2-패스 색인 기법과 규칙 기반 질의 처리 기법을 이용한 고속, 고성능 질의 응답 시스템

(A Fast and Powerful Question-answering System using 2-pass Indexing and Rule-based Query Processing Method)

김 학 수 † 서 정 연 ††
(Harksoo Kim) (Jungyun Seo)

요 약 본 논문은 2-패스 점수 부여 방법에 기초한 정답 후보 색인기를 이용하여 고속, 고정밀의 질의 응답을 실현하는 한국어 질의 응답 시스템을 제안한다. 제안한 정답 후보 색인기의 색인 과정은 다음과 같다. 먼저, 대상 문서에 포함된 모든 정답 후보들을 추출한다. 그리고, 2-패스 점수 부여 방법을 이용하여 각 정답 후보와 밀접하게 연관된 주변 내용어들에게 점수를 부여한다. 마지막으로 데이터베이스에 각 정답과 점수가 부여된 내용어들을 역파일 형태로 저장한다. 사용자의 질의에 포함된 의도(질의 유형)을 파악하기 위해서는 수동으로 구축된 lexico-syntactic 패턴을 이용한다. 이러한 색인 방법과 질의 처리 방법을 이용하여, 제안된 질의 응답 시스템은 빠른 응답 시간을 보장하고 정확률을 향상시킨다.

키워드 : 정답 후보 색인기, 2-패스 점수 부여 방법, 규칙 기반 질의 처리, 질의 응답 시스템,

Abstract We propose a fast and powerful Question-answering (QA) system in Korean, which uses a predictive answer indexer based on 2-pass scoring method. The indexing process is as follows. The predictive answer indexer first extracts all answer candidates in a document. Then, using 2-pass scoring method, it gives scores to the adjacent content words that are closely related with each answer candidate. Next, it stores the weighted content words with each candidate into a database. Using this technique, along with a complementary analysis of questions which is based on lexico-syntactic pattern matching method, the proposed QA system saves response time and enhances the precision.

Key words : predictive answer indexer, 2-pass scoring method, rule-based query processing, question-answering system

1. 서 론

기존의 정보검색(information retrieval, IR)은 사용자의 질문에 대한 응답으로 대량의 문서를 검색하고 순위화하는데 초점을 맞추어 왔다. 그러나, 많은 사용자들은 명확한 의도를 가지고 질문을 하며, 정보 검색 시스템이 대량의 문서를 찾아주기 보다는 정답들을 곧바로 찾아 제시해 주기를 바란다[1]. 이러한 요구를 만족시키기 위하여 질의 응답(question answering, QA)이라는 개념이 출현했으며, 많은 연구들이 AAAI[2]와 TREC[3]을 중심으로 수행되어 왔다. 질의 응답 시스템은 대용량의 문서를 검색하고, 검색된 텍스트로부터 부적당한 구(phrase)나 문장을 제거한다. 이러한 필터링(filtering) 과정 덕분에 사용자는 검색된 문서를 모두 읽지 않고도 빠른 시간 내에 자신이 얻고자 하는 정답에 접근할 수 있다. 그러나 지금까지의 질의 응답 시스템에 대한 연구는 WWW(world wide web)와 같은 실제 필드(field)에서 나타날 수 있는 다음과 같은 문제를 간과해 왔다.

사용자들은 가능한한 빨리 정답을 찾기를원한다. 만약 질의 응답 시스템이 수 초 내에 응답이 없다면 사용자들은 해당 시스템의 유용성에 의심을 품을 것이다.

- 사용자들은 다양한 구문 형태(syntactic form)를 이용하여 자신의 의도를 표현한다. 이러한 이유로 질의 응답 시스템이 응용 영역에 상관없이 항상 좋은 성능을 보인다는 것은 매우 어려운 일이다. 즉, 실용적인 질

† 비 회 원 : 다이퀘스트 연구소 선임연구원
hskim@diquest.com
†† 종신회원 : 서강대학교 컴퓨터학과 교수
seojy@ccs.sogang.ac.kr
논문접수 : 2002년 5월 27일
심사완료 : 2002년 9월 3일

의 응답 시스템을 구현하기 위해서는 쉽게 응용 영역을 바꿀 수 있는 방법이 마련되어야 한다.

- 질의 응답 시스템이 사용자의 모든 질문에 답변을 제시할 수는 없다. 미리 정의된 정답 범주(answer category)에 속한 질문에 대해서만 답변을 제시할 수 있다. 그러므로, 실용적인 질의 응답 시스템은 응용 영역에 따라 변하는 범주를 쉽게 할당하거나 삭제할 수 있어야 한다.

이러한 문제를 해결하기 위해서 본 논문에서는 정답 후보 색인기를 이용한 실용적인 한국어 질의 응답 시스템(이하 MAYA)을 제안한다. MAYA는 실시간 응답(real-time response)과 영역 이식성(domain portability)과 같은 실질적인 문제를 해결하는데 초점을 둔 시스템이다. MAYA는 영역 사전(domain dictionary)과 규칙을 추가함으로써 쉽게 새로운 정답 범주를 할당하거나 삭제할 수 있다. 또한, 검색 엔진과 완전히 분리된 고정밀의 정답 후보 색인기를 가지고 있기 때문에 검색 시에 신속하게 답변을 제시할 수 있다.

