Knowledge Base의 구축/개선 기술동향

金 在 熹
(正 會 員)
延世大學校 工科大學 電子工學科 助教授

Ⅰ. 서론 (Knowledge Base의 배경)

최근에 이르러 인공지능의 연구는 여러 분야에서 괄목할만한 성장을 보이고 있는데, 이 중에서도 특히 주목을 받고 있는 것은 expert system이나 knowledge-based system일 것이다. 이 논문은 특별히 knowledge-based system의 핵심이 되는 knowledge base를 구축하고 이를 개선하기 위한 여러 형태의 연구동향을 서술한 것이다.

엄밀히 말하자면, knowledge-based system과 expert system은 의미하는 것이 약간 다르다. Knowledge-based system은 지능적 성격을 보이는 인공지능시스템의 일종으로써, 다루는 응용분야에 관한 문제분야의 특별한 지식(domain knowledge)을 시스템의 다른 지식(general problem-solving knowledge)과 분리시켜 놓은 것이다. 시스템의 다른 지식은, 지식을 다루는 방법, 문제를 풀이하는 방법, 그리고 시스템과 사용자와의 연결에 관한 것등의 문제분야에 국한되지 않는 일반적인 지식을 의미한다. 일반적으로, 전자는 knowledge base로 구성되고, 후자는 inference engine으로 구성되어, knowledge-based system은 이러한 두 부분으로. 나뉘어 진다. Expert system은 knowledge-based system의 일종으로써, 특별히 의사, 법률가, 교수등의 전문가의 지식을 필요로하는 실사회의 어려운 문제를 다루게 된다.[1] 여기에서는 expert system에서의 knowledge base에 관한 문제를 취급할 것이지만, 대부분의 내용은 보다 포괄적인 knowledge-based system에 대하여서도 적용될 것이다.

일반적으로 expert system은 그림 1과 같은 과정을 거쳐 개발된다. 즉, 전문가의 충고를 이해하고, 이를 프로그램화 하며, 실험을 거쳐 지식을 개선하는

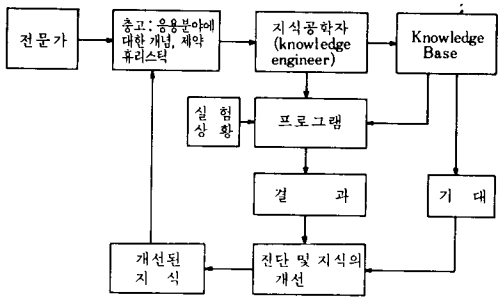

그림 1. Expert system의 개발과정

작업을 반복한다. 이를 각각 지식의 획득(knowledge acquisition), 지식의 프로그래밍(knowledge programming), 지식의 개선(knowledge refinement)으로 구분하기도 하나,[2] 여기에서는 이러한 과정들을 모두 합쳐 지식획득이라고 일컬으며, 앞서의 협의의 지식획득과 프로그래밍을 지식의 추출(knowledge extraction)이라고 표현할 것이다.

고전적으로는, 지식의 획득은 지식공학자(knowledge engineer)가 문제분야의 전문가와 면담을 통하여 이루어 진다. 이러한 경우 knowledge base를 구성하는 지식공학자가 문제분야의 여러 개념 및 제약과 전문가가 사용하는 해법등을 이해하여야 하므로, 오랜 기간이 소요될 뿐더러 다음과 같은 문제 때문에 결과의 시스템이 불완전한 성능을 보이게 된다.

(1) 전문가가 자신의 생각 및 지식을 정확히 분석 못함.

(2) 전문가가 문제분야의 각 경우에 대한 지식을 모두 전달하지 못함.

(3) 전문가가 제시한 법칙이, 잘못된 가정으로 인하여 올바른 결론을 내리지 못함.

(4) 지식공학자가 전문가의 충고를 소홀히 하였거나, 이해하지 못함.

이러한 여러 이유로 인하여 지식의 획득은 expert system의 개발에 대한 소요기간과 경비의 큰 부분을 차지하게 된다. Expert system의 수요가 급증함에 따라, 지식공학자의 공급은 부족하게 되어, 지식의 획득이 expert system의 개발에 큰 난제로 등장하여 소위 말하는 'knowledge aquisition bottleneck'[3],이라는 용어가 생겨나게 되었다. 그러나 지식획득의 어려움에 대한 인식과 이의 극복을 위한 시도는 expert system의 초기에서부터 진행되어 왔고, 콤퓨터가 지식공학자의 역할을 상당부분 대치하여 주는 자동적 지식획득에 관한 연구가 상당히 활발하다. 이 논문에서는 이러한 연구의 현황에 대하여 살펴 볼 것이다. 지식공학자에 의한 수동적 지식획득에 있어서, 제기되는 문제와 지식공학자의 역할에 대하여서는 [4]를, 문제분야의 전문가와의 면담방법 및 단계에 관하여서는 [1], [5]를 참고하기 바란다.

Ⅱ장에서는 지식공학자가 knowledge base를 구성하는 경우에 편의를 제공하는 knowledge base editor에 관하여 살펴 보았다. 이러한 knowledge base editor는 스스로 학습(learning)의 기능을 구비하고 있지는 못하지만, 일반적인 용도의 text editor와는 달리 expert system의 knowledge base를 편집하는데 유용한 것이다. Ⅲ장에서는 지식공학자의 관여없이 문제분야의 전문가가 직접 knowledge base를 구성하기 위한 지식추출에 관한 연구를, Ⅳ장에서는 일단 구성된 knowledge base의 내용을 개선하기 위한 연구를 서술하였다.

Ⅱ. Knowledge Base Editor

Expert system의 개발을 쉽고 빠르게 하기 위하여, 여러 분야에 걸쳐 다양한 expert system 개발기구가 제공되고 있다. 이 들은 inference engine을 제공하고 있으므로, 사용자는 이 들이 요구하는 형태대로 knowledge base 만을 구성하면 된다. 그러나, 이러한 개발기구들은 inference engine 이외에도 다양한 지원기능들을 제공한다. 이러한 지원기능의 하나로써, 대부분의 개발기구들은 knowledge base를 편집하기 위한 editor를 제공하는데, 간단한 경우에는 수동적인 작업으로써 knowledge base를 편집하기 위한 보통의 text editor를 제공한다. 그러나 좀더 정교한 경우에는, knowledge base의 특성에 맞게 여러 형태의 기능을 갖는 knowledge base editor를 제공한다.

