A HYBRID ANALOGICAL LEARNING SYSTEM AND ITS APPLICATION IN EMPLOYMENT ACCIDENTS DOMAIN

Mohammed El-Sedeek[1], Khaled Shaalan[2], Said Mabrouk[3], Ahmed Rafea[4]

Abstract

Analogy is one of the central inference methods in human cognition. Several analogy methods have been developed, they were different in the technique used to establish the analogy, estimate the similarities, and transform the concepts and knowledge from the source situation to the target situation. All of these methods were used separately in different learning approaches. In this paper, we propose a learning system based on hybrid model of analogy, learning is done using different analogy strategies, according to the learning task. Using one model of analogy in a learning system will constrain the learning task, while the possibility of using multi-methods of analogy in one model will contain two main features. First, it will integrate different models of analogy in a hybrid learning system. Second, it will help in selecting and applying the suitable analogy method according to the given learning task and the given knowledge. The developed learning system consists of four main modules: Knowledge Base, User Interface, Retriever and Learning Modules. A set of real cases from the domain of employment accidents is used to demonstrate the learning capability of the system.

Key Words: Machine Learning, Learning by Analogy, Case-Based Learning.

يعتبر التشابه أحد الطرق الأساسية لإستنتاج المعرفة الإنسانية. وقد تم تطوير العديد من طرق التشابه والتى تعمد على تقنيات مختلفة لقياس هذا التشابه ونقل المعرفة والمفاهيم من حالات المصدر السابق دراستها إلى الحالات المستهدفة المراد إيجاد حل لها. كل هذه الطرق والأساليب كانت تستخدم بصورة منفصلة عن بعضها البعض فى منهجيات مختلفة لإنشاء طرق للتعلم. وفى هذا البحث نقترح نظام تعلم مبنى على نموذج مهجن من الطرق المختلفة للتعلم بالتشابه وذلك طبقاً لمهمة التعلم المستهدفة. فإستخدام طريقة واحدة للتعلم بالتشابه فى نظام تعلم سوف يحد من عملية التعلم ، بينما إمكانية إستخدام نظام تعلم متعدد الطرق للتشابه سوف يكون له مميزات عديدة ، أهمها أنه سوف ينشىء نظام متكامل من طرق التعلم بالتشابه ، كما أنه سوف يساعد على إختيار طريقة التعلم بالتشابه المناسبة لحل المشكلة المستهدفة وذلك حسب مهمة التعلم المطلوبة وكذلك المعلومات والخصائص المعطاة والمتوفرة فى قاعدة المعلومات نظام التعلم المطور يتكون من أربع مكونات رئيسية وهى: قاعدة المعرفة ، واجهة المستخدم ، مسترجع الحالات المشابهة ، وحدة التعلم. والنظام تم تطبيقه على حالات واقعية فى مجال إصابات العمل لتوضيح إمكانية التطبيق العملى لنظام التعلم المطور.

1. ISSR, Cairo University, M.Sc.of Computer Science, sedeek@hotmail.com
2. Computer Science Dept., Faculty of Computers and Information, Cairo Univ., shaalan@mail.claes.sci.eg
3. Central Lab for Agriculture Expert Systems (CLAES), Ministry of Agriculture, said51@hotmail.com
4. Computer Science Dept., American University in Cairo, rafea@aucegypt.edu

1. Introduction and Background

From the beginning of AI, researchers have sought to understand the process of learning and to create computer programs that are able to construct new knowledge or to improve already possessed knowledge by using input information. Machine learning [1], is a branch of AI concerned with the study and computer modeling of learning processes. Some things we might call "learning" could also be called "problem solving" or "reasoning". Different learning strategies have been issued: rote learning, learning from instruction, learning from examples, learning from observation, and learning by analogy.

Learning by analogy [2] is one of the central inference methods in human cognition, in which, we transfer knowledge applicable in one domain to perform a similar task in another domain or to extract knowledge from past successful problem-solving situations that bear a strong similarity to the current situation. The ability to learning by analogy is particularly important, because it permits the extension of knowledge of a target domain by virtue of its similarity to a base domain via a process of analogical learning. The general procedure for analogical learning involves coping structure from the base to the target in which missing information is generated, and substitutions are made for items for which analogical correspondences have already been found. The elaboration of analogies is achieved by using a metric measuring the distance between conceptual descriptions of objects. Many researchers have done some pioneering work in analogical learning. The most notable ones are briefly described below.

Carbonell [3] proposed a computational model of learning by analogy based on an extension of means-ends-analysis (MEA). He presented an analogical inference engine and provided a framework for automated example generation that enables one to apply learning-from-examples techniques in order to acquire generalized plans. Thus the system was capable of integrating skill refinement and plan-acquisition processes. Russel [4], considered the conditions under which propositions inferred by analogy are true or sound. They concerned with normative criteria for analogical transfer rather than a descriptive or heuristic theory. Their goal was to provide a reliable, programmable strategy that will enable a system to draw conclusions by analogy only when it should. They proposed a method for generating correct generalizations and analogical inferences, given correct determination rules. Winston [5] investigated analogy as a powerful mechanism for classifying and structuring episodic descriptions. He devoted his work to study the learning by analogy via a computational model. He deals mainly with representing story plots (such as *Romeo and Julie*, and *Macbeth*) and the determination of some measure of plot similarity. These stories are represented via a network of frames, i.e. a graph, and using a constrained matcher, a value representing the number of similar components is determined. Vrain and Kodratoff [6], describe how to apply the analogical process to incremental similarity-based learning. They illustrated the analogy

and showed how relationships between background knowledge enable to use analogy in the domain of concepts formation from a set of examples, and showed the importance of dissimilarities in the process of analogy. They, finally, raised a set of problems about this approach: how to get the most useful analogies, how to improve the generalization to cover the new example, and how to use an analogical process to do incremental learning.

However, poor analogies do not support learning - or worse, they cause "negative learning" by adding incorrect or unsound information to the target domain. Markman [7] and others point out that analogy is too profligate an inference mechanism, and constraints on inferences are necessary. Markman also notes that the one-to-one mapping constraint also acts to constrain the inference set, when n-to-m mappings might be generated. Because analogies use domains that are rarely fully isomorphic [8] computational models can easily over-generate inferences.

A considerable body of recent research has shown that similarity comparisons can involve a process of structural alignment [9]. This view characterizes knowledge as structured hierarchies encoding objects, object attributes, relations between objects and relations between relations. Given these representations it is assumed that similarity comparisons involve the alignment of relational structure to find the most structurally consistent match between two systems of concepts, that satisfies the constraints of parallel connectivity; if two relations match, their arguments must match and one-to-one mapping; that each item in one structure may only be mapped to one other item [10, 11]. Indeed, structural alignment has been mooted as a unified account of a diverse range of phenomena including similarity, analogy, metaphor and concept combination [12, 13, 14].

Several analogy methods have been developed: transformational analogy, derivational analogy, proportional analogy, determination-based analogy, computational analogy, analogy by rendition, and predictive analogy. These methods of analogy process are different in the technique used to establish the analogy, estimate the similarities, and transform the concepts and knowledge from the source situation to the target situation. All of these methods are used separately in different approaches; using one model of analogy in a learning system will constrain the learning task. In our work, we propose a learning system based on hybrid model of analogy, in which we integrate different methods of analogy in one model. The possibility of using multi-methods of analogy in one model will contain two main features. First, it will integrate different models of analogy in a hybrid learning system. Second, it will help in selecting and applying the suitable analogy method according to the given learning task.