본 논문의 구성은 다음과 같다. 먼저, 2장에서 질의 응답 시스템에 대한 기존의 연구를 살펴보고, 3장에서는 실용적인 질의 응답 시스템을 구성하기 위한 효과적인 정답 후보 색인 방법과 질의처리 방법을 제안한다. 4장에서 제안된 시스템의 유용성을 평가하기 위한 실험을 한 후, 5장에서 결론을 맺는다.

2. 관련 연구

질의 응답 방법론은 크게 텍스트 조각 추출 방법(text-snippet extraction method)과 명사구 추출 방법(noun-phrase extraction method)으로 나뉘어 진다[4]. 텍스트 조각 추출 방법은 질문에 대한 정답을 포함하고 있을 것 같은 텍스트의 단락이나 문장 또는 문장의 일부를 추출하는 것으로 지난 TREC QA 부문에 참가한 시스템들[5, 6]이 TREC의 평가 기준에 따라 일반적으로 사용한 방법이다. 명사구 추출 방법은 한정된 클래스(closed-class)에 속한 사용자 질문에 대해서 구체적인 정답구(answer phrase)을 찾아주는 것이다. 이 방법은 일반적으로 단답형 정답(noun-phrase answer)만을 찾아서 제시할 수 있다는 단점을 가지고 있다.

ExraAns[7]는 텍스트 조각 추출 방법을 사용하는 대표적인 질의 응답 시스템으로 정답을 포함하는 문서 내의 절이나 구를 찾아 준다. 그러나, ExtraAns는 제한된 영역의 구문적(syntactic), 의미적(semantic) 정보를 사용하기 때문에 응용 영역을 바꾸기가 쉽지 않다. FALCON[8]도 역시 대표적인 텍스트 조각 추출 시스템 중에 하나이다. 그 시스템은 구문, 의미, 화용(pragmatic) 지식 등을 이용하여 매우 정확하게 정답을 추천해 준다. FALCON이 비록 높은 정확률을 보이지만 의미망(semantic net)과 같은 영역 의존적 지식을 구축하는 것이 현실적으로 매우 어렵기 때문에 실용적인 질의 응답 시스템으로는 적합하지 않다.

MURAX[9]는 대표적인 명사구 추출 시스템으로 품사 태거(POS tagger), lexico-syntactic 패턴 매칭(pattern matching)을 위한 유한 상태 인식기(finite-state recognizer)와 같은 비교적 저급의 언어 지식(shallow linguistic knowledge)을 이용하여 정답을 추천한다. 유한 상태 인식기는 사용자의 질의 의도를 파악하고 올바른 정답 후보를 결정하는데 사용된다. TREC에 참가한 몇몇 시스템들은 사용자의 질의 의도를 파악하기 위해서 MURAX가 사용한 것과 비슷한 유한 상태 인식기를 사용한다[4]. 그러나, 이러한 시스템들은 검색 시에 정답 후보들을 찾아서 점수를 부여하고 상위 몇 개를 추천하기 때문에 응답 시간이 매우 길다는 단점이 있다. 이런 단점은 극복하기 위해서 GuruQA[10, 11]는 검색 전에 미리 정답 후보들을 찾아 색인하는 방법(predictive annotation)[10, 11, 12]을 사용한다. 비록 GuruQA가 빠른 시간에 정답을 제시하고 좋은 성능을 보이지만 여러 문서에 걸쳐서 관찰될 수 있는 유용한 정보들을 이용하지 못한다. 즉, GuruQA는 색인의 범위를 최소 한 문장에서 최대 한 문서 내로 한정을 하기 때문에 동일한 정답이 여러 문서에서 나타났을 경우에 각각의 문서에 존재하는 정보를 서로 공유하지 못한다.

3. MAYA의 접근 방법

MAYA는 일반적인 정보검색 시스템과 쉽게 연결할 수 있는 독립된 컴포넌트(component)로 디자인 되었다. 즉, MAYA는 [그림 1]에서 보듯이 독립된 색인 엔진(indexing engine)과 검색 엔진(searching engine)으로 구성된다.

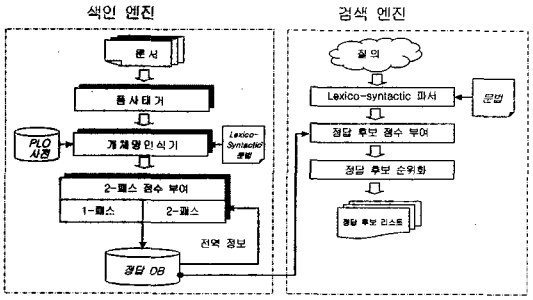


그림 2 MAYA의 시스템 구성도

3.1 정답 후보 색인

정답 후보 색인 과정은 후보 찾기 단계와 점수 부여 단계로 이루어진다. 정답 후보를 찾기 위해서 본 논문에서는 사용자의 질의 유형을 105개의 의미 범주(semantic category)로 구분하고, 그것에 따라 정답 유형(answer type)을 분류한다. [표 1]에서 보듯이 105개의 의미 범주는 2개의 계층으로 이루어진다.

