Argument Mining Using Highly Structured Argument Repertoire

Safia Abbas[1] and Hajime Sawamura[2]

[1] Graduate School of Science and Technology, Niigata University
8050, 2-cho, Ikarashi, Niigata, 950-2181 JAPAN
safia@cs.ie.niigata-u.ac.jp
[2] Institute of Natural Science and Technology,
Academic Assembly,Niigata University
8050, 2-cho, Ikarashi, Niigata, 950-2181 JAPAN
sawamura@ie.niigata-u.ac.jp

Abstract. Argumentation theory is considered an interdisciplinary research area. Its techniques and results have found a wide range of applications in both theoretical and practical branches of artificial intelligence, education, and computer science. Most of the work done in argumentation use the on-line textual data (i.e. unstructured or semi-structured) which is intractable to be processed. This paper reports a novel approach to build a Relational Argument DataBase (RADB) with managing tools for argument mining, the design of the RADB depends on the Argumentation Interchange Format Ontology(AIF) using "Walton Theory". The proposed structure aims to: (i) summon and provide a myriad of arguments at the user's fingertips, (ii) retrieve the most relevant results to the subject of search, (iii) support the fast interaction between the different mining techniques and the existing arguments, and (iv) facilitate the interoperability among various agents/humans.

1 Introduction

Argumentation theory, or argumentation, embraces the arts and sciences of civil debate, dialog, conversation, and persuasion. It studies rules of inference, logic, and procedural rules in both artificial and real world settings. Argumentation is concerned primarily with reaching conclusions through logical reasoning, that is, claims based on premises. Its techniques and results have found a wide range of applications in both theoretical and practical branches of artificial intelligence, education, and computer science [8, 7, 11]. Among other things, we are mainly concerned with argument mapping, analysis and formal computational argumentation frameworks, where many efforts have been most devoted as far as we see in the literature[9].

Argument mapping (e. g., Compendium, Araucaria, Rationale, etc.) aims at improving our ability to articulate, comprehend and communicate reasoning, by producing diagrams of reasoning and argumentation for especially complex arguments and debates. It is greatly anticipated that it helps students learning critical thinking methods as well as uses promoting critical thinking in the daily life. On the other hand, the main concern in formal computational argumentation frameworks is to formalize methods in which the final statuses of arguments are to be decided semantically and/or dialectically [8].

Argument mining and argument discovery technologies are particularly needed to summon the arguments, and help the user who is looking for specific subject or piece of information over a large-scale database. In this paper, we present a novel approach to sustain argument analysis, retrieval, and re-usage from a relational argument database (highly structured argument repertoire). Different mining techniques are used to provide a myriad of arguments at the user's fingertips, refine the user's analysis background, and reveal more relevant search results. This work mainly concerns with mining arguments through RADB, which summarizes the argument dataset. Such representation supports the fast interaction between the different mining techniques and the existing arguments, and facilitates the interoperability among various agents/humans. In addition, the framework outlines are illustrated, where a mining classifier agent is integrated with an intelligent tutoring system (ITS).

Such integration assists in deepening the understanding of negotiation, decision making, critical thinking and improving the analysis and intellectual process of students.

The paper is organized as follows. Section 2 illustrates the design and the implementation of the relational argument database (highly structured argument repertoire) together with different mining techniques. Section 3 motivates our work and compares it to the most relevant work performed in the field. Finally, conclusions and future work is presented in Section 4.

2 Relational Argument DataBase

A relational database can be defined as a set of information reformulated and categorized into a set of files (tables) that can be accessed, gathered (queried), and manipulated in different manners. According to the AIF ontology, arguments can be represented semantically in the form of nodes connected with directed edges in a directed graph known as argument network[1]. If the cyclic problem (the same information node (I-node) refines more than one scheme node (S-node)) is avoided, the arguments can semantically be represented as directed tree, which can be structured in the form of well-established relational database, and annotated as relational argument database (RADB).

