RIPPER를 이용한 전문가시스템의 규칙 생성 모듈 구현

Implementation of a Rule Generation Module for Expert System using RIPPER

김 군오, 김 진상
계명대학교 컴퓨터공학과

Koon O Kim, Jin Sang Kim
Department of Computer Engineering, Keimyung University

요 약

전문가시스템 개발에 있어서 지식획득 병목현상(knowledge acquisition bottleneck)은 해결해야 할 큰 걸림돌중 하나이다. 지식획득을 위한 여러 과정을 단순화하고 자동화함으로 지식공학자의 작업을 최소화하면서 전문지식을 쉽고 빠르게 획득할 수 있도록 지식획득시스템을 설계·구현한다면 전문가시스템의 대중화는 지금보다 쉽게 이루어질 것이다. 본 연구는 지식획득시스템 설계와 구현을 위한 연구의 일환으로 기계학습의 한 방법인 RIPPER(Repeated Incremental Pruning to Produce Error Reduction)를 이용하여 규칙을 생성하고 생성된 규칙을 JESS(Justification based Expert System Shell)에서 처리하도록 하였다. 규칙을 생성하기 위한 데이터는 Bohanec이 1997년도에 만든 자동차 평가 데이터베이스(Car Evaluation Database)를 사용하여 실험하였으며, 1700여 개의 레코드에서 약 40개의 규칙이 생성되었고, 생성된 규칙은 지식베이스의 정당성을 위반하지 않으면서 실행되었다.

1. 서론

전문가시스템(expert system) 개발에 있어서 가장 큰 걸림돌로 작용하고 있는 문제점 중 하나는 지식획득 병목현상(knowledge acquisition bottleneck)이다. 즉, 기존의 전문가시스템 개발에서 지식획득 작업을 수행하기 위해서는 인간전문가(domain expert)로부터 전문지식을 추출하여 시스템에 사용되어지는 지식으로 변환하는 작업을 수행하는 지식공학자(knowledge engineer)의 역할이 필수적이며 전문가시스템을 개발하는 구성원 중 지식전문가가 이 역할을 수행해야만 한다. 그러나 이제까지의 경험상으로 시스템 개발비용을 낮추거나 인간전문가를 비롯한 일반 사용자들이 보다 편리하고 쉽게 전문가시스템을 이용하기 위해서는 지식공학자의 역할을 시스템이 대신 처리하던지 아니면 그 역할의 비중을 최소화 시켜야하는 필요성이 대두되었다. 이러한 필요성에 대한 연구의 일환으로 최근 전문 지식에 대한 인지학 측면에서의 새로운 해석이 시도되고 있으며 새로운 전문가시스템 개발 방법들이 활발히 연구되고 있다. 그 중에서도 전문지식이 오랜 경험을 통하여 축적된다는 것에 착안한 사례기반 학습방법이나 많은 예제들을 기반으로 하여 올바른 개념을 학습하는 것으로 지식획득에 중점을 두는 귀납적 학습방법 등이 대표적인 예이다.

전문가시스템은 인공지능분야 중에서 상업적으로 성공한 사례들이 가장 많이 발표되는 분야이지만

이러한 상업적인 성공에도 불구하고 전문가시스템의 대중화는 이루어지지 않고 있다. 그 이유는 앞에서 설명한 지식획득이 어렵다는 점이 가장 큰 이유라 할 수 있으며 만약 이러한 문제점을 해결한다면 지금보다 더 큰 상업적 성공은 물론 대중화에도 크게 기여하게 될 것이라 본다[KCE97]. 본 연구의 목적은 지식획득 병목현상이라 부르는 지식베이스로의 지식획득에 관한 여러 과정을 단순화하고 자동화함으로 지식공학자의 부담을 줄이면서 전문지식을 쉽고 빠르게 획득할 수 있도록 규칙 기반 전문가시스템 지식베이스에다 규칙 생성 모듈을 설계·구현하는 것이다. 규칙 생성 모듈의 구현을 위해서는 기계학습 기법을 이용하는데, 이것을 이용하면 외부로부터 필요한 지식을 습득하고 획득된 지식들을 이용하여 지식베이스를 자체적으로 구축할 수 있다.

