한국어 텍스트의 개체 URI 탐지: 품사 태깅 독립적 개체명 인식과 중의성 해소

김영식○, 함영균, 김지성, 황도삼, 최기선
KAIST, KAIST, KAIST, 영남대학교, KAIST
twilight@kaist.ac.kr, hahmyg@kaist.ac.kr, jiseong@kaist.ac.kr, dshwang@yu.ac.kr, kschoi@kaist.edu

A Non-morphological Approach for DBpedia URI Spotting within Korean Text

Youngsik Kim○, Younggyun Hahm, Jiseong Kim, Dosam Hwang, Key-Sun Choi
KAIST, KAIST, KAIST, Yeungnam University, KAIST

요 약

URI spotting (탐지) 문제는 텍스트에 있는 단어열 중에서 URI로 대표되는 개체(entity)에 해당되는 것을 탐지하는 것이다. 이 문제는 두 개의 작은 문제를 순차적으로 해결하는 과제이다. 즉, 첫째는 어느 단어열이 URI에 해당하는 개체인가를 인식하는 것이고, 둘째는 개체 중의성 해소 문제로서 파악된 개체가 복수의 URI에 해당할 수 있는 의미적 모호성이 있을 때 그 URI중 하나를 선택하여 모호성을 해소하는 것이다. 이 논문은 디비피디아 URI를 대상으로 한다. URI 탐지 문제는 개체명 인식 문제와 비슷하나, URI(예를 들어 디비피디아 URI, 즉 Wikipedia 등재어)에 매핑될 수 있는 개체로 한정되므로 일반적인 개체명 인식 문제에서 단어열의 품사열이 기계학습의 자질로 들어가는 방법론과는 다른 자질을 사용할 수 있다. 이 논문에서는 한국어 텍스트를 대상으로 한국어 디비피디아 URI 탐지문제로서 SVM을 이용한 개체경계 인식 방법을 제시하여, 일반적 개체명 인식에서 나타나는 품사태거의 오류파급효과를 없애고자 한다. 또한 개체 중의성 해소 문제는 의미모호성이 주변 문장들의 토픽에 따라 달라지므로, LDA를 활용하며 이를 영어 디비피디아 URI탐지에서 쓰인 방법들과 비교한다.

주제어: 한국어 디비피디아 URI 탐지, 텍스트 어노테이션, 링크드 데이터, 개체명 인식

1. 서론

임의의 텍스트에서 오픈지식베이스인 LOD (linked open data) 개체를 자동으로 인식할 수 있는 능력은 웹상의 비정형 데이터를 링크드 데이터와 연결하여 지식베이스를 넓히는 데에 중요한 능력이다. 영문 디비피디아 URI 탐지의 예로서 DBpedia Spotlight[1]과 같은 과제에서 연구된 바 있다. 이 논문에서는 한국어로 된 지식 베이스를 넓히기 위하여 LOD 속의 한국어 디비피디아 URI 탐지에 관한 연구를 보고하고자 한다. DBpedia Spotlight를 여러 언어에 대해 동작하게 만들기 위한 연구는 있었지만[2], 아직 DBpedia Spotlight은 특정 언어에 대해서만 동작한다.

디비피디아 URI 탐지 과정은 개체 경계 인식과 개체 중의성 해소의 두 단계로 이뤄져 있다. 개체 경계 인식 단계에서는 텍스트 속에서 디비피디아 개체에 해당하는 단어열들을 검출하고, 개체 중의성 해소 단계에서는 검출한 단어열이 각각 어떤 디비피디아 URI와 연결되는 지를 구분한다.

영어 디비피디아 URI 탐지 관련 연구에서는 개체 경계 인식 단계를 그다지 깊게 다루지 않는 경향이 있다. DBpedia Spotlight만 봐도 개체 경계 인식을 단순히 사전 기반의 문자열 검색을 통해 해결한다. 영어에서는 이런 방법이 충분할 수 있으나, 한국어처럼 개체명의 평균 길이가 짧은 언어에서는 단순 문자열 검색보다 더 적합한 알고리즘이 필요하다. 이 논문을 통해 그 이유에 대해 논하고, 대안으로 SVM을 이용한 개체 경계 인식 알고리즘을 제시한다.

개체 중의성 해소 단계에서는 한국어 위키피디아 전체를 대상으로 생성한 LDA 토픽 모델을 이용한 방법을 영어 DBpedia Spotlight 연구에서 사용한 여러 베이스라인과 TF*ICF (Inversed Candidate Frequency) 문서 유사도에 기반을 둔 알고리즘과 비교하였다.

또한, 두 단계 모두에서 한국어 품사태거를 통해 얻은 품사 정보를 이용하는 방법과 그렇지 않은 방법의 성능을 비교함으로써 한국어 디비피디아 URI 탐지 과정에서 품사태거의 필요성 유무를 조사하였다.

2. 관련 연구

한국어 디비피디아 URI 탐지라는 특정 문제에 관한 이전 연구를 찾을 수 없었으나, 전통적인 개체명 인식(NER)에 대해 Hidden Marcov Models를 이용하거나 [3], statistical ML과 규칙 기반 알고리즘의 혼합을 이용하는 연구 [4] 등 관련 연구가 여럿 존재한다. 디비피디아 URI 탐지문제는 전통적인 개체명 인식과 다른데, 전통적인 개체명 인식에서는 개체에 장소나 단체 등의 클래스를 정해주는 반면 디비피디아 URI 탐지에서는 개체에

특정 URI를 붙여준다. 예를 들어, '로마' 라는 어휘에 대해 일반적인 개체명 인식 과제에서는 'City' 등의 클래스로 분류하지만, 디비피디아 URI 탐지 과제에서는 '로마, 이탈리아', '로마, 조지아', '로마, 일리노이' 등의 URI를 태깅해야 한다.

