장거리 의존 문제를 해결하기 위한 표층 텍스트 패턴의 확장

이미연 차정원* 박승수
이화여자대학교 과학기술대학원 컴퓨터학과
{ailmy, sspark}@ewha.ac.kr
창원대학교 컴퓨터통신공학부*
jcha@changwon.ac.kr

Expansion of Surface Text Patterns for Long-Distance Dependency

Mee-Yeon Lee Seung-Soo Park
Dept. of Computer Science and Engineering, Ewha Womans University

Jeongwon Cha
School of Computer Information & Communication, Changwon National University

요 약

본 논문에서는 질의 응답 시스템에서 정답 추출을 위해 사용되는 표층 텍스트 패턴을 장거리 의존 문제에도 적용 가능하도록 확장하는 방법을 제안한다. 기존의 패턴 추출 시스템들의 패턴을 구성하고 있는 단어들간의 연속성과 불연속성에 대한 정보를 나타내도록 패턴 형태를 확장함으로써 장거리 의존 문제를 해결한다. 본 논문에서 제안한 형태의 패턴을 TREC-10의 질의를 이용해서 웹 데이터로 실험하여 정확도와 TREC의 평가 기준인 MRR을 사용해서 기존 시스템들과 성능을 비교했다.

1. 서 론

정보 추출이나 질의 응답 작업을 수행하기 위해서는 방대한 양의 외부 지식이나 부가적인 도구들이 필요하다. 하지만 간단한 패턴들만을 이용함으로써 이런 요구 사항들을 크게 줄일 수 있고 성능을 높일 수 있다는 연구 결과들이 나오고 있다[1].

패턴은 특정한 정보를 담고 있는 문장들이 일반적으로 어떻게 구성되는지를 표현하는 구문론적 형태를 말하며, 추론 규칙(inference rules), 정규 표현(regular expressions), 사전(dictionary)과 같은 용어로도 불린다. 예를 들어, "X (Y", "Y, X", "X was born in/on Y", "X invents Y", "when X discovered Y" 등을 패턴이라고 한다[2].

이러한 패턴들은 심도 있는 자연어 처리가 필요하지 않고 안정적이며 어느 정도의 성능을 보인다는 장점이 있는 반면 다음과 같은 단점이 있다. 첫째, 연속된 특정 표현이나 심볼로 구성되는 경우가 대부분이다. 따라서 다양한 변이 형태나 장거리 의존 등에는 대처를 하지 못한다는 단점을 갖는다.

(1) **Mozart** was **born** in Salzburg in 1756.
(2) **Mozart** was **born** into a world filled with music in **January 27, 1756** in the town of Salzburg, Austria.
(3) ...in the form of widespread bureaucratic abuses : graft, nepotism...
(4) ...John Wayne Airport in Santa Ana, Calif., that were paved by…

패턴 "X was born in Y"의 경우에, Y는 년도가 정답임에도 불구하고 예문 (1)과 같은 문장에서는 패턴에 의해서 "Salzburg"가 선택될 수 있다. 하지만 예문 (2)에서는 앞에서 언급한 패턴을 적용할 수가 없다. 둘째, 과적용이 일어날 가능성이 있는 패턴들도 있다. "Y, X" 패턴의 경우가 그러한 종류이다. 예를 들어, "What is nepotism?"에 대한 답을 찾을 경우 "Y, X"라는 패턴은 예문 (3)에서도 적용이 되어 틀린 결과를 내놓는다. 셋째, 추가 정보가 없다면 Y는 오직 하나의 단어로만 바뀐다. "Where is John Wayne airport?"라는 질의가 주어졌을 경우, 예문 (4)에 "X in Y" 패턴이 적용되어서

하나의 단어만 추출된다면 해답은 "Santa"가 된다.

