International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 1 Issue 6, August - 2012

# Discovering Patterns in Interactions between Humans and Animals by Using Tree Based Mining

Palivela Hemant
Computer Science and Engineering
EWIT, VTU
Bangalore, India

VijayKumar S
Computer Science and Engineering
EWIT, VTU
Bangalore, India

Sharadha K A
Computer Science and Engineering
EWIT, VTU
Bangalore, India

Prashanth G
Computer Science and Engineering
EWIT, VTU
Bangalore, India

Kalpana Patil
Computer Science and Engineering
EWIT, VTU
Bangalore, India

*Abstract*— **This paper defines a way such that we can predict the behavior of animals when they interact with humans. By using the capturing technique we are proposing a method for mining out the relevant output from the trees that will be generated from the collection of various sub trees. Human – Animal interactions such as giving commands to pet animals like dogs, cats etc can be visualized in this paper. Commands like sit, run, catch, stand, jump etc will be the nodes defined in the tree. Tree based mining will help us to analyze the patterns from the behavior flow of the tree formulated from the recordings.**

***Keywords-component; Interaction flow, interaction pattern, tree-based mining,***

## I. INTRODUCTION

Wildlife species interact with each other in numerous, complex ways. Fortunately, we can make some general statements about these interactions. This enables us to better understand the role species play within their communities and how individual species can positively or negatively affect the species around them.

Of the various types of interactions between species, most involve resources and consumers. A resource, in ecological terms, is something (such as food, water, habitat, sunlight, prey) that is required by an organism to perform a vital function such as grow or reproduce. A consumer is an organism that consumes a resource (such as predators, herbivores, or detritivores). Thus, most interactions between animals involve one or more competitor species vying for a resource.

Species interactions can be categorized into one of four basic groups based on how the participating species are affected by the interaction. These include:

- Competitive interactions
- Consumer-resource interactions
- Detritivore-detritis interactions
- Mutualistic interactions

Competitive interactions are interactions involving two or more species that are competing for the same resources and all species involved in the competitive interaction are negatively impacted. Competitive interactions are in many cases indirect, two species consume the same resource but, they do not directly interact with each other, instead they impact each other by the effect they have on the resource.

Consumer-resource interactions are interactions in which individuals of one species consumes individuals of another species. Examples of consumer-resource interactions include predator-prey interactions and herbivore-plant interactions. These consumer-resource interactions affect the species involved in different ways, the resource species is negatively impacted while the consumer species is positively impacted. Detritivore-detritis interactions involve a species that consumes the detritis (dead or decomposing organic matter) of another species. The detritivore-detritis interaction is a positive interaction for the consumer and has no impact on the resource. Finally, mutualistic interactions are interactions in which species, resource and consumer, benefit from the interaction. Populations of animals interact with each other and their environment in a variety of ways. One of the primary interactions a population has with its environment and other populations is due to feeding behavior.

The consumption of plants as a food source is referred to as herbivore and the animals that do this consuming are called herbivores. There are different types of herbivores. Those that feed on grasses are referred to as grazers. Animals that eat leaves and other portions of woody plants are called browsers, while those that consume fruits, seeds, sap, and pollen are called frugivores.

Populations of animals that feed on other organisms are called predators. The populations on which predators feed are called prey. Often, predator and prey populations cycle in a complex interaction. When prey resources are abundant, predator numbers increase until the prey resources wane. When prey numbers drop, predator numbers dwindle as well. If the environment provides adequate refuge and resources for prey, their numbers may again increase and the cycle begins again.

The concept of competitive exclusion suggests that two species that require identical resources cannot coexist in the same location. The reasoning behind this concept is that one of those two species will be better adapted to that environment and be more successful, to the point of excluding the lesser species from the environment. Yet we find that many species with similar requirements do coexist. Because the environment is varied, competing species can use resources in different ways when competition is intense, thus allowing space for one another.