앞서 언급한 전통적인 개체명 인식에 관련된 두 연구에서는 한국어에서의 개체 경계 인식의 어려움에 대해 언급하지만, 정작 개체 경계 인식 과정과 개체 분류 과정에 대한 분리를 하지 않았다. 이 논문에서는 한국어에서의 개체 경계 인식을 독립적으로 다룰 필요가 있다는 것을 실험을 통해 제시한다.

3. 문제 정의

3.1 데이터 집합

원본 위키피디아 덤프는 2014년 1월 26일자 한국어 위키피디아 덤프를 이용하였고, [1]Wikipedia Extractor를 이용하여 덤프에서 모든 문서 및 문서의 평문에 포함된 링크의 정보를 추출했다. 이렇게 추출하여 얻어낸 원본 데이터 집합은 덤프 내의 모든 문서 $Article_1$, $Article_2$, ..., $Article_N$에 대해 다음 정보로 구성되어있다:

- $Article_x$의 제목 $Title_x$
- 연속된 문자 c_{x1}, c_{x2}... 로 이뤄진 $Article_x$ 의 평문 $Text_x$
- $Article_x$의 링크들 $Link_{x1}$, $Link_{x2}$... 각 링크 $Link_{xl}$ 는 또한 다음 정보를 가지고 있다:
  - 문자 단위의 오프셋으로 링크의 시작과 끝을 나타내는 $LinkStart_{xl}$와 $LinkEnd_{xl}$
  - 링크의 단어열 $Surface_{xl}$
  - 링크의 URI인 $LinkURI_{xl}$

데이터 셋의 모든 링크는 위키피디아 커뮤니티에 의해 수동 어노테이션 된 링크들이므로, (아래의 개체의 정의에 따른 예외처리 후) 데이터 집합 내 모든 링크를 정답 집합의 일부로 보았다.

3.2 개체의 정의

개체의 범위를 한국어 디비피디아로 한정짓고, 따라서 모든 한국어 위키피디아 문서의 URI만을 개체로 인정하였다. 이 과정에서 다음의 특수한 URI에 대해서는 예외처리가 필요하였다:

- 리다이렉션: 일부 위키피디아 URI는 고유 문서를 가지고 있지 않고, 자동으로 다른 URI로 리다이렉션 된다. 이런 URI는 리다이렉션 되는 URI로 대체하였다.
- 동음이의어: 일부 위키피디아 URI의 문서는 특정 개체를 의미하지 않고 같은 낱말을 공유하는 동음이의어들의 링크만 가진 경우가 있다. 이 URI들은 특정 개체를 가리키고 있지 않으므로, 개체로 인정하지 않았다. 따라서 데이터 집합 중 이런 URI로 링크된 링크들 역시 제외하였다.

3.3 개체 경계 인식의 범위

데이터 집합 내의 모든 링크의 단어열로 구성된 사전을 제작하고, 이 사전 내의 포함된 단어열만을 개체가 가질 수 있는 단어열의 범위로 제한하였다. 사전 SurfaceDict를 $\{C|\exists x \exists l(C = Surface_{xl})\}$의 집합으로 정의하고, 각 문서의 평문 $Text_x$에 있을 수 있는 모든 개체 경계의 집합 $Candidates_x$를 $\{(start,end)|c_{x\,start} \cdots c_{x\,end} \in SurfaceDict\}$로 정의하였다.

이 정의에 따라 $Text_x$에서 개체 경계 인식의 결과로 뽑힌 개체들은 $Candidates_x$의 부분집합이 되고, 또한 서로 위치상으로 겹치는 개체가 없어야 한다.

한국어에서는 영어의 대문자 사용처럼 어떤 단어열이 개체를 나타내는지를 알 수 있는 명백한 단서가 없기 때문에, 사전을 사용하지 않고 임의의 단어열이 개체에 해당하는지 알아내는 일은 현재 이 연구의 범위 밖이다.

3.4 개체 중의성 해소의 범위

앞서 개체의 범위를 한국어 위키피디아 문서의 URI로 제한했으므로, 개체 중의성 해소 단계에서 각 개체에 정해줄 수 있는 URI의 범위도 자연히 제한된다.

개체 경계 인식 단계처럼, 사전 없이 개체의 중의성 해소하는 일은 어렵고, 현재 이 연구의 범위 밖이다. 따라서 데이터 집합에서 1 개 이상의 링크에서 나온 (단어열, URI) 관계만을 개체 중의성 해소의 범위로 정했다. SemanticMeaning이라는 사전을

$$\left\{(s,uri)|s \in SurfaceDict \wedge \atop \exists x \exists l(Surface_{xl} = s \wedge LinkURI_{xl} = uri)\right\}$$

라는 집합으로 정의하고, 단어열 s를 가진 개체의 중의성 해소용 URI의 범위 $URICandidates_s$를 $\{uri|(s,uri) \in SemanticMeaning\}$의 집합으로 한정하였다.

