Dynamic Rule Mining for Argumentation Based Systems

Wardeh, M., Bench-Capon, T., Coenen, F.
The University of Liverpool
Liverpool L69 3BX, UK

Abstract

Argumentation has proved to be a very influential reasoning mechanism particularly in the context of multi agent systems. In this paper we introduce PADUA (Protocol for Argumentation Dialogue Using Association Rules), a novel argumentation formalism that dynamically mines Association Rules (ARs) from the case background as a means to: (i) generate the arguments exchanged among dialogue participants, and (ii) represent each participant's background domain knowledge, thus avoiding the traditional knowledge base representations. Dialogue participants mine ARs from their own case data and then use these rules as arguments and counter arguments. This paper fully describes the PADUA formalism and proposes a suite of dynamic ARM algorithms to provide support for the argumentation process.
Keywords: Dynamic rule mining, argumentation.

1 Introduction

Argumentation is an increasingly influential reasoning mechanism, particularly in the context of multi agent systems. One specific model of argumentation is a *Persuasion Dialogue* [11] during which each participant tries to persuade the other participant(s) of their own thesis, by offering arguments that support this thesis. Despite the increasing use of argumentation in a variety of applications, many of the studies in the literature give little importance to the background knowledge the dialogue participants rely on in their efforts to persuade each other. The focus has been on the protocols and the use of argumentation as a means of communication rather than on the content to be communicated. Typically some form of knowledge base is assumed to provide the participants with the necessary domain knowledge. This knowledge base is used to produce arguments according to some underlying argumentation model. (Note that each participant's knowledge base is similar, but not necessarily the same.)

The work described here does not assume that such a knowledge base has been constructed. Instead arguments are mined directly from some set of records providing examples relating to a particular domain. The repository of background knowledge used by each participant is considered to be a binary valued data set where each record represents a previous case and each column an attribute taken from the global set of attributes described by the background knowledge. Given this formalism we can apply *Association Rule*

Mining (ARM) [1] techniques to the data set to discover relations between attributes, expressed in the form of *Association Rules* (ARs), which in turn can be used to support the argumentation process. This approach offers a number of advantages over the knowledge based approach:
- It enjoys general applicability as it does not require the generation of specialised knowledge bases.
- It employs an automatic rule generation process using a "tried and tested" data mining technique and consequently avoids the need for reference to a domain expert.
- The proposed approach avoids the need for any knowledge re-engineering because it works directly with the case data.
- The advocated method generates knowledge "on the fly" according to the requirements of the participant in question (again because it operates with the raw case data).
In addition the approach provides for a natural representation of the participants experience as a set of records, and the arguments as ARs. The advocated approach also preserves the privacy of the information that each participant "knows", therefore it can be used in domains which involve sensitive data.

This paper introduces PADUA (Protocol for Argumentation Dialogue Using Association Rules); a novel argumentation formalism that implements the approach advocated above using a "just in time" approach to ARM; i.e. particular groups of ARs, that conform to some particular requirement as dictated by the PADUA protocol, are mined dynamically as required. The advantage is that the system avoids the overheads associated with the generation of the complete set of ARs represented in the case base. The PADUA system supports three different forms of dynamic ARM request:
1. Find a subset of rules that conform to a given set of constraints.
2. Distinguish a given rule by adding additional attributes.
3. Generalise a given rule by removing attributes.

The rest of this paper is organised as following: Section 2 gives some necessary background information. Section 3 introduces the PADUA protocol and its main components. Section 4 describes the dynamic ARM algorithms developed to support the PADUA protocol. In Section 5 an example of the PADUA systems operation is presented together with some analysis and discussion. Finally some conclusions are drawn in 6.

2 Previous Work

2.1 Argumentation, Dialogue and Dialogue games

Argumentation is a form of nonmonotonic defeasible reasoning, in which participants interact to decide whether to accept or reject a given statement. During the process of argumentation, each participant forms and asserts arguments that contradict or undermine arguments proposed by the other participant(s). The basic idea behind this *argumentational reasoning* is that a statement is acceptable if it can be argued successfully against attacking arguments.

