th} and the j^{th} classes, the weighted SVM model solves the following two-class classification problem:

$$\begin{aligned} \min_{\mathbf{w}^{ij}, b^{ij}, \xi^{ij}} \quad & \frac{1}{2}(\mathbf{w}^{ij})^T\mathbf{w}^{ij} + C_t^i \sum_{y_t=i} (\xi^{ij})_t + C_t^j \sum_{y_t=j} (\xi^{ij})_t \\ s.t. \quad & (\mathbf{w}^{ij})^T\phi(x_t) + b^{ij} \geqslant 1 - \xi_t^{ij}, \quad if \ \ y_t = i, \\ & (\mathbf{w}^{ij})^T\phi(x_t) + b^{ij} \leqslant -1 + \xi_t^{ij}, \quad if \ \ y_t = j, \\ & \xi_t^{ij} \geqslant 0. \end{aligned} \tag{3}$$

Here, x_t is the t^{th} data instance, y_t is its class label. $\phi(x_t)$ is a feature mapping to x_t. $\mathbf{w}^{ij}$ is the model parameter, b^{ij} is the bias term, and ξ^{ij} is the slack variable denoting the classification error. C_t^i and C_t^j are the weights for the t^{th} instance of i^{th} and j^{th} classes respectively. C_i^t and C_t^j are derived using the similarity function learned from the previous step; and they reflect the confidence values of the data instances x_t interpreted as being from the i^{th} class (*i.e.* activity) in the target domain. In other words, the pseudo training data point x_t is from the i^{th} class with confidence value of C_t^i. Therefore, in Eq. (3), the first term makes sure that in training the support vector machine the margin is maximized; the second and third terms controls the weighted classification errors for both classes. Intuitively, if the weight C_t^i is higher, the pseudo training data instance

x_t from the i^{th} class are more trusted in training the SVM model.

After the optimization in Eq. (3) is solved, [4] uses a voting strategy for multi-class classification. In particular, each binary classification for the i^{th} and the j^{th} classes is considered to be a vote. Then, the votes can be cast for all data instances x_t, and in the end each x_t is designated to be in a class with maximum number of votes. In case that two classes have identical votes, the one with the smallest class index is simply chosen.

Cross-Domain Activity Recognition (CDAR) Algorithm

Finally, we summarize our CDAR method in Algorithm 1. As shown in Algorithm 1, at the first 3 steps, we extract the Web pages for each activity from both domains, and apply the information retrieval technique to transform the Web pages into tf-idf vectors. At step 4, after having the Web data (*i.e.* a set of tf-idf vectors) for each activity, we compute a similarity matrix for each pair of activities between the source domain and the target domain. At step 5, based on the learned similarities, we generate the pseudo training data by relabeling each training instance with the activity labels from the target domain. Each relabeled training instance is assigned with some confidence (weight), which equals to the similarity between its original activity label (from the source domain) and the newly given activity label (from the target domain). At step 6, we train the CDAR model using a weighted Support Vector Machine. At step 7, we use the trained weighted SVM classifier to performing testing on the target domain's data.

Algorithm 1 Algorithm for CDAR

```
Input: Source domain has $T_1$ labeled training data $\mathcal{D}_{src} = \{(x_{src}^{(i)}, y_{src}^{(i)})\}_{i=1}^{T_1}$, where $y_{src}^{(i)} \in \mathcal{A}_{src}$. Target domain does not have any labeled training data; instead it has $T_2$ test data $\mathcal{D}_{tar} = \{(x_{tar}^{(j)}, y_{tar}^{(j)})\}_{j=1}^{T_2}$, where $y_{tar}^{(j)} \in \mathcal{A}_{tar}$ are the ground truth labels for testing only.
Output: Predicted labels on the test data in target domain.
begin
  1: For each activity $u \in \mathcal{A}_{src}$, extract a list of Web pages from some search engine (such as Google);
  2: For each $u$'s Web pages, apply information retrieval technique and transform each Web page to a tf-idf vector, and form a Web data set $\mathfrak{D}_u^w$;
  3: Similar to the above two steps, extract a Web data set $\mathfrak{D}_v^w$ for each activity $v \in \mathcal{A}_{tar}$;
  4: For any two activities $u \in \mathcal{A}_{src}$ and $v \in \mathcal{A}_{tar}$, calculate the similarity $sim(u, v) = MMD^2(\mathfrak{D}_u^w, \mathfrak{D}_v^w)$ using the maximum mean discrepancy in Eq.(2);
  5: Generate pseudo training data as follows: each training instance $(x_{src}^{(i)}, y_{src}^{(i)})$ with $y_{src}^{(i)} = u$ where $u \in \mathcal{A}_{src}$, is relabeled to get a set of pseudo training data as $\{(x_{src}^{(i)}, v_j, sim(u, v_j))|v_j \in \mathcal{A}_{tar}\}_{j=1}^{|\mathcal{A}_{tar}|}$ with $sim(u, v_j)$ as the confidences.
  6: Train the model CDAR with a weighted SVM [4] on the generated pseudo training data.
  7: Testing by the trained weighted-SVM classifier.
end
```