When two interacting species, for example predator and prey, evolve together, they can influence the evolution of the other. This is referred to as co evolution. Sometimes co evolution results in two species that influence (both positively and negatively) from each other, in a relationship referred to as symbiosis. The various types of symbiosis include:

- Parasitism - one species (parasite) benefits more than the other species (host)
- Commensalism - one species benefits while a second species is neither helped nor injured
- Mutualism - both species benefit from the interaction.

The Human-Animal Bond is the dynamic relationship between people and animals in that each influences the psychological and physiological state of the other. Human-animal interaction has profound physiological consequences. People, in the contact with animals experience a decrease in blood pressure, reduced anxiety, and a general feeling of well-being. By observing the behavior of animals, children learn to be more nurturing and perhaps better parents to their own children. The therapeutic value of animals for socially isolated individuals in nursing homes, hospitals, hospices, and prisons has been documented. People in the presence of animals are often perceived to be more happy and healthy. Wild, zoo, farm, and companion animals are an integral component of our culture and socioeconomic environment. Animal welfare, or the humane care of animals, is a societal responsibility. However, there is little information available that relates animal health and welfare to such factors as genetics, stress, environment, and husbandry practices. A better understanding of the determinants of animal well-being is needed to optimize the comfort, health, performance, and sometimes survivability of all animal species [1].

Research confirms what most of us instinctively know to be true: the presence of animals in people's lives has a significant positive influence on the social, emotional and physical well-being of people. Our companion animals can ease loneliness and calm the emotions; they can make us laugh and make us feel needed; and they can soothe us in times of illness or hardship. Many of our companion animals have been trained to provide mobility and independence for those in need. There is a very strong bond between humans and animals.This relationship between humans and animals is referred to as Human-Animal Interaction (HAI).

There are many groups, small and large, formal and informal, that provide opportunities for enhancing HAI through their endeavours. As a result, the field of Human-Animal Interaction has grown considerably, due in no small part to the work of many of these groups in Australia. The groups listed in this Directory play a special role in fostering our understanding of the human-animal bond. That companion animals have a unique place in the human world is undeniable, as companion animals play an irreplaceable part in the enrichment of people's lives [2].

## II. RELATED WORK

The human – animal interactions have to be captured in a multimedia system using rear view. For the human interactions we already have systems based on image processing and computer visualization. Meetings span a range of informational and group activities. Examples include staff meetings, design discussions, project reviews, videoconferences, presentations, and classes. Usually handwritten notes, augmented with presentation material that's either hand copied or obtained from the speaker form the basis of meeting records. Some cases require more detail, so the meeting is recorded on audio or video. Those who attended the meeting along with those who missed it use these records for review, retrieving facts and details. Sometimes their activities are more involved. They study the records, prepare reports, and create meeting summaries [3].

In spring 2007 the PWN Waterleidingbedrijf Noord-Holland started with a pilot project with European bison (wisent) on the 200 ha sand dune area Kraansvlak, a closed-off area near the Kennemer Dunes (near Overveen / Haarlem), accompanied by scientific research.

Its aim is to acquire knowledge and experience with bison in the Dutch situation, the food strategy and the effects of the bison on the sand dune landscape, dune dynamics and dune vegetation. Besides the interaction between wisent and landscape, also the co-existence with other grazers as well as human visitors is being studied. After an acclimatization period, during which the animals could only be watched from a viewpoint or through the fencing, excursions into the area will be organized. The project will be evaluated after 5 years [4].

## III. Mining using tree based interaction

**Definition 1**:

A Tree is specified for the activity of denoting the interaction of the animals, animals with humans and set of different scenarios. The tree is defined as T= (V, E) where V is the set of labeled vertices, E are the edges that are denoted by {e1, e2, e3, e4…, en} € E. The interactions between the two labeled vertices are set in the form of links or edges. The pattern mining is done using the tree based mining algorithms.
Each node has a labeling function L attached as {l1, l2, l3, l4……, ln} € L and the labeling function defines words like sit, run, catch, stand, jump, bark. The mapping is done in the form V→L where V is the set of vertices. The left child of the parent is the interaction that is performed earlier and the right one after that.
Here L (Commands) = {sit, run, catch, stand, jump, bark**}**
Now let us define the whole working of the process in a tree based fashion. With each tree's root being the initial state and the transitions are triggered by the root node and the respective children are formed in the context of the interaction. Command incurs two different things either the animal will listen but the chances are that it may not reply. The study here is the human interactions with ecological beings other than the humans and will formulate the new work that defines the interaction.