Persuasion Dialogue is used to model this type of argumentation. The literature includes plenty of examples of dialogical argumentation that fall into several areas such as distributed planning [15], education [13], and modelling legal reasoning (a survey can be found in [6]). In their well known typology of dialogue types [14], Walton and Krabbe defined Persuasion Dialogue to be initiated by a conflict in the participants' points of view, the main goal of the dialogue is to resolve this conflict by verbal means, while each participant tries to persuade the other(s) of its own point of view. Their typology of dialogue types classifies five other primary dialogue types, besides persuasion, which are negotiation, information seeking, deliberation, enquiry and Eristic dialogue. This categorisation is based upon: the information each participant has at the commencement of the dialogue; the goals of each individual participant; and the goals of the dialogue itself.

Formal dialogue games have been used successfully to model most of the atomic dialogue types in the Walton and Krabbe typology including persuasion [10, 12, 14]. *Formal Dialogue Games* are defined as interactions between two or more players, where each player moves by making utterances, according to a defined set of rules known as a *Dialogue Game Protocol*. Each move has an identifying name associated with it and some statement which contributes to the dialogue. Players keep on exchanging such moves until the dialogue terminates, according to some termination rules. Dialogue games comprise the following components [11]:
1. *Commencement Rules*: Rules that define the circumstances under which the dialogue commences.
2. *Locutions*: Rules indicating what utterances are permitted at every stage of the dialogue.
3. *Combination Rules*: Rules that describe the dialogical contexts under which particular locutions are permitted or not, or obligatory or not.
4. *Commitments*: Rules that define the circumstances under which participants express commitment to a proposition.
5. *Termination Rules*: Rules that define the circumstances under which the dialogue ends.

2.2 Dynamic Association Rule Mining

The original objective of *Dynamic ARM* (D-ARM), also sometimes referred to as *On-line* ARM, was to address the increasing computational requirements for exploratory ARM (usually involving manual parameter tuning). Subsequently D-ARM has been used in the context of dynamic data mining applications where repeated ARM invocations are required to obtain different sets of rules either with different content (attributes or consequents) or different thresholds. The fundamental idea is to summarise the dataset so that all information required for future association rule mining is encoded in an appropriate data structure that will facilitate fast interaction.

D-ARM was, arguably, first proposed by Amir et al. [4] who used a trie data structure to store the data set and conducted experiments using the (sparse)

Reuters benchmark document set. Although Amir et al. allowed questions such as "find all the ARs with a given support and confidence threshold" to be answered, their system could not answer questions such as "find the association rules that contain a given item set". The approach by Amir et al. is essentially not dissimilar to later approaches to ARM, such as TFP [7] and FP-growth [8], that used an intermediate (summarising) data structure within the overall ARM process (P-trees and FP-trees respectively) although these later approaches did not explicitly considered the advantages with respect to D-ARM that their data structures offered.

The term On-line ARM was introduced by Aggarwal and Yu in 1997 in a technical report. In a subsequent publication, Aggarwal and Yu [2], the authors state that "The idea of on-line mining is that an end user ought to be able to query the database for association rules at differing values of support and confidence without excessive I/O or commutation". Aggarwal and Yu define an adjacency lattice, where two nodes are adjacent if one is a superset of the other, and use this structure for fast (on-line) rule generation. The lattice contains only itemsets whose support is greater than some minimum and consequently only ARs with support above this value can be generated. Hidber [9] also generate a lattice but the user can influence its growth by reducing the support threshold as the algorithm proceeds.

2.3 T-trees

As noted above, for D-ARM to operate successfully a summarising structure is required. In the work described here a T-tree (Total support tree) [7] is used. A T-tree is a "reverse" set enumeration tree data structure; Set enumeration trees impose an ordering on items and then enumerate the itemsets according to this ordering and T-trees are reverse in the sense that nodes are organized using reverse lexicographic ordering; the reason behind this reverse ordering is that T-tree differs from typical set enumeration trees in that the nodes at the same level at any sub-branch of the tree are organized as into 1D arrays so that array indexes represent column numbers, hence "reverse" version of the tree enables direct indexing based on the attribute (column) number.

An example of the T-tree structure is given in Figure 1. In the figure each record in the data set includes the items set x or y, these are the class attributes for the cases and are what the competing PADUA players will wish to establish. It should also be noted that each branch of the T-tree contains the itemsets rooted at a particular end itemset, thus all itemsets involving the class x (y) are contained in one branch of the T-tree (other branches are required for calculating individual AR confidence values). The implementation of this structure is illustrated in Figure 2, where each node in the T-tree is implemented as an object comprised of a support value and a reference to an array of child T-tree nodes.