본 논문에서는 기계학습의 한 방법인 RIPPER (Repeated Incremental Pruning to Produce Error Reduction)를 이용해 지식을 획득하고, 획득된 지식을 JESS(Justification based Expert System Shell)라는 전문가시스템 쉘이 읽어들여 문제를 해결할 수 있도록 구현하고자 한다. 2절에서는 기계학습을 이용해 지식을 획득하는 기본 알고리즘에 대해서 살펴보고, 3절에서는 귀납적 학습방법인 RIPPER를 이용한 지식획득 그리고 JESS 시스템과의 연동을 위한 설계 및 구현에 대해 기술하고자 한다. 4절에서는 본 논문에서 설계·구현한 규칙 생성 모듈과 JESS시스템에 실제 데이터를 주어 실험한 결과를 기술하고자 한다. 그리고 마지막 5절에는 향후 연구과제와 보완점에 대해 기술하겠다.

2. 기계학습을 이용한 지식획득

2.1 결정트리 학습

결정트리(Decision tree) 학습은 이산적인 값을 가진 타겟함수로 근사화시키는 방법의 학습이며 결정트리 학습으로 학습된 타겟함수는 결정트리의 형태를 가지게 되고 이 결정트리는 'if-then' 형태의 규칙으로 쉽게 변환이 가능하다. 귀납학습 알고리즘들 중에서 가장 널리 알려진 결정트리 방법을 이용하여 전문가시스템 지식베이스를 위한 규칙 생성 기법의 기본 이론으로 삼고자 한다. 결정트리의 표현 방식을 보면 먼저, 사례들은 루트노드에서 단말노드까지 아래로 내려가면서 정렬되고 각 노드는 사례의 속성을 테스트하는 것이며 각 가지(branch)는 해당 속성이 가질 수 있는 값을 나타낸다. 하나의 사례는 루트노드에서 출발하여 속성을 테스트하고 속성의 값에 따라 해당하는 가지로 따라가게 되는데 이러한 처리 과정을 반복하게 된다. 결정트리 학습에 속하는 대부분의 알고리즘들은 가능한 결정트리 공간에서 하향식의 greedy 탐색을 이용하는 핵심 알고리즘의 변형이며 이 핵심 알고리즘은 ID3 알고리즘에서 처음 제시되었고 그 이후 ID3 알고리즘에 의해 설계된 Hunt의 개념학습시스템(Concept Learning System)을 기반으로 C4.5가 개발되었다[Quin93]. 결정트리 학습 시 루트노드에서 어떤 속성을 테스트할 것인가를 결정하는 것은 매우 중요한 사항인데 이 질문에 대한 답은 각 사례 속성들을 통계적으로 테스트하여 가장 많은 훈련예제를 분류할 수 있는 속성을 찾아 결정하는 것이다.

ID3에서는 어떤 속성이 최선의 것인지 선택할 때 정보이득(Information Gain)으로 결정하게 되는데 정보이득을 계산해 내려면 엔트로피(Entropy)를 먼저 계산해야만 한다. 엔트로피는 훈련예제 집단의 불순도를 재는 측도이고 정보이득은 훈련예제를 분류할 때 속성의 효과성을 잴 수 있는 측도를 말한다. 먼저, 엔트로피를 계산하기 위해 S를 훈련예제들의 샘플이라고 가정하고 $P_{\oplus}$는 S에서 긍정적 예제의 비율로, $P_{\ominus}$는 부정적인 예제의 비율로 각각 가정한다면 S의 엔트로피를 구하는 식은 다음과 같다 :

$$Entropy(S) = -P_{\oplus} \log_2 P_{\oplus} - P_{\ominus} \log_2 P_{\ominus}$$

어떤 속성의 정보이득이란 그 속성으로 예제를 나눌 때 얻을 수 있는 엔트로피의 감소치를 말하는데 훈련예제의 집단 S에 대한 속성 A의 정보이득을 Gain(S,A)로 표시하고 다음과 같은 식으로 구한다 :

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

위 식에서 Values(A)란 속성 A의 가능한 모든 값의 집합이고 S_v는 속성 A의 값이 v인 S의 부분집합을 나타낸다. 그리고 첫째 항 Entropy(S)는 원 집단의 엔트로피를 나타내고 둘째 항은 속성 A로 S를 나눈 후 기대되는 엔트로피의 값을 나타낸다. 이와 같은 방법으로 각각의 정보이득을 계산하여 훈련예제를 분류하므로 서 어느 속성이 더 유리한지를 판단할 수 있다.

