Building Reliable Activity Models Using Hierarchical Shrinkage and Mined Ontology

Emmanuel Munguia Tapia[1], Tanzeem Choudhury[2] and Matthai Philipose[2]

[1] Massachusetts Institute of Technology
1 Cambridge Center 4FL
Cambridge, MA, 02142, USA
emunguia@mit.edu

[2]Intel Research Seattle
1100 NE 45th St., 6th Floor
Seattle, WA, 98105, USA
{tanzeem.choudhury, matthai.philipose}@intel.com

Abstract. Activity inference based on object use has received considerable recent attention. Such inference requires statistical models that map activities to the objects used in performing them. Proposed techniques for constructing these models (hand definition, learning from data, and web extraction) all share the problem of *model incompleteness*: it is difficult to either manually or automatically identify all the possible objects that may be used to perform an activity, or to accurately calculate the probability with which they will be used. In this paper, we show how to use auxiliary information, called an ontology, about the functional similarities between objects to mitigate the problem of model incompleteness. We show how to extract a large, relevant ontology automatically from WordNet, an online lexical reference system for the English language. We adapt a statistical smoothing technique, called shrinkage, to apply this similarity information to counter the incompleteness of our models. Our results highlight two advantages of performing shrinkage. First, overall activity recognition accuracy improves by 15.11% by including the ontology to re-estimate the parameters of models that are automatically mined from the web. Shrinkage can therefore serve as a technique for making web-mined activity models more attractive. Second, smoothing yields an increased recognition accuracy when objects not present in the incomplete models are used while performing an activity. When we replace 100% of the objects with other objects that are functionally similar, we get an accuracy drop of only 33% when using shrinkage as opposed to 91.66% (equivalent to random guessing) without shrinkage. If training data is available, shrinkage further improves classification accuracy.

1 Introduction

Automated reasoning about human activity is central to a variety of pervasive computing usage models and applications. Usage models include activity-aware actuation,

proactive reminding, automated activities-of-daily-living (ADL) monitoring and prompting, embedded health assessment, computer supported coordinated care giving, and task monitoring and prompting in the workplace. Specific applications that have been proposed include the automated control of HVAC and home entertainment systems based on current user activity, automated filling of medical forms about activities of elderly users, delivery of information about care recipients' behavior via shared scheduling tools, and the semi-automated evaluation of student performances of standard medical procedures. For these applications to be practical, the underlying activity recognition module often needs to detect a wide variety of activities (people may routinely perform dozens to hundreds of relevant activities a day, for instance) performed in many different ways, under many different environmental conditions; the particular aspects of the activity that are of interest (e.g. user motion, task progress, object usage or space usage) also vary widely across applications. Such robust recognition across a variety of activities and their variations has proved to be difficult to engineer.

A central challenge underlying activity recognition is that of bridging the gap between conventional sensors and informative high-level features such as objects used, body motion and words spoken. The most common approach is to use a few (typically one per room or user) very rich sensors such as cameras and microphones which can record very large quantities of data about the user and their environment. Although in principle the data captured by these sensors should be as useful as that captured by the key human senses of sight and hearing, in practice the task of extracting features from rich low-level representations such as images has proved to be challenging in unstructured environments. A popular alternate approach is to use specialized sensors (of the order of one per user) such as accelerometers and location beacons to get precise information about a particular small set of features related to the user, such as limb-movement and user location. The simplicity, however, comes at a price: by ignoring the environment of the user, these sensors limit the number of activities they can discriminate between. The inability to distinguish between opening a dishwasher and opening a washing machine can be a deciding factor in discriminating between the corresponding activities.

Recent years have seen the emergence of a third approach to sensing that may be termed *dense* sensing. Exploiting advances in miniaturization and wireless communication, this approach attaches sensors directly to many objects of interest. The sensors are either battery-free wireless stickers called Radio Frequency Identification (RFID) tags[1-3] or small wireless sensor nodes powered by batteries[4, 5]. The sensors transmit to ambient readers the usage of the objects they are attached to by detecting either motion or hand-proximity to the object. Further, since each sensor has a unique identifier, information about the object that does not change (such as its color, weight or even ownership), which would conventionally have to be discerned by sensors, can be associated in a directly machine readable way with the object. The reliable sensing of detailed object use enabled by dense sensing has a few advantages. First, for very many day-to-day activities, the objects used serve as a good indicator as to which activity is being performed. Second, the objects used remain fairly invariant across different ways of performing these activities. Third, since the sensors detect the features quite well regardless of most environmental conditions, activity recognition can

be robust to changes in these conditions. Finally, objects used can serve as a powerful cue as to other aspects of interest: if a hammer or a knife is known to be in use, the space of possible user motions is highly constrained.

Systems based on dense sensors model activities in terms of the sequence of objects used, typically using generative Bayesian representations such as Hidden Markov Models (HMM's)[6-8] or Naïve Bayesian models[9]. Models for individual activities in these representations are generated in one of three ways. The simplest, and least scalable, approach is to construct the model by hand: an application designer can simply list the objects expected to be used in an activity of interest, along with the probability of use. A conventional alternative is to learn the model by performing the activity in a variety of exemplary ways, labeling traces of objects used during the performances with the corresponding activities, and using supervised machine learning techniques to learn the corresponding model. A final approach is to note that the model is essentially a probabilistic translation between the activity name and the names of objects used, and to mine large text corpora such as the web to obtain these translations. The approaches are not mutually exclusive. For instance, both hand-made and web-mined models can be used as priors which are further customized using observed data.

