Knowledge Acquisition from Corresponding Domain Knowledge Transformations

Michael Armella[1], Isaí Michel Lombera[2], Stuart H. Rubin[3], Shu-Ching Chen[1], Gordon Lee[2]
[1]*Distributed Multimedia Information Systems Laboratory*
School of Computing and Information Sciences
Florida International University, Miami, FL 33199, USA
[2]*Dept. of Electrical & Computer Engineering*
San Diego State University
5500 Campanile Drive San Diego, CA 92182, USA
[3] *SPAWAR Systems Center (SSC)*
53560 Hull Street, San Diego, CA 92152-5001, USA
[1]*{marme003, chens}@cs.fiu.edu,* [2]*{glee, imichel}@mail.sdsu.edu,* [3]*stuart.rubin@navy.mil*

Abstract

The capability to efficiently retrieve knowledge in response to specific user queries offers the potential to create decision support systems of unprecedented utility, i.e., systems which can accelerate the learning process. This paper presents such an architecture, the Type 2 Knowledge Amplification by Structured Expert Randomization (T2K) system. This system differs from traditional expert systems in the way knowledge rules are matched with queries. The T2K has the ability to acquire knowledge from corresponding domains to answer queries from domains in which the system has less knowledge. This system also solves the word mismatch problem by modifying queries using word substitutions. This is done through creative transformations and optimizations of knowledge rule antecedents and consequents. By pairing rules with identical antecedents or consequents, we are able to induce new rules from existing knowledge without explicit elicitation from the user. The technique presented in this paper attempts to transform both the rules in the knowledge base as well as the query in order to find a matching action for a specified query.

Keywords: Expert System, Transformation, Rule Induction, Knowledge Acquisition

1. Introduction

Expert systems have been designed to deduce knowledge based upon inference. An expert system is usually made up of 3 parts: an inference engine, a knowledge base, and a user interface [5]. In this paper, we focus on the inference engine. The purpose of the inference engine is to interpret the rule base. The inference engine monitors the facts in the rule base and executes the action portion of those that have their condition satisfied [9]. There are two ways an inference engine can accomplish this, forward chaining or backward chaining. A forward chaining system begins with a piece of information and moves forward through the rules in the knowledge base until it reaches a final node or conclusion. A backward chaining system starts with a final node or conclusion and works back until it finds a starting state [4]. The problem with these methods is that they rely solely on comparing the input to the rules in the knowledge base. If the inference engine is unable to find an exact match in the knowledge base it has to give up. We solve this problem through the use of a novel architecture, the Type 2 KASER (Knowledge Amplification by Structural Expert Randomization).

The purpose of the inference engine in a forward chaining expert system is to determine in an efficient manner which rules are available for firing and then select the rules to be fired [12]. In a typical algorithm for forward chaining inference engine [5], values are read from an input and then compared to conditions in the rule base. The conditions are then evaluated and, if the conditions are satisfied, a rule is fired. The inference engine can be thought of as a finite state machine [5]. The inference engine will keep track of the number of conditions that have been satisfied for each rule. Only when a rule's entire set of conditions are meet is its action fired. This is similar to a finite state machine in that each of the rules contains a final state; each additional matching condition progresses the system toward the final state. This technique of forward and backward chaining works well when the queries are focused on a single domain and the knowledge base contains sufficient knowledge in this domain. However, if the queries may come from a number of domains, then the typical inference engine would be unable to match any queries to domains outside of those contained with the knowledge base.

The cycle of an expert system consists of two phases, a select phase and an execute phase [10]. The select phase is the process where the inference engine selects rules whose conditions have been met. The execute phase is the process by which the inference engine interprets the selected rule and draws inferences that alter the system's working memory [10]. Working memory contains information from rules that have been fired during the user's session. The consequents of fired rules may cause values to be inserted or deleted from the knowledge base for the remainder of the user's session.

The inference engine is the part of an expert system that contains the strategies for controlling the selection and application of rules in the knowledge base [11]. The inference engine must select rules to be fired and thus infer knowledge based solely on the facts that have been provided. The inference engine must contain strategies to deal with situations that can occur such as two rules being satisfied at the same time. But the situation of greater concern is what action an inference engine takes when no rules are satisfied. A simple solution that is commonly used is to elicit further information from the user but this does not make use of the knowledge that the system already contains. This is where experts systems fail to perform as well as a human expert. If a human expert does not have enough information to come to a definite conclusion then they will attempt to deduce possible conclusions based on the information/knowledge they currently have and past experience with similar situations. This kind of reasoning is what the T2K is attempting to simulate. This paper will focus on the ability of the T2K transform or create rules to satisfy cases which have not yet been seen but are similar to cases previously seen.