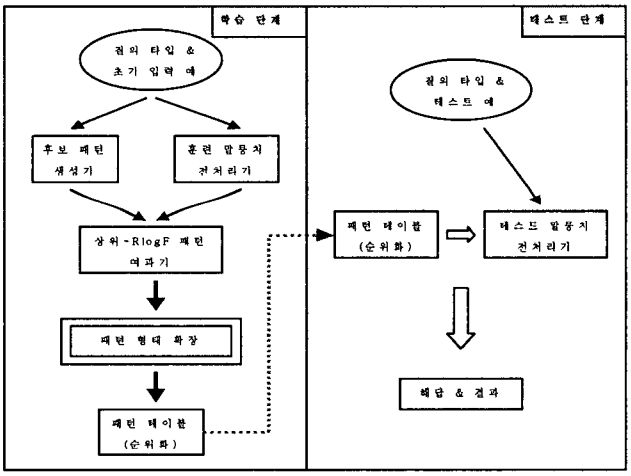
본 논문에서는 표층 패턴의 여러 문제점들 중에서 장거리 의존을 해결하기 위해 연속된 패턴을 확장한다. 즉, 패턴간의 유사성 정보를 바탕으로 패턴을 이루는 각 단어간의 연속성과 불연속성의 정보를 나타내도록 패턴의 형태를 확장한다.

본 논문의 구성은 다음과 같다. 2장에서는 기존의 패턴 추출에 관련된 연구에 대해 알아보고 3장에서는 패턴을 학습하는 과정을 설명한다. 4장에서는 학습된 패턴을 적용하는 과정을 설명하고 5장에서는 실험결과를 보이며 6장에서 결론을 내리고 문제점을 토의한다.

2. 관련 연구

[3]은 훈련 말뭉치로부터 개념 노드(concept node)라고 불리는 패턴을 자동으로 학습하는 AutoSlog 시스템을 제안했다. 하지만 이 시스템은 수동으로 만든 정답이 부착된 훈련 말뭉치가 필요하며 이 말뭉치에 크게 의존한다는 문제점이 있다. [4]에서는 정답이 부착된 훈련 데이터를 필요로 하지 않는 AutoSlog-TS 시스템을 소개했다. 이 시스템은 AutoSlog의 확장 버전으로, 통계적 기법을 사용함으로써 태그가 부착된 텍스트에 대한 의존성을 없앴다. [5]는 말뭉치의 의존 트리에 분포 가설(distributional hypothesis)을 적용해서 트리 내의 유사한 경로를 찾아냄으로써 추론 규칙을 발견하는 DIRT 알고리즘을 소개한다. [2]는 주어진 질의 타입에 대해 웹 문서로부터 패턴을 자동으로 학습하고, 학습한 패턴들의 정확성을 계산하는 알고리즘을 발표했다. 하지만 학습한 패턴이 너무 일반적이고 장거리 의존 문제를 처리할 수 없다는 등의 단점을 내포하고 있다. [6]은 정답이 부착되지 않고 분류되지 않은 말뭉치로부터 패턴을 발견하는 방법론을 제안한다. 이 시스템은 시나리오(관심사 중에서 특정 주제)에 대한 몇 가지 초기(seed) 튜플로부터 출발해서, 말뭉치로부터 만들어낸 후보 패턴들 중에서 특정 최소 한계값 이상의 정확도 값을 갖는 패턴들만 초기 패턴 집합에 추가한 후 과정을 반복함으로써 진행된다. 하지만 여기에는 최소 한계값의 선택과 반복 과정의 종료 조건 선택 등의 문제가 있다. [7]의 Snowball 시스템은 1998년에 Brin이 제안한 DIPRE 알고리즘에[8] 기반해서 일반-텍스트 모음으로부터 특정 시나리오에 대한 관계 테이블을 추출한다. Snowball 시스템에서 주목할 점은 명칭-개체 태그를 사용하고, 패턴과 관계 튜플의 신뢰도를 평가하기 위한 방법과 패턴 추출 시스템을 평가하기 위한 방법론을 제안했다는 것이다. 하지만, 일반-텍스트뿐만 아니라 HTML 데이터에도 적용할 수 있도록 하고, 패턴의 생성과 일치가 명칭 개체와 그 주변의 명사구까지 고려할 수 있도록 확장하는 것이 필요하다. [9]는 최대 엔트로피에 기반한 질의-응답 시스템에 명칭-개체 태거(Named-Entity Tagger)를 추가함으로써 표층 텍스트 패턴의 성능을 높이고 유용함을 보인다. [10]은 [2]의 한계점 중 하나를 해결한다. 즉, 질의어(Question Term)와 해답어(Answer Term) 사이에 특정 질의에 의존적인 단어가 있을 경우, 그 패턴은 다른 질의에 적용되기 어렵다는 점을 극복하기 위해, 지명 사전(gazetteer)과 명칭-개체 태거를 사용해서 패턴 형태와 문서들을 일반화한다. 하지만 이 접근법은 다른 질의 타입에 대한 확장이 부족하다는 단점을 보인다.