예를 들자면, EMYCIN[6] 의 경우에는 automatic bookkeeping의 기능이 있는데, 이는 knowledge base의 지식을 변화시켰거나 새로운 지식을 삽입시킨 사람과 삽입한 일시에 관한 정보를 자동적으로 기록하여 추후에 참고자료로 삼게 한다. 이러한 기능은 여러 지식공학자가 공동으로 knowledge base를 구성할 때 유용하다.

또 어떤 knowledge base editor는 sysntax checking을 행하여 준다. 즉, knowledge base의 표현형태에 위배되는 입력에 대하여, 잘못을 지적하고 또 무엇이 잘못되었는지 등을 알려 준다. 이러한 기능은 차후에 잘못된 부분을 수정하는 것보다 시간적인 절약효과를 준다.

KuBIC[7] 의 경우에는, 새로이 추가되는 지식(이 경우에는 object 라고 불리운다)에 대하여, 이의 분류학적(taxonomic) 위치가 현존하는 knowledge base의 어디에 해당하는지를 찾아내어 적합한 곳에 삽입시켜 준다.

TDE[8] 는 TEST(trouble shooting expert system tool)[9] 에 관계된 knowledge base editor로써, 특별히 window에 의한 편집기능을 제공하고 있다. TDE에서는 frames들의 semantic network으로써 주요개념을 표시하는데, 세 가지 화면에 의하여 knowledge base의 편집을 도와 준다. 즉, 첫번째 화면은 knowledge base에 저장된 개념들간의 계층적 관계나 인과관계를 보여 주고, 두번째 화면은 특별히 관심이 있는 개념과 관계된 여러 법칙 및 프로시듀어들을, 세번째 화면은 각 개념의 속성을 보여준다. 이러한 화면들은 knowledge base와 직접 대응이 되어 있어, 화면에서의 변화는 곧 knowledge base의 변화를 의미한다. 이 밖에도 TDE에는 검색을 위한 기능들이 있는데, 예를 들자면 다음과 같은 것들이다.

(1) String search : Knowledge base에 저장된 사물(object)에 대하여 이의 이름을 이용하여 검색을 행한다. 이 경우, 사용자는 찾고자하는 사물의 종류를 규정할 수 있도록 한다.

(2) Network browsing : 이는 knowledge base 내의 사물을 찾기 위한 것으로써, 현재의 사물에 대하여 제시된 관계를 갖는 사람들을 검색하도록 한다.

(3) Pattern matching : 이는 search template에 의하여 규정된 속성값을 찾도록 한다.

(4) 이 밖에도 knowledge base에는 inverse link가 구비되어 있어서, focus의 이동이 쉽고, 또 이를 이용하여 사용자에게 knowledge base를 변경하였을 때 영향을 받는 부분을 알려 준다.

이제 또 다른 expert system 개발기구의 하나인 GURU[10] 에서 제공하는 knowledge base editor에 대하여 살펴 보자. GURU 에서는 rule에 의하여 지식을 표현하여 knowledge base를 구성하는데, rule을 입력 시키는 방법으로써, text editor에 의한 방법과 사용자와의 interactive한 대화를 이용하는 두 가지 방법을 제공한다. Text editor에 의한 방법은, rule의 형태대로 text file을 생성하는 고전적인 방법이다. 다른 interactive한 방법은, window에 의하여 화면에 rule의 형태를 보여주고, 커서(cursor)를 움직여서 편집하고자 하는 rule의 요소를 바꾸어 지시하면서 입력시키는 방법인데, 하나의 rule의 내용을 일시에 파악할 수도 있고, 또 제공되는 다양한 기능에 의하여 여러 개의 rule을 효과적으로 편집할 수 있는 편의를 제공한다. 여기에서 제공되는 기능들중 중요한 것은 다음과 같다.

(1) CREATE : 새로운 rule을 생성하여 knowledge base에 첨부시킨다.

(2) BROWSE/EDIT : 전체 rule 들을 살펴 보면서, 필요에 따라 rule을 수정한다.

(3) LOOKUP/EDIT : Rule의 이름에 의한 탐색을 행한다.

이러한 기능을 실현하기 위하여, GURU는 그림 2와 같은 화면을 제공한다. 즉, 하나의 rule과 관련된 여러 요소들을 window 별로 분할하여 보여 주므로, 지식공학자는 rule에 대한 내용을 쉽게 파악할 수 있다.

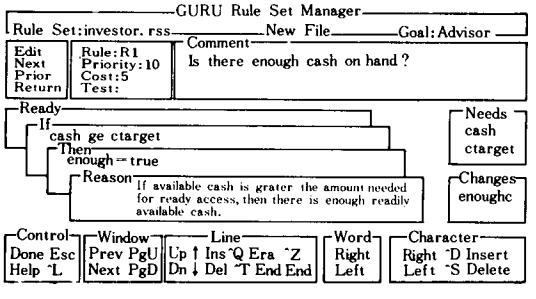

그림 2. GURU에서의 화면예

CREAT의 경우에는 Ready, If, Then, Reason등의 window가 비어있어 이를 채우게 된다. BROWSE/EDIT 기능은 Next와 Prior에 의하여 수행되는데, Next를 선택하면 다음 rule에 대한 화면이 제공되어 이를 편집할 수 있고, Prior는 앞의 rule에 대한 화면이 제공된다. BROWSE/EDIT는 특정한 스트링을 포함한 rule을 찾는 기능도 제공한다. 이 때는 찾는 스트링을 지닌 rule을 화면에 보인다. LOOKUP/EDIT 기능은 rule의 이름에 의하여 rule을 선택하는 방법을 제공한다.