2.1 Design and Implementation

This subsection describes the building blocks for the relational arguments data bases. We consider the AIF ontology [1, 6] with some restrictions (such that no edge emanates from I-node to I-node), and Walton schemes [4] for arguments analysis. Any argument scheme based on Walton theory can be represented as shown in *Fig.*1. which represents a general skeleton for the different schemes

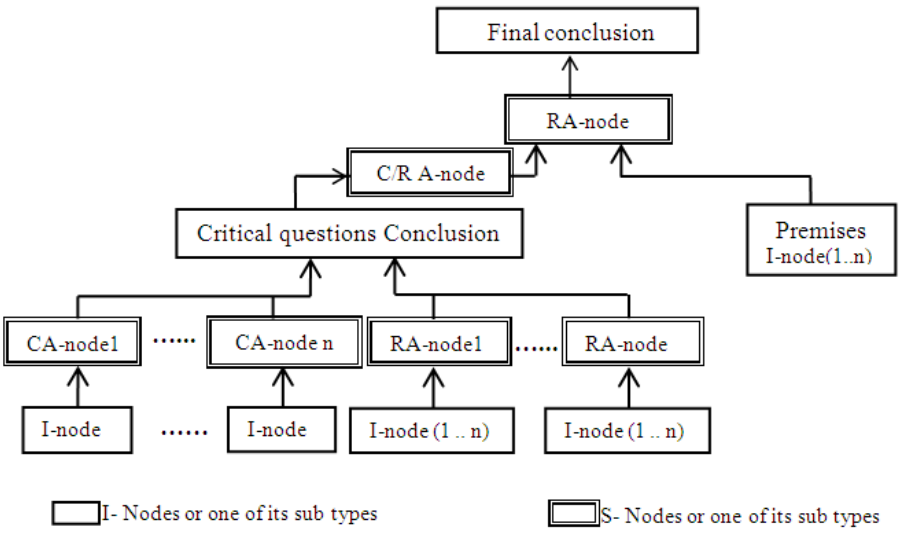
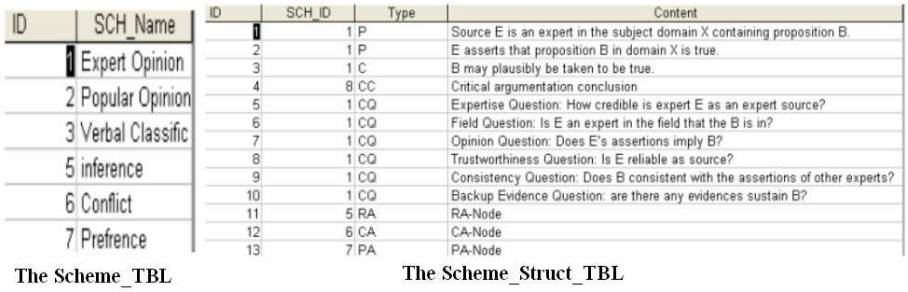


Fig. 1. Argument network representation for different Walton schemes

description in Walton theory[4]. The premises block gathers the different premises types (majors, minors). The critical question's conclusion block assembles the result of the different critical questions together with the results of the different presumption questions that are to be exposed in a specific scheme.

Considering the canonical representation for the schemes, we pose some sentiments about the relational database baselines. In our design, we gather the different scheme information into three basic files (tables): Scheme_TBL, Scheme_Struct_TBL and Data_TBL. First, the scheme kind is formulated in Scheme_TBL *Fig.* 2, in which rows act as records of data for different schemes, and columns as features (attributes) of records. The Scheme_TBL creates an identification number(ID) for each scheme name ($Scheme\_Name$), where this ID plays a role of primary key for the table and foreign key in the others. In addition, any ID attribute will stand for the same function in

all files. Scheme_Struct_TBL assembles the different information associated with different schemes. Such that, *Scheme_Id* stands for the foreign key of *Scheme_TBL*, indicating the scheme concerned. The *Content* field contains the details of the associated information (premises, conclusion,... etc.). The *Type* field has four values, P for premises, C for conclusion, CQ for critical question and CC for critical argument conclusion. For instance, the expert opinion scheme[4] can be represented as shown in Scheme_Struct_TBL of *Fig.* 2.