2.2 귀납적 기계학습

기계학습에는 귀납적인 학습방법, 연역적인 학습방법, 사례기반의 학습방법으로 나뉘어진다. 귀납적 학습방법은 많은 예제들을 기반으로 올바른 개념(concept)을 추출하는 것으로 지식획득에 중점을 두는 것이고, 연역적 학습방법은 문제를 해결하는 경험으로부터 학습하는 것으로 기존의 지식이 재정의되어 변형된 형태로 존재하며 유사한 문제 발생 시 쉽게 결정 내리는 방법이다. 사례기반 학습방법은 일정한 분야에 적용할 수 있는 대표적인 사례들을 기억하고 있다가 새로운 문제를 해결하는 과정에서 유사한 사례들을 이용해서 문제를 해결하는 방법이다[CP99].

앞에서 언급한 ID3와 C4.5는 결정트리를 이용하여 분류모델(classification)을 형성하는 귀납적 기계학습 방법의 대표적인 알고리즘이다[Quin93]. 귀납적 방식은 {<속성1-값1>, <속성2-값2>, ... , <속성n-값n>}의 형태로 표현되는 많은 예제들로부터 이를 효과적으로 설명할 수 있는 개념을 구하는데 유용하며 속성(attribute)과 값(value)으로 이루어진 쌍의 집합 즉, case를 이용하여 결정트리를 생성한다.

C4.5는 ID3 알고리즘을 기반으로 설계된 개념학습 시스템을 기반으로 개발되었는데 ID3가 가지고 있던 문제점들 즉, 이산값(Discrete value)인 데이터 집합의 값들을 학습할 수 없었던 점과 특정 속성의 값이 구체적으로 알려져 있지 않는 값들을 처리할 수 없었던 점, 그리고 post-prunning방법을 이용하여 오버피트(overfit) 문제점을 해결하였다. 또한 C4.5의 여러 장점을 계승한 RIPPER라는 시스템이 Cohen에 의해 개발되었다. RIPPER가 C4.5를 포함한 여러 규칙학습시스템과 비교할 때 가지는 장점으로는 첫째, 큰 노이즈 데이터 집합에서 빠르다($O(n\log n^2)$)라는 점이고, 두 번째로 좋은 결과를 얻을 수 있다는 점이다. 예를 들어 37개의 벤치마크 테스트를 통해서 22개가 C4.5규칙보다 에러율(error rate)이 낮거나 같았다. 세 번째로 RIPPER에서 생성되는 규칙은 이해하기 쉽고 간결하다는 점이다[OJ98].

RIPPER는 separate-and-conquer 접근방법을 사용하여 훈련예제로부터 규칙들을 직접 만들어 낸다. 학습된 규칙집합은 후처리과정에 의해 버려지거나 테스트 데이터 상의 분류의 정확성을 향상시키기 위해 여러 가지 기준을 이용하여 몇 가지 규칙들을 제거한다. RIPPER는 IREP 알고리즘을 확장하여 설계된 것으로 IREP 알고리즘 역시 separate-and-conquer 기법을 사용하고 있으며, 동시에 규칙을 학습하는 대량의 규칙 귀납 알고리즘이다. 이 알고리즘의 처리는 훈련집합이 공집합이 되거나 종료조건이 만족될 때까지 반복된다. 다음의 [표1]은 IREP 알고리즘이다[Coh95, YTCH99].

```
procedure IREP(Pos, Neg)
begin
 Ruleset := ∅
 While Pos ≠ ∅ do
  /* grow and prune a new rule */
  split (Pos, Neg) into (GrowPos, GrowNeg)
   and (PrunePos, PruneNeg)
  Rule := GrowRule(GrowPos, GrowNeg)
  Rule := PruneRule(Rule, PrunePos, PruneNeg)
  /* stopping condition */
  if the error rate of Rule on
   (PrunePos, PruneNeg) exceeds 50% then
     return Ruleset
  else
   add Rule to Ruleset
   remove examples covered by Rule from (Pos,Neg)
  endif
 endwhile
 return Ruleset
end
```

[표1] IREP 알고리즘

IREP는 최대 다음과 같은 크기의 조건을 제거한다:

$$v(\text{Rule}, \text{PrunePos}, \text{PruneNeg}) \equiv \frac{p+(N-n)}{P+N}$$

여기서 P와 N은 PrunePos와 PruneNeg에서의 전체 예제들의 수를 나타내고 p와 n은 규칙에 의해 올바르게 분류된 PrunePos와 PruneNeg의 예제들의 수를 나타낸다.