이러한 window에 의한 방법으로 rule을 편집하는 작업이 끝나면, GURU는 지식공학자가 text editor를 이용하여 입력시킨 결과와 같은 text file의 형태로 변환하여 저장한다.

Ⅲ. 지식추출(Knowledge Extraction)

앞장에서 서술한 knowledge base editor는 knowledge base를 편집하는 지식공학자에게 편의를 제공하는 기능이며, 따라서 사용대상자는 지식공학자가 되며, 이 editor에는 별도의 학습기법은 사용되지 않는다. 그러나, 여기에서 서술할 지식추출은, 보통은 지식공학자의 도움없이 문제분야의 전문가가 직접 knowledge base를 구성하는 경우에 사용할 수 있는 것으로써, 사용대상자는 문제분야의 전문가가 되는 것이 보통이다.

이러한 지식추출의 기능을 제공하는 시스템은 크게 두가지로 구분할 수 있다. 하나는, 아무런 knowledge base의 내용이 없는 상태에서 사용자와의 대화를 통하여 전체 knowledge base를 구성하는 시스템이고, 다른 하나는 기존의 knowledge base에 대하여 추가적으로 지식을 첨부시키는 시스템이다. 후자의 경우는 Ⅳ장에서 논의할 지식의 개선과도 공통되는 성격이 있으나, 여기에서는 개선의 의미가 아닌, 지식의 입력에 대한 경우를 다룬다.

1. 새로운 Knowledge Base의 구성

Knowledge base의 내용은 문제분야에 관한 여러 가지 개념과 문제분야에서 적용되는 여러 법칙들로 구성된다. 따라서 우선적으로 지식추출을 위한 시스템이 행하여야 할 업무는, 문제분야의 여러 개념을 얻어야 하는 것이다. 그런 뒤에, 얻어진 개념들을 기본으로하여, 여러 훈련예(training instances) 들로부터(경우에 따라서는 사용자의 협조도 포함)문제분야에 관한 여러 법칙을 구성하게 되는 것이다.

우선적으로 비교적 단순한 지식추출시스템인 autointelligence[11]에 대하여 살펴보자. 예로써, 환자의 증상으로부터 진단을 행하는 의료용 expert system을 위한 knowledge base를 구성하는 경우를 생각하여 보자. 이 경우 expert system은 다음의 4가지 진단중의 하나를 최종적으로 결론으로 제시하게 된다.

Simple headache
Cold/Flu
Allergy related
Serious Causes

우선, autointelligence는 각 경우에 해당되는 환자의 예를 제시하도록 한다. 예로써 allergy와 관계된 환자의 예는 다음과 같다.(밑줄친 부분은 사용자의 입력이다.)

Allergy related : John Smith
Mary Johnes
David Johnson

Autointelligence는 적당한 3가지 진단을 각각의 환자예와 함께 사용자에게 제시하여 이 들간의 차이를 지적하도록 한다. (3가지씩 비교하는 이유는 심리학적으로 인간은 3개를 비교하는데 탁월하기 때문이다.)

Please think of :

Cold/Flu Sara Davidson
Allergy related John Smith
Simple headache Jane Simpson

Is ONE of them somehow different
from the other two?

만일 사용자가 John Smith를 선택하였다면, autointelligence는 다음의 질문을 제시한다.

What is a trait Which
Characterizes this difference ?
Trait Name : History of Allergy
What is the opposite of History of Allergy ?
Opposite Trait : No history of Allergy

이렇게하여, autointelligence는 history of allergy가 최종진단을 결정하는데 유용한 개념의 하나임을 알게 된다. 그러나 allergy의 경력(history)만이 allergy와 관계된 경우는 아닐 수도 있다. 따라서 같은 진단들에 대한 다른 예들에 대하여서도 같은 질문을 반복한다.

Please think of :

Cold/Flu Sara Davidson
Allergy related Mary Jones
Simple headache Julie Peters

Is ONE of them somehow different
from the other two?

만일 사용자가 Mary Jones를 택하였다면, autointelligence는 다음의 질문을 행한다.

What is a trait which
characterizes this difference?
Trait Name : Young
What is the opposite of Young?
Opposite Trait : Old

(이 경우는 아마도 Mary Jones가 최초로 allergy 현상을 보인 경우에 해당될 것이다.) 이와 유사한 질문을 반복함으로써, autointelligence는 문제분야에서 사용되는 여러 개념을 얻게 된다. 여기까지의 대화에서는 autointelligence는 History of Allergy/No History of Allergy와 Young/Old의 두개의 'trait'을 얻었다. 이제 autointelligence는 각 trait에 대하여 다음과 같은 질문을 행한다.

Please consider
the opposite traits :
Young
Old
Would you like to :
Use these traits
Use ar attribute

이 경우 사용자가 'Use these traits'을 선택하면 autointelligence는 향후 이 trait을 그대로 개념으로 사용할 것이고, 지금과 같이 사용자가 'Use an attribute'를 선택하면, 다음과 같은 질문을 행하므로써, 이 trait들은 대표하는 높은 차원의 개념인 'attribute'를 묻게 된다.

Please enter an attribute name :
Attribute Name : Age

이제, autointelligence는 'Young'과 'Old'는 'Age'라는 개념의 한 경우에 해당되는 것을 알게 되고, Age가 취할 수 있는 값을 묻게 된다.

Is this attribute :
Symbolic
Numeric

사용자가 'Numeric' 을 선택하였으므로, autointelligence는 Age가 취할수 있는 값은 수치임을 알게 된다. (이 경우, 이때까지 사용하였던 Young/Old 의 trait 은 Age라는 개념을 이끌어 내기 위한 것이었을 뿐 향후에는 별 용도가 없게 된다. 그러나 만일 이전의 질문에서 'Use these trait' 을 선택하였었다면, Young/Old trait은 계속 유용하게 된다.) 만일 사용자가 'Symbolic' 을 택하였다면, Age가 취할 수 있는 값을 다시 물었을런지도 모른다.