3. 패턴 학습



[그림 1] 학습 및 테스트 과정

본 논문이 제안하는 시스템은 특정 질의 타입에 대한 패턴 테이블 구축을 목표로 하고, [그림 1]에서 보인 것과 같이 크게 훈련 단계와 테스트 단계로 수행된다. 훈련 과정은 세 단계로 진행된다 : 첫 단계에서는 초기 입력으로부터 후보 패턴을 생성하고, 이들을 기호-패턴과 어휘-패턴으로 분류한다. 초기 입력은 각 질의 타입에 해당하는 질의어와 해답어의 쌍이다. 두 번째 단계에서는 기호-패턴과 어휘-패턴으로부터 각각 RlogF를 [4] 계산하고 상위-RlogF 패턴들을 걸러낸다. 그리고 세 번째 단계에서 장거리 의존 문제를 해결하기 위해서 상위-RlogF 패턴과 나머지 패턴들과의 유사성을 계산해서 상위-RlogF 패턴의 형태를 확장한다. 단, 패턴 확장은 어휘-패턴에 대해서만 수행된다. 특정 질의 타입에

대한 최종 패턴 테이블은 여러 다른 예들에 대한 반복 수행에 의해 생성된다. 시스템의 세부 과정은 장거리 의존 문제를 해결하는 패턴 확장 부분을 제외하면 [2]와 유사하다. 알고리즘의 각 단계를 질의 타입 "BIRTHYEAR"에 대한 예를 통해 설명한다. 이 때 사용된 질의어와 해답어의 쌍과 최종 말뭉치의 문서 개수는 [표 1]과 같다.

[표 1] BIRTHYEAR 질의 타입에 대해 사용된 질의어-해답어 쌍과 실제 사용된 말뭉치의 문서 개수.

질의어	해답어	말뭉치의 문서 개수
Mozart	1756	675
Gandhi	1869	825
Gauss	1777	763
Newton	1642	837

3-1. 1단계 : 후보 패턴 추출

1. 질의어와 해답어로 웹 검색 엔진에 질의를 던진다. BIRTHYEAR 질의 타입에 대해, [표 1]의 첫 번째 행에 있는 (Mozart, 1756)으로부터 질의 "Mozart + 1756"을 검색 엔진에 던진다. 우리는 실험에서 Google 엔진[1]을 사용했다.
2. 검색 엔진이 찾아낸 관련 웹 문서들 중에서 상위 1000개의 문서를 다운 받는다.
3. 웹 문서들로부터 순수한 텍스트들만을 뽑아낸다. 우리는 텍스트만 지원하는 웹 브라우저 Lynx를 사용했다. 이 텍스트 문서 중에서 중복 문서는 제거하고, 문장 분리기를 적용해 문장 단위로 분리한다.
4. 분리된 문장들 중에서 질의어와 해답어를 모두 포함하고 있는 문장들만 모은다.[2] 예를 들어, 질의어와 해답어인 "Mozart"와 "1756"을 모두 포함하고 있는 문장 "Wolfgang Amadeus Mozart was born in 1756 in Salzburg, Austria."이다.
5. positive 문장으로부터 질의어나 해답어로 시작하고 끝나는 positive 절들을 뽑아낸다. 이 때, 한 문장 내에 여러 가지 형태의 절들이 존재할 수 있으므로 그들을 모두 인식해서 추출해야 한다. 또한, 왼쪽과 오른쪽에 각각 한 단어씩을 추가한 형태도 추출함으로써 접미사 트리의 효과도 얻을 수 있도록 한다.
   (1) **Mozart** was born in 1756
   (2) **Mozart** (**1756** -
   (3) **Mozart**'s birth (**1756**
   (4) **Mozart** + **1756**
6. 다음으로, positive 절 중에서 질의어와 해답어를 특정 태그로 교체한다. 즉, "Mozart"를 〈NAME〉으로, "1756"을 〈ANSWER〉로 교체한다. 이 때, 질의어와 해답어에 대한 변이들을 인식해서 모두 하나의 태그로 교체해주는 작업이 필요하다. 다음은 태그가 포함된 positive 절들의 예이다 :
   (1) 〈NAME〉 was born in 〈ANSWER〉
   (2) 〈NAME〉 (〈ANSWER〉 -
   (3) 〈NAME〉's birth (〈ANSWER〉
   (4) 〈NAME〉 + 〈ANSWER〉