4. 개체 경계 인식 실험

4.1 정답 집합

데이터 집합 내의 모든 링크를 정답 집합의 일부로 인정했지만, 그 역이 성립하는 것은 아니다. 일반적으로 위키피디아 문서 내에서 개체로 충분히 볼 수 있지만 링크되어있지 않은 단어열이 굉장히 많다. 간단한 관찰을 통해 1000번 중 1번만 링크되어있는 단어열도 개체로 생각할 수 있다는 것을 알 수 있었다.

이 때문에 수동 어노테이션을 통해 위키피디아 문서의 모든 개체의 위치와 URI에 대한 정보를 가진 정답 집합을 만들 필요가 있었다. 수동 어노테이션은 데이터 집합에 포함된 위키피디아 문서 중 일부를 대상으로, MUC-7 Named Entity Task Definition [6]의 가이드라인을 참조하여 문서의 개체에 해당하는 단어열의 위치 및 URI를

1) http://medialab.di.unipi.it/wiki/Wikipedia_Extractor

부여하는 방식으로 진행하였다.

정답 집합은 3 명의 첨삭자가 55 개의 위키피디아 문서에 대한 어노테이션 작업을 수행함으로써 만들어졌다. 본문이 한 문장으로 이뤄져있거나 링크의 나열로 구성되어있는 등 형태가 특이한 문서들을 걸러내기 위해 어노테이션 대상 문서로 20~50 개의 문장으로 이뤄져있고 링크:문장 개수 비율이 0.5:1 에서 4:1 사이인 것만 사용하였다. 첨삭된 55 개의 문서의 Candidates 합집합은 9,416 개의 개체와 90,221 개의 개체가 아닌 단어열로 이뤄져있다.

4.2 개체 경계 인식 과정 개요

한국어에서는 1~2 문자로 나타낼 수 있는 개체의 수가 매우 많고, 여러 개의 품사를 붙여 쓰기로 쓸 수 있기 때문에 일반적으로 $Text_x$의 모든 개체 경계의 집합 $Candidates_x$에는 그림 1에서 보다시피 서로 위치상으로 겹치는 개체가 굉장히 많다. 정답 집합에서도 개체의 수보다 개체가 아닌 개체 경계의 수가 약 10 배 많은, 그림 1과 비슷한 양상을 보인다.

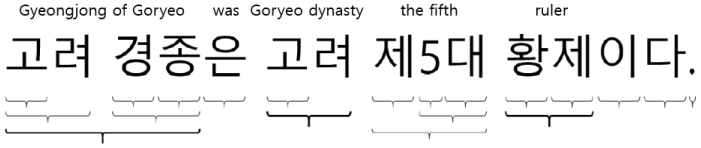

그림 1 "고려 경종은 고려 제5대 황제이다."의 Candidates 내의 모든 개체 경계들을 중괄호로 표시한 예시. 이 중 실제 개체의 개체 경계들은 굵은 중괄호로 구별하였다.

이를 해결하기 위한 가장 간단한 해결책은 먼저 청킹(chunking)을 시행한 뒤, 각 단어열이 개체인지 아닌지 결정하는 것이다. 하지만 한국어에 접두사, 접미사, 합성명사 등이 많이 사용되기 때문에 청킹 과정 자체가 쉽지 않다. 이 연구에서 사용한 최신 한국어 품사태거를 이용한 청킹에 의해 데이터 집합 내 링크의 10 % 이상이 품사와 링크의 경계가 일치하지 않아서 손실되었다는 결과가 이를 입증한다. 따라서 청킹을 이용하지 않는 개체 경계 인식 과정을 이용하였다.

먼저, Candidates 집합 내 모든 경계에 대해 4.3 장에서 설명할 여러 개체 판별 알고리즘 중 하나를 사용하여 경계가 실제 개체의 경계인지 판별한다.

이렇게 추출한 '개체라고 생각되는' 경계들의 부분집합의 경계들끼리 위치상으로 겹칠 수 있기 때문에, 이를 해소해야 한다. 디비피디아 개체는 거의 명사이기 때문에, 이 부분집합 내 겹치는 경계들은 대부분 한 경계가 다른 경계 속에 완전히 포함된 합성명사에 존재한다. 따라서 서로 겹치는 경계에 대해 항상 가장 긴 경계만 사용하였다.

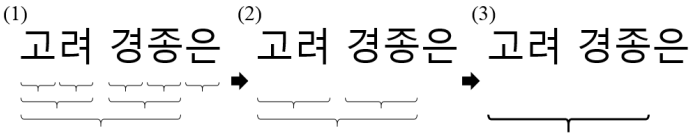

그림 2 4.2 장의 개체 경계 인식 과정. 전체 Candidates 집합으로 시작하여 개체라고 생각되는 경계들의 부분집합을 만든 뒤에, 겹치는 경계들 중 가장 긴 경계만 사용한다.

4.3 개체 판별 알고리즘

베이스라인 개체 판별의 베이스라인으로 Candidates 내의 모든 경계를 개체로 인정하는 방식을 사용하였다. '가' 나 '도' 같은 매우 흔한 단어열들도 Candidates 에 포함되어 있기 때문에 (즉, 이와 같은 단어열을 가진 링크가 데이터 집합 내에 존재하기 때문에) 베이스라인의 precision은 굉장히 낮을 것이라고 예상할 수 있다.