Our domain of application is the employment accidents. The identification of "employment accident" is a challenge. When encountering a new case of employment accident, perhaps we cannot find a direct law to be applied. Then

the expert would compare it with previous situations and find an analogical situation to use it as a guide for the solution of the current case. Since these different previous cases may have different structures and knowledge representations, thus we need to deal with different methods of analogy according to the given knowledge, i.e. we need a hybrid model of analogy.

This Paper is organized as follows: In Section 2 we review some related work on different analogy methods. In Section 3, we describe the hybrid analogy methods of the proposed learning system. Section 4 concludes this paper and gives some final remarks. An appendix presents two experimental examples as a case study is included.

2. Related Work

Analogical learning and case-based learning are sometimes used as synonyms. Case-based learning can be considered a form of single domain analogy, while, analogical learning solves new problems based on past cases from different domains of analogy [15]. Indeed, the analogical learning technique takes different forms of learning methods. Our survey on analogy reveals different models and usage of analogy. Literature that sheds the light on the different methods of analogy process is briefly discussed below.

Solving a problem using a *transformational analogy* [16] is just copying a solution for an old very similar problem to be a solution for our considered problem with some modifications. The concept of the transformational analogy is as: Given a problem P_{new}, target problem, the system is first reminded of a familiar problem P_{old}, source problem, with a solution S_{old}, where P_{old} and P_{new} are so similar that S_{old} is an approximate solution for P_{new}. Then, the system modifies selected components in S_{old} to obtain a candidate solution S_{new} for P_{new}. Thus, in transformational analogy the entire solution S_{old} is transferred to S_{new} and modified to fit the specifications of P_{new}. In transformational analogy, the modified knowledge typically is associative, and often is based on domain specific heuristics [2]. Transformational analogy does not look at how the problem was solved; it only looks at the final solution. It looks for a similar solution and copies it to the new situation making suitable substitutions where appropriate, e.g. it generates proofs in geometry and simple number theory from proofs of related theorems.

Derivational analogy [17] represents a problem solving plan as a hierarchical goal structure, showing how and why each goal was decomposed into sub-goals i.e. a history of the problem solution. A derivation history of a problem is a tree showing the top-down decomposition of a goal into sub-goals terminating in a subroutine. Solving a new problem is by replacing this plan top-down. When the sub-plan for a sub-goal fails, the plan is patched by solving that sub-goal from scratch. The derivational analogy process constructed as: Given the target problem P_{target}, the system will decompose it to some elementary sub-problems

$p_1,p_2,\ldots,p_n$, solving these sub-problems by analogy to get their solutions $s_1,s_2,\ldots,s_n$, then the system recompose these solutions to find the target problem's solution, S_{target}.

The process of derivational analogy consists of:

- Storing derivation histories of solving problems, sub-problems and their justifications in the source problem plan in an episodic memory.
- Retrieving an appropriate case from the memory that shares "significant" aspects with a current problem.
- Transferring the derivation of the retrieved solution to the new problem by replaying relevant parts of the solution and modifying the inapplicable portions in light of the new context of the target problem.

One of the meanings of the Greek word "analogy", from which analogy originates, is proportion [18]. A proportion problem often has the form "A is to B as C is to What?", the familiar geometric-analogy intelligence-test problem is an example of a proportion problem with grouped two dimensional objects for A, B, and C. The *proportional analogy* [18] has a procedure concerned with the relations which having the form "**A** *is to* **B** *as* **C** *is to* **D**". The process is that, given three terms of proportional analogy, the system will generate the fourth term. Associating a description with every object called "natural description", then to know an object is to know it from some perspective under some description. Therefore, syntactic proportional analogy relation can be comprehended from the natural descriptions of its objects and on the other hand, if a re-description of at least one object is necessary, then the analogy relation is interpretive. Thus, what is a syntactic proportional analogy relation for one person could be an interpretive proportional analogy relation for another.

The *Determination-Based analogy* [19] is a type of analogy based on knowledge in the form of determination rules between two relations. Its procedure is concerned with a determination rule between two relations R_1 and R_2 of the form "$R_1(x, y)$ *determines* $R_2(x, z)$". If two entities X_1 and X_2 are characterized by the same entity Y_0, then it is very likely that they are also characterized by the same entity Z_0. We say that R_1 functionally determines the value of R_2, because the value assignment for R_1 is associated with a unique value assignment for R_2 i.e." $\forall x, y\ R1(x) = R_1(y) \Rightarrow R_2(x) = R_2(y)$". Determination rules may be useful in analogy for two reasons: First, in many domains a strong theory may not be available, where as determination rules can be provided, and the system can gain expertise through the acquisition of examples from which it can reason by analogy. Second, even when a strong theory is available, its complete education may be difficult, and it may be easier to elicit knowledge by extracting determination rules.

Another analogy method is the *Computational Analogy*. Here the method for modeling analogy [20] consists of:

- *Matching procedure*: it matches the source problem and the target problem, obtaining pairwise correspondences between the two problems, in order to determine the analogy.
- *Mapping procedure*: the analogy, the pairwise correspondences, is used to map relationships known to hold in the source domain to the target domain.

Then the system maps the solution of the source problem to get the target solution An important issue is the form of knowledge representation to be used by the two algorithms, which provides matching and mapping.

In *Rendition Analogy Method* [18], we do not need to find a direct solution for a problem, but we need to make the strange familiar and the familiar strange. Here, the analogy procedure is "Seeing the target problem as the source problem". In the *analogy by rendition*, the similarities between the target and the source do not exist prior to viewing one as another; the analogy process creates the similarity, e.g. seeing the paintbrush as if it were a pump, viewing the paintbrush as a pump created the similarities that were not there before. Whenever we notice that two objects are similar, we can do so only with respect to their existing ontology and descriptions. However, in analogy by rendition a new ontology and a new level of description are created for the target problem. The major research problem facing this mode of analogy lies in explaining how the new perspective is created. When we perceive any situation, we do so in terms of a conceptual system. In viewing one site as another, we forcibly interpret the conceptual system associated with the source situation in the context of the environment of the target situation. The target situation is re-described by using the terminology of the source system of symbols. This mode of analogy is closely related to models as in, scale model of a ship, wind tunnel, Bohr's model of atom.

Many frameworks for analogy research have arisen, and are typically multi-phase models operating primarily in a sequential manner. Many of these frameworks refer to an *evaluation* phase, but supply little detail on its operation. These frameworks are notable by their lack of an explicit verification activity, which operates on the candidate inferences mandated by an analogy. None propose a validation activity based on the soundness of candidate inferences. For example, Kokinov [21] identifies phases of *retrieval, mapping, transfer, evaluation* and *learning*; Forbus, Gentner, Markman *et al* [22] decompose analogy into *retrieval, mapping* (alignment and projecting inference) and *abstraction*. Hummel and Holyoak's [10] Lisa model encompasses phases of *access, mapping* and *induction*. However, throughout this paper we use the framework:

Representation → Retrieval → Matching → Refinement → Learning

In this section, different methods of learning by analogy are discussed. For each method, a definition, and an evaluation of the method are presented. To use an intelligent analogical-based learning system in real application, such as the employment accident domain, the system must contain several analogical

methods, in a hybrid manner. To achieve this goal and avoid the weakness of a single analogy method in some situations, we proposed a learning system based on a hybrid model of analogy.