비슷하게, autointelligence는 심볼값을 갖는 attribute인 'Sex,' 'Temperature,' 'History of Allergy'를 얻게되고, 반면 'Sneezing/Not-Sneezing'은 trait 자체를 사용할 것임을 알게 된다. 이제 autointelligence는 각 환자예에 대하여, 모든 attribute나 trait에 대한 값을 요청하게 된다.

```
Please enter values for :
                John Smith
Age : 24
Sex : Male
Temperature : Normal
Sneezing : 80
History of Allergy : Yes
```

여기에서 attribute인 것들은 취할 수 있는 값중 하나를 얻게 되지만, 'Sneezing' 과 같은 trait 인 경우에는 0～100 (100은 100% 확실함) 사이의 확신값 (certainty value) 을 갖게 된다. 각 예에 대하여 이러한 값을 얻은 뒤에, 각 예에 대하여 전문가가 생각하는 최종 진단의 확신값을 요청하게 된다.

```
Please think of : John Smith
Please enter a number
Between 0 and 100 as
Your Confidence in :
      Simple headache : 20
      Allergy related : 80
      Cold/Flu : 50
      Serious Causes : 10
100 means total confidence
0 means no conficlence at all
```

이 확신값은 John Smith가 각 최종진단에 대하여 얼마나 적합한가를 반영하는 값이다. 이러한 최종진단에 대한 확신값은 Mary Jones, Sara Davidson 등의 모든 예에 대하여 얻게되고, 각 환자예에 대한 특성들의 값과 진단의 확신값을 이용하여 새로운 환자의 특성값들에 대하여 적절한 진단을 제시하는 법칙을 만들게 된다. 예를 들자면 autointelligence는 다음과 같은 법칙을 제시한다.

```
Diagnosis is Allergy related
If
   History of Allergy=Yes and
   Sneezing>60 and
   Temperature=Normal ;
```

이와 유사한 업무를 행하는 시스템으로는 ROGET[12]이 있는데, ROGET은 개발할 expert system의 개념구조 (conceptual structure) 를 얻도록 도와준다. ROGET에서의 개념구조는 후에 행할 문제분야의 추론종류와 이를 지원하는 사실을 나타낸다. ROGET은 자체에 문제풀이 업무의 종류(진단, 처방, 설계 등)에 따라 goal, subgoal, evident 등의 속성이 규정되어 있는데, 사용자가 선택한 업무의 종류에 따라, 자체에 설정된 goal, subgoal 등에 대응되는 문제분야의 특정한 개념을 제시하도록 한다.

또, SALT[13] 도 비슷한 업무를 행하는데, 보통의 시스템은 주로 분류문제 (classification problem) 를 다루는 knowledge base의 구성을 위한 것이 대부분이지만, SALT는 구성된 해 (solution) 에 의하여 문제를 풀이하는 시스템을 위한 것이다. SALT는 사용자와 메뉴를 이용한 일종의 구조적인 언어 (structured language) 를 통하여 대화한다.

KNACK[14] 는 제한된 분야에서 사용되는데, 특별히 전자기계적 (electromechanical) 시스템의 설계 및 평가를 행하는 expert system을 위한 지식추출 시스템이고, KNACK 자체에는 이러한 전자기계적 시스템의 평가를 위한 일반지식이 포함되어 있다. 이러한 지식을 이용하여 KNACK는 전문가(사용자) 와 대화를 통하여 문제분야의 개념적 모델을 구성한다. 즉, 전문가는 그의 지식을 샘플레포트의 형태로 제시하고, KNACK는 이를 작은 조각으로 나눈뒤, 다른 응용분야의 경우에 이 조각들을 어떻게 규합시킬 것인지를 전문가에게 문의하여, 이 조각들과 또 이들을 규합시키는 방법을 일반화시키고, 몇가지 예를 제시하여, 전문가가 이를 수정하게 한다. 이 밖에도 KNACK 다음 장에서 논의할 지식의 불일치, 불완전등을 검토한다.

2. 추가적 지식의 첨부

추가적 지식을 삽입할 때는 이미 구축된 knowledge base와 비교함으로써, 보다 다양한 기능을 제공할

수 있다. 다음은 TEIRESIAS[15] 에서 제공하는 기능을 부분적으로 발췌한 것이다.

; 사용자는 새로운 rule을 삽입 시키겠다고 요청하였다.

The new rule will be called RULE 383
IF 1-THE PATIENT'S INFECTION IS PRIMARY BACTEREMIA
and 2-THE SITE IS ONE OF THE STERILE SITES
and 3-
THEN 1-THERE IS EVIDENCE (0.8) THAT THE CATEGORY IS ENTEROBACTERIACEAE
and 2-

; TEIRESIAS는 사용자가 입력한 rule을 knowledge base의 표현형태로 바꾼뒤, 이를 다시 영어로 바꾼다.

This is my understanding of your rule :
RULE 383
IF : 1-Primary-bacteremia is an associated infection for which therapy should cover, and
2-The infection with the organism was acquired while the patient was hospitalized, and
3-The culture was taken from a sterile source, and
4-The organism is not normally found at this site
THEN : There is strongly suggestive evidence (0.8) that tue category of the organism is enterobacteriaceae.
OK ? (yes or no)
++**N

; TEIRESIAS는 저장된 rule이 사용자가 의미한 것이었느냐를 묻는데, 이 경우 사용자는 문제가 있다고 생각한다.

Change premise ?
++**Y
What clause should be changed in premise ?
++** CHANGED 1
Suppose (1) were changed to read :
1 the infection is primary-bacteremia
Would that be OK ?
++**Y

; TEIRESIAS는 primary-bacteremia에 관하여 저장된 의미중 두번째 것으로 해석한 뒤 사용자에게 1의 새로운 형태를 보여주는데, 사용자는 여기에 만족하였다. 그러나 새로운 1에 대하여서는 2가 바뀌어야 할 것을 알고, 이를 사용자에게 묻는다.

Then you'll probably want to delete :
2 The infection with the organism was acquired while the patient was hospitalized
Correct ?
++**Y
Anything else ?
++**D4
Clause (S) 4 deleted
Anything else ?
++**C3
Suppose (3) were changed to read :
3 the site of the culture is one of : those sites that are normally sterile
Would that be OK ?
++**Y
⋮

; 이렇게 하여 일차적인 응답이 모두 끝나면 TEIRESIAS는 정돈된 rule을 보여준다.