RIPPER는 위의 식을 다음의 식으로 대치한 것이다:

$$v^*(\text{Rule}, \text{PrunePos}, \text{PruneNeg}) \equiv \frac{p-n}{p+n}$$

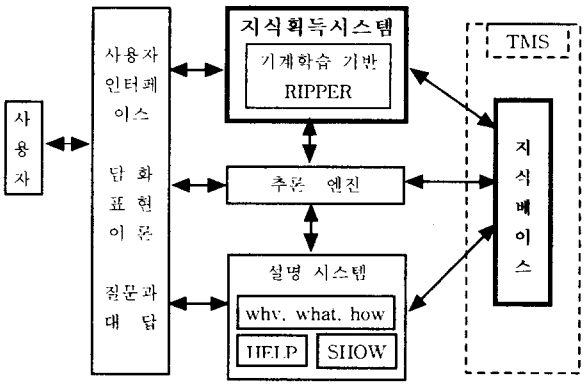
본 논문에서는 여러 규칙학습시스템보다 뛰어난 장점들을 가진 RIPPER를 이용하여 규칙 생성 모듈을 구현하고자 한다. 물론 생성되는 규칙들의 형태를

JESS가 처리할 수 있는 규칙들의 형태로 바꾸어 주어야 하기 때문에 RIPPER의 규칙 생성 방법의 상당부분을 수정하여야만 한다.

3. 구현

3.1 설계

본 연구에서 구현할 규칙 생성 모듈은 지식베이스와 인터프리터를 구분하여 관리하는 시스템에서 지식베이스 부분은 기계학습 이론을 통해 설계된 RIPPER를 사용해 새로운 지식을 획득할 수 있는 시스템이다. 이 연구에서는 [그림1]과 같이 석사학위논문에서 구현된 JESS를 기반으로 새로운 규칙 생성 모듈을 전문가시스템 쉘에 접목시켜 구현하였다[Kim95, KK97a, KK97b, KK99]. 물론 여기서 설계되고 연구된 규칙 생성 모듈은 RIPPER를 기반으로 구현되었고 전문가시스템 지식베이스에 지식공학자의 작업을 최소화하여 새로운 지식을 획득할 수 있고 또한 지식베이스를 효율적으로 관리하는데 적합한 시스템이다. 개발 환경으로는 윈도우 환경에서 에딘버러(Edinburgh) 문법을 기반으로 하는 SWI-Prolog [Apt97]를 사용하여 추론엔진 부분을 포함한 JESS 시스템을 구현하였고, 지식획득 부분은 UNIX 환경에서 ANSI-C로 작성된 RIPPER를 윈도우 환경으로 변경하여 독립된 모듈로 구현하였다. 사용자 인터페이스 부분은 XPCE와 담화표현이론(Discourse Representation Theory) [KK96]을 사용하여 사용자의 시스템 사용환경이 편리하도록 구현하였다.