이들 중에서 기호로만 이루어진 패턴과 어휘 패턴들에 대해 다른 처리 과정이 필요하기 때문에, 후보 패턴을 두 타입으로 분리한다. [표 2]와 [표 3]에 1 단계의 최종 결과인 두 타입의 패턴 예를 보인다.

[표 2] 기호-패턴 (1단계의 결과).

기호-패턴
〈NAME〉 (〈ANSWER〉 -
〈NAME〉 - 〈ANSWER〉
〈NAME〉 + 〈ANSWER〉
〈ANSWER〉-1791) 〈NAME〉
〈NAME〉's (〈ANSWER〉

[표 3] 어휘-패턴 (1단계의 결과).

어휘-패턴
〈NAME〉 was born in 〈ANSWER〉
〈NAME〉's birth (〈ANSWER〉
〈NAME〉 Born : Salzburg, 〈ANSWER〉
〈NAME〉 was born on 〈ANSWER〉 in
〈NAME〉 was born in Salzburg on 〈ANSWER〉

다음 단계에서는 RlogF 값을[4] 사용해서 각 후보 패턴의 신뢰도를 측정한다.

3-2. 2단계 : 상위-RlogF 패턴 추출

1. 질의어만을 검색 엔진에 던진다. 즉, BIRTHYEAR 질의 타입에 대해, [표 1]의 3, 4, 5번째 행의 "Gandhi", "Gauss"와 "Newton"을 질의로 던진다.
2. 각각 상위 1000개의 문서를 다운 받는다. 문서들로부터 HTML 태그는 제거하고, 중복 문서는 제거한다. [표 1]의 마지막 열이 남은 문서의 개수를 나타낸다.
3. 문장 분리기를 사용하여 문장 단위로 분할한다.

1) www.google.co.kr
2) 우리는 이러한 문장을 positive 문장이라고 부른다.

4. 1단계에서 구한 패턴을 각 말뭉치에 적용해서 RlogF 값을 계산한다.

$$R\log F(CP) = \frac{Positive(CP)}{Positive(CP)+Negative(CP)} \cdot \log(Positive(CP))$$