3. The Proposed Hybrid Analogy Methods Learning System

To reuse past experiences, we must both recognize the salient features of the past as well as build a mapping of how that experience may be used in the present situation. In the previous section, we discussed different models of analogy; each one of them has its own strategy to map the experience from the old situations to the new one. Using one model of analogy in a learning system will constrain the learning task, while the usage of hybrid analogy model will have two main features. First, it must integrate different models of analogy. Second, it selects and applies the suitable analogy method according to the given learning task. The major objective of this work is to develop a learning system from analogical examples using a hybrid model of analogy, which will integrate automatic case retrieval and storage, case replay, and modification as required. Learning is done by using different models of analogy strategies based upon the task of learning. The proposed learning model consists of four main modules, as shown in Figure 1. They are: Knowledge Base, User Interface, Case Retriever, and Learning Module.

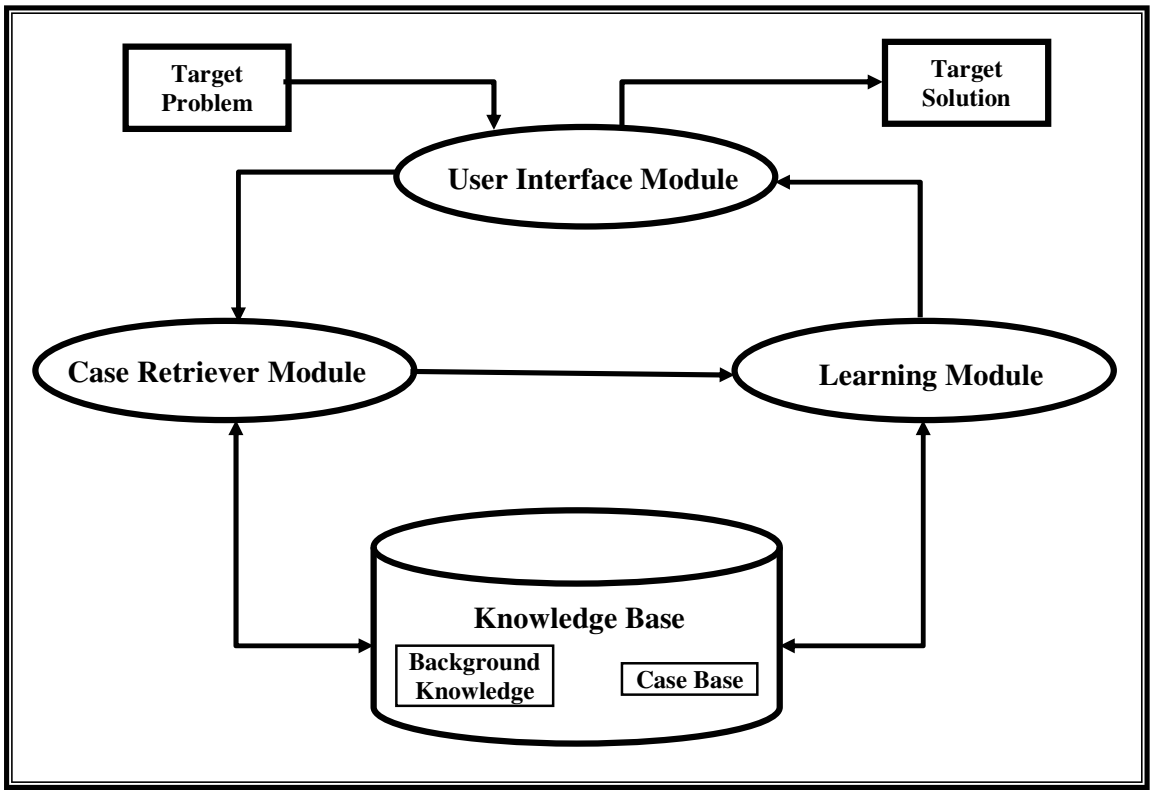

Figure 1: System Architecture

In this section, we will explain the structure and the function of each of these modules. In the description of the system components that follows, we use examples from the employment accidents domain.

3.1 Knowledge Base Module

The knowledge base consists of two main parts: the case base and the background knowledge. The case base contains the previous solved problems and their solutions. The background knowledge contains the required knowledge to find the solution of new problems, such as determination rules, hierarchy relations and if-then relations.

Case Base:

It contains three different structures: category, case, and generalized case. The category is a collection of cases; all of them have common features, which constitute the category, while the generalized case is a set of cases of the same category having some additional common features. Each of the three structures is represented using frames. A frame is a collection of slots (attributes), associated values, and possibly constraints on values; this collection describes some entity. Frames are connected to each other by virtue of the fact that the value of a slot of one frame may be another frame. Three different types of frames may be used to represent the case base structures. These frames are category-frame, case-frame, and generalized-case-frame. Each slot in a frame may be associated with three attributes: value, facet and confidence factor. Value is the abstract value of the slot. Facet is a method, pointer to another frame, or procedure to find the derived value of the slot. Confidence factor is an importance rank, to evaluate the confidence of this slot with respect to the frame entity.

The category structure is represented by the category-frame. Figure 2 illustrates an example of a frame of the category *injury at work* taken from the domain of *employment accidents*.

FRAME : injury-at-work			
slot	value	facet	c-factor
accident-place	work-place(W)	LWP-hierarchy	1.0
accident-time	work-time(T)	LWT-hierarchy	1.0
accident-property	accident-condition(C)	WAC-hierarchy	0.9

Figure 2: Category Frame

In this figure the name of the category frame is injury-at-work. It has three slots: accident-place, accident-time, and accident-property. Each of these slots is associated with the previously discussed three attributes: value, facet, and confidence factor. The value of the accident-place slot will be derived using the work-place(W) procedure. This procedure in turn will use the hierarchy of the legal-work-place, shown in Figure 3, to verify that a given place belongs to, or not, the hierarchy. Similarly, the two other slots, accident-time and accident-property, will use the two procedures, legal-work-time and work-accident-condition in a similar manner.