All three approaches to constructing models suffer from what may be termed the *model incompleteness* problem: the models they produce have objects that are either missing or that have inappropriate probabilities. Incomplete models can, of course, result in faulty inference. Humans who hand-write models typically do not have the patience (and often the judgment) to list all objects that may be used in an activity, especially when alternate or obscure objects need to be considered: the model for "making tea" may mention neither "coffee cup" nor "honey". Further, the probability of use ascribed to unfamiliar objects may be quite skewed. Similarly, given the inconvenience of generating labeled examples of all (or most) possible ways to execute an activity, it is likely that uncommon objects will be missing or under-represented. Finally, when models are mined from the web, the vagaries of the web may result in certain objects (e.g. "cup") being ascribed vastly higher probabilities than others (e.g. "teacup").

The use of objects as the underlying features being modeled suggests a simple approach to countering incompleteness. Intuitively, we can exploit common sense information on which objects are functionally similar. If the model ascribes very different probabilities to two very similar objects, we can "smooth" these probabilities into more similar values. As a degenerate case, if the model omits an object while incorporating very similar ones, we can postulate that the omitted object is likely to be observed in the model. We show below how to realize this idea in a completely unsupervised way and provide evidence that the idea is quite effective. Earlier work [7, 10] has used manually extracted hierarchy to incorporate the notion of object similarity into activity models. In this paper, we show how to extract relevant information on the functional similarity of objects automatically from WordNet, an online lexical reference system for the English language. The similarity information is represented in a hierarchical form known as an ontology. Given the similarity measure provided by the ontology, we formalize the above intuitive notion of smoothing by adapting from statistics a technique called *shrinkage*. Shrinkage is a well established technique

for estimating parameters in the presence of limited or missing training data and has been successfully used in classifying text documents [11, 12], in modeling the behavior of web site users [13], and in service-oriented context-aware middleware applications [14]. We use a mixture of real-world data and synthetic data to evaluate our system. Our results show that our techniques have three benefits. First, mined models that are smoothed recognize activities with significantly higher accuracy than those that are not. Second, models that are learned from data can make do with significantly less training data when smoothing is applied versus when it is not. Third, when faced with test data that contains objects not seen in training data, but that are similar to those in the model, smoothing yields substantially better recognition rates.

This paper begins by describing the procedure for automatically extracting the ontology of objects from WordNet in Section 2. Then, Section 3 covers the algorithm used for performing shrinkage over the ontology of objects. Section 4 shows the results of running simulated experiments over a large ontology of objects, and Section 5 of experiments ran over real sensor data. Finally, Section 6 summarizes the main results and conclusions drawn from this work.

2 Automatic Ontology Extraction from WordNet

WordNet [15] is a hierarchically organized lexical system motivated by current psycholinguistic theories of human lexical memory. WordNet resembles a thesaurus more than a dictionary since it organizes lexical information in terms of word meanings (or senses), rather than word forms. In WordNet, nouns, verbs, adjectives and adverbs are organized into synonym sets called synsets, each representing one underlying lexical concept. For example the noun *couch* in WordNet has three senses or word meanings, and the synset corresponding to the first sense {*couch#1}* defined as *'an upholstered seat for more than one person'* is *{sofa, couch, lounge}*. The sense number in WordNet indicates the frequency of use, where 1 corresponds to the most commonly used.

The real power and value of WordNet relies on the way different semantic relations link the synonym sets (or word senses). Currently, WordNet comprises the following kinds of semantic relations between word meanings: (1) hypernyms, (2) hyponyms, (3) meronyms, and (4) holonyms. Two of these semantic relations are especially important for their usefulness in extracting a semantic hierarchy or an ontology of objects: hyponyms, and hypernyms. Hypernyms are *is-a* relationships were the meaning of a word is a superset of another. For example, *{cooking utensil#1}* is a superset or hypernym of *{pan#1}*. On the contrary, hyponyms are *inverse-is-a* relationships were the meaning of a word is a subset of the meaning of another. Figure 1 shows examples of the hypernyms tree for three everyday objects.

WordNet organizes nouns into a set of 25 semantic primes or unique beginners of separate hierarchies. Five of these unique beginners are particularly important because they encompass all possible natural and man made objects {*non-living things, objects*}, and living organisms commonly used in meal preparation {*living thing, organism*}. These five semantic primes are: *{natural object}*, *{artifact}*, *{substance}*,

{food}, and *{plant, flora}*. All these unique beginners are hyponyms or subsets of the more abstract concept *{entity}*.

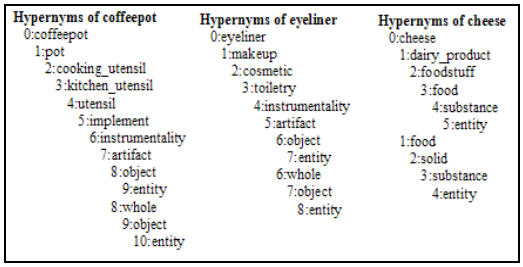


Figure 1. Hypernyms tree for three objects: Coffeepot, eyeliner, and cheese.

Since all the physical objects of interest found in everyday environments are subsets or hyponyms of *{entity}*, a tree-like ontology of objects can be automatically extracted. This follows from the lexical tree (free of circular loops) design imposed over the nouns by the creators of WordNet.