The is my understanding of your rule ;
RULE 383
IF 1) The infection is primary-bacteremia, and
2) The site of the culture is one of : those sites that are normally steril
THEN : There is strongly suggestive evidence (0.8) that the category of the organism is entero bacteriaceae
OK? (yes or no)
++**Y

; 사용자는 최종적인 rule에 만족하였지만, TEIRESIAS는 knowledge base에 저장된 rule과 비교하였을 때, 현재의 rule에는 문제가 있음을 찾아낸다.

I hate to criticize, Dr. Davis, but did you know that most rules about what the category of a organism might be, that mentioned
the site of a culture
the infection

Also mention-

〔A〕-the portal of entry of a organism

Shall I try to write a clause to account for

〔A〕?

++**Y

How about

〔A〕 The portal of entry of the organism is 8i

OK ?

++**Y

⋮

이런 식으로 기존의 knowledge base에 저장된 지식의 내용 및 형태를 이용하여 새로이 삽입되는 지식의 내용 및 형태에 도움을 줄 수 있다. TEIRESIAS에 관한 자세한 내용은 〔16〕을 참조하기 바란다.

지식추출시스템은, 이것이 사용될 expert system과 연관을 지어서 설계하면, 제한된 용도를 갖기는 하지만 보다 쉽게 설계할 수 있다. 바꾸어 말하면, expert system의 구조를 설계할 때, 지식 추출시스템의 개발을 염두에 두는 것이다. 이러한 예로써, 의료진단 expert system MUM과 연관을 지어서, 지식추출시스템의 설계에 관한 연구로는 〔17〕을 참조하기 바란다.

Ⅳ. Knowledge Base의 개선(refinement)

초기의 knowledge base가 구성되면, knowledge base의 미비점을 보완하고 좀 더 효율적으로 다듬어야 하는데, 이 작업이 여기서 논의할 knowledgd base의 개선이다. 편의상 이 작업을 전문가나 지식공학자와의 대화 혹은 실험예에 의하여 개선시켜 나가는 동적개선(dynamic refinement)과 사용자의 개입없이 진행시켜 나가는 정적개선(static refinement)로 구분한다.

1. 정적개선(Static Refinement)

정적개선을 행하는 시스템으로는 우선 CHECK[18]을 생각할 수 있다. CHECK은 rule로써 구성된 knowledge base에서 다음과 같은 문제를 갖는 rule들을 찾아낸다.

1) 중복된 rule들 : 같은 조건에서 같은 결론을 내는 rule들.

예로써, 'p(x) →q(x)'와 'p(y) →q(y)'와 같은 형태

2) 모순된 rule들 : 같은 조건에서 수행되나, 모순된 결과를 낳는 rule들.

예로써, 'p(x) →q(x)'와 'p(x) →not q(x)'의 형태

3) 포함되는 rule들 : 두 rule이 같은 결과를 내지만, 한 rule의 조건이 다른 rule보다 추가적인 제약이 있는 rule.

예로써, '(p(x) ∧q(y)) →r(z)'는 'p(x) →r(z)'에 포함된다.

4) 순환되는 rule들 : rule들이 싸이클(cycle)을 이루면, 이 rule들은 순환한다고 한다.

예로써, 'p(x) →q(x)', 'q(x) →r(x)', 'r(x) →p(x)'는 순환하고, 이들은 시스템의 수행시에 무한한 루프(loop)에 빠지게 된다.

5) 빠진 rule들 : 어떤 object의 속성이 갖을 수 있는 값중 일부가 아무런 rule의 IF절의 의하여 카버(cover)되지 않을 경우, 이런 경우, 시스템은 결론을 못내거나, 잘못된 결론을 내는 수가 생긴다.

6) 도달할 수 없는 절들 : 역방향(backward) 추론에서는, rule의 THEN절은 목표절(goal clause)이나 다른 rule의 IF절과 매칭될 수 있어야 한다. 그렇지 않으면, 이 THEN절은 도달할 수 없다.

7) 막다른 절(deadend clauses) : 어떠한 goal이나 subgoal이 이루어지기 위하여서는, (1)목표절 (goal clause)의 속성을 사용자가 제공하거나 (2)목표절 이 rule들 중 적어도 하나의 THEN절과 매칭될 수 있어야 한다. 만일 이 두가지가 만족되지 않는다면, 이 목표절은 막다른 절이다.

CHECK는 dependency chart를 이용하여 이러한 문제를 찾아내어, 사용자에게 알려준다. CHECK에 이은 노력으로써, ARC[19]에서는 불필요한 IF 조건, 중복되는 rule chains, 포함되는 rule chains, 모순되는 rule chain등과 같이 CHECK로부터 확장된 문제점을 검출한다. 예로써, 중복된 rule들의 확장형태는, 'not p(x) ∧not q(x) →r(x)'와 'not〔p(y) ∨q(y)〕 →r(x)'의 형태이거나, 'A(x) →B(x)', 'B(x) →D(x)'와 'A(y) →C(y)', 'C(y) →D(y)'와 같은 chain을 포함한다. 이 밖에도 CHECK에서는 N개의 rule에 대하여 N(N−1)/2번의 rule의 비교횟수가 소요되지만, ARC에서는 inference graph에 의하여 이 보다 빠른 시간내에 같은 작업을 행할 수 있다. 비슷한 업무로써 〔20〕에서는 정당한 입력이지만 향후 모순된 결론을 유도할 가능성이 있는 것까지 검토하여 낸다.

역시 비슷한 정적개선을 행하는 중요한 시스템으로 MORE[21,22]가 있다. 이 역시 우선적인 knowledge

base를 구성한 뒤, 생성된 rule들을 강화시키는데, 다음과 같은 정책을 사용한다. (MORE는 진단업무를 행하는 시스템을 위한 것으로 hypothesis는 결과의 진단, symptom은 관측된 증상으로 간주하라)

1) Differentiation : 2개의 서로 다른 hypothesis가 초래하는 symptom들이 같으면, 이를 구분하도록 하여, 서로 다른 hypothesis는 서로 다른 symptom들을 갖도록 조정한다.