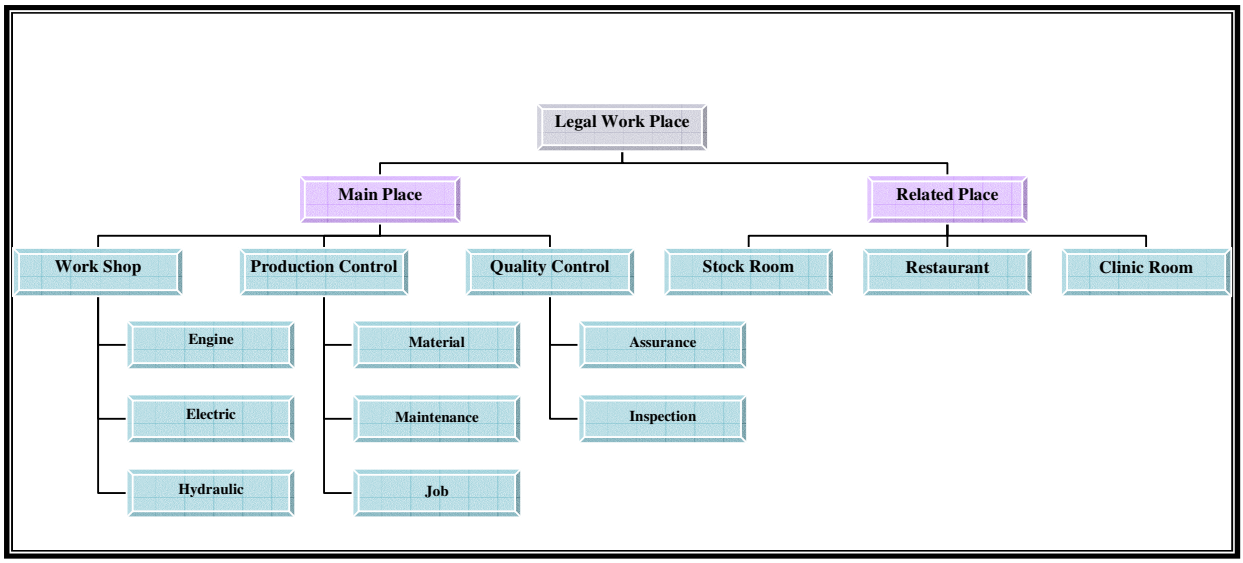

Figure 3: Legal work place hierarchy

The case structure will be represented as a frame. Figure 4 illustrates a typical case frame that belongs to the category of *injury-at-work*, which was shown above. When more than one case belonging to the same category are sharing additional common features, then these cases can be generalized to a general case, taking into consideration that these common features may be derived using hierarchy structures of these features. The common features will be factored out from the frames of the cases into the generalized case. These frames are set to inherit from the generalized case frame. In this sense, the generalized cases can be viewed as subcategories. Their existence will improve the efficiency of both the storage and the retrieval of cases.

FRAME : C05			
slot	a-value	facet	c-factor
a-kind-of	injury-at-work		
accident-place	machine-room	LWP-hierarchy	1.0
accident-time	over-time	LWT-hierarchy	1.0
accident-property	suddenly	WWP-hierarchy	0.9
body-injury	complete-amputation		0.8
organ-injury	right-arm		0.7
inability	partial		0.6

Figure 4: Case Frame

Background Knowledge:

It contains three types of knowledge: hierarchy relations, importance ranks table and equations and rules. The similarity relations among the different features of the cases are represented as hierarchical relations, which control the specialization and the generalization of these features and their values. As an example, the features that represent the related work times are represented in a hierarchical relation, as shown in Figure 5.

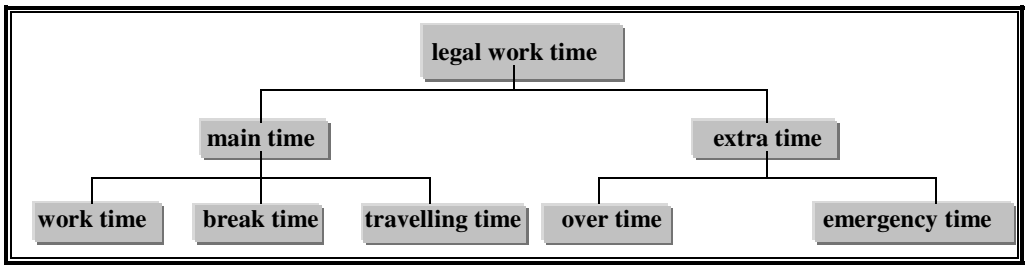

Figure 5: Hierarchy Relation

The importance rank table contains the confidence factors of all features used in the domain as they extracted from the expert. The contents of this table may be modified or grown during the system interactions. As an example, the table shown in Figure 6 represents the importance ranks of some accident features.

Feature	Importance Rank
accident place	1.0
accident time	1.0
accident property	0.9
body injury	0.8
organ injury	0.7
inability ratio	0.8

Figure 6: Importance Ranks Table

The equations and rules are extracted from the domain, which could be used or applied during the learning process. These rules are in the form of proportional rules, determination rules, and if-then rules. The following are examples for such rules:

- Proportional rule:
  polluted acupuncture **is to** *nurse* **as** *poison gas inhalation* **is to** *miner*
- Determination rule:
  inability ratio **determine** *injury compensation*
- If-Then rule:
  if *inability ratio* > 80% **then** *injury pension*

3.2 User Interface Module

The user interface module facilitates communication between the system and the user. Therefore, its main function is to exchange information between the user and the system modules in three manners. First, by accepting the target case and the user request. Second, for browsing the target solution and the learning result for the user. Third, by requesting some decisions from some expert user during the case classification and retrieval processes, when the system has no knowledge to take these decisions.

The user may be a normal user or an expert. The expert user can interact with the system to modify or create a new knowledge in the case base or the background knowledge in various situations that require a decision or in case of failure states. The user interface accepts the target case from the user as a feature list. The output will be a solution of the target problem, an answer of the user request, or a learning knowledge for the user.

3.3 Case Retriever Module

The case retriever module is essentially responsible for the process of retrieving suitable source case from the case base, which will be analogical to the target case. As a partial result, there will be classification process of the target case to some category class in the case base. The case retriever module, as shown in Figure 7, consists of four sub-modules: Feature Classifier, Category Retriever, Matched Cases Extractor, and Most Analogical Case Selector.

Each step of the case retriever algorithm performs a basic task of the retrieving process. First, the feature classifier procedure classifies the features of the target case to extract its category features. Second, the category retrieval procedure uses the target case category features to retrieve its category class from the stored categories in the KB. Third, the matched case extractor procedure searches for the analogical source cases, which belongs to the target case category. Finally, the most analogical case selector applies a metric procedure on each case of the set of analog source cases to select the most analogical one.