CP는 후보 패턴을 의미한다. *Positive*(CP)는 각 후보 패턴의 positive 일치 횟수이고, *Negative(CP)*는 negative 일치 횟수이다. 각 후보 패턴 CP의 middle(두 tag 사이의 context)을 위에서 얻은 말뭉치에 적용한다. CP의 형태가 "**〈NAME〉** middle **〈ANSWER〉**"인 경우에, middle을 포함하는 문장 중에서 middle의 왼쪽에 "**Mozart**"가 있고 오른쪽에 해답어인 "**1756**"이 모두 있을 경우를 positive 일치라고 하고, 왼쪽에 "**Mozart**"가 있지만 오른쪽에 "**1756**"이 없을 경우를 negative 일치라고 한다. CP의 형태가 "**〈ANSWER〉** middle **〈NAME〉**"인 경우에는, "**Mozart**"와 "**1756**"의 위치가 각각 오른쪽과 왼쪽으로 바뀐다는 점을 고려한다. 이렇게 해서 각 후보 패턴의 RlogF 값을 계산할 수 있고, 그 중에서 0 이상의 RlogF 값을 갖는 후보 패턴을 상위-RlogF 패턴으로 분류한다. 각 말뭉치로부터 얻어진 상위-RlogF 패턴들을 합집합 형태로 취함으로써 최종 상위-RlogF 패턴이 결정된다. 이렇게 하여 구한 BIRTHYEAR의 패턴 예는 [표 4]와 [표 5]에 보인다. (BIRTHYEAR 타입에서는 말뭉치 2에 대한 결과가 말뭉치 3, 4에 대한 결과를 모두 포함했다.)

[표 4] 말뭉치 2에 대한 상위-RlogF 기호 패턴 ('Gandhi + 1869')

상위-RlogF 기호 패턴	RlogF 값
〈NAME〉 (〈ANSWER〉	0.909169402008534
〈NAME〉) (〈ANSWER〉	0.333333333333333
〈NAME〉 (〈ANSWER〉 -	0.211328333429488
〈NAME〉 〈ANSWER〉 -	0.00365479439059452

[표 5] 말뭉치 2에 대한 상위-RlogF 어휘 패턴 ('Gandhi + 1869')

상위-RlogF 어휘 패턴	RlogF 값
〈NAME〉 was born on 〈ANSWER〉	2
〈NAME〉 was born in 〈ANSWER〉	1.1531438728791
〈NAME〉 was born in 〈ANSWER〉 to	0.950977500432694
〈NAME〉 was born in 〈ANSWER〉 in	0.935784974019201

3-3. 3단계 : 어휘-패턴의 확장

2 단계에서 선택된 상위-RlogF 패턴들을 확장한다. 어휘-패턴의 경우는 서론에서 보았듯이 중간에 부사나 다른 어구가 나오는 경우가 빈번하다. 따라서 2 단계에서 찾은 패턴들을 다음과 같은 과정을 통해서 장거리 의존에 대처할 수 있는 형태로 확장한다.

1. 상위-RlogF 패턴 중에서 하나를 선택한다.
2. middle중의 각 단어를 나머지 패턴에 적용하여 분리되어 적용될 수 있는 것을 찾는다.
3. 분리될 수 있는 단어는 다른 그룹으로 모은다.

[표 6] 패턴 확장의 예(BIRTHYEAR)

상위-RlogF 패턴	〈NAME〉 was born on 〈ANSWER〉 〈NAME〉 was born in 〈ANSWER〉 〈NAME〉 was born 〈ANSWER〉
그 외 패턴	... 〈NAME〉 was born into a world filled with music on 〈ANSWER〉 〈NAME〉 was born at Salzburg, Austria, in 〈ANSWER〉 ...
확장된 패턴	〈NAME was born on ANSWER〉 〈NAME was born〉 〈on ANSWER〉 〈NAME was〉 〈born〉 〈on ANSWER〉 〈NAME was born in ANSWER〉 〈NAME was〉 〈born in ANSWER〉 〈NAME was born〉 〈in ANSWER〉 〈NAME was born ANSWER〉

예를 들어, [표 6]에 보인 상위-RlogF 패턴 "**〈NAME〉** was born on **〈ANSWER〉**"와 그 외 패턴 "**〈NAME〉** was born into a world filled with music on **〈ANSWER〉**"를 선택하여 이 두 패턴을 비교하여 "**〈NAME〉** was born"까지가 일치하고 'on'은 떨어져서 "on 〈ANSWER〉"가 있다. 이 비교를 통하여 선택된 패턴 내의 모든 단어가 나타나는 패턴들을 찾아내어 그룹을 만들 수 있다. 또한 선택된 패턴에서 어떤 단어들이 같이 나타나는지, 어떤 단어들은 떨어져 나타날 수도 있는지를 알 수 있다. 위의 예에서는 "**〈NAME〉** was born"은 같이 나타나는 빈도가 많고, "on **〈ANSWER〉**"와는 떨어져 나타나는 경우가 많다. 따라서 이 경우에 우리는 이들을 다음과 같이 표기한다.