We need to build the dataset so that it acts as a training set for the new system to be made as a possible input so that the correct decision output will be with utmost accuracy. We make a tree dataset by taking the set T = {T1, T2, T3 ......} where Ti is the individual tree built for one interaction.

Patterns are the sub trees. We will calculate the support of the occurrence of the tree that we will give as an input to the trained data set. The minimum relative support value (#supporting transactions / #total transactions) satisfied by all rules needs to be calculated by defining a support value for it say (8-9 percent). A frequent tree is visualized if the support value of T is more than the threshold value.

Consider that we have some set of trees having the value e*(total dataset) where e is the minimum support for the calculation. We are interested in finding out this particular set of trees.

We can make the trees to be isomorphic by exchanging the nodes like for example Command (Listen) * Acknowledge * Reject and Command (Listen) * Reject * Acknowledge. If the siblings are same then it can be omitted. We will take the isomorphic trees that have support more than e and from that we will take one and discard the rest. A reliable procedure for building the minimal set of hidden, Markovian states that is statistically capable of producing the behavior exhibited in the data -- the underlying process's causal states [5].

## IV. Pattern discovering algorithms

### A. *The actual algorithm*

The FP-Growth Algorithm is an alternative way to find frequent item sets without using candidate generations, thus improving performance. For so much it uses a divide-and-conquer strategy. The core of this method is the usage of a special data structure named frequent-pattern tree (FP-tree), which retains the item set association information.
In simple words, this algorithm works as follows: first it compresses the input database creating an FP-tree instance to represent frequent items. After this first step it divides the compressed database into a set of conditional databases, each one associated with one frequent pattern. Finally, each such database is mined separately. Using this strategy, the FP-Growth reduces the search costs looking for short patterns recursively and then concatenating then in the long frequent patterns, offering good selectivity.
In large databases, it's not possible to hold the FP-tree in the main memory. A strategy to cope with this problem is to firstly partition the database into a set of smaller databases (called projected databases), and then construct an FP-tree from each of these smaller databases.

### B. *FP-Tree structure*

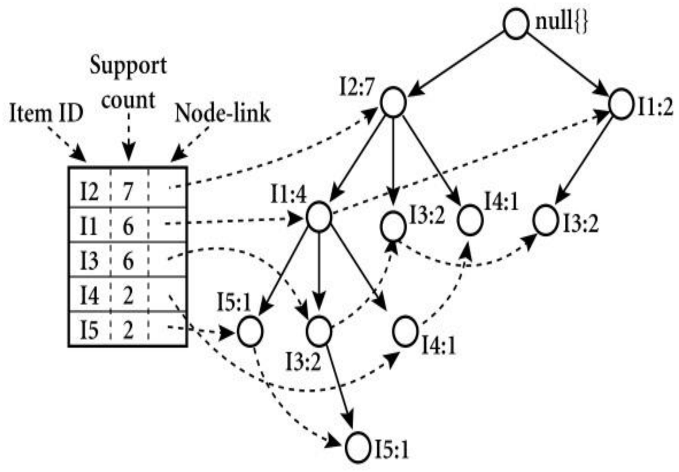
The frequent-pattern tree (FP-tree) is a compact structure that stores quantitative information about frequent patterns in a database. Han defines the FP-tree as the tree structure defined below. One root labeled as "null" with a set of item-prefix sub trees as children, and a frequent-item-header table (presented in the right side of Figure). Each node in the item-prefix sub tree consists of three fields:

- Item-name: registers which item is represented by the node;
- Count: the number of transactions represented by the portion of the path reaching the node;
- Node-link: links to the next node in the FP-tree carrying the same item-name, or null if there is none.

Each entry in the frequent-item-header table consists of two fields:

- Item-name: as the same to the node;
- Head of node-link: a pointer to the first node in the FP-tree carrying the item-name.