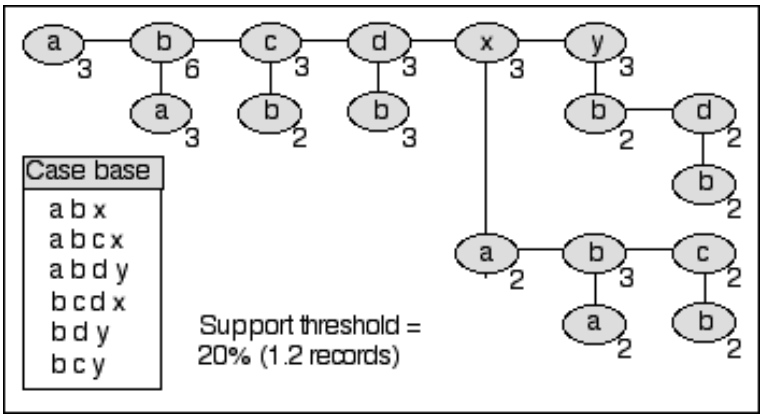

Fig. 1. Example T-tree

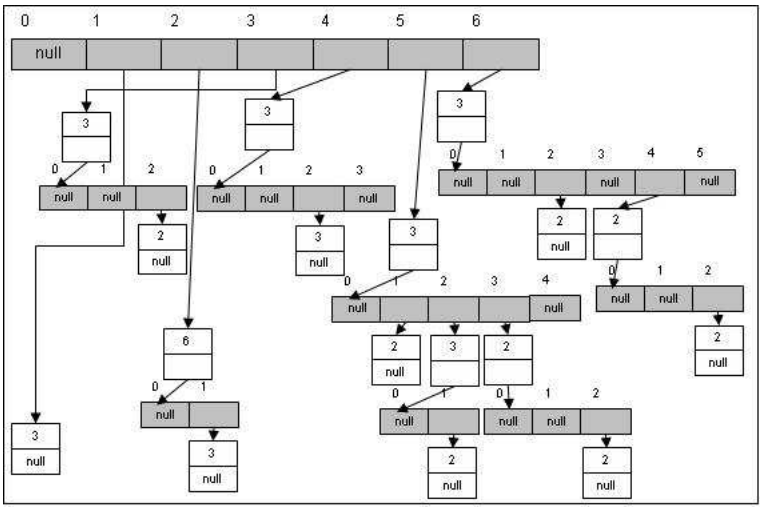

Fig.2. Example T-tree Implementation

The advantages offered by the T-tree structure are as follows:
1. Reduced storage requirements compared to those required by more traditional T-tree structures.
2. Fast look up facilities (by indexing from level to level)
3. Computational advantages because the frequent itemsets with particular consequence (classes) are stored in a single branch of the tree.

These advantages all serve to support D-ARM in general and the requirements for D-ARM supported argumentation in particular. Note that each PADUA player has its own T-tree (set of supported itemsets from which ARs can be obtained) generated from the player's individual case base.

3 PADUA Protocol

3.1 Dialogue Scenario

The proponent starts the dialogue by proposing some AR ($R : P \rightarrow Q$), which premises (P) match the case, and the conclusion (Q) justifies the agent's position. Then the opponent has to play a legal move (see Sub-section 3.4) that would undermine the initial rule proposed by the proponent: four of these moves involve some new rule. This is mined from the player's background database, and represents an attack on the original rule. The turn then goes back to the proponent which has to reply appropriately to the last move. The game continues until one player has no adequate reply. Then this player loses the game, and the other player wins. In fact this is the 'cannot argue any further' situation, and PADUA disallows playing rules that are weaker than previously played ones, which guarantees the dialogues's coherency.

3.2 PADUA Framework

The formal PADUA framework, *Argumentation Dialogue Framework* (ADF), is defined as follows:

$$ADF =< P, Attr, I, M, R, Conf, playedMoves, play > \quad (1)$$

Where P: denotes the players of the dialogue game. $Attr$: denotes the whole set of attributes in the entire framework. I: denotes the instance argued about. M: denotes the set of possible (legal) moves. R: denotes the set of rules that govern the game. $Conf$: denotes the confidence threshold, all the association rules proposed within this framework must satisfy this threshold. $playedMoves$: denotes the set of moves played in the dialogue so far, this set of played moves represents the commitment store of the dialogue system under discussion. Finally, *play*: is a function that maps players to some legal move.