[NAME] was born〉 〈on **[ANSWER]**〉

[표 5]의 패턴으로부터 3 단계를 거쳐 생성된 확장된 형태의 어휘-패턴을 [표 7]에 보인다.

[표 7] 확장된 패턴의 예 ('Gandhi + 1869')

상위-RlogF 확장-어휘 패턴	RlogF 값
〈[NAME] was born〉 〈on [ANSWER]〉	2.18349827271147
〈[NAME] was born on [ANSWER]〉	2
〈[NAME] was born in [ANSWER]〉	1.1531438728791
〈[NAME] was born in [ANSWER] to〉	0.950977500432694
〈[NAME] was born in [ANSWER] in〉	0.935784974019201
〈[NAME] was born on [ANSWER] in〉	0.792481250360578
〈[NAME] was〉 〈born in [ANSWER]〉	0.4
〈[NAME] was born [ANSWER] in〉	0.135853928633242
〈[NAME] was born [ANSWER]〉	0.0766917339058624
〈[NAME] Born : [ANSWER]〉	0
〈[NAME] was born〉 〈in [ANSWER]〉	0

4. 패턴의 적용

학습된 패턴을 사용하여 새로운 질의에 대해 답을 추출한다.

1. 질의 유형을 판별한다.
2. 질의에서 질의어를 인식한다.
3. 질의어로 웹 검색 엔진에 질의를 던지고, 상위 1000개의 문서를 다운 받는다. 문서에서 HTML을 제외한 순수한 텍스트만 남기고, 중복 문서는 제거한다.
4. 각 문장에 위의 패턴 학습 과정에서 생성된 최종 패턴을 적용해서 패턴의 "**〈ANSWER〉**" 태그 자리에 위치한 단어를 해답으로 추출한다. 각 해답을 추출하는데 사용된 패턴의 정확도에 따라 정렬된 형태로 보인다.

5. 실험 및 결과

3개의 question type에 대한 실험을 수행했다: BIRTHYEAR, DEATYYEAR, LOCATION. 각 질의 타입에 대해 패턴을 추출할 때 초기 입력 쌍으로는 4개의 예를 사용했다. (BIRTHYEAR 질의 타입에 대해 사용된 초기 입력 쌍은 [표 1]에 표시 되어 있다). 훈련 과정의 초기 입력 예와 테스트에 사용한 질의는 Deepak이 [2]에서 사용했던 것들과 TREC-9[3], TREC-10[4]을 참고했다. [표 8], [표 9], [표 10]은 각 질의 타입에 대해 우리 시스템이 추출한 최종 패턴 테이블을 보인다 (상위 10개). 패턴들은 정확한 답을 찾을 가능성이 높은 순서대로 정렬되어 있다.

3) 2000, TREC-9 QA Data,
http : //trec.nist.gov/data/qa/t9_qadata.html
4) 2001,TREC-10 QA Data,
http : //trec. nist.gov/data/qa/t2001_qadata.html

[표 8] BIRTHYEAR 타입에 대한 패턴 테이블

순위	패 턴	RlogF 값
1	〈[NAME] was born〉 〈on [ANSWER]〉	2.18
2	〈[NAME] was born on [ANSWER]〉	2.00
3	〈[NAME] was born in [ANSWER]〉	1.15
4	〈[NAME] was born in [ANSWER] to〉	0.95
5	〈[NAME] was born in [ANSWER] in〉	0.94
6	〈[NAME] ([ANSWER]〉	0.91
7	〈[NAME] was born on [ANSWER] in〉	0.79
8	〈[NAME] was〉 〈born in [ANSWER]〉	0.40
9	〈[NAME]) ([ANSWER]〉	0.33
10	〈[NAME] ([ANSWER] ->〉	0.21