어절 기반 DBpedia Spotlight에서 사용된 개체 경계 인식 과정을 재현한 방법으로, Candidates 내의 경계 중 단어열의 바로 앞과 뒤 모두에 공백 또는 특수문자가 있는 경계들만 개체로 인정하는 알고리즘이다.

접두·접미 기반 어절 기반 알고리즘의 확장으로, 경계의 바로 뒤에 오는 문자들이 공백과 특수문자 외에도 데이터 집합 내의 링크 중 약 95% 정도를 커버하는 27 가지의 한국어 단어열 중 하나와 일치하면 개체로 인정하는 알고리즘이다. 이 27 가지 단어열은 모두 '은', '는', '이다' 등 명사구 뒤에 붙는 흔한 접미사들이지만, 이 알고리즘에서는 품사 분석을 하지 않고 단순히 문자열 매칭을 이용한다.

품사 기반 최신 한국어 품사태거를 이용하여 전체 단어열을 품사 단위로 쪼갠 뒤, Candidates 내의 경계 중 경계를 하나 이상의 완전한 품사로 나타낼 수 있고 모든 품사가 명사구(nc, np, nn, nb)나 명사 파생 접미사(xsn)인 경계만 개체로 판별하는 알고리즘이다.

이 알고리즘의 기준으로는 데이터 집합의 완전한 품사들로 표현되는 링크 중 95 % 이상을 선택할 수 있다.

SVM 기반 Support Vector Machine 방식의 여러 가지 커널을 이용하여 정답 집합 중 일부를 가지고 학습한 뒤, 학습된 모델을 가지고 Candidates 내의 각 경계가 개체인지 판별하는 알고리즘이다. SVM에 이용하는 자질은 문자열일 수 없기 때문에[2], 경계의 단어열과 주변 문자들을 이용한 조건부 확률식을 자질로 사용하였다.

두 함수 C(s): '경계 s는 어떤 조건 C를 만족한다'와 E(s): '경계 s는 개체이다'를 정의하면, SVM에 이용할 자질을 계산하는 이상적인 조건부 확률식은 조건 C를 만족하는 경계가 개체일 확률, 즉 $P(E(s)|C(s))$일 것이다. 하지만 정답 집합의 크기가 작기 때문에 이 확률식을 이용하여 만든 자질은 매우 빈약(sparse)할 수밖에

2) Levenshtein distance를 이용하여 문자열 기반 커널을 구현할 수 있으나, 경계의 단어열 자체가 닮았다고 개체일 확률도 비슷한 것이 아니기 때문에 URI 탐지 문제에서는 이용할 수 없다.

없다.

이를 해결하기 위해, 전체 데이터 집합 내의 링크의 분포가 대략적으로 전체 개체의 분포와 비슷할 것이라는 가정을 세우고, 자질값 생성의 범위를 전체 데이터 집합으로 넓혔다. Link(s): '경계 s는 데이터 집합에서 링크로 태깅 되어있다' 라는 함수를 정의하여, 자질값으로 $P(E(s)|C(s))$ 대신 $P(Link(s)|C(s))$를 이용하였다.

SVM 기반 알고리즘에서는 경계의 단어열과 주변 문자들을 기반으로 한 다음 7개의 자질을 이용하였다. 모든 자질은 문자 기반으로, 품사 분석을 필요로 하지 않는다.

1. 경계의 단어열
2. 경계의 접두 단어열 (경계 단어열의 첫 단어와 같은 어절에 있는, 경계의 앞에 붙은 단어열)
3. 경계의 접미 단어열 (경계 단어열의 마지막 단어와 같은 어절에 있는, 경계의 뒤에 붙은 단어열)
4. 경계의 단어열과 접두 단어열
5. 경계의 단어열과 접미 단어열
6. 경계의 단어열과 경계의 바로 앞 어절
7. 경계의 단어열과 경계의 바로 뒤 어절

다음의 SVM 커널을 가지고 실험하였고, 모든 SVM 알고리즘은 scikit-learn 프레임워크 [7]를 이용하였다. 수동으로 정한 패러미터 외에는 모두 이 프레임워크에서 정한 기본값을 사용하였다.

- SVM-1: linear 커널, 두 클래스에 같은 무게
- SVM-2: linear 커널, 두 클래스에 자동으로 무게 부여
- SVM-3: 3-degree polynomial 커널, 두 클래스에 같은 무게
- SVM-4: 3-degree polynomial 커널, 개체:non-개체 클래스에 3:1의 무게 부여
- SVM-5: 3-degree polynomial 커널, 개체:non-개체 클래스에 6:1의 무게 부여
- SVM-6: 3-degree polynomial 커널, 두 클래스에 자동으로 무게 부여
- SVM-7: RBF 커널, 두 클래스에 같은 무게
- SVM-8: RBF 커널, 두 클래스에 자동으로 무게 부여

4.4 성능 측정 방식

각 개체 판별 알고리즘을 이용한 개체 경계 인식의 성능을 측정하기 위해 CoNLL-2003 shared task [8]에 나온 성능 측정 방식을 이용하였다. 정답 집합을 5개의 균등한 셋으로 쪼개어 5-fold cross-validation을 시행하였다.