3.3 PAUDA Players

Each player in a PADUA game ($\forall p \in P = Pro, Opp.$)is defined as a dialogical agent [3]:

$$\forall p \in P : p =< name_p, Attr_p, G_p, \Sigma_p, >>_p > \quad (2)$$

where: (i) $name_p$ is the *player* (*agent*) name, $\forall p \in P$ then $name(p) \in \{pro, opp\}$ (ii) $Attr_p$ is the set of attributes the player can understand (knows about), (iii) G_p: is the set of *goals* (class attributes) the player tries to achieve, G_p is defined as a subset of the attributes set $Attr_p$, i.e. G_p is the set of class attributes this player tries to demonstrate to be true, (iv) Σ_p: is the set of potential dynamic ARs the player can mine or has mined from its T-tree and (v) $>>_p$: represents the preferences order over Σ_p, a definition of this preference relationship is suggested as $>>_p$: $\Sigma_p \times \Sigma_p \rightarrow \{true, false\}$, but the exact implementation of this relation may differ from player to player. Σ_p is defined as follows:

$\forall p \in P : \Sigma_p =< T_p, R_p, Dr_p >$; where (i) T_p is the T-Tree representing the background data set, R_p is the set of association rules previously mined by this player and kept in store (thus $R_p = \{r : r =< Prem, Conc, Conf >\}$ where r is AR) and (iii) Dr_p is a function that maps between legal moves and suitable rules ($Dr_p : Tp \times M \rightarrow R$, where R is the set of all possible association rules).

3.4 PAUDA Legal Moves and Game Rules

The set (M) describes the six possible PADUA moves (M) that a player ($p \in P$) can play. The confidence measurement is used to compare the strength of the rules introduced by these moves, mainly because it's been implemented excessively in literature. PADUA moves are defined as follows.:
1. *Propose Rule*: Propose a new rule with a confidence higher than a given confidence threshold, the permises of this rule should be present in the instance under consideration.
2. *Distinguish*: Undermine the current rule proposed by the other player by adding a new attribute(s) satisfied by the instance, such that the confidence of the proposed rule is reduced below a confidence threshold.
3. *Unwanted Consequences*: Indicate that certain attributes in the consequent of the current rule are not present in the current instance.
4. *Counter Rule*: Propose a new rule, with a confidence value in excess of the previously proposed rule, that contradicts the consequent of the previous rule. The permises of the new rule should be present in the instance under consideration.
5. *Increase Confidence*: Add some new attribute, satisfied by the given instance, to the current rule so that the overall confidence rises.
6. *Withdraw Unwanted Consequences*: Remove unwanted attributes from the previously proposed rule while maintaining an appropriate level of confidence.

Not all of the above moves are "legal" moves at every stage of the game. The legal moves each player ($p \in P$) can play are determined by the following set of rules:
1. *Commencement Rules*: The dialogue always starts with a Propose Rule move played by the proponent.
2. *Locutions and Combination Rules*: Table 1 lists the possible moves that each player can play in respons to a move played by the other player. The next move column lists the legal moves according to desirability, according to one plausible strategy.
3. *Termination Rules*: The dialogue ends when a player can not find a suitable rule in its own data set to respond to a move played by the other player.

Move	*Label*	*Next Move*
1	Propose Rule	3, 2, 4
2	Distinguish	3, 5, 1
3	Unwanted Cons.	6, 1
4	Counter Rule	3, 2, 1
5	Increase Conf.	3, 2, 4
6	Withdraw Unwanted Cons.	3, 2, 4

Table 1. Possible Moves

4 Dynamic Association Rules Generation

The basic idea behind the PADUA approach is to mine ARs dynamically as needed according to: (i) desired minimum confidence, (ii) a specified consequent and (iii) a set of candidate attributes for the antecedent (a subset of the attributes represented by the current case). ARs are generated as required by traversing the T-Trees in such a way so that only the rules that match the content selection criteria of some move $m \in M$ are generated. Three dynamic AR retrieval algorithms were developed to support the PADUA protocol:

1. *Algorithm A*: Find a rule that conform to a given set of constraints.
2. *Algorithm B*: Distinguish a given rule by adding additional attributes.
3. *Algorithm C*: Revise a given rule by removing attributes.