As of September 2005, WordNet 2.1 contains approximately 117,097 noun word forms organized into approximately 81,426 word meanings (synsets) that make WordNet a unique and rich semantic database for recovering complete ontology of objects automatically.

2.1 Ontology Extraction Algorithm

The generation of the ontology or hierarchy of objects can be divided in two steps (1) the generation of the ontology skeleton, and (2) the expansion of the ontology. In order to generate the ontology skeleton, an initial list of objects of interest is required. In the context of our work, this initial list of objects is the list of all objects that appear in the mined activity models (or activity recipes) plus all the objects (RFIDs object labels) found in the sensor traces. The ontology skeleton generation algorithm proceeds as follows: (i) since everyday tangible objects correspond to nouns in natural language, we proceed to search the objects or words of interest in the noun files of WordNet, and (ii) once the noun has been found, we proceed to automatically select the sense of the word by looping through all the senses of the word until finding the first sense that is a hypernym or subset of *{entity}*. As discussed in the previous section, the node *{entity}* includes as subsets all possible natural and man made objects and living organisms commonly used in meal preparation. This guarantees that the selected sense will be the most commonly used sense that is also a physical object.

After the appropriate word sense has been selected, we proceed to find the hypernym tree or superset (parent) nodes of the selected sense of the word (or object). It is important to notice that some words may have multiple parents (who are descendants of *entity*) at the same level of the hypernyms tree, since in the previous step, we only ensured that the leaf node has a unique sense that is a descendant of the *{entity}* node. Figure 1 shows an example of such case for the object *cheese*. In situations when multiple parents are found at the same level in the hypernyms tree, only the first

one is considered for being the most common, and the other ones are discarded. In practice we have found that this does not represent a major problem in extracting the hierarchy of objects. When the ontology is generated, a synonyms file is also generated so that any synonym of a word can be used while performing search operations in the ontology. For example, the synset for the object *cleaner* is *{cleansing_agent, cleanser, cleaner}*.

It is important to note that in order to perform shrinkage, the ontology must not have any loops. Our algorithm generates a tree structured ontology by only selecting word senses that are hyponyms or subsets of the concept node *{entity}* and by ensuring each node has a unique parent. Thus, the node *{entity}* having the single sense: *'that which is perceived or known or inferred to have its own distinct existence (living or nonliving)'* correspond to the root node, and the highest abstraction level of the ontology. Also note that the leaf nodes or most specific terms in the ontology will correspond to the objects provided in the original list.

Once the ontology skeleton has been generated, it is useful to expand the ontology to accommodate for possible objects that might be used while performing an activity, but were not provided in the original list of objects. The expansion of the ontology consists of finding all the ancestor (parents) nodes for all the ontology leaf nodes up to a specified level *MaxParentLevel*. Then, we proceed to find all the hyponyms (children nodes) of those ancestor nodes up to a maximum depth level *MaxChildLevel*. By performing this procedure, we create sibling nodes for the leaf nodes of our original ontology that might appear in sensor traces in the future. Figure 2 shows a simplified version of the pseudo-code for extracting the ontology from WordNet.

```
//GENERATION OF ONTOLOGY SKELETON
 For i:=1 to objectList.length(){
     object = objectList(i);
     word = find_word_in_wordnet_noun_file(object);
     If(!empty(word)){
     For j:=1 to word.getSenses.length(){
          wordsense = word.getSense(j);
          If(wordsense.ishypernym("entity")) break;
     }
     hypernyms = getHypernymsTree(wordsense);
     ontologytree.addNodes(hypernyms);
 }

 //ONTOLOGY EXPANSION
 For i:=1 to ontologytree.getLeafNodes().length(){
     Node = ontologytree.getLeafNode(i);
     ancestors = getHypernymsTree(Node, MaxParentLevel);
     For j:=1 to ancestors.length{
         Hyponyms = getHyponymsTree(ancestors(j), MaxChildLevel);
         ontologytree.addNodes(hyponyms);
     }
 }
```

Figure 2. Simplified version of the pseudo-code for automatically extracting the ontology of objects from WordNet.

3 Shrinkage over the Hierarchy of Objects

Shrinkage [16] is a well established statistical technique for improving parameters values estimated for a given model, when they can not be computed reliably from training data alone. By exploiting the similarity between nodes in a hierarchy, shrinkage estimates new parameter values for child nodes by linearly interpolating the values from the child node to the root node [11]. This represents a trade-off between specificity and reliability. The child node estimate is the most specific (low bias), but high variance (less reliable), and the root node is the most reliable (low variance), but general (high bias). By combining these estimates we can end up with a better and more reliable model.

In this work, we use shrinkage to create improved probability estimates of the leaf nodes of the ontology. Our assumption is that the leaf nodes in our ontology represent $P(o_i \mid a_j)$, the probability estimates of observing an object $o_i \in O$ during the performance of an activity $a \in A$, and that the hierarchy structure characterizes the functional similarity between objects. We denote $\widetilde{P}(o_i \mid a_j)$ the new probability estimates of observing an object given an activity class, and we compute them as follows:

$$\widetilde{P}(o_i \mid a_j) = \lambda^0 P^0(o_i \mid a_j) + \ldots + \lambda^k P^k(o_{i} \mid c_j) = \sum_{l=0}^{k} \lambda^l P^l(o_{i} \mid a_j) \qquad (1)$$

$P^l(o_i \mid a_j)$ denotes the maximum likelihood (ML) probability estimate of a node at level l in the leaf ($l = 0$) to root ($l = k$) path. The interpolation coefficients (weights) are denoted $\{\lambda^1, \lambda^2, \ldots \lambda^k\}$ where $\sum_{l=0}^{k} \lambda^l = 1$. j denotes the activity class number, and i the object used. The ML probability estimates at each node are computed using the following equation:

$$P(o_i \mid a_j) = \frac{N(o_i, a)}{\sum_{s=1}^{|O|} N(o_s, a)} \qquad (2)$$

where $N(o_i, a)$ is the number of times object o_i occurs in activity a, and $|O|$ denotes the set of all possible objects.