4.5 실험 결과

표 1 각 개체 판별 알고리즘의 개체 경계 인식 성능

알고리즘	Precision	Recall	F-score
베이스라인	21.03	92.85	34.28
어절 기반	50.55	35.54	41.69
접두·접미 기반	62.70	78.71	69.74
품사 기반	55.64	91.20	69.11
SVM-4	76.83	85.26	**80.81**

표 1의 결과에 따라 SVM 기반 개체 경계 인식의 성능이 전반적으로 베이스라인, 그리고 규칙 기반 개체 경계 인식보다 좋은 것을 알 수 있다. 특히, 서로 다른 규칙을 사용하는 접두·접미 기반 알고리즘과 품사 기반 알고리즘의 F-score가 비슷하나 SVM 기반 알고리즘의 F-score보다는 낮다. 따라서 규칙 기반 알고리즘으로는 성능에 한계가 있으며, 이 한계를 SVM같은 기계학습 알고리즘을 통해 개선시킬 수 있다.

특히 SVM 기반 알고리즘의 recall이 품사 기반의 알고리즘과 큰 차이를 보이지 않는 점은 고무적인데, 현재의 한국어 품사태거의 품사 정보를 이용하여 감지할 수 있는 개체만큼을 문자 기준 자질만을 사용한 SVM으로도 충분히 감지할 수 있다는 것을 의미하기 때문이다.

SVM 기반 알고리즘 중에서는 SVM-4(3-degree polynomial 커널과 3:1의 무게 부여)의 성능이 가장 좋았으나, 다른 SVM 기반 알고리즘의 성능과 큰 차이는 보이지 않았다.

모든 경계를 다 개체로 인정한 베이스라인의 recall이 100%에서 큰 차이를 보이는 까닭은 명사구와 접두·접미사가 합쳐져 (문맥에 맞지 않는) 또 다른 개체의 단어열을 만드는 경우 때문이다. 예를 들어, '일본' 에 접미사 '도' 가 합쳐져 '일본도' 가 되는 경우에 베이스라인으로 '일본' 이라는 개체를 인식하지 못한다.

5. 개체 중의성 해소 실험

개체 중의성 해소 단계의 시작점은 개체 경계 인식 과정에서 나온 개체 경계의 집합이기 때문에, 개체 경계 인식 과정에서 생긴 오류는 모두 개체 중의성 과정으로 전파된다. 개체 경계 인식 과정에서 뽑힌 개체가 아닌 경계들은 어쩌면 개체 중의성 해소에서 제거될 수도 있겠으나, 이를 통해 성능을 높이기 위해서는 경계의 단어열이 해당 문장에서 가지는 의미를 파악하는 능력이 필요하다.

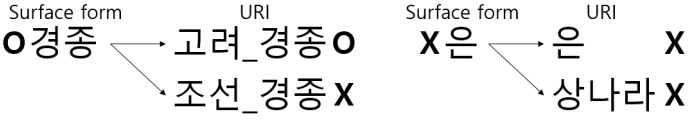

그림 3 개체 경계 인식 과정에서 정확히 선택한 개체를 대상으로 개체 중의성 해소 과정에서 정확한 URI를 연결해야 정답으로 인정되고, 개체 경계 인식 과정에서 잘못 인식된 개체는 중의성 해소 과정에서도 오류로 남는다.

5.1 정답 집합

4.1 장에서 사용한 정답 집합을 그대로 사용하되, 각

개체의 위치 정보뿐만 아니라 URI 정보까지 추가하였다.

5.2 개체 중의성 해소 과정 개요

단어열 s를 가지는 개체가 가질 수 있는 URI의 전체 집합은 3.4 장에서 정의한 URICandidates_s 이다. 따라서 개체 중의성 해소 과정은 모든 개체에 대해 각 개체의 URICandidates 내의 URI 중 하나를 선택하는 일이 되고, 각 URI에 점수를 매기는 기준 및 방식이 개체 중의성 해소 알고리즘이 된다.

개체 중의성 해소 알고리즘이 이용할 수 있는 정보는 개체 및 개체의 맥락 외에 데이터 집합에서 계산할 수 있는 URI별 등장 빈도나 URI별 출현 문서의 집합 등이 있다.

5.3 개체 중의성 해소 알고리즘

이 실험에서 사용한 개체 중의성 해소 알고리즘 및 베이스라인은 LDA말고는 모두 DBpedia Spotlight [1]에서 이용한 것들이다. 이 논문에서 같은 방법을 사용함으로써 실험 결과를 영문 디비피디아 URI 탐지의 결과와 비교할 수 있다.

(모든 개체 중의성 해소 알고리즘은 문서 d에 포함된, 단어열 s를 가진 개체 e를 가정한다)

무작위 베이스라인 이 베이스라인은 각 개체에 대해 URICandidates_s 내의 URI 중 무작위로 하나를 선택한다. 무작위 베이스라인은 개체 하나마다 이 과정을 반복하기 때문에, 같은 문서 내 같은 단어열을 가진 개체들에 서로 다른 URI가 부여될 수 있다.

이 베이스라인의 목적은 정답 집합 내의 개체가 가질 수 있는 중의성의 정도를 측정하는 것으로써, 개체의 평균 중의성의 정도가 높아질수록 자연히 무작위 베이스라인의 성능이 낮아질 것이다.

빈도 베이스라인 단어열 s에 대해 데이터 집합 내에서 같은 단어열을 가진 링크가 흔하게 가지고 있는 URI가 확률적으로 단어열 s를 가진 개체의 옳은 URI일 가능성이 높다. 예를 들어 '로마'라는 단어열을 가진 링크 중 1,423 개가 '로마, 이탈리아'를 향하고 4 개가 '로마, TV시리즈'를 향한다면, 임의의 문서에 나오는 '로마'가 '로마, 이탈리아'를 의미할 확률이 더 높을 것이다.