[그림1] JESS와 지식획득시스템 결합 구조도

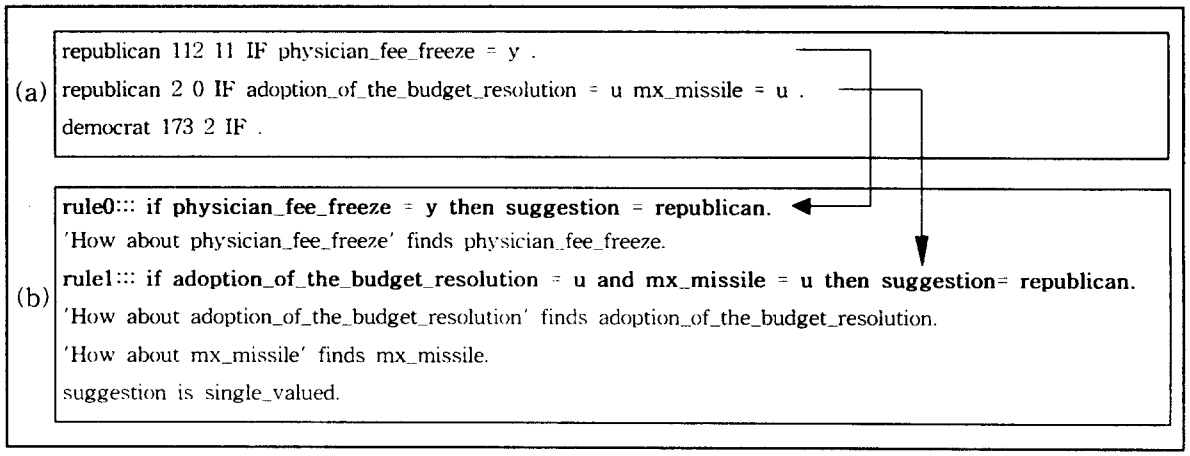
3.2 전문가시스템 쉘

JESS(Justification-based Expert System Shell) [Kim95]는 1995년 계명대 석사학위 논문에서 개발된 전문가시스템 쉘이다. 전문가시스템 쉘이란 전문가시스템에서 지식베이스를 제거한 인터프리터를 말한다[BCR96, LS93]. 즉, 전문가시스템이 규칙들의 집합으로 구성된 지식베이스와 그에 대한 처리를 목적으로 하는 인터프리터가 주된 구성 요소로 되어 있는 경우 지식베이스와 인터프리터는 별개로 구분하는 것이 가능하며, 따라서 특정 영역에 관한 전문지식을 가진 지식베이스와 인터프리터를 결합하면 새로운 전문가시스템이 구성 될 수 있다. 처음 구현 환경은 도스환경에서 LPA-PROLOG를 사용하여 구현되었으나 지금은 윈도우 환경에서 SWI-PROLOG를 사용하여 그 환경을 변환하였다.

JESS는 일반적인 규칙 기반 전문가시스템 쉘과 유사한 구조를 가지고 있으나 지식이 일관성과 완전성을 유지하도록 관리하는 시스템인 TMS(Truth Maintenance System) [KK97a, KK97b]와 자연어 처리를 위한 담화표현이론을 포함하고 있다는 점이 이 시스템의 특징이라 할 수 있다[KK96]. JESS의 TMS는 먼저 규칙들 중에서 중복된 지식이 있는지를 검사하고 상호 모순된 지식 즉, 규칙의 전제가 상반되는 경우에도 똑같은 결론을 유도해 내는 의미 없는 지식들을 찾아내어 삭제하므로 지식베이스의 일관성을 유지하는 기능을 수행한다. 담화표현이론은 먼저 어떤 증상을 간단한 자연어로 입력하여 불필요한 추론과정을 줄여 시스템 효율성을 증대시키는 기능을 수행하고 있으며 또한 Why, What, How 등의 다양한 설명기능들도 가지고 있다.

3.3 규칙 생성 모듈 구현

본 논문의 규칙 생성 모듈 구현을 위해 사용된 RIPPER는 규칙학습시스템으로 Cohen에 의해 UNIX 환경에서 ANSI-C를 사용하여 개발되었으며 그 기본 알고리즘은 IREP 알고리즘이다[Coh95]. RIPPER는 "*.data"파일과 "*.names"파일을 기반으로 새로운 규칙을 생성하여 "*.hyp"파일을 만들어 내는데, 만들어진 규칙의 형태는 JESS 시스템이 처리할 수 있는 규칙 형태와 상이하다. 따라서 RIPPER에서 생성된 규칙들이 JESS에서 그대로 읽혀져 처리되게 하기 위해서는 RIPPER의 규칙 생성부분을 수정하



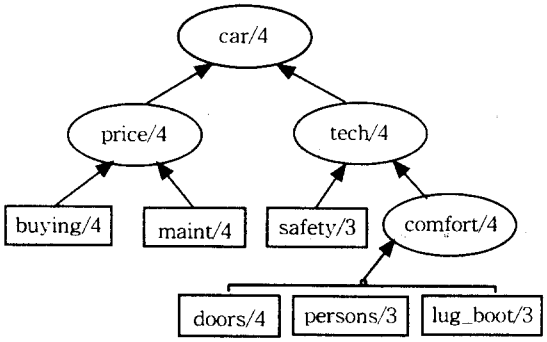
[그림2] 기존 RIPPER 생성된 규칙형태(a)와 RIPPER 수정 후 규칙형태(b)