표 1 의미 범주의 일부

The first layer	The second layer		
animal	bird	fish	mammal
	person	reptile	
location	address	building	city
	continent	country	state
	town		
date	day	month	season
	weekday	year	
time	hour	minute	second
organization	company	department	family
	group	laboratory	school
	team		

첫 번째 계층에 속한 의미 범주들은 두 번째 계층에 속한 것들보다 넓은 의미를 지닌다. 본 논문에서는 105개의 의미 범주를 결정하기 위하여 TREC에 참가한 질의 응답 시스템들의 의미 범주를 참고하였고, 상업용 정보 검색 시스템[13]으로부터 수집한 질의 로그(log)를 분석하였다.

문서로부터 각각의 의미 범주에 속하는 정답 후보들을 추출하기 위해서 색인 엔진은 품사 태거와 개체명 인식기(named entity recognizer)를 이용한다. 개체명 인식기는 PLO 사전이라고 불리어지는 개체명 사전(named entity dictionary)과 패턴 매처(pattern matcher)로 구성된다. PLO 사전은 사람, 국가, 도시, 기관의 이름 외에도 센티미터(cm), 킬로그램(kg)과 같은 단위들을 포함한다. 정답 후보를 추출하는 과정은 다음과 같다. 먼저, 품사 태거가 입력된 문장에 형태소 분석을 한다. 그리고, 개체명 인식기가 그 결과를 입력으로 하여 개체명을 추출하고 의미 범주를 할당 한다. 개체명 추출을 위해서는 정규 표현(regular expression) 형태의 lexico-syntactic 패턴이 이용되고, 의미 범주 할당을 위해서는 PLO 사전이 이용된다. 예를 들어, "야후코리아(사장 염진섭, www.yahoo.co.kr)은 무료 이메일 서비스의 용량을 6메가로 늘렸다."라고 하는 문장이 있을 때, 개체명 인식기는 그 문장으로부터 4개의 정답 후보를 추출한다. PLO 사전을 이용하여 '야후코리아'는 company로, '염진섭'은 person으로, '6메가'는 size로 의미 범주가 결정된다. 그리고, 'www.yahoo.co.kr'과 URL은 패턴 매처에 의해서 의미 범주가 결정된다. 패턴 매처는 정규 표현(regular expression) 형태의 lexico-syntactic 패턴을 이용하여 telephone number, email address와 같은 정형화된 정답 후보들을 추출한다. 또한, 패턴 매처는 '~ 때문에', '~하려면'과 같은 단서 구문을 이용하여 이유(reason)나 방법(method)과 같은 서술형 정답의 후보들도 추출한다. 그러나, 서술형 정답은 그 경계(boundary)를 추출하는 것이 매우 어려우므로 패턴과 매치(match)된 문장 전체를 정답 후보로 추출한다. 정답 후보를 추출한 후, 색인 엔진은 정답 후보와 같은 문맥(context)에 존재하는 내용어들(content words)에게 점수를 부여한다. 본 논문에서는 문맥의 범위를 정답 후보가 속한 문장을 기준으로 앞의 한 문장, 뒤의 한 문장으로 한다. 다시 말해서, 색인의 범위가 되는 문맥 윈도우(context window)의 최대 크기는 3문장이 되는 것이다. 문맥 윈도우의 크기는 정답 후보가 포함된 주변 문장에 어휘 체인이나 대용어가 포함되었는지의 여부에 따라 동적으로 변한다. 예를 들어, 정답 후보를 포함한 문장이 이전 문장과 어휘 체인을 가지고 있고, 다음 문장과는 그렇지 못할 경우에 문맥 윈도우의 크기는 2로 줄어든다. [그림 2]는 문맥 윈도우의 크기 변화를 예로 보여준다.

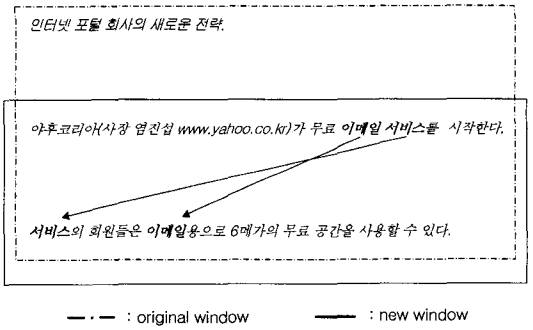


그림 3 문맥 윈도우의 크기 변화

문맥 윈도우의 크기가 결정되면 색인 엔진은 2-패스 점수 부여 방법에 따라 윈도우에 속한 내용어들에게 점수를 부여한다. 첫 번째 패스에서 각 내용어에 지역 점수(local score)를 할당한다. 지역 점수는 한 문서의 해당 문맥 윈도우 내에서 정답 후보와 내용어 사이의 연관 관계를 의미한다. 예를 들어, "야후 코리아(www.yahoo.co.kr)가 새로운 서비스를 시작한다."라는 문장이

있고 정답 후보가 '야후코리아'일 때, 'www.yahoo.co.kr'은 '서비스'보다 '야후코리아'와 더 밀접한 관계를 맺고 있다. 본 논문에서는 이렇게 하나의 문맥 윈도우 내에서 정답 후보와 내용어 사이에 존재하는 연관 관계를 점수화한 것을 지역 점수라고 부른다. 색인 엔진은 다음과 같은 2개의 특징(feature)을 이용하여 지역 점수를 할당한다.