EXPERIMENTAL RESULTS

In this section, we plan to validate the effectiveness of our algorithm through experiments on several real-world datasets. We would plan to answer the following questions we had posed out in our previous sections.

- Firstly, is it possible for us to transfer useful knowledge between source domain and the target domain with our proposed approach on sensor readings?
- Secondly, how accurate can it be when we propagate the labels between two domains using Web knowledge?
- Thirdly, how would the choice of activities in the source domain affect the algorithm performance when testing in the target domain?
- And finally, how would our algorithm parameters, such as the similarity function we use and the number of top ranked pages we pick out in the searching step, affect our algorithm performance?

In this section, we propose to answer these questions to provide a systematic view over the performance of our algorithm.

Datasets, Evaluation Criteria and Implementation Details

In this paper, we use three real-world activity recognition datasets to validate our algorithm. Our first dataset (Amsterdam in short) [1] is from [27] where a dataset is recorded in the house of a 26-year-old man, living alone in a three-room apartment where 14 state-change sensors are installed. Locations of sensors include doors, cupboards, refrigerators and a toilet flush sensor. Sensors were left unattended and collected data for a period of 28 days, resulting in 2120 sensor events and 245 activity instances with seven different activities annotated: "Leave house", "Toileting", "Showering", "Sleeping", "Preparing breakfast", "Preparing Dinner", "Preparing Beverage". The second dataset we use is the MIT PLIA1 dataset [2] [11], which was recorded on March 4, 2005 from 9AM to 1PM in the MIT PlaceLab. The dataset contains 89 different activities and was manually classified into several categories such as "Cleaning", "Yardwork", *etc.* The third dataset is from [17] and is provided by Intel Research Lab (Intel in short), which aims to recognize 11 routine morning activities.

The evaluation criteria we use in this paper is rather standard and simple, since we are omitting all possible sequential information we might use in the dataset and therefore we just calculate the accuracy we achieve on the test data set, in other words, the number of correctly predicted activities divides the total number of activities.

[1] http://staff.science.uva.nl/~tlmkaste/research/software.php

[2] http://architecture.mit.edu/house_n/data/PlaceLab/PLIA1.htm

We also briefly describe how we handle the Web pages we crawled from the search engine. Using the activity names (e.g. preparing breakfast, cleaning misc, etc.) as queries, we submit these queries to Google and then the top K results are retrieved from, where K is a parameter we will tune in the next sections to show the relationship between algorithm performance and the number of Web pages we retrieve for each query. Next, data preprocessing has been applied to the raw data where all letters in the text are converted to lower case and the words are stemmed using the Porter stemmer [20]. Furthermore, stop words are removed. The SVM implementation we used is the LIBSVM package [4].

Algorithm Performance

We report the performance of our algorithm on three datasets with different source domain's activities and target domain's activities. We will also describe the way we choose the activities in the source domain and how we choose our target domain for testing in detail.

In our first dataset, there are only 7 activities. Therefore, we use training data from 3 activities for training (source domain), and using the remaining 4 activities for testing (target domain). All sensor events with their activities in the source domain are used as training data and the rest used as testing data. The process of selecting different activities is repeated ten times and we report the average accuracy and standard deviation we get under different parameters K. Here K is the number of top ranked Web pages we extract from the search engine results. We use a similar way of choosing activities in the source domain and the target domain in the dataset [11], while we use 5 activities in the source domain and the remaining 6 activities in the target domain. Again, the algorithm is repeated ten times to report a mean accuracy and standard derivation value. Detailed results are shown in Table 1 below. The row "Supervised" indicates the accuracy we achieve using SVM classifier with normal 10-fold cross validation on the target domain when we could acquire labeled data on them, in other words, the performance of SVM classifier under the traditional supervised learning setting on the target domain. Such a result could be used as a baseline and an upper-bound to understand how good the performance of our cross-domain activity recognition system is.