ID	SCH_Name
1	Expert Opinion
2	Popular Opinion
3	Verbal Classific
5	inference
6	Conflict
7	Prefrence

The Scheme_TBL

ID	SCH_ID	Type	Content
1	1	P	Source E is an expert in the subject domain X containing proposition B.
2	1	P	E asserts that proposition B in domain X is true.
3	1	C	B may plausibly be taken to be true.
4	8	CC	Critical argumentation conclusion
5	1	CQ	Expertise Question: How credible is expert E as an expert source?
6	1	CQ	Field Question: Is E an expert in the field that the B is in?
7	1	CQ	Opinion Question: Does E's assertions imply B?
8	1	CQ	Trustworthiness Question: Is E reliable as source?
9	1	CQ	Consistency Question: Does B consistent with the assertions of other experts?
10	1	CQ	Backup Evidence Question: are there any evidences sustain B?
11	5	RA	RA-Node
12	6	CA	CA-Node
13	7	PA	PA-Node

The Scheme_Struct_TBL

Fig. 2. The main structure of different schemes

The Data_TBL table contains all users' transactions. This table gathers all users' analysis for different arguments. The table consists of : the *Stru_Id* that serves as foreign key for the Scheme_Struct _TBL's *ID*, and refers to a specific part of the scheme details, the *Content* attribute contains a portion of the analyzed text that fulfills the referred fixed scheme part, the *Type* attribute, which holds three values only, 1 for the supported node, 0 for rebuttal node, -1 for undetermined value that denotes neither supported nor rebuttal node. Since we consider any argument network as a kind of directed rooted tree, the *Child_Of* attribute points to the parent of each node, whereas the root node has no parents (0 refers to no parent). The *level* attribute refers to the level of each node in the tree, such that the value 0 indicates the root node. Finally, the *argumentation_no* attribute contains the number of the analyzed argument context.

For example, the following context below from Araucaria[2] repository database[2, 3, 12] is reanalyzed based on expert opinion scheme as shown in *Fig.*3 and *Fig.*4. *"Eight monthold Kyle Mutch's tragic death was not an accident and he suffered injuries consistent with a punch or a kick, a court heard yesterday. The baby, whose stepfather denies murder, was examined by pathologist Dr James Grieve shortly after his death. Dr. Grieve told the High Court at For far the youngest was covered in bruises and had suffered a crushed intestine as well as severe internal bleeding. When asked by Advocate Depute Mark Stewart, prosecuting, if the bruises could have been caused by an accident, he said "No. Not in a child that is not walking, not toddling and has not been in a motor car." Dr. Grieve said the injuries had happened "pretty quickly" and would be "difficult for an infant to cope with". The lecturer in forensic medicines at Aberdeen University told the jury that the bruises could have been caused by a single blow from a blunt instrument, like a closed hand. Death, not accident, court told, "Evening Telegraph", Monday, September 13, 2004, p.11"*

Regarding to the canonical representation for Waltons schemes presented in *Fig.*1, the given context could be analyzed as shown in *Fig.*3 based on expert opinion scheme [1]. Moreover, this analysis will be devolved through transaction records, as shown in *Fig.*4, to the structured data base (RADB) revealing the different parties of the analysis.

2.2 Framework Overview

Yun Chi et al. [13] surveyed the current algorithms used for mining frequent subtrees from databases. They focused on two main components of these algorithms, the candidate generation step and the support counting step. They revealed that there is no single best tree mining algorithm. Some algorithms offer a better time efficiency, while others require less memory. So every time we manipulate