2) Path differentiation : 어떤 symptom S가 2개의 hypothesis H_1과 H_2로부터 기인한다면, 이 두 path중에서 겹치지 않는 곳에 중도 symptom을 찾도록 하여, 이 symptom의 weight를 높인다.

3) Path division : Hypothesis H에 직접 연결된 symptom S에 대하여 negative support value를 갖는 rule이 아무 것도 없으면, H로부터 S사이에 새로운 symptomatic event S_2를 삽입하도록 요청한다. 이렇게하여, S_2는 H에 의하여 기인하고, 또 이는 S를 초래한다. 이렇게하면 S_2가 찾아지지 않을 때는 H에 대하여 큰 부정적인 요소가 될 것이다.

4) Sympton distinction : 여러 hypothesis에서 기인하는 같은 symptom이라도 어느 hypothesis에서 기인하였느냐에 따라 symptom의 특별한 성격이 다를 수 있다. 이를 찾도록 한다.

5) Symptom conditionalization : 주어진 hypothesis에 대하여, 관련된 symptom이 일어나는데 영향을 주는 event를 찾도록 한다. 이 event가 있으면 주어진 symptom을 초래하는 hypothesis는 확신율이 높아지고, 없으면 낮아진다.

6) Test differentiation : Symptom이 생겼음을 찾아낼 수 있는 procedure나 관측장비를 결정하도록 함.

7) Test conditionalization : Test의 결과나 정확도의 확신에 영향을 주는 조건을 문의함.

8) Frequency-conditionalization : 예측되는 hypothesis의 존재에 영향을 주는 조건을 찾도록 함.

MORE는 정적인 개선으로써, knowledge base가 상기의 문제를 갖고 있으면, 이를 지적하여 준다. 그러나 실제의 경우, 상기의 대부분은 존재하지 않는 문제일 경우가 많다. 이런 경우, MORE는 사용자와의 대화에 불필요한 시간을 소비하게 된다. 이의 개선책으로 MOLE[23]에서는 differentiation등의 일부만 정적인 개선단계에서 행하고, 나머지는 동적인 개선단계로 넘겨, 전문가가 제공하여 주는 실험예에 대하여, 이 실험예의 답과 시스템의 답이 일치하지 않는 경우에만 관계된 부분의 개선을 행한다.

2. 동적개선(Dynamic Refinement)

동적개선의 대표적인 예로써는, 실험예에 대하여 전문가와의 대화를 통하여 knowledge base를 개선시켜 나가는 Teiresias[24]를 생각할 수 있다.

Teiresias는 사용자가 제공한 예에 대하여 처리를 한다. 만일 제시된 결론에 대하여 사용자가 동의하지 않게되면, 결론에 이르는 추론사슬을 교사가 추적하도록 한다. 만일 추론사슬이 잘못되었다면, 여기에 사용되었던 rule들 중에는 사용하지 말았어야 할 것이 있든지, 아니면 사용되지 않았던 rule들 중에서 사용되었어야 할 것이 있을 것이다. Teiresias는 상위 goal에서 세부 goal에 이르기까지 이 추론사슬을 따라가며 교사가 문제가 되는 부분을 찾아내도록 한다. 즉, 다음 중 하나가 생겨날 것이다 :

(1) 어떤 rule의 모든 조건이 만족되지만, 이 rule의 결론인 P가 실제로 이 조건들로부터 얻어지지 말았어야 하는 경우

(2) 유도되었어야 할 결론 P가 있는데, 이를 결론으로 저장하고 있는 모든 rule들은 각각의 조건중 일부가 만족되지 못하여 사용되지 못한 경우

첫번째 경우에는 rule을 특수화(specialize) 시켜야 한다. 즉, P를 결론으로 제시하였던 rule은 이 경우에 사용되지 못하도록 조건을 강화시켜야 한다. 두번째 경우에는 rule을 일반화 시켜야 하는데, 현재 상황에서 P를 결론으로 하는 새로운 rule을 첨가하거나 혹은 P를 결론으로 제시하는 rule의 조건을 약화시킨다. 비록 Teiresias는 이 두가지중 어느 것을 행하여야 하는지를 알려주지만, 어떤 특수화나 일반화가 적합한 지를 추정하여 주지는 않는다. 단지 교사가 제시하도록 요청할 뿐이다.

비슷한 방법으로 knowledge base를 개선시켜 나가는 것으로는 LAS[25]가 있는데, Teiresias에서는 추론이 실패한 요인과 이에 대한 수정을 사용자가 행하여야 하나, LAS에서는 한 걸음 더 나아가 이러한 실패요인에 대한 가능한 이유를 dependency network에 의하여 제시하여 준다.

이러한 Teiresias나 LAS에 의한 방법의 장점으로는, knowledge base의 개선작업이, 통상의 expert system의 사용시에 일어나므로, 전문가들이 별도로 시간을 투자할 필요가 없고, 또 특별한 문제를 다룰 때 생겨나므로, knowledge base의 작은 부분에 초점을 맞출 수 있다는 것이다.

Seek[26,27,28]는 전문가가 제공한 test case (실험예)들에 의하여 knowledge base의 개선을 행하는 프로

그램이다. 각 test case는 환자의 증상들과 이에 대한 전문가의 최종진단을 포함하고 있다. 우선적으로, 저장된 모든 test case들에 대하여 knowledge base의 수행결과를 그림 3 과 같은 표로 제시하여 준다.