- 빈도수(frequency): 문맥 윈도우에 속한 내용어들의 빈도수. 색인 엔진은 빈도수가 높은 내용어에 더 높은 점수를 부여한다. 예를 들어, [그림 2]에서 '이메일'이 '회원'보다 높은 점수를 받는다.
- 거리(distance): 정답 후보와 내용어 사이의 거리. 색인 엔진은 정답 후보와 가까운 거리에 있는 내용어에게 높은 점수를 부여한다. 예를 들어, [그림 2]에서 '염진섭'이 정답 후보일 때, '사장'이 '서비스'보다 높은 점수를 받는다.

색인 엔진은 단어들 사이의 IS-A 관계나 문장 성분(grammatical role)과 같은 정보를 사용하지 않는다. 왜냐하면 실제 응용 영역에서는 웹 문서와 같이 테이블(table)이나 이미지(image)를 포함한 문서들이 많이 존재하며, 문장 분할도 어려울 만큼 자유롭게 기술된 문서들이 많아서 위와 같은 고급 정보를 추출하는 것이 상대적으로 어렵기 때문이다.

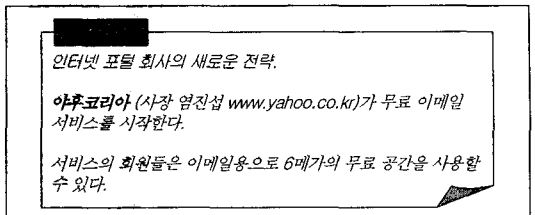
색인 엔진은 2단계에 걸쳐서 지역 점수를 계산한다. 먼저, [식 1]을 이용하여 정답 후보와 내용어 사이의 거리 가중치(distance weight)를 계산한다.

$$distw_{d,k}(a_i, w_j) = \frac{c}{\log(dist(i,j)) + c} \quad (1)$$

[식 1]에서 $distw_{d,k}(a_i, w_j)$는 문서 d의 k번째 문맥 윈도우에 j번째 위치한 내용어 w의 거리 가중치이다. $dist(i,j)$는 i번째 위치한 정답 후보 a와 j번째 위한 내용어 w사이의 거리이다. c는 실험 상수이다. 거리 가중치 계산이 끝나면, 색인 엔진은 동적 프로그래밍 기법(dynamic programming method)에 따라 기술된 [식 2]를 이용하여 문맥 윈도우에 속한 동일한 내용어들의 거리 가중치를 모두 더한다. [식 2]를 이용하여 빈도수가 높은 내용어들이 더 높은 점수를 부여 받는다.

$$\begin{aligned} &LS^{n}_{d,k}(a_i, w_{pos(n)}) \\ &= distw_{d,k}(a_i, w_{pos(n)}) + (1 - distw_{d,k}(a_i, w_{pos(n)})) \\ &\quad \times LS^{n-1}_{d,k}(a_i, w_{pos(n-1)}), where\ LS^{0}_{d,k}(a_i, w_{pos(0)}) \\ &= 0 \end{aligned} \quad (2)$$

[식 2]에서 $LS^{n}_{d,k}(a_i, w_{pos(n)})$은 문서 d의 k번째 문맥 윈도우에 동일한 형태의 내용어가 n개 존재할 때 n번째 내용어의 지역 점수이다. $pos(n)$은 n번째 내용어의 위치이다. [식 2]를 재귀적으로 풀고 나면 i번째 정답 후보 a_i와 k번째 문맥 윈도우에 속한 내용어 w 사이의 지역 점수 $LS_{d,k}(a_i, w)$가 계산된다. [그림 3]은 지역 점수 계산 과정의 예를 보여준다.



정답후보	처리 과정
야후코리아	1. '야후코리아'와 인접한 두 문장에 존재하는 '서비스' 사이의 거리를 계산한다. dist(1, 7)=6, dist(1, 9)=8
	2. 거리 가중치를 계산한다. *distw*(*Yahoo Korea1, service7*)= 1/(log(6)+1)=0.358 *distw*(*Yahoo Korea1, service9*)= 1/(log(8)+1)=0.325
	3. 두 거리 가중치를 합한다. *LS*(*Yahoo Korea1, service*)= 0.358+(1.0-0.358)*0.325=0.567

그림 3 지역 점수 계산의 예

두 번째 패스는 가상 문서(pseudo-document) 생성, 전역 점수(global score) 계산, 그리고 점수 합산 단계로 나뉘어 진다. 첫 번째 단계에서 색인 엔진은 정답 후보를 기준으로 가상 문서를 생성한다. 가상 문서는 모든 문서를 대상으로 하여 동일한 정답 후보를 포함하는 문맥 윈도우들에 속한 내용어들로 구성된다. 가상 문서는 해당하는 정답 후보로 구분되어 이름 붙여진다. [그림 4]는 가상 문서의 예를 보여준다.