Algorithm A (Fig 3) is used to find a new rule (moves 1, 2, 5 and 6) given (i) a current instance (I), (ii) a desired class attribute ($c \in G_p$) and (iii) a desired confidence threshold. The algorithm attempts to minimise the number of attributes in the rule. The algorithm operates by generating candidate itemsets, using the input values, in a level-wise manner; starting with the 2-itemset level in the T-tree(one attribute from the case and the class attribute). For every generated itemset ($S = (A \cup c)$), the T-tree is traversed for the node representing this itemset, if such a node exists, rules of the form ($P \rightarrow Q \cup c$)such that ($P \cup Q = A$), are generated, the algorithm returns the first rule that satisfies the given confidence threshold, otherwise the generation process continues until the entire T-tree has been processed.

Algorithm B (Fig 4) is used to distinguish an input rule $r = (P \rightarrow Q)$. The algorithm operates as follows: (i) generate the candidate itemsets ($P \cup Q \cup a_i$) where ($a_i \in I/(P \cup Q)$), (ii) search the T-tree subbranches for the node representing this itemset, (iii) if such a node exists generate a rule of the form $\acute{r} = ((P \cup a_i) \rightarrow Q)$ if the rule confidence is lower than the input rule confidence return the rule, otherwise traverse through the subtree which root is the candidate itemset for a rule of the form $\acute{\acute{r}} = (\acute{P} \rightarrow \acute{Q})$ that satisfies the conditions listed in the algorithm.

```
Algorithm A (inputs: instance I, class c,
             input T-Tree T,confidence threshold conf)
begin
 ∀s(a_i ∪ c : a_i ∈ I) ∈possible frequent 2-itemset
 if node(s) ∈ T
  generate rule r_dist(a_i ∪ P → Q)
  if r.confidence ≥ conf
   return r
  else
   level = 2
   while (no rules found) and (level ¡ T.max-level)
    ∀s(a_1, ..., a_level ∪ c : a_i ∈ I) ∈
    possible frequent(level+1)-itemset
    if node(s) ∈ T
     if ∃ association rule r(P → Q):
      c ∈ Q and (P ∪ Q = s) and r.confidence ≥ conf
      return r
     else
      Level++
end
```

Fig.3. Algorithm A - Propose New Rules.

```
Algorithm B (inputs: rule r(P → Q), instance I,
             class c,input T-Tree T,
             confidence threshold conf)
begin
 I_sub = I/(P ∪ Q)
 ∀ possible frequent itemset
 s(a_i ∪ P ∪ Q) : a_i ∈ I_sub
 if node(s) ∈ T
  generate rule r_dist(P ∪ a_i → c)
  if r_dist.confidence ≥ r.confidence
   return r_dist
  else
   traverse the sub T-Tree T(s)
   for every child node n ∈ T(s)
    if ∃ association rule r_dist(Ṕ → Q́):
    c ∈ Q́ and (Ṕ ∪ Q́ = n) and (Ṕ ⊆ I)
    and (r_dist.confidence ≤ conf)
     return r_dist
end
```

Fig.4. Algorithm B - Distinguish Rule.

In order to withdraw some unwanted consequences (X) from an input rule ($r = P \rightarrow Q \cup X$), *Algorithm C* (Fig 5) tries first to produce a rule ($\acute{r} = P \rightarrow Q$). If such a rule satisfies the confidence threshold, then the algorithm returns this rule, otherwise, the candidate itemsets are generated and rules are produced and tested in a very similar manner to *Algorithm B* to produce the