3.1 Determining Mixture weights

The weights $\{\lambda^1, \lambda^2, \ldots \lambda^k\}$ used during shrinkage balances the influence of the nodes containing specific information but little training data, with those nodes containing more generic information but larger amounts of training data. The mixture of weights can be computed in one of the following ways: (1) uniformly where all the weights are equal (2) by applying the Expectation-Maximization algorithm (EM) as in [11] to find the weights that maximize the likelihood of the data or (3) using heuristics schemes that are a function of the rank (level) of the node in the ontology [13].

Since the goal of this work is to have a completely unsupervised approach to activity recognition where no sensor traces are available, we decided to estimate the weights using the following heuristics: (1) $\lambda^{level} = 1/c^{level}$, and (2) $\lambda^{level} = e^{-c \cdot level}$, where c is a constant. These heuristics correspond to exponentially decaying functions that

will assign large weights to nodes in the neighborhood of the leaf node, and low weights to the generic nodes found in the upper levels of the ontology.

```
//ASSIGNING COUNTS TO LEAF NODE IN ONTOLOGY
ontology.setLeafNodeCounts(modelsObjectProbs*Factor);
//COMPUTE MAXIMUM LIKELIHOOD COUNTS FOR INTERNAL NODES
 internalNodes = ontology.getInternalNodes();
 For node:=1 to internalNodes.length(){
    inode = internalNodes(node);
    childrenLeaves = getChildrenLeafNodes(inode);
    inode.setMLCount(getCountsSum(childrenLeaves));
 }

 //OBTAIN LEAF NODES SMOOTHED COUNTS BY SHRINKAGE
 leaves = ontology.getLeafNodes();
 For leaf:=1 to leaves.length(){
    lnode = leaves(leaf);
    nodes = getNodes2RootNode(lnode);
    For l:=0 to nodesPath.length()-1{
        lambda = ComputeHeuristics(level);
        If(level==0) //if leaf node
            SmoothCount = lambda*lnode.getCounts();
        Else{ //if internal node
            //substract node counts to reduce dependency
            counts = nodes(l).getCounts()-nodes(l-1).getCounts();
            smoothCount = smoothCount + lambda*counts;
        }
    }
    lnode.setCounts(smoothCounts);
  }
```

Figure 3. Pseudo-code for performing shrinkage over the ontology of objects.

The use of shrinkage over the ontology of objects in our unsupervised approach provides two main benefits: (1) it improves the probability estimates in the leaf nodes by taking advantage of the functional relationship of objects represented by the ontology. The effect of this improvement is a reduction in the number of training examples required to achieve a desired accuracy. If the number of training examples is kept constant, an increased accuracy will be observed by performing shrinkage; (2) shrinkage provides robustness when objects not present in the activity models are used while performing an activity. This effect is achieved by creating object observation probability estimates for those objects not present in the models by shrinking them towards the objects present in the models using the ontology. This means that we are able to compute educated probability estimates for unseen objects when it was not previously possible.

The pseudo-code for performing shrinkage over the ontology of objects is shown in Figure 3, and consists on the following steps: (1) set the object observations (counts) for each leaf node by converting object probabilities to counts by multiplying them by a factor (2) compute the maximum likelihood counts for all the internal (non-leaf) nodes and (3) compute the smoothed count (shrinkage) for all the leaf nodes using equation 1. The counts are converted back to probabilities by normalizing them.

4 Experimental Results: Effect of Limited or Missing Data

In this experiment, we test the effectiveness of shrinkage over a large ontology of objects when we have limited training data or missing objects. We use Hidden Markov Models (HMMs) to parameterize the activities and assume that the objects used during an activity appear on the leaf nodes of the ontology. This assumption is plausible since usually one interacts with a specific instance of an object during an activity and not the broader abstract category. HMMs are a particular type of dynamic Bayesian Networks (DBNs) consisting of three parameters: (1) prior probabilities for each state π , (2) a state transition probability matrix T , and (3) the observation probabilities for each state B . The observation matrix represents the object observation probabilities for a given activity. Our experimental results show that shrinkage over the HMM object emission probabilities helps not only in reducing the number of training examples required to achieve a given accuracy, but also in providing robustness when objects not present in the activity models are used.