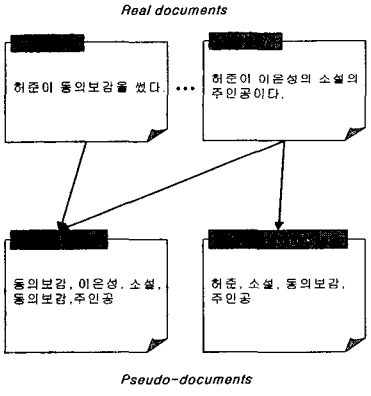


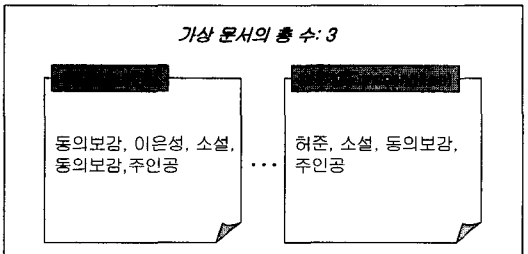
그림 4 가상 문서의 예

다음 단계에서 색인 엔진은 [식 3]을 이용하여 각 정답 후보들(가상 문서들)의 전역 점수를 계산한다. 전역 점수는 각 정답 후보와 그 후보를 포함하는 여러 문맥 윈도우들에 속한 내용어들 사이의 연관 관계를 의미한다.

$$GS(pseudo_{d_a}, w) = \begin{cases} \left(0.5+0.5\frac{tf_w}{Max_{tf}}\right)\frac{\log(N/n)}{\log(N)}, & \text{if } tf_w>0 \\ 0, & \text{if } tf_w=0 \end{cases} \quad (3)$$

[식 3]은 대상 문서가 실제 문서가 아니라 가상 문서라는 것을 제외하고는 잘 알려진 $TF \cdot IDF$ 수식[14]과 동일하다. 그러므로, TF 컴포넌트인 $(0.5+0.5 \cdot (tf_w/Max_{tf}))$는 가상 문서 $pseudo_{d_a}$에 포함된 내용어 w의 정규화된 빈도수를 의미한다. IDF 컴포넌트인 $\log(N/n)/\log(N)$은 내용어 w를 포함하는 가상 문서들의 정규화된 역문헌 빈도수를 의미한다. $GS(pseudo_{d_a}, w)$는 정답 후보 a와 내용어 w 사이의 전역 점수를 의미한다. 좀더 자세히 설명하면, tf_w는 가상 문서 $pseudo_{d_a}$에 포함된 내용어 w의 빈도수이고, Max_{tf}는 가상 문서 $pseudo_{d_a}$에 포함된 내용어들의 최대 빈도수이다. n은 내용어 w를 포함하는 가상 문서들의 수이고, N은 가상 문서의 총 수이다. [그림 5]는 전역 점수 계산 과정의 예를 보여준다.

마지막 단계에서 색인 엔진은 [식 4]를 이용하여 지역 점수와 전역 점수를 합한다.



정답후보	처리과정
허준	GS(*허준, 동의보감*)= (0.5+0.5*(2/2))*(log(3/2)/log(3))= 0.369
	GS(*허준, 이은성*)= (0.5+0.5*(1/2))*(log(3/2)/log(3))= 0.277
	GS(*허준, 소설*)= (0.5+0.5*(1/2))*(log(3/2)/log(3))= 0.277
	GS(*허준, 주인공*)= (0.5+0.5*(1/2))*(log(3/2)/log(3))= 0.277

그림 5 전역 점수 계산의 예

$$S_{d,k}(a_i, w) = \frac{\alpha \cdot LS_{d,k}(a_i, w) + \beta \cdot GS(pseudo_{d_a}, w)}{\alpha+\beta} \quad (4)$$

[식 4]에서 $LS_{d,k}(a_i, w)$는 정답 후보 a_i와 문서 d의 k번째 문맥 윈도우에 포함된 내용어 w 사이의 지역 점수이고, $GS(pseudo_{d_a}, w)$는 전역 점수이다. α와 β는 가중치 인수(weighting factor)이다. 두 점수를 합친 후에 색인 엔진은 정답 후보의 위치, 점수와 함께 내용어들을 역파일 형태로 데이터베이스에 저장한다.

3.2 규칙 기반 질의 처리

검색 엔진은 PLO 사전과 수동으로 구축된 lexico-syntactic 패턴을 이용하여 사용자의 질의 의도를 파악한다. 구축된 lexico-syntactic 패턴의 개수는 의미 범주당 평균 7.44개이며, 상업용 정보검색 시스템[13]으로부터 수집된 자연어 질의 로그(log)를 분석하여 구축되었다. 사용자의 질의 의도를 파악하기 위한 과정을 자세히 살펴보면 다음과 같다. 먼저, 검색 엔진은 PLO 사전을 이용하여 입력된 질의를 적당한 형태로 변환한다. PLO 사전은 단어들의 의미 표지(semantic marker)를 담고 있으며, 질의어들은 패턴 매칭에 앞서 의미 표지로 변환된다. 예를 들어, "누가 야후코리아의 사장이죠?"라는 질문은 "%who j @company j @position jp ef sf"로 변환된다. 예제에서 '%who', '@company', '@position'이 의미 표지이다. PLO 사전에 없는 단어들은 어휘 형태 그대로 유지되며, 기능어들은 품사 태그(tag)로 대체된다. 사용자 질의를 변환한 후에 검색 엔진은 [표 2]와 같이 수동으로 구축된 lexico-syntactic 패턴과 변환된 질의를 매칭하여 105개의 의미 범주 중에 하나로 질의를 분류한다. 분류된 결과를 기초로 하여 검색 엔진은 정답 후보의 대상을 해당 범주로 한정한다.