빈도 베이스라인은 이를 토대로 URICandidates_s 내의 URI 중 데이터 집합 내 단어열 s를 가진 가장 많은 링크의 URI인 것을 선택한다.

TF*ICF 이 알고리즘은 TF*ICF로 계산한 문서 유사도를 이용하여 단어열 s를 가진 링크 중 문서 d와 평균적으로 가장 비슷한 위키피디아 문서들에 포함된 링크들이 향하는 URI를 선택한다.

일반적으로 특정 단어열을 가진 링크가 등장하는 문서의 집합 자체가 위키피디아 전체 문서에 비해 작기 때문에 URI별 IDF(Inverse Document Frequency) 지표의 변별력이 떨어지는 단점이 있다. ICF(Inverse Candidate Frequency)는 전체 문서의 집합이 아닌 단어열 s를 가진 링크를 포함한 문서만을 이용함으로써 이 문제를 해결한다.

일단 URICandidates_s 내의 각 URI u에 대해 위키피디아의 문서 중 단어열 s와 URI u를 가진 링크가 포함된 문서들의 평문을 단순히 이어붙이는 식으로 만든 거대한 문서 $\text{TextOfSense}_{s,u}$를 하나씩 만들었다. 문서 간 유사도를 계산하기 위해 문서 d와 모든 TextOfSense 문서를 bag-of-words 형태로 변환한 뒤에, 각 단어의 TF*ICF 점수를 계산하여 이를 벡터로 변환하였다. 마지막으로 이렇게 변환된 모든 $(d, \text{TextOfSense}_{s,u})$ 벡터쌍의 cosine을 계산하여, d와 가장 비슷한 TextOfSense 문서에 해당되는 URI를 선택하였다. 이 과정에서 단어 w의 ICF 점수는 다음 공식에 의해 계산되었다.

$$ICF(w) = \log\left(\frac{TextOfSense\text{ 문서 개수}}{w\text{를 포함한 }TextOfSense\text{ 문서 개수}}\right)$$

문서를 bag-of-words 형태로 변환하기 위한 두 가지 단어 선정 방식을 이용하였다. 첫 번째 방식은 품사태거를 이용하여 모든 연속된 명사구를 단어로 선정하여 bag-of-nouns를 만드는 방식이며, 두 번째 방식은 개체 경계 인식 과정을 통해 얻은 문서의 개체들을 단어로 선정하여 bag-of-surfaces를 만드는 방식이다. 두 가지 방식을 이용한 TF*ICF 알고리즘을 각각 **TF*ICF-nouns**와 **TF*ICF-surfaces**라고 지칭한다. 이 중 TF*ICF-surfaces 알고리즘은 품사태거를 이용하지 않는다.

LDA 이 알고리즘은 TF*ICF 알고리즘처럼 문서 간 유사도를 기반으로 URI를 선택하나, LDA를 한국어 위키피디아 전체에 대해 실행시켜 얻은 각 문서의 토픽 분포를 기준으로 유사도를 계산한다.

이 논문에서는 LDA를 사용하기 위해 mallet 2.0.7 프레임워크 [9]를 이용하였는데, 각 문서를 bag-of-words 형태로 변환한 뒤, mallet의 입력 형식에 맞추기 위해 bag-of-words 안의 각 단어를 로마자로 변환하였다. 문서별 토픽분포 토픽 수를 300 개로 설정하여 LDA 알고리즘을 통해 계산하였다. 마지막으로 URICandidates_s 내의 각 URI u에 대해 위키피디아의 문서 중 단어열 s와 URI u를 가진 링크가 포함된 문서들의 문서 d와의 KL-divergence를 계산하여, KL-divergence 값이 가장 낮게 나오는 문서에 포함된 URI를 선택한다.

TF*ICF 알고리즘에서처럼 문서를 bag-of-words 형태로 변환하는 두 가지 방식 모두를 사용하였으며, 여기서는 각각 **LDA-nouns**와 **LDA-surfaces**로 부르기로 하였다.

5.4 성능 측정 방식

4.4 장에서 사용한 5-fold cross-validation과 같은 정답 집합에 대해, 4.3 장의 모든 개체 판별 알고리즘을 이용하여 각각 뽑아낸 개체의 부분집합을 가지고 각 개체 중의성 해소 알고리즘을 적용하였다. 총 12가지 개체 판별 알고리즘과 6 가지 중의성 해소 알고리즘을 가지고 테스트 셋에 대해 총 72 회의 URI 탐지를 실행하였다.

5.5 실험 결과

표 2 SVM-4 개체 경계 인식 알고리즘을 통해 추출한 개체 부분집합을 기반으로 한 각 개체 중의성 해소 알고리즘의 성능

알고리즘	Precision	Recall	F-score
무작위 베이스라인	53.68	49.55	51.52
빈도 베이스라인	76.81	70.92	73.72
TF*ICF-nouns	76.76	70.85	73.66
TF*ICF-surfaces	76.65	70.77	73.57
LDA-nouns	78.83	72.79	75.66
LDA-surfaces	78.42	72.41	75.27

가장 성능이 좋았던 SVM-4 개체 경계 인식 알고리즘에서 감지한 개체들에 대한 개체 중의성 해소 알고리즘 실행 결과가 표 2에 나와 있다. 개체 경계 인식과 개체 중의성 해소 과정을 모두 거친 결과이기 때문에, 표 2의 결과는 곧 한국어 URI 탐지 자체의 실험 결과이다.