Many other techniques have been proposed for the creation of rules for inference engines. Techniques such as rule pruning [6] and dependency trees [7] are useful but still run into the problem of answering questions for domains for which only sparse knowledge is contained within the knowledge base. In [8], a technique is presented to solve the problem of word mismatch. People often use different words in their queries than authors use in their documents [8]. Xu and Croft show that by analyzing local and global contexts, words could be matched to alternative words with the same meaning, thus redefining the query. In this paper, we propose a technique to transform not only the query, but also the rules in order to more accurately reflect knowledge.

The T2K is an extension to the work done in the KASER system [1]. The original KASER system is based upon the theory of randomization, which takes a larger set of information and reduces it to its simplest form through compression. The KASER is capable of accelerated learning in symmetric domains, which are domains that can be represented in a more compact form. The KASER system is a third generation expert system that computes with words and employs qualitative fuzzy reasoning. The main breakthrough with the KASER system is that it does not suffer from the knowledge acquisition bottleneck; in the KASER, the cost of acquisition decreases as the scale of knowledge increases; whereas, the cost of acquisition increases as the scale of knowledge increases in standard expert systems [1-2].

The T2K continues the advancement of the KASER system by adding the capability to transform rules. The transformation of knowledge is based on the process of relating rules in corresponding domains. The capability to transform rules allows the T2K to take knowledge in a corresponding domain from the original query and apply it to the domain of the query. In this paper, we present the mechanisms used by the T2K, which allows it to transform both the query and rules to allow for proper matching. This technique eliminates the word mismatch problem as well as allowing for processing of queries for which knowledge in the knowledge base is sparse.

The remainder of this paper is organized as follows. In Section 2, we describe the capability of the Type 2 KASER to transform knowledge. Section 3 provides a description of the design of the Type 2 KASER while in Section 4, we discuss the graphical user interface that is used for the Type 2 KASER. Finally, in Section 5, we present some concluding remarks.

2. Type 2 KASER Transformations

The major contribution of the T2K is the use of transformations to induce corresponding rules. These transformations serve to dynamically create and normalize contexts and rule antecedents, which in turn facilitates the transformative induction of new knowledge.

The T2K uses transformations to induce corresponding rules from the rule base. The rule base consists of an array of rules of the form {i, j, k, …} $\rightarrow$ (u v w), where the antecedent consists of a non-empty, sorted set of distinct positive integers and the consequent consists of a non-empty sequence of positive integers – including the normalized insertion (INS) and erasure (ERA) commands and their arguments. All phrases entered into the system are hashed to integers to allow easier processing and storage of the rules. Upon the firing of a rule, the integers are reverse hashed to their original phrases to be displayed. Rules are kept in order from *most likely to be valid* at the top of the array to *least likely to be valid* at the bottom of the array.

The design of the system is such that the user provides the system with an initial context, which is of the same form as rule antecedents, and the system will attempt to match this context with a rule antecedent contained within the rule base. The initial matching of the context and antecedents does not differ greatly from that of other expert systems. The rule antecedent that contains the most matching phrases from the context is considered the most specific rule and the rule antecedent which contains the least matching phrase from the context is considered the least specific. The most-specific rules will be the first to be fired and within this stratum the most-possible rules are preferred.

There are two types of transformations being used within the T2K: creative transformations and optimization transformations. Creative transformations are created by pairing rule antecedents having a common consequent and optimizations are created by pairing rule consequents having common antecedents. Both transformations are performed such that the direction of the transformation is always towards the more likely to be valid.

2.1 Creative Transformation Rules

The procedure for finding and applying creative transformations is as follows: Let R_i and R_j be two distinct rules, where R_j is the more valid of the two; that is, R_j is the topmost rule between R_i and R_j. Let R_iA and R_jA denote the antecedents for R_i and R_j, respectively, such that $R_iA < > R_jA$. This must be true; otherwise the rules are not distinct. Let R_iC and R_jC denote the consequents for R_i and R_j, respectively, such that $R_iC = R_jC$. Then, we may induce the creative transformation $R_iA \rightarrow R_jA$. For example, R1: {1, 2} $\rightarrow$ (4 3 4) and R2: {1, 3, 5} $\rightarrow$ (4 3 4) induces the creative transformation rule, T1: {1, 2} $\rightarrow$ {1, 3, 5}.