[2] http:// araucaria.computing.dundee.ac.uk/

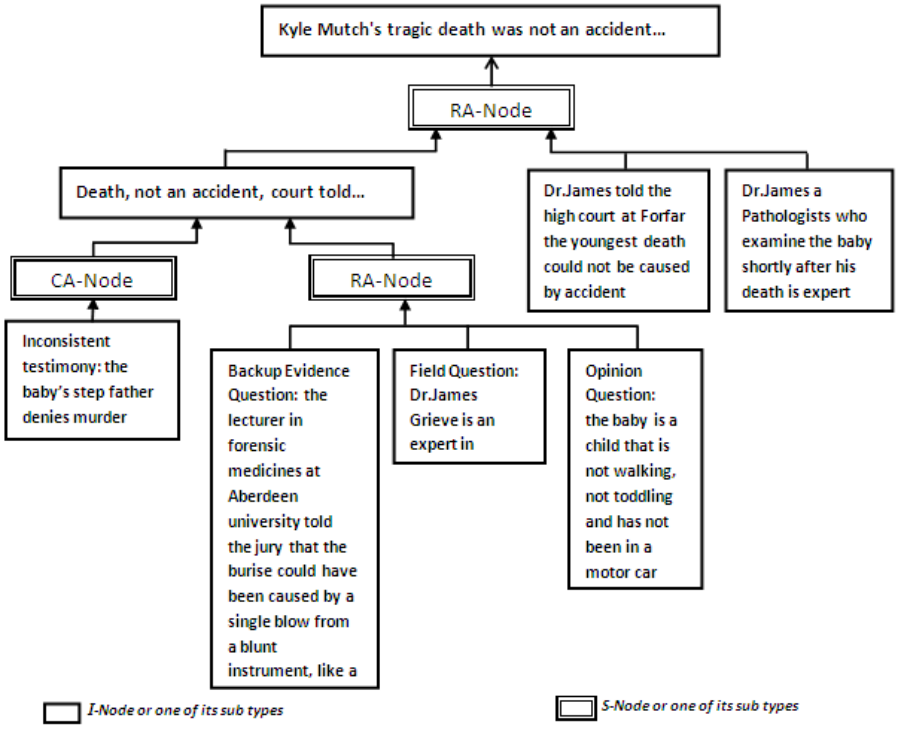


Fig. 3. The analysis diagram of the above context based on expert opinion scheme

ID	Stru_ID	Content	Type	Child_of	level	argumentation_no
1	3	Kyle Mutch's tragic death was not an accident and he suffered injuries	-1	0	0	argument_602
2	7	RA-Node	1	1	1	argument_602
3	2	Dr.James told the high court at Forfar the youngest death could not be cau	1	2	2	argument_602
4	1	Dr.James Grieveis a Pathologists who examne the baby shortly after his	1	2	2	argument_602
5	4	Death, not an accident, court told	1	2	2	argument_602
6	7	RA-Node	1	5	3	argument_602
7	6	Field Question: Dr.James Grieve is an exper in pathology	1	6	4	argument_602
8	7	Opinion Question: the baby is a child that is not walking,	1	6	4	argument_602
9	10	Backup Evidence Question: the lecturer in forensic medicines	1	6	4	argument_602
10	12	CA-Node	0	5	3	argument_602
11	9	conflict from inconsistent testimony: the baby's stepfather denies murder	0	10	4	argument_602

Fig. 4. The transaction records of the above analysis

the proposed RADB we will consider the time and memory consuming.

We draw a preliminary vision for retrieving and mining the RADB, using a framework with ITS component incorporated. The framework as depicted in $Fig.5$ consists of three main components: the *parser* module, the mining $classifier$ agent, and the ITS. The *parser* module receives a statement S from the intended users such as students or agents. the statement is divided by the parser into tokens, then the number of tokens is reduced. Finally the final crucial set of words { I_1, I_2,..., I_n } is sent to the $classifier$ agent. The tokens are reduced if they belong to a look up table containing the set of all unnecessary words like{a, an, the,...,etc }, otherwise it is added to the set of tokens to be sent to the $classifier$ agent. The importance of the *parser* module lies in reducing the set of tokens which in turn will reduce the number of iterations done by the classifier agent, and improve the complexity of the used mining algorithms.

The $classifier$ agent classifies the retrieved contexts depending on the students specification. The agent can classify the retrieved arguments by priority, polarity, scheme name, premises (with/against), and by conclusion. The priority aims to show the retrieved contexts organized by the maximum