K	Amsterdam Acc(Std)	Intel Acc(Std)
K = 5	40.2% (21.7%)	39.8% (19.7%)
K = 10	53.7% (22.8%)	47.3% (20.2%)
K = 20	65.8% (22.1%)	58.1% (20.7%)
K = 50	**66.7% (21.2%)**	**63.2% (23.5%)**
K = 100	66.0% (22.4%)	63.1% (20.7%)
Supervised	72.3% (20.7%)	78.3% (17.6%)

Table 1. Algorithm Performance on Amsterdam and Intel Dataset.

In the MIT PLIA1 dataset, since there are many activities included in this dataset and a taxonomy could be built to describe these activities [8, 11]. Therefore, in MIT PLIA1 dataset, we analyze how the performance will be when we use the activities under the same category as activities in the source domain and construct training data, and then do the testing on another set of activities under the same category in the target domain.

Examples are shown in Figure 1, when we use activities under the node of "Cleaning Indoor" as activities in the source domain and activities under the node of "Laundry" or "Dishwashing" as activities in the target domain. Therefore, we are using all sensor events with activities "Sweeping, Swiftering, Mopping, Vacuuming, Dusting, Making the bed, Putting things away, Disposing Garbage, Taking out trash, Cleaning a surface, Scrubbing, Cleaning misc, Cleaning background" as training data and the testing data might be composing of all sensor events with activities "Washing laundry, Drying laundry, Washing laundry background, Drying laundry background, Folding laundry, Putting away laundry, Ironing, Laundry misc". [3]

The reason for us to choose the source domain's activity set and the target domain's activity set in such a way is as follows: we would like to analyze the performance of our algorithm to transfer from a more "categorized" set of activities to another set of activities, which is more similar rather than chosen at random, as we had done in the previous two datasets. We report the performance of our algorithm tested on different pairs of source activity sets and target activity sets, with mean accuracies and standard deviations calculated over ten independent runs. Results are reported below in Table 2 with various settings of parameter K.

From Table 1 and Table 2, we can observe that our cross domain activity recognition (CDAR) algorithm could achieve comparable performance to supervised learning classifiers when evaluated on the target domain's activities and trained on source domain's activities. Especially, we find that when we evaluate SVM classifier under a supervised learning scenario on the Amsterdam dataset, the accuracy is around 72%, whereas we could achieve a performance of 66% using our cross-domain activity recognition algorithm, which is very close to the upper bound.

We also observe that with the increasing value of K being introduced in the algorithm, the performance generally increases. Such an observation follows our intuition, which means more information is being extracted from the top Web pages and generally the MMD value calculated is more accurate, thereby improving the performance of the algorithm overall. Here we make another special note of how the choice of value K would affect our algorithm performance. From Table 1 and Table 2, we could observe that a good value of K, around 20 or 50, would give us the optimal results. The reason is that, when K is too small, the Web page data is sparse and hence we could not learn much useful information for calculating similarity function and when K is too large, it may probably add noise into the Web page we had crawled and therefore it may degrade the overall performance of our algorithm.

[3]Some activities, although defined in the activity taxonomy, do not appear in the MIT PLIA1 dataset, e.g. "Mopping".

Source	Target	K = 5 Acc (Std)	K = 10 Acc (Std)	K = 20 Acc (Std)	K = 50 Acc (Std)
Cleaning	Laundry	40.4% (21.2%)	53.4% (19.2%)	57.3% (18.7%)	58.9% (20.5%)
Cleaning	Dishwashing	38.9% (22.8%)	43.2% (18.7%)	49.3% (23.0%)	53.2% (20.7%)
Cleaning	Hygiene	42.4% (21.7%)	48.3% (22.8%)	52.4% (17.6%)	58.3% (20.7%)
Cleaning	Information / Leisure	43.1% (23.0%)	45.7% (17.9%)	52.8% (20.7%)	54.9% (22.8%)
Laundry	Cleaning	41.2% (19.2%)	52.8% (20.5%)	53.2% (20.7%)	60.2% (20.0%)
Laundry	Dishwashing	43.2% (20.2%)	53.2% (20.7%)	58.3% (20.5%)	61.2% (21.2%)
Laundry	Hygiene	49.3% (19.5%)	46.3% (19.2%)	49.5% (23.0%)	58.3% (21.4%)
Laundry	Information / Leisure	32.7% (19.2%)	40.2% (20.7%)	48.3% (20.5%)	49.2% (22.6%)
Dishwashing	Cleaning	45.3% (21.2%)	48.3% (20.5%)	53.2% (21.7%)	59.2% (19.2%)
Dishwashing	Laundry	44.8% (20.5%)	50.1% (21.7%)	54.8% (21.9%)	60.8% (22.6%)
Dishwashing	Hygiene	43.2% (20.2%)	52.7% (22.1%)	57.3% (20.7%)	59.2% (16.7%)
Dishwashing	Information / Leisure	39.3% (20.2%)	43.7% (21.7%)	48.3% (18.7%)	50.3% (19.2%)
Hygiene	Cleaning	44.7% (20.5%)	45.8% (20.5%)	49.7% (20.7%)	52.9% (19.5%)
Hygiene	Laundry	41.8% (20.7%)	43.7% (20.2%)	53.8% (23.0%)	53.7% (17.0%)
Hygiene	Dishwashing	42.9% (22.6%)	42.8% (19.2%)	47.1% (22.8%)	51.2% (18.7%)
Hygiene	Information / Leisure	30.7% (19.5%)	33.8% (20.5%)	38.3% (20.2%)	42.3% (21.2%)