3.3 정답 후보 순위 부여

검색 엔진은 입력된 사용자 질의와 정답 후보들 사이에 유사도를 계산하고 그 유사도에 따라 정답 후보의

표 2 Lexico-syntactic 패턴

의미범주	Lexico-syntactic 패턴
person	%who (j|ef)? (%person|@person) j? (sf)* $ (%person|@person) j? %ident j? (sf)* $ (%person|@person) j? (%about)? @req (%person|@person) j? (%ident)? @req (%person|@person) jp ef (sf)* $ %which (%person|@person)
tel_num	(%tel_num|@tel_num) (%num)? j? (sf)*$ (%tel_num|@tel_num) (%num)? j? %what (%tel_num|@tel_num) j? (%about)? @req (%tel_num|@tel_num) j? (%what_num)

순위를 부여한다. 유사도 계산을 위해서는 [식 5]와 같은 p-Norm 모델[15]의 AND 오퍼레이션(operation)을 이용한다.

$$Sim(A, Q_{and}) = 1 - \sqrt[p]{\frac{q_1^p(1-t_1)^p + q_2^p(1-t_2)^p + \ldots + q_i^p(1-t_i)^p}{q_1^p + q_2^p + \ldots + q_i^p}} \quad (5)$$

[식 5]에서 A는 정답 후보이고, t_i는 그 정답 후보의 문맥 윈도우에 포함된 i번째 내용어의 점수이다. q_i는 질의에 포함된 i번째 내용어이고, p는 p-Norm 모델의 p-value이다.

MAYA는 색인 시에 정답 후보와 내용어들 사이에 점수를 미리 계산해 두기 때문에 검색할 때 매우 빨리 정답 후보를 순위화하여 제시할 수 있다. MAYA의 유사도 계산 방법은 전통적인 정보 검색 시스템에서 사용하는 질의와 문서 사이의 유사도 계산법과 유사하나 문서가 아니라 정답 후보를 색인하고 검색하며 순위화한다는 점에서 다르다.

MAYA는 독립된 컴포넌트로 디자인되었기 때문에 정보 검색 시스템과 쉽게 결합될 수 있다. 본 논문에서는 기존 정보 검색 시스템[16]의 성능 향상을 위하여 [식 6]과 같이 독립된 두 시스템의 검색 결과를 합하고, 그 시스템을 Joined-MAYA라고 부른다.

$$Sim(D, Q) = \frac{\alpha \cdot IRSim(D, Q) + \beta \cdot QASim_d(A_{max}, Q)}{\alpha + \beta} \quad (6)$$

[식 6]에서 $IRSim(D, Q)$는 질의 Q에 대한 정보 검색 시스템의 문서 검색 유사도이고, $QASim_d(A_{max}, Q)$는 문서 d에서 나타난 정답 후보들 중에서 가장 큰 값을 가지는 MAYA의 유사도이다. α와 β는 가중치 인자(weighting factor)이다.

4. 실험 및 평가

4.1 실험 데이터

MAYA의 성능을 실험하기 위해서 본 논문에서는 두 종류의 테스트 컬렉션(test collection)을 이용하였다. 하나는 korea.internet.com과 www.sogang.ac.kr로부터 수집된 컬렉션이고, 다른 하나는 한국어 질의응답 시스템 성능 평가 테스트 컬렉션인 KorQATeC 1.0[17]이다. 본 논문에서는 전자를 WEBTEC이라고 부르고, 후자를 KorQATeC이라고 부른다. WEBTEC은 22,448 문서(110,004 KB)로 구성되어 있으며, KorQATeC은 207,067 문서(368,768 KB)로 구성되어 있다. 그리고, 각각의 컬렉션은 50개의 질의 응답 셋(set)을 포함한다.

MAYA의 성능을 평가하기 위해서 TREC에서 사용하는 것과 마찬가지로 각 질의에 대한 첫 번째 정답의 RAR(reciprocal answer rank)을 계산하고, [식 7]과 같이 평균을 구한다[3, 18].

$$MRAR = 1/n\left(\sum_i 1/rank_i\right) \quad (7)$$

[식 7]에서 $rank_i$는 i번째 질문에 대해 응답으로 제시한 것들 중에서 첫 번째로 정답인 것의 순위이다. n은 질의의 총 수이다.

Joined-MAYA의 성능을 평가하기 위해서는 각 질의에 대해 정답을 포함하는 첫 번째 문서의 순위를 역순한 후 평균한 값을 구한다. 본 논문에서는 그 값을 MRDR (mean reciprocal document rank)이라고 부른다.