[표 9] DEATHYEAR 타입에 대한 패턴 테이블

순위	패 턴	RlogF 값
1	〈[NAME] died in [ANSWER]〉	2.67
2	〈[ANSWER]) [[NAME]〉	1.60
3	〈[NAME]'s death in [ANSWER]〉	1.00
4	〈[NAME] in [ANSWER]〉	0.93
5	〈[ANSWER]) [NAME]〉	0.87
6	〈[ANSWER], [NAME] was〉	0.86
7	〈in [ANSWER], [NAME] started〉	0.40
8	〈[ANSWER], [NAME] started〉	0.29
9	〈in [ANSWER], [NAME] was〉	0.24
10	〈[ANSWER]) [NAME] was〉	0.22

[표 10] LOCATION 타입에 대한 패턴 테이블.

순위	패턴	RlogF 값
1	〈[ANSWER]'s [NAME]〉	4.70
2	〈to [NAME] in [ANSWER]〉	3.44
3	〈in [ANSWER]'s [NAME]〉	3.37
4	〈at [NAME] in [ANSWER]〉	2.56
5	〈[NAME] in [ANSWER] and〉	2.34
6	〈from [NAME] in [ANSWER]〉	2.32
7	〈the [NAME] in [ANSWER]〉	2.00
8	〈of [NAME] in [ANSWER]〉	1.58
9	〈of [ANSWER]'s [NAME]〉	1.34
10	〈[ANSWER], including [NAME]〉	0.50

[표 11]에 각 질의 타입의 테스트 질의 개수와 MRR(Mean Reciprocal Rank)[11] 수치를 보인다.

[표 11] 각 질의 타입에 대한 테스트 질의 개수와 MRR.

질의 타입	질의 개수	MRR
BIRTHYEAR	8	0.67
DEATHYEAR	3	0.00
LOCATION	16	0.84

DEATHYEAR 타입의 MRR 값은 [10]에서 지적된 문제점으로 인해 0.00을 나타냈다. 이를 해결하기 위해 후에 명칭-개체 태거를 사용할 계획이다.

6. 결론 및 토의

본 논문에서는 기존의 표층 패턴을 확장하여 장거리 의존 문제에도 적용할 수 있게 하였다. [표 12]를 통해 확장하지 않은 기존의 어휘-패턴과 확장한 패턴들이 추가된 후의 성능을 비교할 수 있다(여러 질의에 대한 결과의 평균값). 각 패턴의 정확도는 다음과 같이 계산되었다.

$$\Pr ecision(CP) = \frac{Positive(CP)}{Positive(CP) + Negative(CP)} \cdot 100$$

[표 12] 확장 전과 확장 후의 어휘 패턴의 정확도 비교

질의 타입	확장 전 정확도	확장 후 정확도
BIRTHYEAR	25.26%	28.88%
DEATHYEAR	1.63%	1.63%
LOCATION	2.81%	2.80%

확장된 형태의 어휘-패턴들은 기존의 패턴이 찾지 못하는 다음과 같은 문장에 적용됨으로써 정확한 해답을 찾아낼 확률을 향상시킨다.

1) 〈[NAME] was born〉 〈on [ANSWER]〉
   - Abraham Lincoln was born in Kentucky, on February 12, 1809
   - Rosa Parks was born in Tuskegee, Alabama on February 4, 1913
2) 〈[NAME] was born〉 〈in [ANSWER]〉
   - Rosa Parks was born on this date in 1913.
   - Abraham Lincoln was born on February 12 in 1809
3) 〈[NAME] was〉 〈born in [ANSWER]〉
   - Mahatma Gandhi was a child in the town of Porbandar, India, where was born in 1869

[표 12]에서 LOCATION 타입의 확장 후 정확도가 확장 전보다 낮은 것은 확장된 패턴이 너무 일반적인 단어들만을 포함하고 있기 때문이다. 예를 들어, "〈ANSWER〉 and the 〈NAME〉"과 같은 패턴이 "〈[ANSWER] and〉 〈the [NAME]〉"으로 확장되면, middle의 단어가 너무 일반적이기 때문에 과적용의 결과로 정확도가 떨어지게 된다.