Creative transformation rules must not be right recursive. The set on the left side of the transformation rule must not be embedded in the set on the right of the transformation rule, that is, $R_iA \not\subset R_jA$. For example, {1} $\rightarrow$ {1, 2}, or {2, 3} $\rightarrow$ {1, 2, 3} may not be applied because the set on the left is embedded in the set on the right.

2.2 Optimization Transformation Rules

The procedure for finding and applying optimization transformations is as follows. Let, R_i and R_j be two distinct rules, where R_j is the more valid of the two. Let R_iA and R_jA denote the antecedents for R_i and R_j, respectively, such that $R_iA = R_jA$; and let R_iC and R_jC denote the consequents for R_i and R_j, respectively, such that $R_iC < > R_jC$. Notice that this forms a non-deterministic rule pair as there exists an antecedent with two possible consequents. If a pair of rules exists that meet these conditions, then, we may induce an optimization rule, $R_iC \rightarrow R_jC$. For example, R1: {1, 2} $\rightarrow$ (4 3 4) and R2: {1, 2} $\rightarrow$ (3 4 5) induces the optimization rule, O1: (4 3 4) $\rightarrow$ (3 4 5).

Similar to creative transformations, the optimization transformations may not be right recursive either. Optimization transformations differ from creative transforms in that the left and right sides of the optimization transformation rule are not sets, but rather sequences. Thus, an optimization rule is right recursive if the sequence on left is embedded in the sequence on the right. For example, (1) $\rightarrow$ (1 2), or (2 3) $\rightarrow$ (1 2 3) may not be applied because the sequence on the left is embedded in the sequence on the right. However, the optimization rule, (1 3) $\rightarrow$ (1 2 3) may be applied because it is not a right recursive sequence.

2.3 Transformation Example

An illustration of the power of the T2K can be seen in the following example. Given the rules {airplane, explosives, terrorists} $\rightarrow$ (al-Qa´-ida used TNT to bring down a commercial airliner) and {airplane, bombs, terrorists} $\rightarrow$ (al-Qa´-ida used TNT to bring down a commercial airliner), we can see that the antecedents {airplane, explosives, terrorists} and {airplane, bombs, terrorists} both have the same consequent. Thus, we can induce the creative transformation rule {airplane, explosives, terrorists} $\leftrightarrow$ {airplane, bombs, terrorists}. Take note that although it appears we are interchanging explosives and bombs, one can not make this generalization for all cases. Only in the context of airplane and terrorists can we make the case that the phrases *explosives* and *bombs* may be interchanged.

Suppose that we were now given the rule {airplane, explosives, lighters, terrorists} $\rightarrow$ (Issue a Red Alert). By the creative transformation rule that was previously generated we can induce the new rule {airplane, bombs, lighters, terrorists} $\rightarrow$ (Issue a Red Alert). The advantage of this methodology is that it can be used to induce context-sensitive knowledge.

3. Type 2 KASER Design

The T2K differs from a traditional expert system in its unique ability to transform rules to create new rules based on the knowledge already contained within the knowledge base. The goal of the T2K system is to be able to simulate intelligence and be able to link related rules to each other.

The T2K has been developed as two separate modules, the T2K engine and the T2K graphical user interface. The T2K graphical user interface (GUI) provides all interaction between the user and the T2K engine. The T2K engine contains all rule processing algorithms.

All knowledge acquisition takes place through the T2K GUI. Rules entered by the user are inserted at the top of the rule base, thus making the last entered rule to be the most valid. This is done because recently entered information is more likely to be of use to the user than older information. Once the knowledge acquisition has been completed, the user may query the system. Similar to other expert systems, the user enters a query for which a response is required. The query is entered as a set of terms or phrases to be matched by the T2K; this set of terms is referred to as the context. The T2K provides an action to be taken for any given context that is entered into the system. The procedure for the T2K is summarized in Figure 1.