Additionally the frequent-item-header table can have the count support for an item. The Figure 1 below show an example of a FP-tree.



Algorithm 1: FP-tree construction

Input: A transaction database DB and a minimum support threshold e
Output: FP-tree, the frequent-pattern tree of DB.

Method: The FP-tree is constructed as follows
.
1. Scan the transaction database DB once. Collect F, the set of frequent items, and the support of each frequent item. Sort F in support-descending order as FList, the list of frequent items.
2. Create the root of an FP-tree, T, and label it as "null". For each transaction Trans in DB do the following:
3. Select the frequent items in Trans and sort them according to the order of FList. Let the sorted frequent-item list in Trans be [ p | P], where p is the first element and P is the remaining list. Call insert tree ([ p | P], T ).
4. The function insert tree ([ p | P], T ) is performed as follows. If T has a child N such that N.item-name = p.item-name, then increment N 's count by 1; else create a new node N , with its count initialized to 1, its parent link linked to T , and its node-link linked to the nodes with the same item-name via the node-link structure. If P is nonempty, call insert tree(P, N ) recursively.
5. By using this algorithm, the FP-tree is constructed in two scans of the database. The first scan collects and sort the set of frequent items, and the second constructs the FP-Tree [8].

TABLE I
Notations

| Notations | Description |
|---|---|
| TD | Dataset of interaction trees |
| ITD | Full set of isomorphic trees to TD |
| t | A tree |
| t^k | K- subtree |
| c^k | Set of candidates with K nodes |
| F^k | A set of frequent K Subtrees |
| e | Threshold minimum support |

Algorithm 1 used [8]:

FITM (TD, e) (Frequent interaction tree pattern mining)
Input: a tree database TD and a support threshold e
Output: all frequent tree patterns with respect to e
Procedure:
1. Scan database TD; generate its full set of isomorphic trees, ITD
2. Scan database ITD; count the number of occurrences for each tree t
3. Calculate the support of each tree
4. Select the trees whose supports are larger than e and detect isomorphic trees; if m trees are isomorphic, select one of them and discard the others
5. Output the frequent trees.

Algorithm 2 used [8]:
FISTM (TD; e) (Frequent interaction sub tree pattern mining)
Input: a tree database TD and a support threshold
Output: all frequent sub tree patterns with respect to
Procedure:
1. i←0
2. scan database TD, calculate the support of each node
3. select the nodes whose supports are larger than e to form F1
4. i←i + 1
5. for each tree ti in Fi, do
6. for each node t1 in F1, do
7. join ti and t1 to generate C^(i+1)
8. Sub tree Support Calculating (TD; t^(i+1)) //Calculate the support of each tree in C^ (i+1)
9. if there are any trees whose supports are larger than e, then select them to form F^(i+1) and return to Step (4)
10. else output the frequent sub trees whose supports are larger than e.

11. Algorithm 3 [8]:

Sub procedure.

Sub tree: - Support Calculating (TD, st)

```
1.  count← 0
2.  supp(st) ← 0
3.  for each tree t € TD do
4.  create sub trees S of t with any item s € S; |s|  = |st|
5.  flag ←false
6.  for each item s € S do
7.  generate isomorphic trees IS of s
8.  for each item is 2 IS do
9.  if tsc(is)=tsc(st) then
10. count ←count + 1
11. flag  ← true
12. break
13. if flag = true then
14. break
15. supp(st) count=|TD|
16. return supp(st)
```

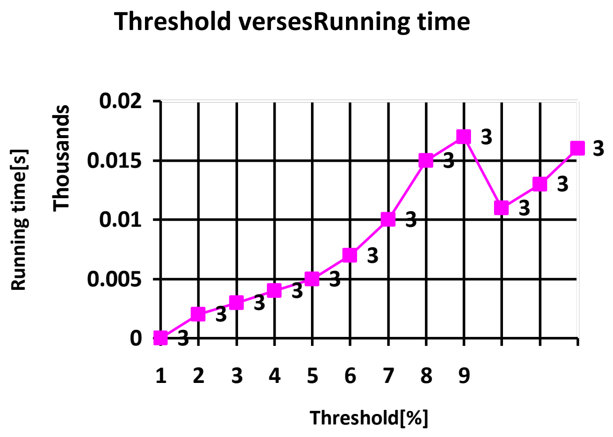
## V. Results

To deal with data transactions in a unified way, Weka provides several data types to serve this purpose; these data objects can be analogized to database terms such as transaction, table, record and field; they can be categorized into several layers as follows [7]:

1. Data Source: the source where we obtain the data; usually a data file;
2. Instances: the collection of data transactions, or database;
3. Instance: a single transaction, or record;
4. Item: a unique value for a field; this is an abstract class, its subclass such as  Nominal Item or
5. Numerical Item should be used in practice for nominal items or numerical items respectively.

Association rule mining uses NominalItem as its data structure; the built-in "equals()" function is used to determine where two items belong to the same field (i.e. getAttribute() returns the same value) and their values (or precisely, the index of their values) are also the same. Note that the frequency (or "support") of the two items are not compared.  Item's innate "compareTo()" function compares their frequencies and attribute name with descending order, that is, the natural order of Items is the descending order on their frequencies. In Weka, an Item's "value" is represented by an "index" of the value domain; the real meaning of this index can only be obtained by referencing the underlying Attribute object. Class "Attribute" contains attribute information of a data field, including its name and value domain; e.g., suppose the attribute's value domain is {"Suraj", "Vishal", "Ravi"}, a nominal item with a value index of 0 corresponds to "Li", and a value index of 1 corresponds to "Wang". Besides the above mentioned 4 layers of data objects, an "Item Set" object represents the collection of one or more data items; its inner structure is an array of integers, each of which represents the value index of one item; the size of this array is the length of this Item Set. Apriori-like algorithms use horizontal representation for transactions, in which the basic data element is "Instance" (aka transaction or record); FP-Growth algorithm uses vertical presentation of data, i.e., it uses data "Items" to construct the FP-trees; its implementation requires the ability of tree computation.

In the figure below we have taken the x axis as threshold and y axis as the running time.



## Conclusion

In this paper we have defined a method by which we can define a training system for the human and animal interaction and can also develop an ecosystem based training model for all the living beings a dog training system is a simple example. By taking the training set of the particular dog or a dolphin for example we can first let the trainer know the average behavior of the animal for some set of commands and then we can go further with the actual training procedures. In the next phase of this paper we will be working on the modification of the existing algorithm.

## References

[1] http://en.wikipedia.org/wiki/Category:Human-animal_interaction

[2] Australian Companion Animal Council Incorporated, 2006.

[3] Patrick Chiu, Ashutosh Kapuskar, Sarah Reitmeier, and Lynn Wilcox FX Palo Alto Laboratory, "*Room with a Rear View Meeting Capture in a Multimedia Conference*": IEEE, 2000, pp. 123–135.

[4] Workshop bison and Human Interactions in combination with field excursion, Kraansvlak, Overveen, The Netherlands, 13 October 2009

[5] An Algorithm for Pattern Discovery in Time Series, C Shalizi, K Shalizi… - Arxiv preprint csLG, 2002.

[6] T. Menendez, S. Achenbach, W. Moshage, M. Flug, E. Beinder, A. Kollert, A. Bittel, and K. Bachmann, "Prenatal recording of fetal heart action with magnetocardiography" (in German), *Zeitschrift für Kardiologie*, vol. 87, no. 2, pp. 111–8, 1998.

[7] Guang-li Yu, Ying Zhan and Shui Wang, "Analysis on Weka Foundation Classes and Algorithm Extending Method", Journal of Nanyang Institute of Technology, vol. 1, no. 6, pp. 9-11, 2009.

[8] Tree-Based Mining for Discovering Patterns of Human Interaction in Meetings, Zhiwen Yu, Senior Member, IEEE, Zhiyong Yu, Xingshe Zhou, Member, IEEE, Christian Becker, Member, IEEE, and Yuichi Nakamura, Member, IEEE, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 24, NO. 4, APRIL 2012.