rule ($\acute{r} = P \rightarrow Q \cup Y$) where ($X \cap Y = \phi$). A player ($p \in P$) may apply *Algorithm C*, both as a defender or attacker of some thesis under discussion. *Algorithm C* therefore takes the status of the player into consideration, so that if the player is defending its point of view the algorithm search for rules which confidence is equal or higher than the input rule. On the other hand, if the player is attacking its opponent's thesis, then the returned rules confidence must not be higher than the confidence of the input rule.

```
Algorithm C (inputs: rule r(P → Q ∪ X), instance I,
            class c,input T-Tree T,
            confidence threshold conf,player role role)
begin
 if (node(P ∪ Q) ∈ T)
  generate rule r_with(P → Q)
   if (role = defender) and
       (r_with.confidence ≥ r.confidence)
     return r_with
   else
    if (role = opponent) and
       (r_with.confidence ≤ r.confidence)
     return r_with
 else
   I_sub = I/(P ∪ Q)
   ∀s(a_i ∪ P ∪ Q : a_i ∈ I_sub) ∈
   possible frequent itemset
   if node(s) ∈ T
    ∀ association rule r_with(P → Q ∪ a_i)
     if (role = defender) and
        (r_with.confidence ≥ r.confidence)
      return r_with
     else
     if (role = opponent) and
        (r_with.confidence ≤ r.confidence)
      return r_with
    else
     traverse the sub T-Tree T(s)
     for every child node n ∈ T(s)
      if ∃ association rule r_with(Ṕ → Q́):
      c ∈ Q́ and (Ṕ ∪ Q́ = n) and (Ṕ ⊆ I)
       if (role = defender) and
          (r_with.confidence ≥ r.confidence)
        return r_with
       else
       if (role = attacker) and
          (r_with.confidence ≤ r.confidence)
        return r_with
end
```

Fig.5. Algorithm C - Withdraw Unwanted Consequences

```
Instance: [adoption-budget-resolution=y, physician-fee-freeze=y.
religious-groups-in-schools=n, el-salvador-aid=y,
superfund-right-to-sue=y, immigration=y, export-act-south-africa=y.]
(1)  proponent  Propose Rule
{physician-fee-freeze=y}->{className = democrat} 92.94%

(2)  opponent   Distinguish
{physician-fee-freeze=y, export-act-south-africa=y}->
{className = democrat, immigration=y} 53.33%

(3)  proponent  Propose Rule
{physician-fee-freeze=y, el-salvador-aid=y}->{className =
democrat} 92.68%

(4)  opponent   Counter Rule
{religious-groups-in-schools=n}->{className = republican} 93.33%

(5)  proponent  Distinguish
{adoption-budget-resolution=y, religious-groups-in-schools=n}->
{synfuels-corporation-cutback=y, className = republican} 37.17%

(6)  opponent   Unwanted Consequences
{synfuels-corporation-cutback=y} not in the case

(7)  proponent  Propose Rule
{physician-fee-freeze=y, immigration=y}->{className = democrat}
93.75%
...
(12) opponent   Propose Rule
{adoption-budget-resolution=y, religious-groups-in-schools=n}->
{className = republican} 96.0%

(13) proponent  Distinguish
{adoption-budget-resolution=y,religious-groups-in-schools=n,
export-act-south-africa=y}->{className = republican} 50.0%

(14) opponent   all moves fail
     proponent  wins --> class = democrat
```

Fig.6. Example Dialogue

5 Experimentation and Analysis

The dynamic AR algorithms (figures 3, 4, 5) were tested using the congressional voting records data set [5]. This dataset includes votes for each of the U.S. House of Representatives members of Congress (in the 1984 US congressional elections) on the 16 key votes identified by the CQA. The congressional voting records database contains 435 instances, among which 45.2% are Democrats

and 54.8% are Republicans. The dataset, original comprising 17 binary attributes (including the class attribute) was normalised to 34 unique numerical attributes (numbered from 1 - 34) each corresponding to certain attribute value. The last two values (33 and 34) represents the two classes *Republican* and *Democrat* respectively). This dataset was horizontally divided into two equal size datasets, each of which was assigned to a player in PADUA framework. A T-tree was built for each player using a 10% support threshold, a minimum support threshold of 75% was adopted.

In the dialogue shown in (Figure 6) the two players (the proponent and the opponent) used the protocol rules discussed earlier (Sub-section 2.6), where the moves to be played are determined by the content of the moves played last. In the example the proponent (*pro*) starts the dialogue game by putting forward rule (1) ({`physician-fee-freeze =y`} `->` {`className = democrat`} `92.94%`), to establish that the given case falls under class *Democrat*, in the next move the opponent (*opp*) distinguishes the previous rule by adding the attribute (`export-act-south-africa=y`) to its premises, which causes the confidence to drop below the acceptable threshold. The dialogue continues with rules being proposed, distinguished and rejected for having unacceptable consequences, until step (12) where *opp* proposes a counter rule concluding that the class of the example case is *Republican*, with 96.0% confidence, but *pro* successfully distinguishes this rule in the following move (13). At this point the opponent has no valid moves to play, and thus the proponent wins this game, and persuades the opponent that the example case represents a *Democrat* candidate.