Table 2. Algorithm Performance on MIT PLIA1 Dataset

Therefore, our experimental results on three datasets could validate the effectiveness of our algorithm.

Choice of Similarity Functions

The Maximum Mean Discrepancy (MMD) function in our proposed approach seems to be an ad hoc choice, and it is natural for one to ask the question about whether it is chosen deliberately to improve the performance of our algorithm or whether other similarity functions would also achieve comparative performance, compared to MMD? To answer this question, we also evaluated our algorithm using another similarity function that is also popular in calculating the similarity in information retrieval, i.e. the cosine similarity [15].

In short, the cosine similarity is defined as the cosine of the angle between two vectors, defined as:

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{|A||B|}, \tag{4}$$

where A and B are two vectors and for text matching. Such a value is easy to calculate based on the Webp ages we crawled. In our case, we generate A and B by merging the term-frequency (tf) vectors of the documents for the two candidate activities. For example, given the K Web pages extracted for some activity a_A, we will add up all the corresponding K tf vectors to get such a vector A. Similarly, we can get the vector B for some activity a_B, so that we can compute the Eq. (4). Using similar approaches except the MMD function replaced with the cosine similarity function, we report our results for the Amsterdam dataset [27] and the Intel dataset [17] in Table 3.

In Table 3, the left two columns are results using cosine similarity functions and the right two columns are results using MMD functions, which is the same as results reported Table 1. We could see that although the results using cosine similarity functions are slightly worse than the results we report using MMD functions, it still achieves comparable performance to MMD. Therefore, we could make the conclusion that by incorporating other "meaningful" and "reasonable" similarity functions, our cross-domain activity recognition algorithm could still achieve reasonable performance.

In addition, one particular characteristic of our algorithm is that, the standard deviations of the experimental results are large. The reason is that in each round of training and testing, it is possible that we are selecting quite different sets of activities. In some rounds the activities drawn in the source domain are more similar to the activities in the target domain compared to the other rounds, thus making the prediction accuracy not as stable.

Discussion of Experiments

Here we briefly discuss and summarize some characteristics of the results we report from our experiments.

- Firstly, our algorithm CDAR, could successfully transfer knowledge between source domain and target domain and thus solve the cross-domain activity recognition task. From Tables 1, 2 and 3, we could see that in most cases, our algorithm could achieve an accuracy of more than 60% when evaluating on the test domain activity set. Such a performance is rather promising since (i) we did not use any sequential information, which had proved to be of great help in traditional activity recognition tasks; (ii) we did not use any labeled data in the target domain, such a feature is especially effective in real world use, since the activity recognition system will be trained on a predefined set of activities and then when users use such an activity recognition system, he may perform activities outside of the predefined activity set. Therefore, such an algorithm that could perform cross-domain activity recognition and does not acquire training data in the target domain would be especially useful.
- Secondly, the number of K Web pages we extract from the search engine results would affect the algorithm performance but would be rather close to the optimal results or converge when K is larger than 50. This means that

	Cosine Similarity		MMD Similarity	
K	Amsterdam Acc(Std)	Intel Acc(Std)	Amsterdam Acc(Std)	Intel Acc(Std)
K = 5	48.7%(19.5%)	42.4% (20.5%)	40.2% (21.7%)	39.8% (19.7%)
K = 10	53.2%(20.7%)	45.3% (20.2%)	53.7% (22.8%)	47.3% (20.2%)
K = 20	58.3% (20.2%)	52.1% (20.7%)	65.8% (22.1%)	58.1% (20.7%)
K = 50	62.1% (19.2%)	57.3% (19.0%)	66.7% (21.2%)	63.2% (23.5%)
K = 100	**65.3% (16.7%)**	**62.3% (21.7%)**	**66.0% (22.4%)**	**63.1% (20.7%)**

Table 3. Algorithm Performance on Amsterdam and Intel Dataset with Cosine Similarities and MMD Similarities.

we could approximate the underlying similarity score between different activities by using around 50 related Web documents. Another advantage is that the cost of performing such a Web search would not be heavy to put into real world usage.