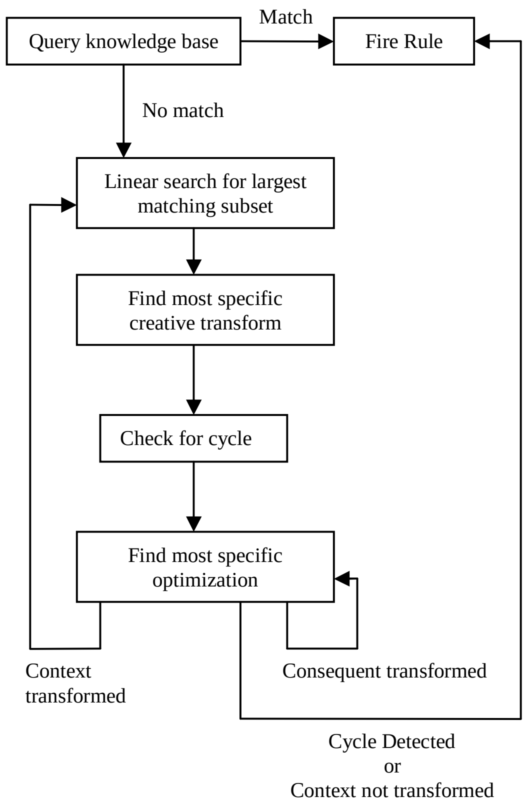

Query knowledge base
Match
Fire Rule
No match
Linear search for largest matching subset
Find most specific creative transform
Check for cycle
Find most specific optimization
Context transformed
Consequent transformed
Cycle Detected or Context not transformed

Figure 1. Flow Diagram of Type 2 KASER

If there is a direct match between the context and an antecedent in the rule base, then the consequent for the matching rule is returned and displayed as the action to be taken; this is similar to standard forward chaining. If there is no match, this is where the T2K differs from other expert systems; the T2K will attempt to induce new rules to match the given context. The T2K will first try to match the context or a subset of the context with an antecedent in the rule base.

There are $\binom{n}{r} = \frac{n!}{r!(n-r)!}$ combinations of antecedent subsets of length r in a context of length n. If all rules in the rule base have approximately the same validity (referred to as possibility), then each of these subsets, from longest, most-specific, to shortest, most-general, would be hashed to see if the context can fire a rule in the base. But this would not be efficiently done using hashing due to the fact that the rule base must be searched in sequential order for the first most valid and most specific rule; for this reason a linear search must be performed.

It can be argued that more-specific rules have less of a chance of contextual error. This offsets any reason to fire higher, more general and relatively more possible rules in lieu of lower, more specific, and less possible ones. Thus, it is the most-specific rule, rather than the most-valid rule that is chosen, amongst the equally most-specific candidate rules, the highest most-possible rule will be fired, i.e. the longest matching rule highest in the rule base. A top down linear search is performed on the rule base. As antecedents are found that match a subset of the context, the row number and length of the antecedent are stored. The first found highest and longest antecedent that matches the context is considered to be the most-valid, most-specific rule to be fired.

At this point, no transformations have taken place and the context is untransformed. It is necessary to minimize the number of general creative transformations of the context so as to minimize the introduction of combinatoric error. The following procedure is used to accomplish this: first the most-specific creative transform of the context is made. Rule antecedents are checked from the bottom to the top of the rule base to find the lowest most-specific subset of the latest version of the context which has the same consequent above it. The higher top-most antecedent, having the same consequent, replaces the previously matched subset in the context to form a creative transformation, as previously described. The top-down linear search is repeated if a creative transformation was made, with the newly transformed context. The new matching rule will overwrite the previous matching rule if the antecedent is more specific; that is, if the antecedent is of a greater length than the previous matching antecedent. This process will continue until no further creative transforms are possible, or a cycle is found in the transformed context.

Transformations can potentially fire in an infinite sequential cycle. In order to detect this problem, it is required that the context be saved in distinct temporary hash tables until it is found, if ever, that the contextual state has been previously saved in a sequence consisting of more than one contextual state. In other words, such repetition can be most conveniently detected by temporarily hashing the most-recent state vector until it is found, if ever, that it has been previously saved in a sequence consisting of more than one state.