여 JESS 시스템이 처리할 수 있는 규칙 형태로 출력되도록 하던지 아니면 JESS의 추론엔진 부분이 RIPPER가 생성해내는 규칙 형태를 인식할 수 있도록 수정해야만 한다. 본 연구에서는 전자의 방법을 이용하여 RIPPER의 규칙 생성 부분을 수정하였다. 다음의 [그림2]는 기존의 RIPPER에서 생성된 규칙형태([그림2] (a))와 RIPPER의 소스 수정 후 생성되는 규칙형태([그림2] (b))를 비교한 것이다. [그림2]에서는 모두 2개의 규칙이 만들어지는데 [그림2] (b)에서는 rule0과 rule1외에도 세 개의 질의문과 한 개의 문장이 자동적으로 더 생성되도록 하였다. 이러한 문장들은 JESS시스템에서 추론할 때 필요한 사항들이며 RIPPER에 의해 생성된 [그림2] (b) 형태의 규칙들과 질의문들은 특별한 수정 없이 바로 JESS 시스템에 적재된 후 추론 가능하도록 하기 위함이다.

4. 실험

4.1 자동차 평가 데이터베이스

본 논문에서 규칙 생성 모듈 테스트에 사용되는 예제로는 Bohanec이 1997년도에 만든 자동차 평가 데이터베이스(Car Evaluation Database)를 사용할 것이다. 이것은 DEX시스템의 실험을 위해 개발된 간단한 계층 결정 모델이다[Boh97, ZBBD97]. [그림3]은 자동차 평가 데이터베이스의 속성 구조를 나타낸다. [그림3]에서 car는 자동차의 만족성을 나타내고, price는 차량 전체 가격을, tech는 기술적인 특성들을 나타낸다. price에는 buying과 maint가 있는데



[그림3] 자동차 평가 데이터베이스 속성 구조

buying은 차량의 가격을 나타내고 maint는 유지보수 비용을 나타낸다. tech에는 자동차의 안정성을 나타내는 safety와 편의성을 나타내는 comfort로 구성되며, 다시 comfort는 자동차 문의 개수를 나타내는 doors와 탑승 인원을 나타내는 persons, 그리고 화물 적재 공간을 나타내는 lug_boot로 구성된다. 즉, 이 모델은 세 개의 중간 개념들(PRICE, TECH, COMPORT)로 구성된다. [표2]는 이 모델에서 사용된 각 개념들이 가지는 값들을 나타낸다.

CAR	unacc, acc, good, v_good
. PRICE	v_high, high, med, low
. . BUYING	v_high, high, med, low
. . MAINT	v_high, high, med, low
. TECH	poor, satisf, good, v_good
. . COMFORT	bad, acc, good, v_good
. . . DOORS	2, 3, 4, 5_more
. . . PERSONS	2, 4, more
. . . LUG_BOOT	small, med, big
. . SAFETY	low, med, high

[표2] 각 속성들의 값

자동차 평가 데이터베이스는 위의 구조적인 정보들이 제거된 예제들을 포함하는데, 자동차와 직접적으로 관련된 6가지 입력 속성들(buying, maint, doors, persons, lug_boot, safety)을 포함한다. 그리고 unacc 클래스의 개수는 1210(70.023%), acc 클래스의 개수는 384(22. 222%), good 클래스의 개수는 69(3.993%), v_good 클래스의 개수는 65(3.762%), 총 1728개의 사례들(instances)로 구성된 데이터 파일을 포함한다.

4.2 실험

자동차 평가 데이터베이스에서 사용된 6가지 속성을 기술한 names파일과 1768개의 데이터를 가진 data파일을 가지고 본 논문에서 구현된 규칙 생성 모듈을 테스트한다. [그림4]에서처럼 수정된 RIPPER를 사용하여 data파일을 학습하면 학습된 규칙이 생성되며 그 규칙들은 파일로도 생성된다.