- Thirdly, the choice of similarity functions would not affect our algorithm performance that much as long as a reasonable similarity function that approximates the underlying similarity would be used. In our experiments we have already validated this conclusion through the usage of both MMD and cosine similarity functions and from the comparison between these two functions, we arrive at such a conclusion.

- Fourthly, whether the cross-domain activity recognition performance would be successful or not depends not only on the algorithm but on some underlying characteristics of the activity set. In Table 2, we could note that some activity pairs perform worse than others, for example, when we are transferring knowledge from activities under the subcategory of "Hygiene" (Source Domain) to activities under the subcategory of "Information / Leisure" (Target Domain), we could only achieve an accuracy of around 42% at the best. (Corresponding to the last row in Table 2.) Such a result might be of the reason that there is a relatively larger distance between "Hygiene" activities and "Information/Leisure" activities. In contrast, we could also find that when we use activities in the "Dishwashing" subcategory as the source domain and activities in the "Cleaning" or "Laundry" subcategory as the target domain, the accuracy we achieve is much higher, suggesting that these two kind of activities have closer relationships, which also follows our intuition. Thus, one interesting topic to study is to determine when we could successfully "transfer the knowledge" between the source domain and the target domain, or, if we had known the activity set in the target domain in advance, what source domain would be best for us to use to ensure that the algorithm performance can be guaranteed? One possible choice can be using some activity ontology and shrinkage based approach to combine data from similar activities for training the classifier [23].

CONCLUSION AND FUTURE WORK

In this paper, we have proposed a simple yet effective approach to solve the cross-domain activity recognition problem. The basic setting of our problem lies in that we have labeled training data in the source domain but no labeled training data in the target domain; and our goal is to predict the activities in the target domain. Furthermore, the activities in the source and target domain do not overlap, which means that traditional supervised learning approaches cannot be applied under this scenario. Our proposed approach makes use of top ranked Web pages returned by search engines to mine the similarities between the activities in the two domains. Such an assumption makes it possible for us to "interpret" the labeled data in the source domain using the label space of the target domain. We validate our approach in several real-world sensor-based activity recognition datasets, and achieve results comparable to those of traditional activity recognition algorithms based on supervised learning.

In the future, we aim to extend our work in several directions which would dig deeper alongside the problem of cross-domain activity recognition. Firstly, we aim to discuss cases where not only the labeled space, but also the feature spaces in two domains are different, so that an innovative approach for bridging the two feature spaces must be proposed as well. Secondly, since we omit the sequential information in the current approach for simplicity reasons, we plan to propose new approaches for cross-domain activity recognition which could make better use of sequential information. Thirdly, we aim to investigate into the intrinsic question of "whether the activities in two domains can be transferred or not" and "if we cannot acquire training data for activities in the target domain, what activities would be best for us to use as training data in the source domain"? Such questions are the possible directions along which we could extend in the future.

ACKNOWLEDGEMENT

We thank the support from Microsoft Research Asia.

REFERENCES

1. K. Borgwardt, A. Gretton, M. Rasch, H. Kriegel, B. Schölkopf, and A. Smola. Integrating stuctured biological data by kernel maximum mean discrepancy. In *Proc. of the 14th International Conference on Intelligent Systems for Molecular Biology*, pages 49–57, 2006.

2. H. H. Bui. A general model for online probabilistic plan recognition. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI 2003)*, pages 1309–1318, 2003.

3. R. Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.

4. C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001.

5. O. Etzioni, M. J. Cafarella, D. Downey, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates. Methods for domain-independent information extraction from the web: An experimental comparison. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI)*, pages 391–398, 2004.

6. T. Evgeniou and M. Pontil. Regularized multi-task learning. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 109–117, Seattle, Washington, USA, August 2004. ACM.