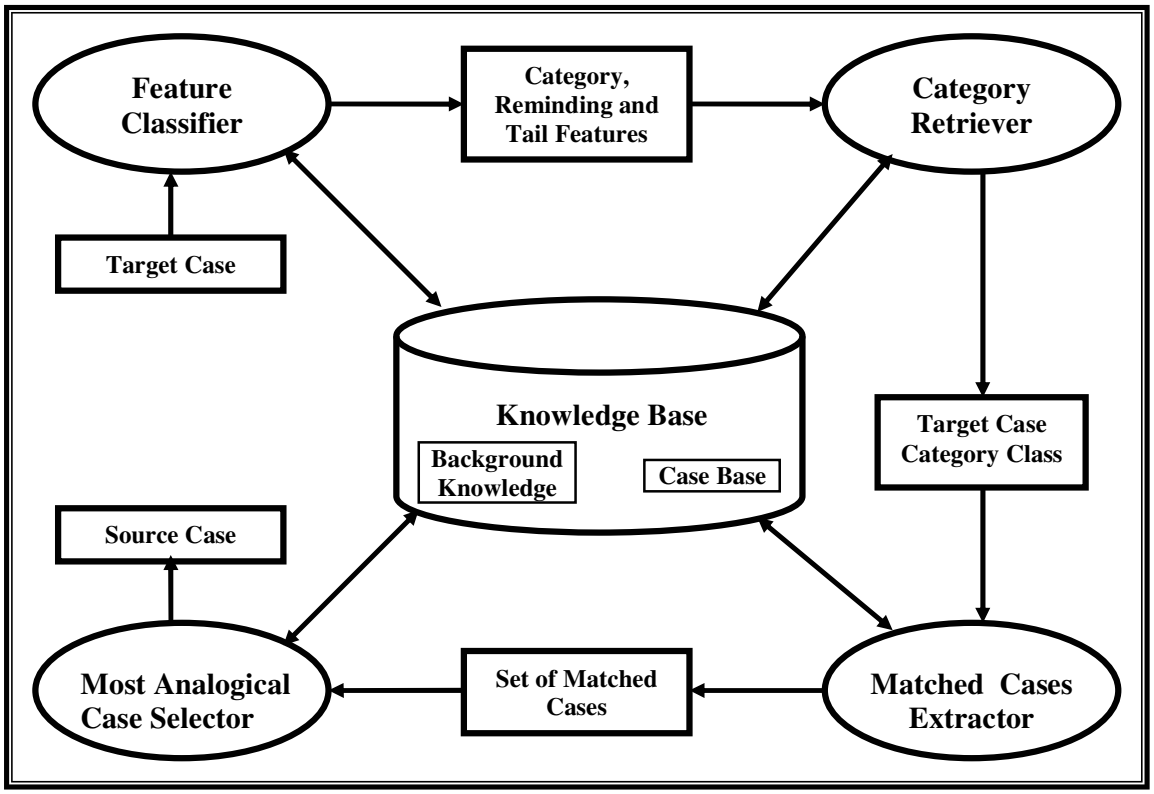

Figure 7: Case Retriever Module

Feature Classifier:

The target case, received from the user interface module, is a set of different features, which describes the case. Each feature has a confidence factor, called feature importance rank, which is used to evaluate the importance of the feature with respect to its case. For Example if *feature1* has an importance rank greater than the importance rank of *feature2*, then *feature1* is seemed to be more valuable for its case than *feature2*. Thus, the importance ranks can be used to differentiate between the features of the case and classify them to different classes of features.

Measuring the difference between cases: The process of retrieving an analogical case (source case) to the target case is based on finding the common and the different features between the two cases. This commonality is occurred either by identical features or by analogical features. In case of identical features, the common features can be found by applying the normal intersection process between the two sets of features. While in case of analogical features, the intersection process must take into consideration the background knowledge about the analogy relations among the features. This is what will be called, the *generalized intersection* between two sets of features.

For example, if we have the two sets of features:
L1 = {F1, F2, F3}, L2 = {F2, F4, F5}, and we have in the background knowledge that F1 and F5 are analogical to each other, then:
Generalized Intersection (L1, L2) = {F2, F5}, Difference (L1, L2) = {F3}, and Difference (L2, L1) = {F4}.

Category Retriever:

The set of category features resulted from the feature classifier procedure is used as a guide to retrieve the suitable category class of the target case from the stored classes of categories in the KB. This procedure retrieves the category class, which has the maximum common features with the target case. These common features may be identical or analog to each other. Thus, the procedure will use the generalized intersection procedure to find the category class of the maximum intersections with the target case.

Matched Cases Extractor:

The reminding features, obtained from the feature classifier procedure, are the case features that are common with the reminding features of the analogical cases. The matched cases extractor procedure applies the generalized intersection procedure on the target case reminding features and all cases, which belong to its category class in the case base. Thus, it extracts a set of similar or near-similar cases for the target case. This set of cases can be filtered again by applying the generalized intersection, once more, on the target case tail features and the tail features of the set of cases to extract the matched cases of our target.

Most Analogical Case Selector:

From the previous procedure, matched cases extractor, a set of matched cases for the target case is generated. From this set of matched cases, the most analogical case selector selects the most matching one to the target case. This is done using the importance ranks, which is associated with the features of the source cases, to calculate a matching rank for the case as a whole. Then, it gets the case of the highest matching rank, which is considered as the most analogical source case to our target case. The case matching rank is calculated using the average sum, a linear metric relation, depending on the values of the features ranks.

3.4 Learning Module

The learning module is responsible for performing the learning process and finding the required solution for the given problem.

Different Learning Situations: The output resulted from the case retriever module is the analogical source case of the target case and its category class. In analyzing this output, one of the following situations can be encountered:

- *An analogical source case, with a classified category for input target case.*
- *A classified category for the input target case, with non-similar case.*

• *Neither analog source case, nor classified category for input target case.*

In the first situation, we have analogical source case for our target case. Thus, we must scan the features of the two cases to extract the difference between them. This difference will lead us to find the solution of the unknown features of the target case. It may also lead to create, or modify some features of the source case. Consequently, some relations in the background knowledge may be created or modified. If the target and source case have the same features and their values are typical or analog to each other, we can generalize the two cases to a generalized case. In the second situation, we get the category class of the target case but no analogical source case was found. Thus we can store this target case as a new case in the retrieved category. Finally, in the third situation, there is no category in the KB that can be common with the category features of our target case. In this situation, an interaction with domain expert to enter the knowledge of a new category can take place. Then, this new category class is created and the given target case is stored as the first case in this category. In all previous situations, it may be needed to modify some existing knowledge or to create a new one. The learning algorithm performs the different learning situations during the process of analyzing the retrieved analogical case and its comparison with the target case. The procedure gets, as input, the retrieved target case with its category, and the source case. Then, it produces the target solution and the learned knowledge. To summarize, the following different seven learning problems can be encountered:

- Finding target case's solution.
- Creating a new category .
- Refining the source case.
- Generalizing cases.
- Creating a new case.
- Refining existing category.
- Refining the knowledge base.

Analogy Method Selection: The selection process of the analogy method to be applied in the learning task is done according to the type of the matched features and the relations between them as they appeared in the knowledge base, e.g. hierarchy relation, proportional relation or determination rules. The learning task in any of the previous seven learning situations can be done using one of the following methods of analogy:

- *Transformational analogy*, by transferring the solution of the features from the source situation to the target situation, if the features are related to each other in a form of hierarchy relation.
- *Proportional analogy*, by simulating features properties from the source situation to the target situation, if the features are related to each other in a form of proportional relation.
- *Determination-based analogy*, by using some determination rules, between two analog situations features, if the features are related to each other in a form of determination rule.

The learning module is responsible for performing the different previous discussed learning situations. Figure 8 shows the essential components of the learning module, which are: Analogical Case Analyzer, Expert Requester, Case/Category Creator, Case Refiner, Case Generalizer, and Knowledge Refiner.

Learning Submodules:

Analogical Case Analyzer:

The analogical case analyzer procedure receives the retrieved source case and its category class, resulted from the case retriever module. Depending on this output, one of the following procedures is performed:

- Expert request procedure: If no retrieved category was found for the target.
- Case/Category Creator procedure: When there is no analogical case.
- Case Refiner procedure: When an analogical case of a classified category exist.