[그림4] 지식획득

위 실험에서 생성된 규칙은 rule0에서 rule39까지 총 40개의 규칙이 생성되었으며 각 규칙의 질의문과 필요한 정보가 자동 생성되었다. [그림5]는 규칙 생성 모듈에서 생성된 규칙들의 일부분을 나타낸다.

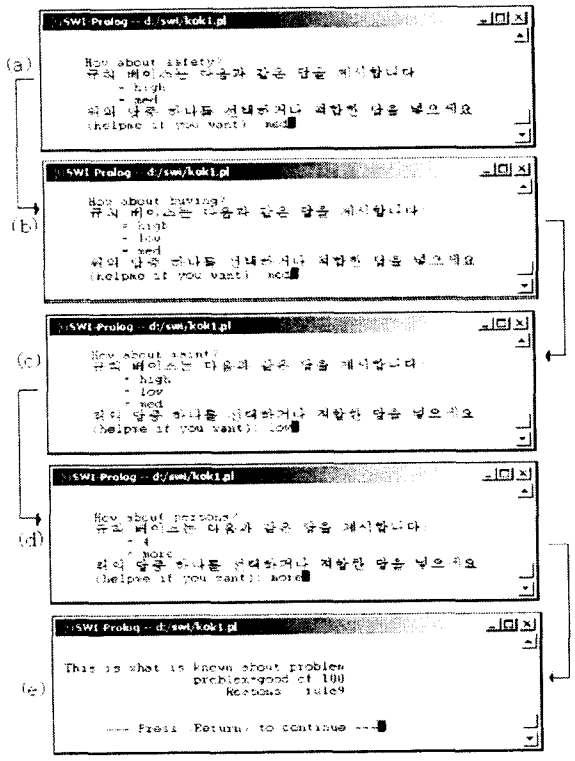
```
rule0::: if safety - high and buying - low and lug_boot- big
         and persons - 4 then suggestion - v_good.
rule1::: if safety - high and buying - med and maint - med
         and persons - 4 and lug_boot - big
         then suggestion - v_good.

rule9::: if maint - low and buying - med and persons-more
         and safety - med then suggestion - good.
rule10::: if buying - low and maint - low and persons-more
         and safety - high then suggestion - good.

rule38::: if safety - med and doors - more_5 and persons- 4
         and lug_boot - med then suggestion - acc.
rule39::: if buying - low and persons - more and safety-med
         and maint - high then suggestion - acc.

  'How about safety' finds safety.
  'How about doors' finds doors.
  'How about persons' finds persons.
  'How about lug_boot' finds lug_boot.
  'How about buying' finds buying.
  'How about maint' finds maint.

  suggestion is single_valued.
```

[그림5] 생성된 규칙들

생성된 규칙은 수정 없이 바로 JESS시스템에 적재되어 추론할 수 있다. JESS시스템의 load 명령을 사용하여 [그림5]의 생성된 규칙의 파일을 불러온 후 search 명령을 사용하면 추론이 시작된다. [그림6]은 rule9의 결과 good이 유도되는 과정을 나타낸다. JESS시스템은 사용자에게 자동차의 안전도, 가격, 세 번째로 자동차의 유지보수 비용, 마지막으로 최대 탑승인원에 대한 질문을 차례로 하는데 만약 사용자가 이러한 시스템의 질문에 med(보통)([그림6] (a)), med(보통)([그림6] (b)), low(낮음)([그림6] (c)), 그리고 more(4인 이상)([그림6] (d))라고 차례로 대답하였다면 이러한 사실들을 바탕으로 JESS시스템은 [그림6] (e)에서처럼 추론결과는 good으로, 추론 근거는 rule9에 의해서라고 결론을 내린다.