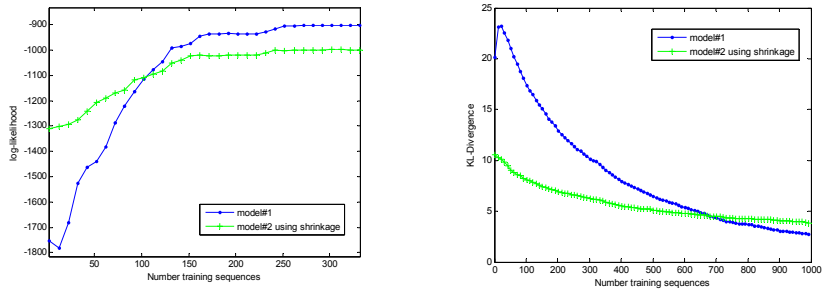
The ontology used in this experiment was generated from a list of 815 objects used in performing household activities. The list was obtained from objects appearing in the mined activity models, and sensor traces used in [1]. The ontology consists of 4188 nodes, 815 leaf nodes, and has a maximum depth of 14. The results presented in this section are based on simulated sensor traces (i.e. sampled from a true model that we create and not from actual observations from people). However, the ontology contains representative information about objects used during performing everyday activities. In the next section we will present results on using shrinkage in real sensor traces obtained from multiple individuals.

The experiment proceeds as follows: We first create a true activity model *model#0* represented by a 3 state HMM (3 subtasks in activity) with random prior, transition, and observation matrices. Next, we generate training data by sampling n number of sequences from *model#0*. We learn the model parameters from the training data in two ways: (i) by computing the maximum likelihood estimate of the prior (π) , transition (T) , and observation (B) matrices (*model#1*) and (ii) by re-estimating the observation matrix (B) using shrinkage (*model#2)* and $\lambda^l = e^{-3.5 \cdot l}$. We measure the closeness of the learned models (*model#1, model#2)* to the true model *model#0* by computing the Kullback-Leibler (KL) divergence between the observation matrices of *model#0*, and *model#1,* and *model#2*, respectively. The KL divergence $\mathrm{D}(\mathrm{p} \parallel \mathrm{q})$ is a measure of the similarity between two probability distributions p and q . The smaller the KL divergence, the more similar the compared distributions are. Finally, we compute the log-likelihood for models *#1* and *#2* on a test dataset sampled from the true model *#0*.

4.1 Reducing the number of training examples by shrinkage

The plots in Figure 4 were generated by iteratively increasing n , the number of training sequences, to learn parameters for *models #1*and *#2*. Figure 4a shows the log-likelihood computed over the test sequences (50 of length 13) using the learned mod-

els. The higher the log-likelihood, the better the model explains the test dataset, which in turn leads to higher accuracy. By inspection of Figure 4a, we note that 70 training examples are required by *model#1* to achieve the same log-likelihood that *model#2* achieves using a single training sequence. This is an important result because it shows that shrinkage can dramatically reduced the number of training examples required to achieve a specific log-likelihood. When n is greater than 100, the log-likelihood of *model#1* is higher than *model#2* for the specific test dataset. However, Figure 4b shows that the KL divergence is lower for *model#2* up to when $n = 675$. This signifies that shrinkage *model#2* is a closer match to the true *model#0* and will explain new test data more often, when trained on less than 675 example sequences. When we have enough representatives training examples the maximum likelihood (ML) solution will converge to the true model and shrinkage will not improve the parameter estimates anymore. Although, depending on the complexity of the model the number of training data required to have a reliable ML solution may be huge.



(a) (b)

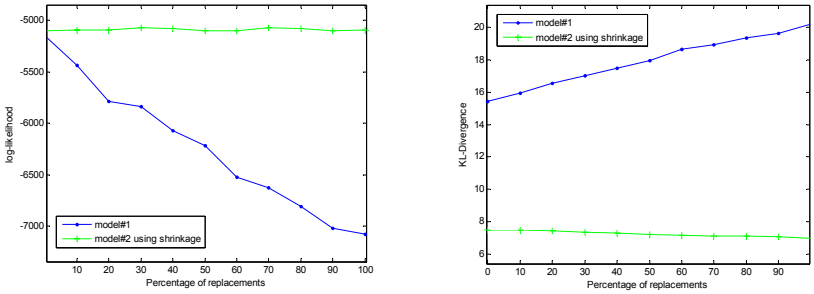
Figure 4. (a) Log-likelihood, and (b) KL-Divergence between the baseline HMM model, learned model, and learned model using shrinkage over the ontology.

4.2 Robustness to unseen objects by shrinkage

Often it might be the case that the initial model specifies the use of an object during an activity (e.g. use of *teacup* while *making tea*) which is later substituted by a functionally similar object (e.g. *mug*). If the activity model does not incorporate the similarity between a *teacup* and a *mug* then the model won't be able to correctly identify the activity *making tea* when a mug is used. In this experiment, we simulate the use of objects not present in the activity models by modifying the observations in the sequences sampled from *model#0* in the previous experiment. The modification consists of replacing $m\%$ of observations by observations of one of their randomly selected sibling nodes in the ontology.

This simulates the effect of having observed the sibling nodes (objects) in the sequences rather than the original leaf nodes. Once the replacements have been performed, we proceed to learn the transition, and observation matrices from the training sequences for models *#1* and *#2*. Figure 5 shows the resulting plots for the likelihood over the test sequences and the KL divergence when the percentage of replaced ob-

servations is modified from 0% to 100%. The fact that the likelihood is always greater, and the KL divergence smaller for *model#2* than for *model#1* corroborates the usefulness of shrinkage when unseen objects in our models are used.



(a) (b)

Figure 5. (a) Log-likelihood, and (b) KL-Divergence between the baseline HMM model, learned model, and learned model using shrinkage over the ontology.