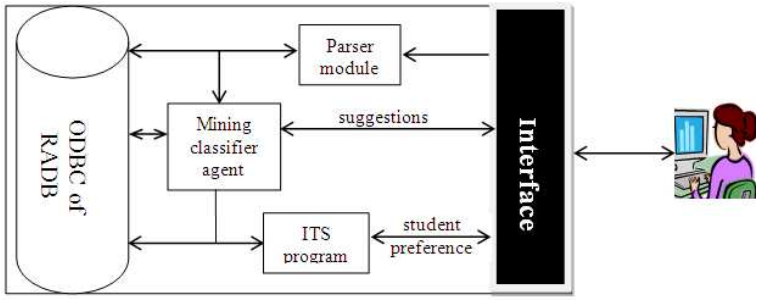


Fig. 5. Framework outline

support number based on the classification mining technique AprioriTid [10, 5]. Polarity classifies the retrieved arguments in to two classes, support class and against class, using the text mining techniques. Scheme name retrieves the desired contexts depending on a specific scheme name determined by the student. Premises (with/against) retrieves arguments by searching only in the different premises, and conclusion retrieves and classifies the arguments by searching only in the different conclusions. The classifier agent receives the set of crucial words { I_1,I_2, ..., I_n } from the parser module and the search type from the student, then retrieves and classifies the documents that are relevant to the student's search statement from the RADB using the multi-term text phrases approach[5] such that T={ T_1,T_2,...,T_m } is the collection of raw documents, I={ I_1,I_2,,I_n } is a set of words appearing in T. T'={ T'_1, T'_2,..., T'_m } is the set of documents, where T'_i contains a set of multi-term phrases I'={ I'_1, I'_2,..., I'_n }, $I'_i = I_j \bigcup I_{j+q} \bigcup .... \bigcup I_k$, where $1 \leq j \leq k \leq n$, $q \in$ [1,2,..,k-j], and I_i can appear in I'_i repeatedly. The importance of this classifier agent lays in manging the different mining techniques in order to: (i) direct the search towards hypotheses that are more relevant to the user's needs, (ii) add flexibility to the retrieving process to suit the users aims (iii) offer a myriad of arguments at users fingertips.

After the classifier agent exposed the pertinent contexts to the student, the student picks up one context among them. The student preference then delegates to the *ITS* program. The program exposes the corresponding context, and gives the student the ability to analyze the selected argument based on a specific chosen scheme. Finally the program negotiates with the student about the way of analysis through mining techniques to (i) provide constrains that guide the argument analysis process based on scheme structure and pre-existing arguments, (ii) refine the user's underlying classification, (iii) provide an analysis background to the different users, (iv) deepen the understanding of negotiation, decision making, (v) develop critical and intellectual thinking of students, and improve the analysis ability.

2.3 Illustrative Example

Suppose the student wants to know anything about Iraq war, so he/she come up with a statement "the destructive war in Iraq". First the parser module will divide the statement into tokens {the, destructive, war, in, Iraq}, such that I_1=the, I_2=destructive, I_3=war, I_4=in, I_5=Iraq, then access to the data base through the ODBC connection to compare each token with the lookup table entities and reduce the number of tokens, after checking the lookup table the tokens will be I_1=destructive, I_2=war, I_3=Iraq. So the output will be the item sets {destructive, war, Iraq}. Now the classifier should find the set of raw documents T= {T_1,T_2,...,T_n }. Assume the conclusion is the search criteria, so the classifier will use the mining AprioriTid algorithm [10, 5] to make all the possible combination of the item sets and classify the result depending on the support number for each combination. Firstly, the algorithm will calculate the support number for each single token, and select the tokens that have support number greater than *minsup*, that is a number specified by the student, however in our case we will take the *minsup*=1 so any token appears at least once will be considered. Since we assume that the student choose to search by conclusion then the support number for each token can be counted by the number of transactions resulted from the following select statements.

Select argument_no from Data_TBL where
Stru_id = 3 *and Content like '%destructive%';*

Select argument_no from Data_TBL where
Stru_id = 3 *and Content like '%war%';*
Select argument_no from Data_TBL where
struc_id = 3 *and Content like '%Iraq%';*

The out put of this step will be the set of ordered pairs L_1 = {(destructive, 5), (war, 10), (Iraq, 20)}, where the ordered pair is of the form (the token, the support number), and the set A_1={argument_801, argument_509,...} which contains the non repetitive arguments (argument_no) that contains these tokens. Secondly, the algorithm consequently builds the super set C_k = apriori_gen(L_{k-1}) for all possible combinations of the tokens. *Fig.*6. Shows the first iteration for C_2=apriori_gen(L_1).

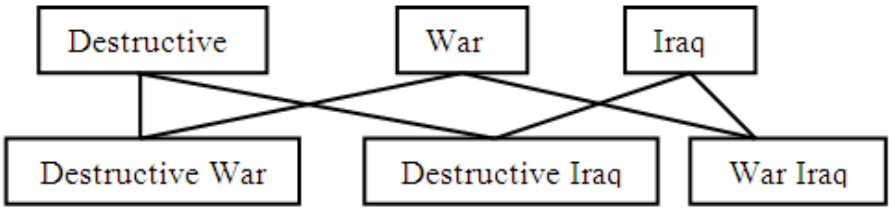


Fig. 6. The super set C_2 of the singleton token set L_1

Then the support number for each combination is checked through the set A_1. Suppose that the support number for the item set "War Iraq" is 0, which is less than the minsup=1, so this item set is neglected. The output of this iteration will be L_2= {(Destructive war, 3), (Destructive Iraq, 5)}, and the set A_2={argument_509,...}. Finally, the last iteration of our example will out put the set L_3={(Destructive war Iraq, 1)} and the set of arguments A_3={ argument_509}. Suppose that A_3 had more than one argument_no, we add function to the algorithm to check the counter argument of each argument_no and order the arguments depending on the possessed counter arguments, such that the argument that contains more cons is the weakest.