Once a rule has been acquired, the rule consequents are checked from the bottom of the rules array to the top to find the lowest most specific sequence, if any, which is embedded by the latest version of the consequent to be optimized and having the same antecedent above it. This higher topmost consequent, having the same antecedent, replaces the previously matched embedded sequence in the consequent to form an optimization as previously described. The process is iterated until no embedding can be so replaced, or a cycle is detected, the last offending optimization skipped, and otherwise run to conclusion. The validity of a dynamic transformative knowledge space is thus maximized.

The possibility that a fired sequence of rules is correct is simple to compute as a function of each fired rule's relative validity.

Let m denote the number of rules in the knowledge base. Let n denote the number of distinct rules in the fired sequence. Let r_i denote the relative position or row number from the top for the ith fired rule in the sequence of length n, where, $1 \le r_i \le m$

Then,

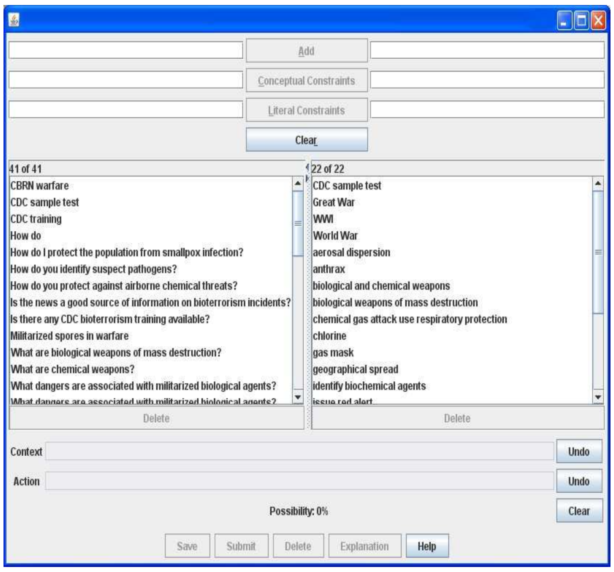
$$possibility = \left\lceil \min_{i=1}^{n} \left(\frac{m - r_i + 1}{m} \right) \right\rceil$$

expressed as a percent, where a result of 1.0 or 100 percent is to be displayed as 99 percent to better reflect the inherent potential for error. A result of 0 will be automatically displayed as 1 percent to better reflect the inherent potential for a correct chance result.

The T2K contains many of the same features and functionality of common expert systems. The process of transforming and creating new rules is where the T2K differentiates itself from other expert systems. The T2K will always attempt to return a result to a users query even despite a lack of knowledge in the domain of the query.

4. The Graphical User Interface

The T2K graphical user interface that has been used by the T2K is a modified version of the user interface that can be seen in [3]. As can be seen in Figure 2, the layout of the user interface remains the same, but the underlying mechanisms have been modified to better suit the needs of the T2K.



41 of 41

CBRN warfare
CDC sample test
CDC training
How do
How do I protect the population from smallpox infection?
How do you identify suspect pathogens?
How do you protect against airborne chemical threats?
Is the news a good source of information on bioterrorism incidents?
Is there any CDC bioterrorism training available?
Militarized spores in warfare
What are biological weapons of mass destruction?
What are chemical weapons?
What dangers are associated with militarized biological agents?
What dangers are associated with militarized biological agents?

22 of 22

CDC sample test
Great War
WWI
World War
aerosal dispersion
anthrax
biological and chemical weapons
biological weapons of mass destruction
chemical gas attack use respiratory protection
chlorine
gas mask
geographical spread
identify biochemical agents
issue red alert

Possibility: 0%

Figure 2. T2K graphical user interface

The design methodology of the T2K GUI serves the goal of being able to rapidly enter contexts, rule antecedents, and rule consequents for processing by the T2K engine [3]. The goal of T2K GUI is to provide a fluid man-machine interface between the T2K and the user.

There are two lists being used in the design of the T2K GUI. The list on the left-hand-side is used to display contextual keywords and phrases (CKP). The list on the right-hand-side is used to display action phrases (AP). The CKP list is used to select the antecedents and the AP list is used to select the consequents during rule construction [3]. As the user enters new entries into either list, the phrases are hashed to integers to be used by the T2K system. The T2K GUI will always display the full phrases to the user; but the T2K engine will use the integers to compress the representation of the phrases and rules.