5 Experimental Results: Performance on Data Collected from Multiple Individuals

In this section, we show the benefit of incorporating high level information into activity models using shrinkage over an ontology and measure the performance in real sensor traces. To get the initial models, we adopt the procedure followed in [6] to mine activity models from the web, compute object observation probabilities, and to perform inference using hidden Markov models. We extend the work done in [6] by showing how to improve the quality of the mined models without requiring additional training data and how to deal with novel unseen objects.

Data Collection

The sensor data used in this experiment has already been used in [1, 6], thus, allowing us to compare our results against this work. In this data collection, over one hundred everyday objects in a real home were instrumented with passive RFID tags. Objects tagged include silverware, cooking utensils, hygienic products, and furniture among others. Over a period of six weeks, nine non-researcher subjects spent a single 20-40 min session to collect data by carrying out 14 activities of their choice out of a provided list of 65 activities of daily living (ADLs) while wearing a glove equipped with an RFID reader. In practice, the subjects selected to perform only the 26 activities shown in Figure 7.

5.2 Mining Activity Models from the Web

Given a set of activities A, the authors of [4] mine the list of objects O used for each activity a, and their corresponding usage probabilities $P(o \in O \mid a \in A)$ from the web. The primary assumption underlying the mining process is that textual description of activities on the web reflects the performance of activities in everyday life. The mining process mainly consists in the following steps: (1) First, find instructional or "how to" web pages $\widetilde{P}$ that contain a detailed description on how to perform each activity in A. (2) Second, extract the set of objects mentioned in each page by identifying nouns phrases (using a part of speech tagger), these nouns will be hypernyms or subsets of *{object}* or *{substance}* in WordNet. For each extracted object, the probability that the extraction denotes a physical object is computed as $w_{i,p} = p(object \mid noun)p(noun)$. In this equation, $p(noun)$ is the probability that the last word of the noun phrase is a noun as assigned by the POS tagger, and $p(object \mid noun)$ is computed by dividing number of occurrences of noun senses that are hypernyms of *{object}* or *{substance}* by the total number of occurrences of all noun senses. It is possible for a single object to have multiple weights by appearing several times in a single page, the final weight used is the average weight $\hat{w}_{i,p}$. Finally, the object probabilities $p(o_i \mid a)$ are computed as the fraction of pages in which the object o_i appeared weighted by its average extraction score on each page, i.e.:

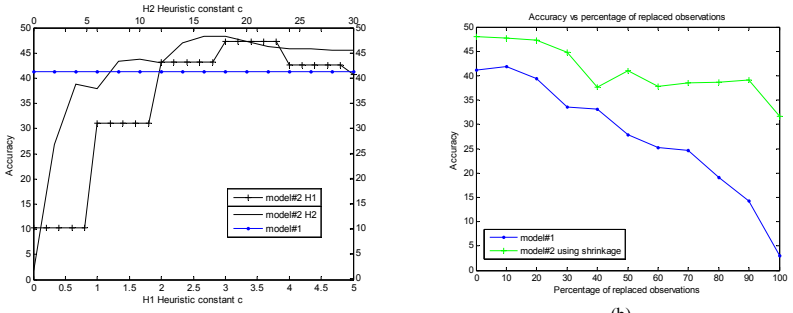
$$p(o_i \mid a) = \frac{1}{\left|\widetilde{P}\right|} \sum_{p} \hat{w}_{i,p}$$

The common sense information mined (activity recipes, and object observation probabilities) is compiled into an HMM for the task of activity inference. Each activity A is represented as one internal state in the HMM, and the object usage probabilities mined are used as the set of observations for each state $B_{ji} = P(o_i \mid a_j)$. For the transition matrix T, an expected activity duration $\gamma = 5$ is assumed, thus, all self-transition probabilities are set to $T_{jj} = 1 - 1/\gamma$. The remaining probability mass is uniformly distributed over the transitions to all other states. Finally, the prior state probabilities π are set to the uniform distribution over all activities. Using this representation, the classification task simply consists of inferring the most likely sequence of internal states by running the Viterbi algorithm over the sequences of observations. For more details about mining models from the web please see [4].

5.3 Improving Object Probabilities by Shrinkage

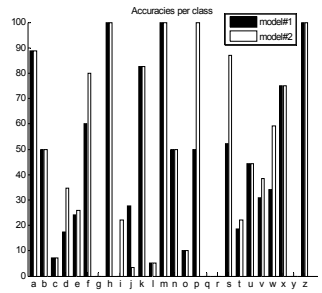
This experiment demonstrates the usefulness of shrinkage in improving the classification accuracy. First, we proceeded to generate the ontology from the list of 68 objects in the mined models and the sensor traces in [6]. Then, we construct two HMM models, *model#1* as described in Section 5.2, and *model#2* by performing shrinkage over the observation matrix of *model#1*. Finally, we search over the values of c to find the optimal value for the two heuristic functions (H1) $\lambda^{level} = e^{-c \cdot level}$, and (H2) $\lambda^{level} = 1/c^{level}$.

The plots in Figure 6a show the results for various values of c. In these plots, we observe that the maximum accuracy obtained is 48.35%, located at $c = [16,18]$ for heuristics (H2). This accuracy represents an improvement of 15.11% over the accuracy obtained using *model#1* (42%). This is an important result, because in [6] the authors also describe a procedure to learn from the sensor traces. Based on the segmentation obtained using mined models, new model parameters are learned using 126 sensor traces, which improve the accuracy of *model#1* by 19.2%. Here we have shown that just performing shrinkage and without using sensor data whatsoever, we achieve an improved accuracy of 15.11%. Consequently, we believe learning for sensor traces will further improve the accuracy. Figure 7 presents the accuracy per class results before and after performing shrinkage. Table 1 presents the confusion matrix as computed over the 65 segmented examples of the 26 ADLs.