실험은 모든 개체 경계 인식 알고리즘을 가지고 하였으나, 전반적으로 개체 경계 인식 알고리즘의 성능과 그 후의 개체 중의성 해소 알고리즘의 성능이 비례하는 것 외에 특이사항은 없었다.

무작위 베이스라인과 빈도 베이스라인의 F-score가 각각 51.52와 73.72인데, F-score 80.81인 SVM-4의 결과에서부터 시작했다는 점을 고려하면 둘 다 굉장히 높은 수치임을 알 수 있다. 대조적으로 영문 DBpedia Spotlight 연구 [1]에서의 두 베이스라인의 성능이 각각 17.77, 55.12에 불과하다. 이런 큰 차이는 DBpedia Spotlight 연구에서 사용한 테스트 데이터의 중의성의 정도가 특별히 높았던가, 아니면 전반적인 영어/한국어 위키피디아 개체의 중의성의 차이에서 비롯한 것이라 예상한다.

LDA 관련 알고리즘의 성능이 다른 알고리즘에 비해 미세하게 좋은 것을 알 수 있지만, 이 차이가 의미 있는 것인지에 대해서는 추가적인 연구가 필요하다.

6. 결론 및 추후 연구

이 논문에서 한국어에 대한 디비피디아 URI 탐지를 위한 두 단계인 개체 경계 인식과 개체 중의성 해소 모두를 해결하기 위해 여러 알고리즘을 비교하였다. 실험 결과를 통해 한국어의 여러 언어적 특성 때문에 한국어에서의 개체 경계 인식 문제가 충분히 독립적으로 다룰 가치가 있을 만큼 어렵고, 또한 전반적인 URI 탐지 과정에서 품사태거를 통해서 얻은 품사 정보가 품사태거를 URI 탐지 파이프라인에 추가할 만큼의 효과를 주지 않는다는 사실을 알 수 있다. 아직은 부족한 품사태거의 성능 및 소요 시간을 고려하면 이는 바람직한 결과라 할 수 있다.

이 논문에서 사용한 정답 집합이자 테스트 집합은 모두 한국어 위키피디아의 문서에서 추출한 데이터로 이뤄져 있다. 하지만 디비피디아 URI 탐지의 목적이 일반적인 텍스트에 개체 정보를 태깅 하는 것인 만큼, 신문 기사 등 외부 데이터로 실험하여 개체 중의성 해소 과정 중 TF*ICF 및 LDA 알고리즘이 베이스라인에 비해 어떤 성능을 보이는지 조사해야 한다.

이렇게 만들어진 URI 탐지 파이프라인을 이용하여 추후 한국어 개체명 인식 과제를 위한 silver-standard 코퍼스 구축, 그리고 한국어 텍스트를 입력으로 하여 디비피디아를 위시한 LOD 개체 URI를 태깅 하는 웹서비스를 만들 계획이다. 이를 위해서는 파이프라인의 실행 시간과 필요 메모리에 대한 고려가 필요하다.

사사

본 연구는 미래창조과학부 및 정보통신기술연구진흥센터의 정보통신•방송 연구개발사업의 일환으로 수행하였음 [10044494, WiseKB: 빅데이터 이해 기반 자가학습형 지식베이스 및 추론 기술 개발]

참고문헌

[1] Mendes, Pablo N., et al. "DBpedia Spotlight: shedding light on the web of documents." Proceedings of the 7th International Conference on Semantic Systems. ACM, 2011.

[2] Daiber, Joachim, et al. "Improving efficiency and accuracy in multilingual entity extraction." Proceedings of the 9th International Conference on Semantic Systems. ACM, 2013.

[3] Chung, Euisok, Yi-Gyu Hwang, and Myung-Gil Jang. "Korean named entity recognition using HMM and CoTraining model." Proceedings of the sixth international workshop on Information retrieval with Asian languages-Volume 11. Association for Computational Linguistics, 2003.

[4] Seon, Choong-Nyoung, et al. "Lightweight Named Entity Extraction for Korean Short Message Service Text." KSII Transactions on Internet and Information Systems (TIIS) 5.3 (2011): 560-574.

[5] Florian, Radu, et al. A statistical model for multilingual entity detection and tracking. IBM THOMAS J WATSON RESEARCH CENTER YORKTOWN HEIGHTS NY, 2004.

[6] Chinchor, Nancy, and Patricia Robinson. "MUC-7 named entity task definition."Proceedings of the 7th Conference on Message Understanding. 1997.

[7] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

[8] Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: language-independent named entity recognition. In Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003 - Volume 4 (CONLL '03), Vol. 4. Association for Computational Linguistics, Stroudsburg, PA, USA, 142-147. DOI=10.3115/1119176.1119195 http://dx.doi.org/10.3115/1119176.1119195

[9] McCallum, Andrew Kachites. "MALLET: A Machine Learning for Language Toolkit." http://mallet.cs.umass.edu. 2002.