Therefore, the conclusions corresponding to the retrieved arguments are organized, such that the $argument_1$ is highly relevant to the issue of search rather than $argument_n$ as shown in *Fig.* 7. When the student pickup one of the classified output conclusions the ITS will access to the database to retrieve the corresponding context and then the context is exposed to the student giving him/her the ability to analyze. Furthermore, the ITS will negotiate with the student partially (step by step hints)

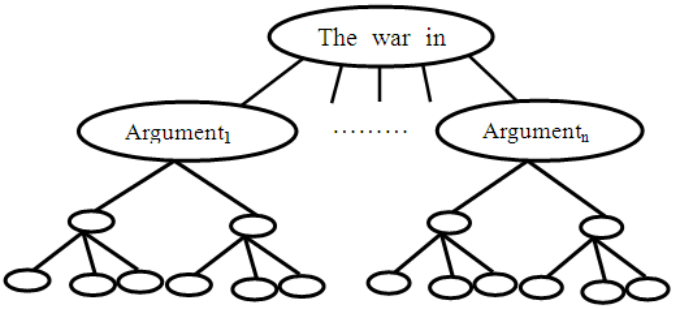


Fig. 7. The argument retrieval output

or totally (compare the student whole analysis with the original one retrieved from the repository) as discussed in the next section, in order to improve his/her analysis skill.

3 Motivation

In this paper, firstly we introduce a novel approach to retrieve the information using mining techniques based on RADB, which is a highly structured repertoire gathers the argument dataset, such that all needed information is encoded in an appropriate form. This structure facilitates fast interaction, and enjoys general applicability since it does not require a specialized knowledge. The idea is to mine the pre-existing arguments in order to (i) direct the search towards hypotheses that are more relevant to the users needs, even with more than one word in the search statement, (ii) add flexibility to the retrieving process to suit the users aims (iii) offer a myriad of arguments at users fingertips (iv) provide an analysis background to the different users. Secondly, we assemble the different retrieving techniques in a *Classifier agent* to be merged with an ITS. The agent based intelligent tutoring system aims to (i) provide constrains to guide the argument analysis process based on scheme structure and pre-existing arguments, (ii) refine the users' underlying classification, (iii) deepen the understanding of negotiation, decision making, develop critical and intellectual thinking of students, and improve the analysis ability.

I. Rahwan presents the ArgDf system [1, 6], through which users can create, manipulate, and query arguments using different argumentation schemes. Comparing ArgDf system to our approach, both of them sustain creating new arguments based on existing argument schemes. The available argument schemes are listed, enabling the user to choose the scheme to which the argument belongs. Details of the selected argumentation scheme are then retrieved from the repository, and the generic form of the argument is displayed to the user to guide the creation of the premises and conclusion. For example, querying the ArgDF repository to extract the name of the schemes can be done through the following RQL query:

$SelectScheme,\ PresumptiveInferenceScheme - hasSchemeName$
$FromScheme\ :\ kb : PresumptiveInferenceScheme\ kb : hasSchemeName$
$PresumptiveInferenceScheme - hasSchemeNameusingnamespace$
$rdf\ =\ http://www.w3.org/1999/02/22 - rdf - syntax - ns\#,$
$rdfs = http://www.w3.org/2000/01/rdf - schema\#,$
$kb = http://protege.stanford.edu/kb\#.$

whereas, in our approach, querying the RADB to extract the name of the schemes is done through the following SQL query:

$SELECT\ SCH\_Name,\ ID\ FROM\ [Scheme\_TBL]$

Consequently, to extract the details of the selected scheme, the following SQL query is performed:

$SELECT\ Content\ FROM\ [Scheme\_Struct\_TBL]\ WHERE$
$Id\_of\_sel\_scheme =\ [Scheme\_Struct\_TBL].SCH\_ID.$

In addition, the ArgDf system guides the user during the creation process based on the scheme structure only, the user relies on his efforts and his background to analyze the argument. However, in our approach, the user is not only guided by the scheme structure but also by crucial hints devolved through mining techniques. Accordingly, the creation process is restricted by comparing the contrasting reconstruction of the user's analysis and the pre-existing one. such restriction helps in refining the user's underlying classification.