(a) (b)

Figure 6. (a) Accuracy results after performing shrinkage using different constant values c in the heuristics, (H1) $\lambda^{level} = e^{-c \cdot level}$, and (H2) $\lambda^{level} = 1/c^{level}$ and (b) Accuracy vs. percentage of replaced observations using *model#1* and *model#2*.



A	adjust thermostat	N	make peanutbutter jelly sandwich
B	boil water in microwave	O	
C	brew a pot of tea	P	make a snack
D	brush your hair	Q	play solitaire
E	brush your teeth	R	put on make up
F	change a baby's diaper	S	read a magazine
G	clean a toilet	T	shave your face
H	clean the bathroom	U	take vitamins
I	clean the kitchen	V	use microwave
J	do laundry	W	use the telephone
K	dress a baby	X	use toilet
L	drink water	Y	vacuum carpets and floors
M	load and run a dishwasher	Z	wash your hands
			watch tv

Figure 7. List of the 26 ADLs activities collected and accuracy per activity results before and after performing shrinkage using the best heuristic and c value found.

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	8	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
b	0	5	0	0	0	0	0	3	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c	1	8	1	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
d	0	0	0	8	0	0	8	0	0	0	1	0	2	0	0	2	0	0	0	0	0	0	2	0	0	0
e	12	0	0	0	16	2	3	12	0	0	0	0	4	0	0	0	4	0	0	0	0	4	5	0	0	0
f	1	0	0	0	0	40	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
g	2	0	0	0	0	0	0	0	0	0	0	0	7	0	0	0	0	0	0	0	0	0	0	0	0	0
h	0	0	0	0	0	0	0	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
i	0	0	0	0	0	0	0	9	4	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	2	0
j	0	0	0	0	0	0	8	0	0	1	0	0	14	0	0	0	0	0	0	0	0	0	6	0	0	0
k	0	0	0	0	0	0	0	3	0	0	19	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
l	0	3	0	0	0	0	0	13	3	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m	0	0	0	0	0	0	0	0	0	0	0	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0
n	1	0	0	0	0	0	1	0	4	0	0	2	11	31	2	0	0	0	0	0	8	0	0	0	2	0
o	0	0	0	0	0	0	3	4	8	0	0	0	0	3	2	0	0	0	0	0	0	0	0	0	0	0
p	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0
q	0	0	0	7	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0
r	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
s	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	1	0
t	0	1	0	0	0	0	0	0	0	0	0	19	0	0	0	0	0	0	0	6	1	0	0	0	0	0
u	0	4	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	4	0	0	0	0	0
v	1	0	0	0	0	0	19	0	1	0	0	0	0	0	0	0	0	0	0	0	0	15	3	0	0	0
w	12	0	0	0	0	0	4	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	26	0	0	0
x	2	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	9	0	0
y	5	0	0	1	0	0	0	0	0	0	0	0	5	0	0	0	3	0	0	0	0	0	5	0	0	0
z	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	35

Table 1. Confusion matrix for the classification of the 26 ADLs using the shrinkage model (*model#2*). The letters are the same as the ones used in Figure 7. Rows indicate the hand-labeled class and columns indicate the predicted class label.

5.4 Robustness to Unseen Observations by Shrinkage

In this section, we provide experimental results showing that shrinkage improves model robustness when objects not found in the activity models are present in the sensor traces.

The experiment is performed as follows: first, the ontology generated in the previous experiment was expanded to *MaxParentLevel*=1, and *MaxChildLevel* =1 as described in Section 2.1. This guarantees that sibling nodes will exist for each leaf node in the ontology tree. Secondly, the observation matrix is extended to include all the new leaf nodes in the ontology that were not originally present. Thirdly, two HMM models were generated, *model#1* as described in Section 5.2, and *model#2* by performing shrinkage over the observation matrix of *model#1*. Then we proceed to replace *m*% of the observations in each sensor trace for a randomly selected sibling of the original observation in the ontology. Figure 8 shows three examples of the original, and modified ADLs sensor traces. The modified sequences are then concatenated into a single sequence, and the hidden sequence of states is computed running the Viterbi algorithm using models *#1*, and *#2*. The overall accuracy is computed as the number of observations whose inferred label matched ground truth divided by the total number of observations. Similarly the accuracy per activity is calculated by dividing the number of observations inferred correctly for each activity divided by the total number of observations for each activity.

Activity	Original and Replaced Traces
Brushing teeth	Original: light toothpaste floss light Replaced: light tooth_powder floss lamp
Watching TV	Original: remote magazine remote magazine Replaced: remote newspaper remote newspaper
Watching TV BAD EXAMPLE	Original: television couch remote couch Replaced: television sofa water cooler lawn chair

Figure 8. Example sequences where 50% of the observations were replaced

Figure 6b shows a plot comparing the overall accuracy versus the percentage of replaced observations for the two models. From this plot we can observe that (1) the accuracy of *model#2* is always greater than that of *model#1*, and (2) when 100% of the observations are replaced, the accuracy of *model#2* drops only 33% (from 48% to 32%) when the accuracy for *model#1* drops 91.66% (from 42% to 3.8%, which is equivalent to random guessing).