Expert Requester:

If there is no retrieved category for the input target case; that is, there is no such category which includes our target case, then a need for help from domain expert will be required. This can be performed through the expert requester procedure. The expert is asked to take the decision for creating a new category and to enter all its required knowledge. The system receives this knowledge and interacts with the expert through the user interface module.

Case/Category Creator:

The case/category creator procedure is responsible for the process of creating new cases using the target case, and the process of creating new categories, through domain expert interaction. The procedure gets the new knowledge of a case or a category and transforms it in a suitable format compatible with the knowledge format stored in the KB.

Case Refiner:

In case of retrieving an analogical case for our target case, a refining process is required to copy or reuse some features in the source case to the target case, resulting in a solution for the requested or unknown features of the target case. On the other hand, in case of retrieving an analogical case for the target case, it may contain more detailed information than the source case. Thus, the target case can be used to improve the stored source case. The case refiner procedure makes a comparing process between the two cases, target and source cases, this process will result the following two sets of features:

- Source Candidate Features: these are the features of the source case, which are not found in the target case features.
- Target Candidate Features: these are the features of the target case, which are not found in the source case features.

The set of source candidate features can be used to find the unknown features of the target case or to refine its knowledge by reusing or transforming these

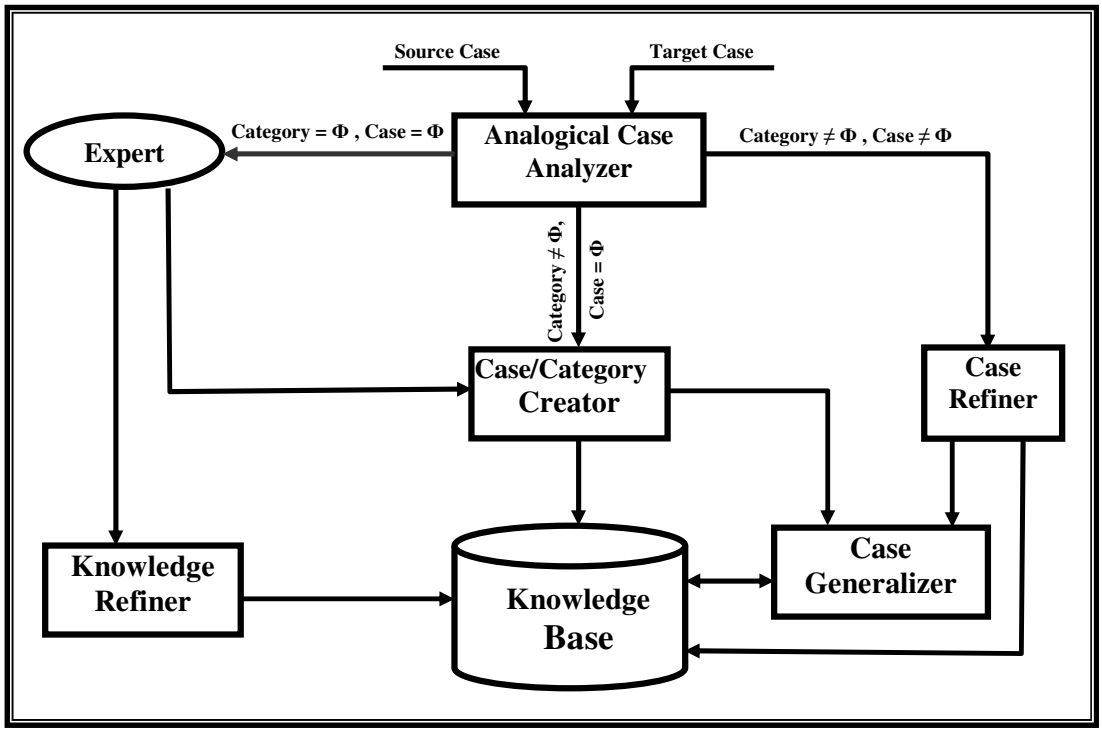

Figure 8: Learning Module

features to the target case using the analogy rules that are related to the features in the KB. Similarly, the set of target candidate features can be used to refine the source case knowledge. The target case may contain more detailed information than the source case. Thus the target case can be used to improve the stored source case. According to both of source/target candidate features and their relevance to the background knowledge, which could be a proportional relation, determination rule, or hierarchical relation, one of the three analogical learning methods will be selected i.e. proportional analogy, determination-based analogy, and transformational analogy.

Case Generalizer:

The case generalizer procedure is used to generalize two typical analog cases that have common features with the same or analogical values, to get a more general case. These generalized cases can be viewed as subcategories. Storing common cases as subcategories may improve the efficiency of the storing and retrieval processes.

Knowledge Refiner:

In case of new category creation, new case creation, or refining an existing case, some knowledge may be added or modified in the KB, to satisfy the new case situation. This is done using the knowledge refiner procedure. The knowledge refiner algorithm may refine a case or category frame by adding new slots. It may also modify the structure of a hierarchy and add a new analogy or determination-based relation into the KB.

4. CONCLUSION

The learning by analogy approach is concerned mainly with the previous experience in solving similar new problem situations, instead of solving it from scratch. The objective of using the learning by analogy was to reduce the time and the exerted effort when solving a new problem. This paper has described a new learning by analogy system that integrates different methods of analogy in one hybrid model. Using one method of analogy in a learning system constrain the learning process. The different structures of the background knowledge and the given target cases features will require different methods of analogy, thus we developed a hybrid learning model to get the benefits of the different methods of analogy and to provide the suitable analogy method for every target case. The domain of employment accidents used to demonstrate the capability of the proposed learning system on a two real cases from the domain.

5. References

[1] J.G.Carbonell & Ryszad S. & Tom M. , "*An Overview of Machine Learning,*" Machine Learning, An Artificial Intellgence Approach, Pub. Morgan Kaufmann, Palo Alto, 1986.

[2] D.M.Boase-Jelinek & D.Milech, *"Role of Analogical Reasoning as a Tool for Training,*" Proc. 5th International Conference on Human-Computer Interaction, Orlando, Florida, Vol. II, Ed. G.Salvendy & M.J.Ssmith, Pub. Elsevier Science, Amsterdam, 1993.

[3] J.G.Carbonell, "*Learning By Analogy: Formulating and Generalizing Plans from Past Experience,*" Machine Learning, An Artificial Intellgence Approach, Pub. Morgan Kaufmann, Palo Alto, 1986.

[4] Stuart J. Russel, "*Analogy and Single-Instance Generalization*," Proc. 4th International Workshop on Machine Learning, Univ. California, Irvine, pub. Morgan Kaufmann, Los Altos, 1987.

[5] P.H. Winston , "*Learning and Reasoning by Analogy*," Communications of the ACM, (23), 1980.

[6] Christel Vrain, Yves Kodratoff, "*The Use of Analogy in Incremental SBL,*" Proc. Knowledge Representation and Organization in Machine Learning, LNAI 347, Ed. K.Morik, Pub. Springer-Verlag, 1987.

[7] Markman, A. "Constraints on Analogical Inference", Cognitive Science, 21, 4, 373-418, 1997.