In the ArgDf system, searching existing arguments is revealed by specifying text in the premises or the conclusion, as well as the type of relationship between them. Then the user can choose to filter arguments based on a specific scheme. Whereas, in our approach, searching the existing arguments is not only done by specifying text in the premises or the conclusion but also by providing different strategies based on different mining techniques (as explained in subsection 2.2). This method guarantees the retrieval of the most convenient hypotheses relevant to the subject of search.

4 Conclusions and Future Work

In this paper, we present a novel approach of building a highly structured argument repertoire (RADB) that uses different mining techniques to support argument analysis, retrieval, and re-usage. The paper also introduced an educational framework that utilizes the RABD. The proposed structure aims to: (i) summon and provide a myriad of arguments at the user's fingertips, (ii) retrieve the most relevant results to the subject of search, (iii) support the fast interaction between the different mining techniques and the existing arguments, and (iv) facilitate the interoperability among various agents/humans. Our attempt enjoys certain advantages when compared to others, especially with respect to the search of pre-existing arguments. The results obtained are very promising, where highly relevant and convenient arguments are obtained, especially when the search statement is in this form: "the destructive war in Iraq". Future work mainly concerns with the implementation of the rest of the framework components.

References

1. S. Modgil I. Rahawan C. Reed et.al. C. Chesnevar, J. McGinnis. Towards an argument interchange format. In *The Knowledge Engineering Review*, volume Vol. 00:0, pages 1–25. Cambridge University Press, 2007.
2. et.al. D. Walton, G. Rowe. Araucaria as a tool for diagramming arguments in teaching and studying philosophy. In *Teaching Philosophy*, volume Vol. 29, pages 111–124, 2006.
3. C. Reed G. Rowe and J. katzav. Araucaria: Making up argument. In *European Conference on Computing and Philosophy*, 2003.
4. M. Godden and D. Walton. Argument from expert opinion as legal evidence: Critical questions and admissibility criteria of expert testimony in the american legal system. In *Ratio Juris*, volume Vol 19, pages 261–286, 2006.
5. H.Ahonen-Myka. finding all maximal frequent sequences in text. In *Proceeding of ICML99 workshop*, 1999.
6. F. Zablith I. Rahawan and C. Reed. The foundation for a world wide argument web. In *Artificial Intelligence Conference (AAAI)*. published in the Artificial Intelligence Journal, April 04, 2007.
7. M. Baker J. Andriessen and D. Suthers. Arguing to learn confronting cognitions in computer-supported collaborative learning environments. In *Kluwer Academic Publishers, Dordrecht/Boston/London*, volume Vol.1, 2003.
8. C. Reed J. Katzav and G. Rowe. Argument research corpus. In *M.-P. Huget (ed.), Communication in Multiagent Systems,Lecture Notes in Computer Science,Springer Verlag, Berlin, Germany*, volume Vol.2650, pages pp. 269–283, 2003.
9. H. Prakken and G. Vreeswijk. Logical systems for defeasible argumentation. In *D. Gabbay and F. Guenther, editors, Handbook of Philosophical Logic*, pages 219–318. Kluwer, 2002.
10. R. Srikant R. Agrawal. Fast algorithms for mining rules. In *Proceeding of the 20th VLDB Conference Santiago, Chile*, 1994.
11. I. Rahawan and P.V. Sakeer. Representing and querying arguments on semantic web. In *Computational Models of Argument, P.E. Dunne and T.J.M. Bench-Capon (Eds.), IOS Press*, 2006.
12. C. Reed and G. Rowe. Araucaria: Software for argument analysis, diagramming and representation. In *International Journal on Artificial Intelligence Tools*, volume Vol.13, page pp.983, 2004.
13. R.R. Muntz. et.al. Y. Chi. Frequent subtree mining-an overview. In *Fundamenta Informaticae*, pages pp.1001–1038, 2001.