6 Conclusions

In this paper, we have presented a completely unsupervised approach to activity recognition that uses activity models automatically mined from the web in combination with shrinkage over an object ontology extracted from WordNet. The novelty of this approach relies on the fact that high level information is incorporated using shrinkage which provides the following benefits: (1) an improved accuracy by re-estimating the object observation probabilities of the mined models. We achieve an improvement of 15.11% in the overall accuracy in Section 5.3. (2) An approach to activity classification that requires no real sensor traces or training data, however, if training sequences are available, shrinkage can further improve accuracy. This is shown in Section 4.1 by simulation, where shrinkage reduces the number of training examples required to achieve a particular log-likelihood value from 70 to 1. Model parameters learned using shrinkage are closer to the true model as measured by the KL divergence between the true model, and the learned model. (3) And finally, the ability to reason about objects that are not present in the mined activity models but are used while performing an activity. This is achieved by estimating observation probability for the objects not present in the models by shrinking them towards the objects found in the models using the ontology. This is exemplified by showing that when 100% of the observations in real sensor traces are replaced, accuracy drops 91.66% for a model not using shrinkage, and only 33%, when shrinkage is used.

References

[1] M. Perkowitz, M. Philipose, D. J. Patterson, and K. Fishkin, "Mining Models of Human Activities from the Web," in *Proceedings of The Thirteenth International World Wide Web Conference (WWW '04)*. New York, USA, 2004.

[2] K. Fishkin, M. Philipose, and A. Rea, "Hands-On RFID: Wireless Wearables for Detecting use of Objects," in *Proceedings of the Ninth Annual IEEE International Symposium on Wearable Computers (ISWC '05)*. Osaka, Japan.

[3] A. Feldman, E. Munguia-Tapia, S. Sadi, P. Maes, and C. Schmandt, "ReachMedia: On-the-move Interaction with Everyday Objects," in *Proceedings of the Ninth Annual IEEE International Simposium on Wereable Computers (ISWC '05)*. Osaka, Japan, 2005.

[4] E. Munguia-Tapia, S. S. Intille, L. Lopez, and K. Larson, "The Design of a Portable Kit of Wireless Sensors for Naturalistic Data Collection," in *Proceedings of the 4th International Conference on Pervasive Computing (PERVASIVE '06)*., Dublin, Ireland: Springer-Verlag, 2006., to appear

[5] E. Munguia-Tapia, N. Marmasse, S. S. Intille, and K. Larson, "MITes: Wireless Portable Sensors for Studying Behavior," in *Proceedings of Extended Abstracts Ubicomp 2004: Ubiquitous Computing*. Vienna, Austria, 2004.

[6] D. Wyatt, M. Philipose, and T. Choudhury, "Unsupervised Activity Recognition Using Automatically Mined Common Sense," in *The Twentieth National Conference on Artificial Intelligence (AAAI 05')*. Pittsburgh, Pennsylvania, 2005.

[7] E. Munguia-Tapia, T. Choudhury, M. Philipose, and D. Wyatt, *Using Automatically Mined Object Relationships and Common Sense for Unsupervised Activity Recognition*, Technical Report IRS-TR-05-014, Intel Research Seattle, Seattle, WA, May 2005.

[8] J. Lester, T. Choudhury, N. Kern, G. Borriello, and B. Hannaford, "A Hybrid Discriminative/Generative Approach for Modeling Human Activities," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI '05)*, 2005.

[9] E. Munguia-Tapia, S. S. Intille, and K. Larson, "Activity Recognition in the Home Setting Using Simple and Ubiquitous Sensors," in *Proceedings of PERVASIVE 2004*, vol. LNCS 300, B. Heidelberg, Ed.: Springer-Verlag, 2004, pp. 158-175.

[10] D. Patterson, D. Fox, H. Kautz, and M. Philipose, "Fine-Grained Activity Recognition by Aggregating Abstract Object Usage," in *Proceedings of The Ninth Annual IEEE International Symposium on Wearable Computers (ISWC '05)*. Osaka, Japan, 2005.

[11] A. McCallum, R. Rosenfeld, T. Mitchell, and A. Ng, "Improving Text Classification by Shrinkage in a Hierarchy of Classes," in *Proceedings of the 15th International Conference on Machine Learning (ICML-98)*, J. W. Shavlik, Ed.: Morgan Kaufmann Publishers, San Francisco, US, 1998, pp. 359-367.

[12] D. Freitag and A. K. McCallum, "Information Extraction with HMMs and Shrinkage," in *Proceedings of the AAAI '99 Workshop on Machine Learning for Information Extraction*, 1999.

[13] C. R. Anderson, P. Domingos, and D. Weld, "Relational Markov Models and their Application to Adaptive Web Navigation," in *In Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2002, pp. 143-152.

[14] T. Gu, H. K. Pung, and D. Q. Zhang, "A Service-oriented Middleware for Building Context-aware Services," *Journal of Network and Computer Applications (JNCA '05)*, vol. 28, pp. 1-18, 2005.

[15] G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. Miller, *Introduction to WordNet: An On-line Lexical Database*, 1993.

[16] C. Stein, "Inadmissibility of the Usual Estimator for the Mean of a Multivariate Normal Distribution.," in *Proceedings of the Third Berkeley Symposium on Mathematical Statistics and Probability.*: University of California Press, 1955, pp. 197-206.