To create rules, the user is able to enter multiple phrases from either list. A user may only create rules from phrases that have been entered into the GUI lists. This ensures that all phrases are properly hashed before they are used to create rules. A rule is created by providing a set of antecedents to be used as the context and a sequence of consequents to be used as the action. The unique feature of the T2K GUI is the ability to select semantic phrases based on natural language conceptual specifications. During a typical system run, there are a large number of phrases from which the user will select. The methodology of the T2K GUI addresses the problem of how to rapidly retrieve the desired semantic phrases in real-time for contextual specification [3].

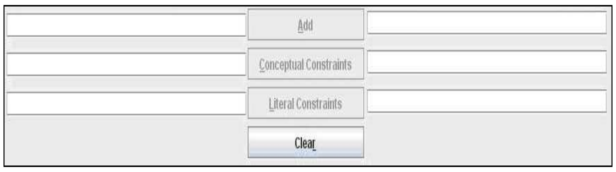


Figure 3. Entry bar

The rapid retrieval of semantic phrases is accomplished through the use of filtering. Conceptual constraints can be associated with phrases as they are entered into the system using the entry boxes, as seen in Figure 3. These constraints are used to filter phrases and provide a streamlined method to allow the user to search through phrases. An example of a conceptual constraint is “colors”; this would allow only phrases that have been tagged with the constraint “colors” to be displayed, such as “red”, “green” and “blue” [3]. The user may also enter literal constraints, which can also be seen in Figure 3. These literal constraints filter phrases so that only those that contain the constraint will be displayed for the user. Literal constraints are more specific constraints that limit the actual words within the phrases that must be present for the phrase to be displayed [3]. These two constraint mechanisms allow the user to effectively narrow the search space for selecting phrases to be used to create rules.

The user uses this GUI to populate the knowledge base with rules. A rule is entered by selecting a phrase from the CKP list and using the add button, seen in Figure 3, to add the phrase to the context. The same is done for selecting phrases from the AP list to be added to the action. Once the user has selected all the phrases to be used in the rule, the save button (seen in Figure 4) is used to add the rule to the knowledge base, with the context becoming the antecedent and the action becoming the consequent of the new rule.

Once the elicitation of knowledge is complete, the user can then proceed to submit queries to the system. In order to build a query, the user must use the CKP list to select phrases and add them to the context, as can be seen in Figure 4. The user then submits this context to the T2K to see if a matching action can be found.

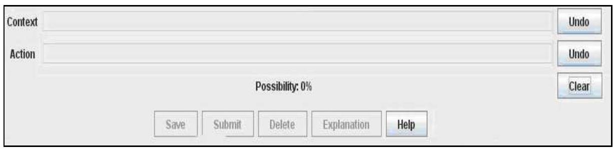
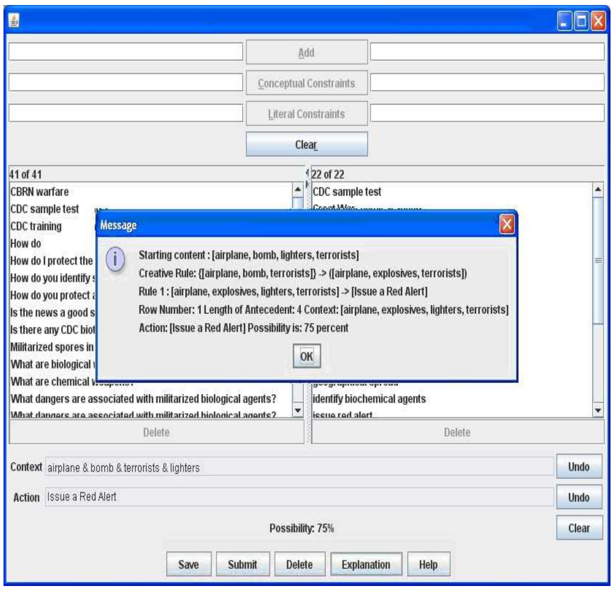


Figure 4. Query bar

Explanation functionalities will also be provided to the user, as can be seen in Figure 5, to allow the user to see the transformations that were performed in order to produce the fired rule. The explanation functionality provides the user with details about which transformations and rules where fired in order to find the action that is displayed.