[8] Holyoak K. J. Novick L. Melz E. "Component processes in Analogical Transfer: Mapping, Pattern completion and Adaptation", in Analogy, Metaphor and Reminding, Eds. Barnden and Holyoak, Ablex, Norwood,

[9] Markman, A.B. & Gentner, D.. Structure mapping in analogy and similarity. In P. Thagard (Ed.), *Mind readings*. Cambridge, MA: MIT Press, 1998.

[10] Hummel, J. E. Holyoak, K. J. "Distributed Representation of Structure: A Theory of Analogical Access and Mapping", Psychological Review, 1997.

[11] Veale, T. & Keane, M.T. The competence of sub-optimal structure mapping on 'hard' analogies. *IJCAI'97: The 15th International Joint Conference on Artificial Intelligence.* Morgan Kaufmann, 1997.

[12] Costello, F. & Keane, M.T. Efficient creativity: Constraints on conceptual combination.*Cognitive Science*, 2000.

[13] Gentner, D., Holyoak, K.J., & Kokinov, B. *The Analogical Mind.* Cambridge, MASS; MIT Press, 2001.

[14] Keane, M.T. & Costello, F. Why Conceptual Combination is Seldom Analogy. In D. Gentner, K.J. Holyoak, & B. Kokinov (Eds.), *T h e Analogical Mind*. Cambridge, MASS; MIT Press, 2001.

[15] Agnar Aamodt & Enric Plaza, "*Case-based reasoning: Foundational issues, methodological variations, and system approaches*," AI-Communications, Vol 7, No.1, 1994.

[16] Ashok K. Goel. "*AI in design: Design, Analogy, and Creativity*," IEEE, Intelligent systems & their applications, Vol.12-No.3, 1997.

[17] Jack Mostow, "*Design by Drivational Analogy: Issues in the Automated Replay of Design Plans,*" Machine Learning Paradigms and Methods, Ed. J.G.Carbonell.

[18] Bipin Indurkhya, "*Modes of Analogy,*" Proc. Analogical and Inductive Inference International Workshop, Reinhardsbrunn Castle,GDR, LNAI 397, Ed. K.P.Jantke, Pub. Springer-Verlag, 1989.

[19] Manfred Kerber, "*Some Aspects of Analogy in Mathematical Reasoning,*" Proc. Analogical and Inductive Inference International Workshop, Reinhardsbrunn Castle,GDR, LNAI 397, Ed. K.P.Jantke, Pub. Springer-Verlag, 1989.

[20] Ken Wellsch & Marlene Jones, "*Computational Analogy,*" 7th European Conference on Artificial Intelligence ,Advances in Artificial Intelligence-II, Brighton, U.K., Ed. B.Boulay, D.Hogg, L.Steels, Pub. Elsevier Science, Amsterdam, 1986.

[21] Kokinov, B. N. "A Model of Reasoning by Analogy", in Analogy, Metaphor and Reminding, Eds. Barnden and Holyoak, Ablex,. Norwood, NJ: 1994.

[22] Forbus, K. Gentner, D. Markman, A. Ferguson, R. "Analogy just looks like high level perception", Journal of experimental and Theoretical Artificial Intelligence, 1999.

6. Appendix: Two Case Studies

The following two real cases, from the domain of employment accidents, are used to demonstrate the learning capability of the system.

Case Study 1

Given the following target case: "An employment injury, at the daily work time, due to the collapse of the stockroom over him when he was bringing some spare parts to his machine, and this result in a complete amputation for his left leg and a total inability". For this target case the frame that represents its features with its values and importance ranks, is shown in Figure 9

FRAME: Target Case1			
Slot	**Value**	**Facet**	**Confidence Factor**
accident_place	stockroom	LWP hierarchy	1.0
accident_time	daily_work_time	LWT hierarchy	1.0
accident_action	collapse	WAC hierarchy	0.9
body_injury	complete_paralysis	motion_organ hierarchy	0.8
inability	total		0.7
organ_injury	right_leg		0.6
work_activity	bring_spare_parts		0.3

Figure 9: Target Case1 frame

Find:

1- Type of category that covers this case.
2- Inability_pension and different compensation for the injured employment.

Solution:

Step 1:

The *user interface* module gets the target case features and passes it to the *case retriever* module, which has as a first procedure the *feature classifier* module. Using the importance ranks of each feature the *f*eature classifier module will classify the target case features into three sets, which are:

Category features =
{(accident_place, stockroom) ,
(accident_time, daily_work_time),
(accident_action, collapse)}

Reminding features =
{(body_injury, complete_amputation),
(organ_injury, right_leg),
(inability, total)}

Tail features =
{(work_activity, bring_spare_parts)}

Step 2:

The second procedure of the case retriever module is the *category retriever*, which gets the obtained category features and applies the *generalized intersection* procedure on it with all the category classes stored in the case base.

Considering the *legal_work_place*, and *legal_work_time* hierarchies, which are shown in Figures 3, 5, and 10. The *stockroom* is a *related_work_place*, which is consequently a *legal_work_place*. Also, *daily_work_time* is a *main_time*, which is consequently a *legal_work_time*. Considering the *work_accident_condtion*, hierarchies the *collapse* is a *suddenly_accident,* which is consequently a proper *accident_condtion.* Thus, the three category features of the target case are equivalent to the three features of the category class *injury_at_work.* Therefor, the applying of *generalized intersection* procedure result the ***injury_at_work*** category class, which is the first requirement.

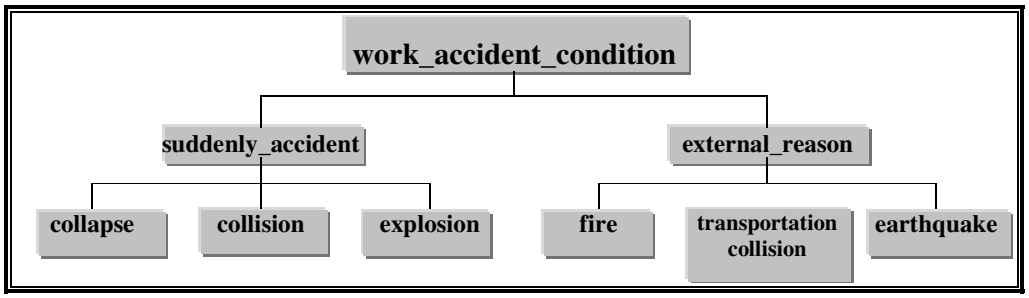

Figure 10: Work accident condition hierarchy

Step 3:

The third procedure of the case retriever module is the *matched cases extractor*, which gets the target case reminding features, {(body_injury, complete_amputation), (organ_injury, right_leg), (inability, total)}, and the retrieved category class, injury_*at_work*. Then, it applies the generalized intersection procedure with all the cases that belong to the retrieved category class stored in the case base. Now, consider the following cases, *C01, C02, C03,* and *C04* which belong to the retrieved category class and their reminding features are:

C01:

{(body_injury, complete_paralysis),
(organ_injury, right_leg), (inability, partial)}

C02:

{(body_injury, complete_bone_break),
(organ_injury, right_leg), (inability, total)}

C03:

{(body_injury, deep_wound),
(organ_injury, right_leg), (inability, partial)},

C04:

{(body_injury, complete_amputation),
(organ_injury, left_arm), (inability, partial)},