7. M. R. Hodges and M. E. Pollack. An 'object-use fingerprint': The use of electronic sensors for human identification. In *Proceedings of the Ninth International Conference on Ubiquitous Computing (UbiComp 2007)*, pages 289–303, 2007.

8. D. H. Hu, S. J. Pan, V. W. Zheng, N. N. Liu, and Q. Yang. Real world activity recognition with multiple goals. In *Proceedings of the Tenth International Conference on Ubiquitous Computing (UbiComp 2008)*, pages 30–39, 2008.

9. D. H. Hu and Q. Yang. Cigar: Concurrent and interleaving goal and activity recognition. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI)*, pages 1363–1368, 2008.

10. D. H. Hu, X.-X. Zhang, J. Yin, V. W. Zheng, and Q. Yang. Abnormal activity recognition based on hdp-hmm models. In *IJCAI*, pages 1715–1720, 2009.

11. S. S. Intille, K. Larson, E. M. Tapia, J. Beaudin, P. Kaushik, J. Nawyn, and R. Rockinson. Using a live-in laboratory for ubiquitous computing research. In *Proceedings of the Fourth International Conference on Pervasive Computing (Pervasive 2006)*, pages 349–365, 2006.

12. H. Kautz. *A Formal Theory of Plan Recognition*. PhD thesis, University of Rochester, 1987.

13. L. Liao. *Location-Based Activity Recognition*. PhD thesis, University of Washington, 2006.

14. L. Liao, D. Fox, and H. A. Kautz. Learning and inferring transportation routines. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI)*, pages 348–353, 2004.

15. C. D. Manning, P. Raghavan, and H. Schutze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

16. L. Mihalkova, T. Huynh, and R. J. Mooney. Mapping and revising markov logic networks for transfer learning. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence*, pages 608–614, 2007.

17. D. J. Patterson, D. Fox, H. A. Kautz, and M. Philipose. Fine-grained activity recognition by aggregating abstract object usage. In *Proceedings of the Ninth IEEE International Symposium on Wearable Computers (ISWC 2005)*, pages 44–51, 2005.

18. M. Perkowitz, M. Philipose, K. Fishkin, and D. J. Patterson. Mining models of human activities from the web. In *Proc. of the 13th international conference on World Wide Web (WWW)*, pages 573–582, 2004.

19. M. E. Pollack, L. E. Brown, D. Colbry, C. E. McCarthy, C. Orosz, B. Peintner, S. Ramakrishnan, and I. Tsamardinos. Autominder: an intelligent cognitive orthotic system for people with memory impairment. *Robotics and Autonomous Systems (RAS)*, 44(3-4):273–282, 2003.

20. M. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

21. R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng. Self-taught learning: Transfer learning from unlabeled data. In *Proceedings of the 24th International Conference on Machine Learning*, pages 759–766, Corvalis, Oregon, USA, June 2007.

22. M. Sugiyama, S. Nakajima, H. Kashima, P. V. Buenau, and M. Kawanabe. Direct importance estimation with model selection and its application to covariate shift adaptation. In *Advances in Neural Information Processing Systems 20*. MIT Press, 2008.

23. E. M. Tapia, T. Choudhury, and M. Philipose. Building reliable activity models using hierarchical shrinkage and mined ontology. In *Pervasive*, pages 17–32, 2006.

24. E. M. Tapia, S. S. Intille, and K. Larson. Activity recognition in the home using simple and ubiquitous sensors. In *Pervasive Computing*, pages 158–175, 2004.

25. D. L. Vail, M. M. Veloso, and J. D. Lafferty. Conditional random fields for activity recognition. In *Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2007)*, pages 1–8, 2007.

26. T. van Kasteren, G. Englebienne, and B. Kröse. Recognizing activities in multiple contexts using transfer learning. In *AAAI Fall 2008 Symposium: AI in Eldercare*, Washington DC, USA, 2008.

27. T. van Kasteren, A. K. Noulas, G. Englebienne, and B. J. A. Kröse. Accurate activity recognition in a home setting. In *UbiComp*, pages 1–9, 2008.

28. S. Wang, W. Pentney, and A. maria Popescu. Common sense based joint training of human activity recognizers. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2237–2242, 2007.

29. D. Wyatt, M. Philipose, and T. Choudhury. Unsupervised activity recognition using automatically mined common sense. In *Proceedings, The Twentieth National Conference on Artificial Intelligence (AAAI 2005)*, pages 21–27, 2005.

30. J. Yin, X. Chai, and Q. Yang. High-level goal recognition in a wireless lan. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI)*, pages 578–584, 2004.