5. 결론

전문가시스템 개발에 있어서 큰 걸림돌로 작용하는 지식획득 병목현상을 해결하기 위해 본 논문에서는 기계학습 중 귀납적 학습방법인 RIPPER를 이용하여 규칙 생성 모듈을 설계·구현하였다. 그리고 구현된 규칙 생성 모듈의 실험을 위해 Bohanec이 만든 자동차 평가 데이터베이스를 사용하여 40개의 규칙을 생성해 내었으며 이 생성된 규칙을 JESS에 적재하여 실행시켰다. 1728개의 레코드로부터 생성된 규칙들은 레코드들 사이의 공통된 특징들로 잘



[그림6] JESS시스템의 추론과정

분류되어 유도된 규칙들이었으며 JESS에서 적재되어 실행될 때 지식베이스의 정당성을 위반하지 않고 실행되었다. 그러나 귀납적 학습방법에서 규칙을 생성해내기 위해서는 대량의 데이터가 필요하고 이러한 데이터를 만드는 것 또한 적지 않은 노력이 필요하다. 이러한 문제를 해결하기 위해서는 어떤 특정 분야의 데이터를 정형화되어있는 데이터 형태로 바꾸어주는 기술의 개발이 필요하다. 그러나 현재 웹 상에서나 주위에 존재하는 많은 데이터들은 정형화되어 있지 않는 데이터들이며 그러한 데이터들은 자동으로 RIPPER가 처리할 수 있는 데이터 형태로 정형화시키는 연구가 필요할 것으로 생각된다.

또 다른 연구 방향으로는 본 논문에서 사용한 RIPPER 자체를 향상시키는 작업인데 최근 관련 논문에 의하면 RIPPER를 개선한 SLIPPER(Simple Learner with Iterative Pruning to Produce Error Reduction)를 Cohen이 개발하여 실험해 본 결과 RIPPER보다는 20배, C4.5규칙보다는 22배정도 에러율을 낮추었다고 보고되고 있다[CS99].

참고문헌

[CP99] 최은영, 박영택. "Case Selection을 이용한 기계학습". 1999년도 한국정보과학회 춘계 학술 발표 논문집. 1999.

[KCE97] 강병호, 폴 콤프톤, 그렌 에드워드. "Knowledge Engineer을 배제한 Case를 이용한 전문지식의 획득". 1997.

[Kim95] 김군오. 정당화를 기반으로 한 전문가 시스템 쉘. 계명대학교 일반대학원 석사학위 논문. 1995.

[KK96] 김군오, 김진상. "담화 표현 이론을 이용한 전문가 시스템 쉘의 사용자 인터페이스". 1996년도 한국정보과학회 춘계 학술 발표 논문집. 1996.

[KK97a] 김진상, 김군오. "JTMS를 기반으로 한 전문가 시스템 쉘의 설명시스템". 산업기술 연구소 논문 보고집. 계명대학교. 1997.

[KK97b] 김군오, 김진상. "LTMS 기능을 가진 전문가 시스템 쉘". 1997년도 한국정보과학회 추계 학술 발표 논문집. 1997.

[KK99] 김군오, 김진상. "전문가 시스템 쉘을 이용한 자동차 진단 시스템 구현". 1999년도 한국정보과학회 춘계 학술 발표 논문집. 1999.

[Apt97] Krzysztof R. Apt, From Logic Programming to Prolog, Prentice Hall, 1997.

[BCR96] R. Boswell, S. Craw, R. Rowe. "Refinement of a Product Formulation Expert System". In Proceedings of the ECAI-96 Workshop on Validation, Verification and Refinement of KBS. 1996.

[Boh97] Marko Bohanec. http://www-ai.ijs.si/BlazZupan/car.html. 1997.

[Coh95] William W. Cohen. "Fast Effective Rule Induction". Machine Learning: Proceedings of the Twelfth International Conference. 1995.

[CS99] William W. Cohen, Yoram Singer. "A Simple, Fast, and Effective Rule Learner". AAAI-99. 1999.

[LS93] G. Luger, W. Stubblefield. Artificial Intelligence. The Benjamin/ Cummings Publishing Company, Inc. 1993.

[OJ98] T. Oates, D. Jensen. "Large Datasets Lead to Overly Complex Models: an Explanation and a Solution". 1998.

[Quin93] J. Ross Quinlan. C4.5: PROGRAMS FOR MACHINE LEARNING. Morgan Kaufmann Publishers. 1993.

[YTCH99] J. Yang, A. Tiyyagura, F. Chen, V. Honavar. "Feature Subset Selection for Rule Induction Using RIPPER". 1999.

[ZBBD97] B. Zupan, M. Bohanec, I. Bratko, J. Demsar. "Machine Learning by Function Decomposition". ICML-97. 1997.