Message

Starting content : [airplane, bomb, lighters, terrorists]
Creative Rule: ([airplane, bomb, terrorists]) -> ([airplane, explosives, terrorists])
Rule 1 : [airplane, explosives, lighters, terrorists] -> [Issue a Red Alert]
Row Number: 1 Length of Antecedent: 4 Context: [airplane, explosives, lighters, terrorists]
Action: [Issue a Red Alert] Possibility is: 75 percent

Context: airplane & bomb & terrorists & lighters
Action: Issue a Red Alert
Possibility: 75%

Figure 5. Explanation functionality

5. Conclusions

In this paper, we have presented a new system, the T2K, which brings together elements not found in current expert systems. The capability to transform knowledge allows the T2K to answer a large number of queries, while still minimizing the amount of knowledge that must be stored.

This system has the capability to perform not only in the explicit domain in which it was provided knowledge, but also in corresponding domains, which it may have sparse knowledge about. We have succeeded in creating an expert system that is able to induce new knowledge from related knowledge without the knowledge explicitly elicited from the user to the system.

Use of the T2K shows that an expert system can have intelligence to answer questions about subjects for which it has not been strictly informed. In this sense it can replicate the intelligence of an actual human expert, who would be able to answer questions from a relevant domain.

The main purpose of the T2K is not to develop a new technique for inferencing but a new technique for creating new knowledge from existing knowledge. The T2K does not have a significant advantage over expert system which will have queries focused in a single domain. The T2K excels greatest in applications where the query domain maybe from a wide range of domains.

6. Acknowledgments

This research was supported, in part, by DTRA contract number BA08MSB008. The statements, findings, conclusions, and recommendations are those of the author(s) and do not necessarily reflect the views of the sponsoring agency. This work was produced, in part, by a U.S. government employee as part of his official duties and no copyright subsists therein. It is approved for public release with an unlimited distribution.

7. References

[1] S.H. Rubin, S.N.J. Murthy, M.H. Smith, and L. Trajković, "KASER: Knowledge Amplification by Structured Expert Randomization," *IEEE Transactions on Systems, Man, and Cybernetics Part B*, vol.34, no.6, 2004, pp. 2317-2329.

[2] S.H. Rubin, R.J. Rush Jr., J. Murthy, M.H. Smith "KASER: A Qualitatively Fuzzy Objectoriented Inference Engine," *Fuzzy Information Processing Society, 2002. Proc. of the NAFIPS 2002 Annual Meeting of the North American*, New Orleans, Louisiana, USA, Jun 27-29, 2002, pp 354-359.

[3] I.M. Lombera, J. Patel, S. Rubin, S.C. Chen, G. Lee, "A Graphical-User Interface In Support of a Cognitive Inference Architecture," *Proc. of the ISCA 21st International Conference on Computer Applications in Industry and Engineering (CAINE 2008)*, Honolulu, Hawaii, USA, November 12-14, 2008, pp. 274-279.

[4] A. Kandel, *Fuzzy Expert Systems*, CRC Press, 1991.

[5] N.L. Griffin, F.D. Lewis, "A rule-based inference engine which is optimal and VLSI implementable," *Proc. of the Tools for Artificial Intelligence, 1989. Architectures, Languages and Algorithms, IEEE International Workshop on*, Fairfax, Virginia, USA Oct 23-25, 1989, pp. 246-251.

[6] W.W. Cohen, "Fast effective rule induction," *Proc. of the Twelfth International Conference on Machine Learning*, Tahoe City, California, USA, July 9-12, 1995, pp. 115-123.

[7] D. Lin, P. Pantel, "Discovery of inference rules for question-answering," *Natural Language Engineering*, vol. 7, no.3, Cambridge University Press, 2001, pp. 343-360.

[8] J. Xu, W.B. Croft, "Query expansion using local and global document analysis," *Proc. of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, Zurich, Switzerland, August 18 – 22, 1996, pp. 4-11.

[9] M. Togai, H. Watanabe, "Expert System on a Chip: An Engine for Real-Time Approximate Reasoning," *IEEE Expert*, vol. 1, no. 3, 1986, pp. 55-62.

[10] F. Hayes-Roth, "Rule-based Systems," *Communications of the ACM*, vol. 28, no. 9, 1985, pp. 921-932.

[11] S. J. Biondo, *Fundamentals of Expert Systems Technology*, Intellect Books, 1990.

[12] K. Fordyce, G. Sullivan, "Boolean Array Structures for a Rule-Based Forward Chaining Inference Engine," *Proc. of the International Conference on APL: APL in transition*, Dallas, Texas, USA, May 10 - 14, 1987, pp. 185-195.