	True Positives	False Positives
Mixed Connective Tissue Disease	9/33 (27%)	0(00%)
Rheumatoid Arthritis	42/42(100%)	10(13%)
Systemic Lupus Erythematosus	12/18 (67%)	4 (04%)
Progressive Systemic Sclerosis	22/23 (96%)	5(04%)
Polymyositis	4/ 5 (80%)	1(01%)
Total	89/121(74%)	

그림 3. 전체 수행 결과의 요약

그림 3 에서 true positive는, 예로써 SLE(systemic lupus erythematosus)의 경우, 전체 18개의 test case들 중 12개가 정확하게 진단되었음을 의미하고, false positive는 다른 진단을 내려야 하는데 시스템이 SLE로 잘못 진단내린 것이 4 번임을 나타낸다. 그림 3 의 결과를 보면, RA(rhe umatoid arthritis)의 경우, 모든 test cases에 대하여 올바른 진단을 내렸지만, RA로 잘못 진단을 한 경우가 10회나 되므로, RA에 관계된 rule들은 아마도 보다 특수화를 시켜야 할 것이다.

Seek는 각 rule에 대하여서도 test cases에 의한 수행결과를 그림 4 와 같이 요약하여 준다. 이는 rule 72가 43개의 test cases에서 수행되었는데, 이 중 13 case들에 대하여서는 전문가의 결론과 일치하였고, 7 case들에 대하여서는 전문가의 결론과는 틀렸음을 보여 준다.

Rule 72: :2 or more Majors for RA
2 or more Minors for RA
No Exclusion for RA
→ Probable Rheumatoid arthritis

43 Cases: in which this rule was satisfied.

13 Cases :in which the greatest certainty in a conclusion was obtained by this rule and it matched the expert's conclusion.

7 Cases: in which the greatest certainty in a conclusion was obtained by this rule and it did not match the expert's conclusion.

그림 4. Rule의 수행결과

Test case들에 의한 수행결과는 단순한 암시일 뿐이고, 본격적으로 rule에 대한 개선작업은 잘못 진단된 case를 분석함으로써 가능하여 진다. 그림 5 는 하나의 잘못 진단된 case에 대한 분석결과를 보여 준다. 즉, 전문가는 PSS(progressive systemic sclerosis)라고 진단하였는데, 시스템은 RA로 진단하였다. Seek에서는 3 가지의 확신값을 사용하는데, 가장 높은 것부터 나열하면, definite, probable, possible의 3 가지이다. 그림 5 에 의하면, 이 case에서 전문가와 같은 결론을 내리는 rule중 가장 근접하였던 rule은 rule 111이다. 실제로 rule 111의 경우, 모든 조건들은 만족되었지만, 결론의 확신값이 possible로써, 이 경우에 같이 만족되었던 rule 72(오진이었음)의 결론 확신값인 probable보다 낮았기 때문에 rule 72 가 시스템의 결론으로 채택되었던 것이다. 혹은 rule 112도 이 case와 관계가 있는데, rule 112는 조건이 모두 만족되지는 못하였지만, 이 rule이 만족되었다면, 제시되는 결론은 rule 72의 결론의 확신값보다 낮지 않으므로, 전문가와 같은 진단이 내려졌을 것이다. 따라서 그림 5 의 결과로부터 우리는 rule 72의 결론의 확신값을 약화시키든지(특수화), 아니면 rule 112를 보다 일반화시키면, 이 case에 올바른 진단을 하여 줄 수 있음을 알 수 있다.

이 밖에도 Seek는 그림 6 과 같이 동일한 결론(진단)

1. *Expert's* conclusion : Progressive systemic sclerosis
2. *Model's* conclusion : Probable rheumatoid arthritis

Strongest satisfisd rule for the expert's conclusion:

rule 111: 1 or more majors for pss (1 majors satisfied)
1 or more minors for pss (3 minors satisfied)
→ Possible progressive systemic sclerosis

Rule for the model's conclusion:

rule 72: 2 or more majors for ra (2 majors satisfied)
2 or more minors for ra (3 minors satisfied)
no exclusion for ra (satisfied)
→ Probable rheumatoid arthritis

Rules for expert's conclusion wioth potential weight greater than or equal to the model's conclusion:

rule 112: requirement 1 for probable pss (not set)
no exclusion for probable pss (satisfied)
→ Probable progressive systemic sclerosis

그림 5. 잘못 진단된 case의 분석

을 내리는 test case들에 대한 통계도 분석한다. 이는 앞서 그림 5와 같은 단일 case에 대한 분석들을 규합시킨 것인데, 예로써 rule 55는 mixed connective tissue disease에 관하여 일반화 시켜야 할 경우가 7 cases, 특수화 시켜야 할 경우가 없었음을 말하여 준다. rule56 같은 것은 8 case들에 대하여 일반화 시켜야함을 보여준다.

Mixed Connective Tissue Disease

Rule	Certainty	Generalization	Specialization
54.	Possible	2	0
55.	Possible	7	0
56.	Possible	8	0
57.	Probable	2	0
58.	Probable	2	0

그림 6. 같은 진단의 test case들의 통계

V. 결 론

이상으로 우리는 knowledse based system에서 핵심이 되는 knowledge base의 구축 및 개선에 관한 최근의 연구동향 및 기술에 대하여 살펴 보았다.

Knowledge base editor는 학습부분을 지니지는 않은 것으로써, 현재의 text editor가 제공하지 못하는 기능을 지원하여 줌으로써, 지식공학자가 보다 편리하고 체계적으로 knowledge base를 편집할 수 있다. 아직 많은 expert system 개발기구가 기존의 text editor를 빌어와서 사용하고 있으므로, 이 부분의 연구는 대상이 되는 개발기구에 맞추어서 진행되어야 할 것이다.

지식추출은 knowledge base가 없는 상태에서 새로운 개념을 포함한 지식을 획득하는 것과 기존의 knowledge base의 내용을 이용하여 새로운 지식을 추가시키는 두가지로 구분하였는데, 전자의 연구는 비교적 expert system 개발기구에 대하여 독립적으로 진행시킬 수 있으므로, 상업용 시스템의 출현이 시작되고 있다.

마지막으로 제시한 지식의 개선은, test case들이나 전문가와의 대화에 의한 동적개선과 자체의 knowledge base만을 갖고 knowledge base를 개선시키는 정적개선의 두가지로 구분하였는데, 후자의 연구가 비교적 규모가 작으며, 이에 대한 많은 알고리듬들이 현재 제시되고 있다.