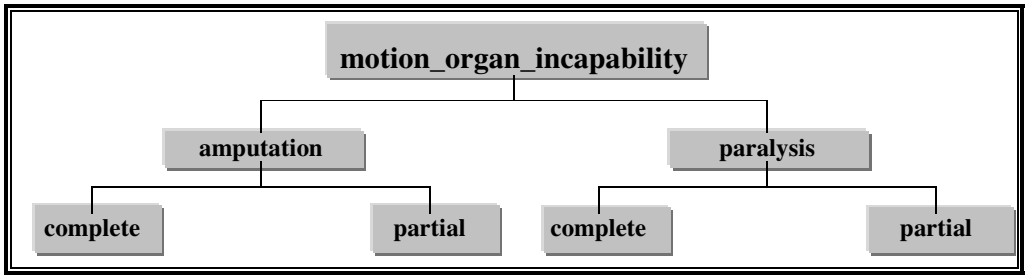

Figure 11: Motion organ incapability hierarchy

Considering the *motion_organ_incapability* hierarchy, which is shown in Figure 11, that describes the incapability of motion organ, the *amputation* of a motion member is equivalent to *paralysis* of it. Also, both of the right leg and the left arm are members of motion. Where, the inability feature is common between them but with different values, total and partial, thus the *case retrieval module* will return the two cases, *C01* and *C04* as two analogical source cases to the target case.

Step 4:

The fourth procedure of the *case retriever* module is the *most analogical case selector*. This procedure gets as input the set of analogical source cases, *C01* and *C02*, for our target case, then using the importance ranks of the features of the target case it finds the most analogical source case with the target case. Since the ranks of *amputation* and *leg* features are 0.8 and 0.6, thus:

Rank of amputation > Rank of leg

Therefore, the matching of "amputation" feature between our target case and C04 is more preferable than the matching of "leg" feature between the target case and C011. Thus the most analogical selector will consider *C04* to be the most analogical source case that can be retrieved.

Step 5:

The *case retriever* module will pass the retrieved analogical source case *C04* to the *learning module*, which has as a first procedure, the *analogical case analyzer*. This procedure will explore *C04* and discover that it is a proper case, thus it will divert the case to the *case refiner* procedure. Our retrieved analogical case, *C04,* has the following tail features:

{(salary_compensation,100%), (transportation_charge,100 LE),
(inability_ratio, 60%), (inability_pension, 30%)}

The *case refiner* procedure will compare the tail features of both the target and the retrieved case and find that, the retrieved case has features that can be suggested to be new features for the target case with little modification. Using the background knowledge we find that the learned knowledge are:

- The category name is *injury_at_work*.

- The *salary_compensation* is a fixed ratio 100%, thus it is transferred as it is to the target case.
- The *transportation_charge* is variable to each case, so it is transferred as a new feature without suggested value to the target case.
- The *inability_ratio* depending on the organ, which has amputated, from the background we find it is 80% for the left leg, so the *inability_ratio* feature transferred with value 80% as a new feature to the target case.
- Finally, the *inability_pension* is a fixed ratio 30%, thus it is transferred as is to the target case.

Case Study 2

Now, consider adding a new source case, shown in Figure 12, to the case base and running the system on the target case, described in case study 1. Then, the resulted matched case will be this new one. The reason is that both of the source and target cases have a higher degree of similarity than found in the former source case in case study 1. As a matter of fact, all the known features of the target case are found in the source case and their values either identical or analogical. Also the unknown features of the target case can directly be matched with the source case features. That is, Target case features – Source case features = Φ, and

Source case features – Target case features = Target unknown features

Thus the learning module will transfer the missing feature values from the source to the target case, then the case generalizer will create a new generalized case to comprise them as a sub-categories. This is shown in Figure 13.

slot	value
a_kind_of	?
accident_place	stockroom
accident_time	daily_w_time
accident_property	suddenly
accident_descrip.	collapse
body_injury	comp_amput..
organ_injury	left_leg
disability	?
injury_ratio	?
injury_pension	?
injury_compens.	**?**

Target Case

slot	value
a_kind_of	injury_at_work
accident_place	machine_room
accident_time	over_time
accident_property	suddenly
accident_descrip.	machine_falling
body_injury	comp_amput.
organ_injury	right_arm
disability	total
injury_ratio	100 %
injury_pension	130 %
injury_compens.	30 %

Source Case

Figure 12: Target/Source case frames

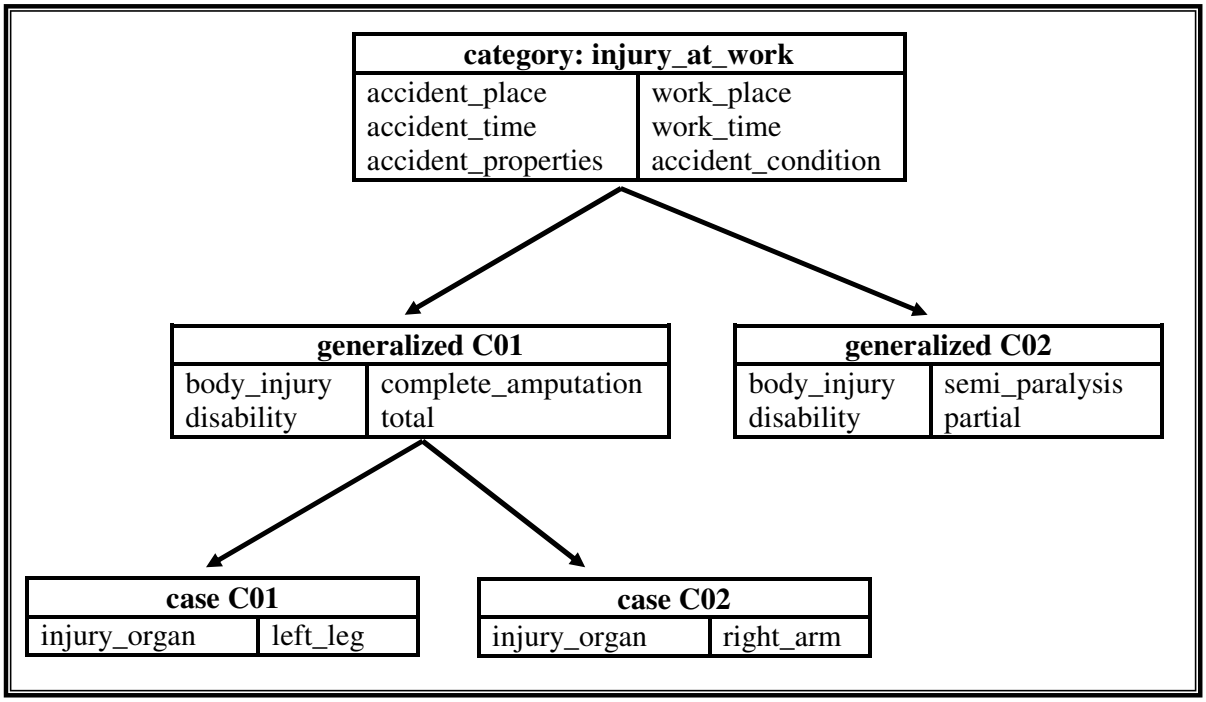

Figure 13: Case Generalizing