본 연구에서는 간단하게 패턴을 확장할 수 있는 방법을 제안하였고 실험을 통해 어느 정도 성능도 향상시킬 수 있음을 증명했다. 하지만 확장이 어휘-패턴에만 한정되기 때문에 확장될 만한 어휘-패턴이 추출되지 않는 질의 타입에 대해서는 이 방법을 적용할 수 없고 패턴이 너무 일반적인 단어들만으로 구성되는 경우에는 역효과를 보인다는 한계점이 있다.

또한, 질의 타입에 따라 추출된 패턴의 정확성이 높음에도 불구하고 MRR 값이 낮은 경우가 있는데 그 이유는 웹 문서 자체의 해답어 포함률이 현저히 낮기 때문으로 분석되었다. 즉, 패턴이 적용되기만 한다면 정확한 답을 추출할 가능성이 높지만, 패턴이 적용될 만한 문장이 아예 나타나지 않는 경우에 대한 대안이 필요하다.

그리고 해답어에 대한 제약이 없기 때문에 오류를 많이 생성시킬 수 있다. 이 같은 문제점과 앞에서 언급했던 DEATHYEAR 타입에 대한 시스템의 한계점을 해결하고 질의어와 해답어의 다양한 변이를 고려하기 위한 추가적인 대안이 필요하다. 따라서 앞으로 명칭-개체 태거를 사용해서 의미 태깅을 한다면 이러한 문제점과 서론에서 제시했던 패턴의 일반성과 과생산 문제를 해결할 수 있을 것으로 기대된다.

참고 문헌

[1] Soubbotin, M.M., S.M. Soubbotin, "Patterns of Potential Answer Expressions as Clues to the Right Answer", Proceedings of the TREC-10 Conference, pp.175-182, 2001

[2] Deepak Ravichandran, Eduard Hovy, "Learning Surface Text Patterns for a Question Answering System", Proceedings of 40th ACL 2002, pp.41-47, 2002

[3] Ellen Riloff, "Automatically Constructing a Dictionary for Information Extraction Tasks", Proceedings of the 11th National Conference on Artificial Intelligence, pp.811-816, 1993

[4] Ellen Riloff, "Automatically Generating Extraction Patterns from Untagged Text", Proceedings of 13th National Conference on Artificial Intelligence(AAAI-96), pp.1044-

1049, 1996

[5] Dekang Lin, Patrick Pantel, "DIRT - Discovery of Inference Rules from Text", Proceedings of 7th ACM SIGKDD International Conference on Knowledge Discovery and DataMing 2001, pp.323-328, 2001

[6] Roman Yangarber, Ralph Grishman, Pasi Tapanainen, Silja Huttunen, "Unsupervised Discovery of Scenario-Level Patterns for Information Extraction", Proceedings of the Conference in Applied Natural Language Processing ANLP-NAACL 2000, pp.282-289, 2000

[7] Eugene Agichtein, Luis Gravano, "Snowball : Extraction Relation from Large Plain-Text Collections", Proceedings of 5th ACM International Conference on Digital Libraries, pp.85-94, 2000

[8] Sergey Brin, "Extracting Patterns and Relations from the World-Wide Web", Proceedings of the 1998 International Workshop on the Web and Databases(WebDB'98), pp.172-183, 1998

[9] Deepak Ravichandran, Abraham Ittycheriah, Salim Roukos, "Automatic Derivation of Surface Text Patterns for a Maximum Entropy Based Question Answering System", Proceedings of the HLT-NAACL, 2003

[10] Mark A. Greenwood, Robert Gaizauskas, "Using a Named Entity Tagger to Generalise Surface Matching Text Patterns for Question Answering", Proceedings of the Workshop on Natural Language Processing for Question Answering(EACL03), pp.29-34, 2003

[11] Voorhees, E., "Overview of the Question Answering Track", Proceedings of the TREC-10 Conference, pp.157-165, 2001