4.2 실험 결과 분석

정답 후보들의 점수를 계산하기 위해서 MAYA는 [식 4]와 같이 지역 점수와 전역 점수의 합을 사용한다. [식 4]의 가중치 인수를 실험적으로 결정하기 위하여 각 인수값의 변화에 따라 MAYA의 성능을 평가하였다. [표 3]은 가중치 인수값의 변화에 따른 MAYA의 성능을 보여준다. [표 3]에서 볼드체(boldface)로 표시된 MRAR이 실험에서의 최고값이다. 이 실험을 기초로 하여 본 논문에서는 [식 4]의 α와 β를 0.1과 0.9로 설정한다.

본 논문에서는 MAYA의 성능을 평가하기 위해서 이경순2000[19]과 Kim2001[12]을 MAYA와 비교하였다. 비교 대상 시스템들이 WEBTEC에 대한 결과를 제시하지 않기 때문에 KorQATeC만을 이용하여 실험하였다.

표 3 가중치 인자값에 따른 MAYA의 성능

		WEBTEC	KorQATeC	TOTAL
1.0	0.0	0.354	0.506	0.435
0.9	0.1	0.341	0.506	0.430
0.8	0.2	0.350	0.520	0.444
0.7	0.3	0.365	0.524	0.452
0.6	0.4	0.379	0.526	0.462
0.5	0.5	0.388	0.515	0.466
0.4	0.6	0.388	0.516	0.471
0.3	0.7	0.385	0.519	0.461
0.2	0.8	**0.405**	0.524	0.471
0.1	0.9	0.395	**0.540**	**0.473**
0.0	1.0	0.349	0.475	0.438

표 4 질의 응답 시스템의 성능 비교

System	이경순2000 (object)	이경순2000 (50-byte)
MRAR	0.322	0.456
MRAR-1	0.322	0.456
System	Kim2001 (object)	MAYA (object)
MRAR	0.485	0.540
MRAR-1	0.539	0.600

[표 4]에서 보듯이 MAYA의 성능은 다른 비교 대상 시스템보다 좋은 것을 알 수 있다. 이 사실은 MAYA의 점수 부여 방법이 단순하지만 매우 유용하다는 것을 보여준다. [표 4]에서 이경순2000 (50-byte)는 정답 그 자체를 제시하는 것이 아니라 정답을 포함하는 주변 50-byte를 제시하는 시스템이다. MRAR-1은 정답을 제시하지 못한 것들을 제외한 MRAR이다. MAYA는 5개의 질의에 대해서 정답을 제시하지 못했는데 그 원인은 다음과 같다.

- 규칙 기반의 질의 처리 모듈이 사용자의 의도를 파악하는데 실패했다. 이 문제를 해결하기 위해서는 lexico-syntactic 패턴의 추가가 필요하다.
- 개체명 인식기가 정답 후보를 추출하지 못했다. 이 문제를 해결하기 위해서는 PLO 사전의 엔트리(entry)를 보충하고, 후보 추출을 위한 규칙을 추가하는 것이 필요하다. 또한 개체명 인식기 자체의 성능 향상도 필요하다.

[표 5]는 Joined-MAYA의 성능을 보여준다. [표 5]에서 보듯이 Joined-MAYA는 105개 의미 범주에 속한 질의에 대해서 기존 정보 검색 시스템의 성능을 향상시킨다.

표 5 문서 검색 성능 비교

	IR system	MAYA	Joined-MAYA
MRDR	0.581	0.699	0.720

[표 6]은 기존 정보 검색 시스템[16]과 MAYA의 검색 속도를 비교한 것이다. [표 6]에서 보듯이 정보 검색 시스템의 평균 검색 시간은 듀얼(dual) CPU의 펜티엄(pentium) III 서버에서 0.026초이고, MAYA는 0.048초이다.

표 6 검색 속도 비교

	질의당 응답 시간 (초)	메가 바이트당 색인 시간 (초)
IR system	0.026	26.765
MAYA	0.048	30.542
Incomplete-MAYA	5.300	26.765

이것을 바탕으로 MAYA와 정보 검색 시스템과의 속도 차이는 그렇게 크지 않음을 알 수 있다. Incomplete-MAYA는 색인 엔진 없이 검색 시에 정보 검색 시스템이 제시한 상위 30개의 문서로부터 정답 후보들의 점수를 계산하여 추천하는 시스템이다. 그러므로, 별도의 색인 시간이 필요하지 않다. 그러나, [표 6]에서 보듯이 검색 시간이 MAYA에 비해 110배 정도 더 걸린다. 만약, 정보 검색 시스템의 결과 중에 상위 50개나 그 이상을 사용했다면 더 오랜 시간이 걸릴 것이다. 결론적으로, MAYA는 색인 엔진이 없는 시스템에 비해서 메가 바이트(mega byte)당 26.765초의 추가 색인 시간이 필요하지만 검색 시간이 실제 응용 영역에는 더 적합하고 효율적인 시스템이다.