부록

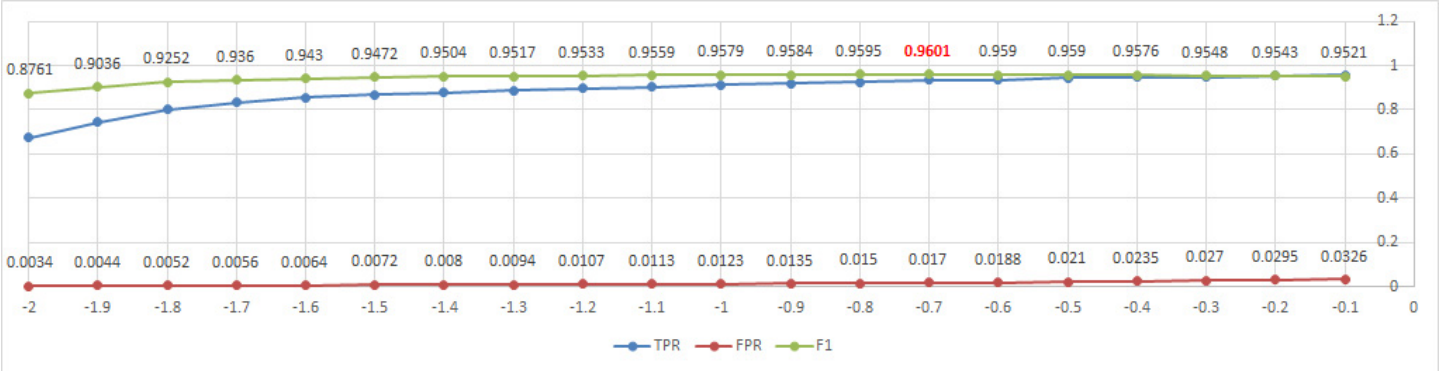

그림 5 햄과 스팸 비율 10:3 환경 실험 결과

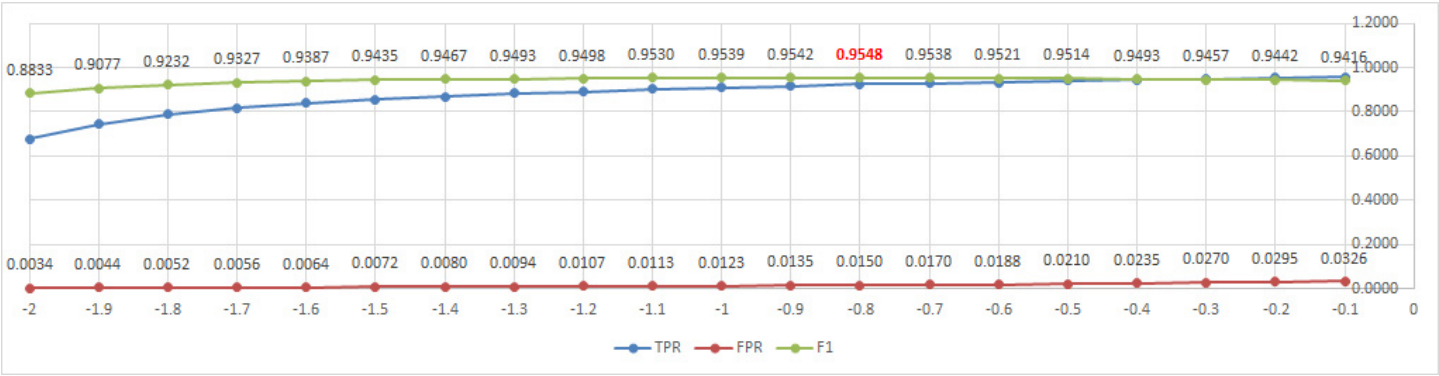

그림 6 햄과 스팸 비율 10:2 환경 실험 결과

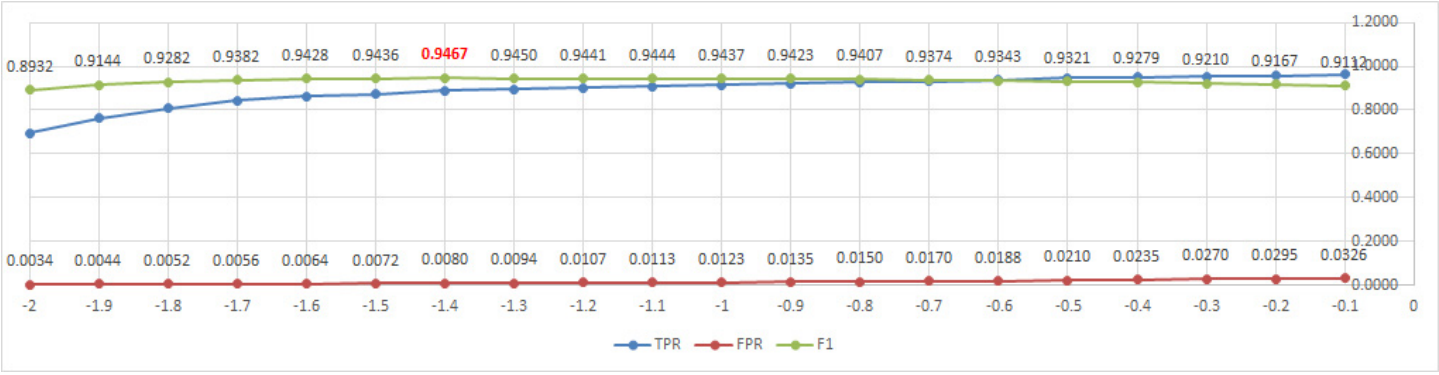

그림 7 햄과 스팸 비율 10:1 환경 실험 결과