향후 규모가 커지고 응용분야가 다양하여지는 expert system의 추세로 볼 때, 이러한 knowledge base의 구성과 개선에 관한 연구는 보다 집중적인 주목을 받아야 할 것이며, 여기에서는 언급하지 management system에 관한 연구도 병행되어야 할 것이다.

參 考 文 獻

[1] Donald A. Waterman, *A Guide to Expert Systems,* Addison-Wesley Publishing Company, U.S.A., 1986.

[2] Phillip Klahr and Donald A. Waterman (eds), *Expert Systems,* Addison-Wesley Publishing Company, U.S.A., 1986.

[3] E. Feigenbaum, "Themes and Case Studies of Knowledge Engineering," in Expert Systems in the Micro-Electronic Age, D. Michie Ed., Edinburgh University Press, 1979.

[4] James F. Brule and F.A. Blount, "Problems of Knowledge Acquisition from Human Experts: A Psychological Perspective," The Second Annual Artificial Intelligence & Advanced Computer Technology Conference, pp. 62-68, 1986.

[5] Anna Hart, Knowledge Acquisition for Expert System, Kogan Page Ltd., 120 Pentonville Road, London N19JN, 1986.

[6] W. Van Melle, E.H. Shortliffe and B.G. Buchmanan, "EMYCIN: A Knowledge Engineer's Tool for Constructing Rule-Based Expert systems, " in B. Buchanan and E. Shortliffe (eds.), Rule-based Expert Systems, Addison-Wesley Publishing Company, U.S..A., 1984.

[7] T. Finin and D. Silverman, "Interactive Classification: A technique for Building and Maintaining Knowledge Bases, IEEE Proceedings Workshop on Principles of Knowledge-Based Systems, IEEE Computer Society Press, pp. 107-114. 1984.

[8] G.S. Kahh, "From Application Shell to Knowledge Acquisition System," IJCAI-87 pp. 355-358.

[9] G.S. Kahn, Al Kepner, and Jeff Pepper, "Test: A Model-Driven Application Shell," AAAI-98, pp. 814-818, 1987.

[10] GURU Reference Manual, Micro Data Base Systems, Inc., P.O. Box 248, Lafayelle, Indiana 47902, U.S.A., 1985.

[11] K. Parsaye and S. Murphree, *Automating the Knowledge Acquisition Process,* Intelligence Ware, Inc., 9800 S. Sepulreda Blrd. Suite 730, LA, LA 90045, U.S.A., 1987.

[12] James S. Bennett, "ROGET: Acquiring the conceptual structure of a diagnostic expert system," IEEE Proceedings Workshop on Principles of Knowledge-Based Systems, IEEE Computer Society Press, pp. 83-88 1984.

[13] S. Marcus, J. McDermott, and T. Wang, "Knowledge Acquisition for Construction Systems, IJCAI-85, pp. 637-639, 1985.

[14] G. Klinker, C. Boyd, S. Genetet, and J. McDermott, "A KNACK for Knowledge Acquisition," AAAI-87, pp. 488-493, 1987.

[15] R. Davis, Applications of Meta Level Knowledge to the Construction, Maintenance and Use of Large Knowledge Bases, Report STAN-CS-76-552, Stanford AI Laboratory, Stanford University, Stanford, Calif., July 1976.

[16] R. Davis, "Interactive transfer of experties: Acquisition of new inference rules," in Artificial Intelligence 12, pp. 121-157, 1979.

[17] T. Gruker and P. Cohen, "Principles of design for knowledge acquisition," Proceedings the Third Conference on Artificial Intelligence Applications, pp. 9-15, 1987.

[18] T.A. Nguyen, "Verifying consistency of production systems," Proceedings the Third Conference on Artificial Intelligence Applications, pp. 4-8, 1987.

[10] T.A. Nguyen, W.A. Perkins, T.J. Laffey, and D. Pecora, "CHECKING an expert systems knowledge base for consistency and completeness," IJCAI-85, pp. 375-378, 1985.

[20] A. Ginsberg, "A new approach to checking knowledge bases for inconsistency and redundancy," Third Annual Expert Systems in Gorernmant Conference, pp. 102-111, 1987.

[21] G. Kahn, S. Nowlan, and J. McDermott, "A foundation F.R knowledge acquisition," IEEE Proceedings Workshop on Principles of Knowledge-Based Systems, IEEE Computer Society Press, pp. 89-96, 1984.

[22] G. Kahn, S. Nowlan, and J. McDermott, "MODE: An Intelligent Knowledge Acquisition Tool," IJCAI-85, pp. 581-584, 1985.

[23] L. Eshelman and J. McDermott, "MOLE: A knowledge acquisition tool that use its head," AAAI-86, pp. 950-955, 1986.

[24] R. Davis, "Teiresias: Applications of meta-level knowledge," in R. Davis and D.B. Lenat (Eds.), Knowledge-Based Systems in Artificial Intelligence, McGraw-Hill, pp. 227-490, 1982.

[25] R. Smith and H. Winston, "Representation and use of explicit justification for knowledge base refinement," IJCAI-85, pp. 673-680, 1985.

[26] P.G. Politakis, *Empirical Analysis for Expert Systems,* Pitman Publishing Inc., MA 02050, U.S.A., 1985.

[27] P. Politakis and S. Weiss, "Using empirical analysis to refine expert system knowledge bases," Artificial Intelligence 22, pp. 23-48. 1984.

[28] A. Ginsberg and S. Weiss, "SEEK I: A generalized approach to automatic knowledge base refinement," IJCAI-85, pp. 367-374.

♣ 用 語 解 說 ♣

Hierarchical model (계층모델)

현실 세계의 실체간의 관련을 방향을 방향을 갖는 나무(有向木)의 형태로 나타낸 데이타 모델이다. 나무 구조는 파일에 의한 실현이 쉽기 때문에 모델에 적합한 질문처리는 매우 빠른 속도로 처리할 수 있는 특색이 있다. 그러나 현실 세계의 데이타를 나무의 형으로 표현하는 것이 곤란한 경우도 있다.