5. 결 론

본 논문에서는 정답 후보 색인기를 이용한 고속, 고정밀의 한국어 질의 응답 시스템을 제안하였다. 정답 후보 색인기는 정답 후보와 인접한 내용어들을 추출하고, 2-패스 점수 부여 방법에 따라 각 후보와 내용어들 사이의 연관 관계를 계산하다. 그리고, 검색 시에는 단순한 계산을 통하여 질의어와 정답 후보들 사이의 유사도를 계산한다. 이러한 방법을 이용하여 제안된 질의 응답 시스템은 검색 시간을 최소화하고 검색 성능을 향상시킨다. 또한, 제안된 시스템은 품사 태거, 개체명 인식기, 규칙 기반 질의 처리, 그리고 *TF·IDF* 가중치와 같은 비교적 간단한 자연어 처리 기술과 정보 처리 기술을 이용하기 때문에 응용 영역 변화에 쉽게 대처할 수 있다.

참고 문헌

[1] Voorhees E. and Tice D. M., "Building a Question Answering Test Collection", In *Proceedings of SIGIR 2000*, pp. 200-207, 2000.

[2] AAAI Fall Symposium on Question Answering, *http://www.aaai.org/Press/Reports/Symposia/Fall/fs-99-02.html*.

[3] TREC (Text REtrieval Conference) Overview, *http://trec.nist.gov/overview.html*.

[4] Vicedo J. L. and Ferrández A., "Importance of Pronominal Anaphora resolution in Question Answering systems", In *Proceeding of ACL 2000*, pp. 555-562, 2000.

[5] Moldovan D., Harabagiu S., Paşca M., Mihalcea R., Goodrum R., Girju R. and Rus V., "LASSO: A Tool for Surfing the Answer Net", In *Proceedings of The Eighth Text REtrieval Conference (TREC-8)*, from http://trec.nist.gov/pubs/trec8/t8_proceedings.html, 1999.

[6] Prager J., Radev D., Brown E. and Coden A., "The Use of Predictive Annotation for Question Answering in TREC8", In *Proceedings of The Eighth Text REtrieval Conference (TREC-8)*, from http://trec.nist.gov/pubs/trec8/t8_proceedings.html, 1999.

[7] Berri J., Molla D., and Hess M., "Extraction

automatique de réponses: implémentations du systéme ExtrAns", In *Proceedings of the fifth conference TALN 1998*, pp. 10-12, 1998.

[8] Harabagiu S., Moldovan D., Pasca M., Mihalcea R., Surdeanu M., Bunescu R., Girju R., Rus V. and Morarescu P., "FALCON: Boosting Knowledge for Answer Engines", In *Proceedings of the Ninth Text REtrieval Conference (TREC-9)*, from http://trec.nist.gov/pubs/trec9/t9_proceedings.html, 2000.

[9] Kupiec J., "Murax: A Robust Linguistic Approach for Question Answering Using an On-line Encyclopedia", In *Proceedings of SIGIR'93*, 1993

[10] Prager J., Radev D., Brown E., and Coden A., "The Use of Predictive Annotation for Question Answering in TREC8", In *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*, http://trec.nist.gov/pubs/trec8/t8_proceedings.html, Gaithersburg, Maryland, 1999.

[11] Prager J., Brown E. and Coden A., "Question-Answering by Predictive Annotation", In *Proceedings of SIGIR 2000*, pp. 184-191, 2000.

[12] Kim H., Kim K., Lee G. G. and Seo J., "MAYA: A Fast Question-answering System Based On A Predictive Answer Indexer", In *Proceedings of the ACL Workshop Open-Domain Question Answering*, pp. 9-16, 2001.

[13] DiQuest.com, *http://www.diquest.com*

[14] Fox E. A., *Extending the Boolean and Vector Space Models of Information Retrieval with P-norm Queries and Multiple Concept Types*, Ph.D. Thesis, CS, Cornell University, 1983.

[15] Salton G., Fox E. A. and Wu H., *Extended Boolean Information Retrieval*, Communication of the ACM, Vol.26, No.12, pp. 1022-1036, 1983.

[16] Lee, G., Park, M., and Won, H., "Using syntactic information in handling natural language queries for extended boolean retrieval model", In *Proceedings of the 4th international workshop on information retrieval with Asian languages (IRAL99)*, pp. 63-70, 1999.

[17] 이경순, 김재호, 최기선, "질의응답시스템의 성능 평가를 위한 테스트컬렉션 구축", *제 12회 한글 및 한국어 정보처리 학술 대회 논문집*, pp. 190-197, 2000.

[18] Voorhees E. and Tice D. M., "The TREC-8 Question Answering Track Evaluation", In *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*, from http://trec.nist.gov/pubs/trec8/t8_proceedings.html, 1999.

[19] 이경순, 김재호, 최기선, "한국어 질의응답시스템에서 개체인식에 기반한 대답 추출", *제 12회 한글 및 한국어 정보처리 학술대회 논문집*, pp. 184-189, 2000.

김 학 수

1996년 건국대학교 전자계산학과 학사. 1998년 서강대학교 컴퓨터학과 석사. 1998년 ~ 현재 서강대학교 컴퓨터학과 박사과정. 2001년 ~ 현재 다이퀘스트 연구소 선임연구원. 관심 분야는 자연어 처리, 대화 시스템, 생략 및 대용어 처리, 화행 분석, 정보 검색, 질의 응답 시스템

서 정 연

정보과학회논문지 : 소프트웨어 및 응용 제 29 권 제 6 호 참조