6 Conclusions

In this paper we have described a novel application of D-ARM to support argumentation, specifically dialogue games. The PADUA system (Dynamic Rule Mining for Argumentation Based Systems) is described. PADUA uses D-ARM to obtain ARs in a "just in time" manner that avoids generating all ARs with a confidence value above a given threshold. ARs are generated by interacting with the T-tree data structure that supports fast interaction times. Three specific D-ARM algorithms are described to either: (i) find a subset of rules that conforms to a given set of constraints, or (ii) distinguish a given rule by adding additional attributes or (iii) revise a given rule by removing attributes. The operation of the system is illustrated with an example taken from Congressional Voting benchmark data set. the approach introduced in this paper enjoy certain advantages when compared with other D-ARM techniques in the literature, specially regarding the generation of argumentation rules, mainly answering questions like "find the association rules that contain a given item set", "mine association rules with specific conclusions"...etc. another important advantage of the suggested techniques is that rules of various confidence thresholds can be mined from the auxiliary T-tree.

References

[1] R. Aggrawal, T. Imielinski, and A. N. Swami (1993). Association rules between sets of items in large databases. In Proc. ACM SIGMOD Int. Conf. on Management of Data, pp 207-216.

[2] C.C. Aggarwal and P.S. Yu (1998). Online Generation of Association Rules.Proc. i4th International Conference on Data Engineering (ICDE'98), IEEE, pp 402-411.

[3] L. Amgoud and S. Parsons (2001). Agent dialogues with conflicting preferences. Proc. 8th International Workshop on Agent Theories, Architectures and Languages, pp 1-15.

[4] A. Amir, R. Feldman and R. Kashi (1997). A New and Versatile Method for Association Generation. Proc. 1st Conf. Principles of Data Mining and Knowledge Discovery (PKDD), Springer LNCS, pp 221-231.

[5] C.L. Blake and C.J. Merz (1998). UCI Repository of machine learning databases http://www.ics.uci.edu/ mlearn/MLRepository.html, Irvine, CA: University of California, Department of Information and Computer Science.

[6] T.J.M. Bench-Capon and H. Prakken (2006).Argumentation. In Lodder, A.R. and Oskamp, A. (Eds),Information Technology and Lawyers: Advanced technology in the legal domain, from challenges to daily routine, Springer Verlag, pp 61-80.

[7] F. Coenen, P. Leng and G. Goulbourne (2004).Tree Structures for Mining Association Rules. Journal of Data Mining and Knowledge Discovery, Vol 8, No 1, pp25-51.

[8] J. Han, J. Pei and Y. Yiwen (2000).Mining Frequent Patterns Without Candidate Generation. Proceedings ACM-SIGMOD International Conference on Management of Data, ACM Press, pp1-12.

[9] C. Hidber (1999). Online association rule mining. Proc ACM SIGMOD international conference on Management of data, pp 145-156.

[10] P. Mcburney and S. Parsons (2001). Representing epistemic uncertainty by means of dialectical argumentation, In Annals of Mathematics and Artificial Intelligence 32, 125-169.

[11] P. Mcburney and S. Parsons (2002). Games That Agents Play: A Formal Framework for Dialogues between Autonomous Agents. In Jo. of logic, language and information, 11(3), pp 315-334.

[12] H. Prakken (2000).On dialogue systems with speech acts, arguments, and counterarguments. Proc, 7th European Workshop on Logic in Artificial Intelligence (JELIA 2000), Springer-Verlag, pp 224-238.

[13] E. Sklar and S. Parsons (2004).Towards the Application of Argumentation-Based Dialogues for Education. Proc. 3rd International Joint Conf. on Autonomous Agents and Multiagent Systems, Vol 3, pp 1420-1421.

[14] D. N. Walton and E. C. W. Krabbe. Commitment in Dialogue: Basic Concepts of Interpersonal Reasoning. SUNY Press, (1995), Albany, NY, USA.

[15] Y. Tang and S. Parsons (2005). Argumentation-based dialogues for deliberation. Proc. 4th Int. Joint Conf. on Autonomous Agents and Multiagent Systems, pp 552-559.