구묶음을 반영한 한국어 의존 구조 말뭉치 생성

남궁영[O†], 김창현[‡], 천민아[†], 박호민[†], 윤호[†], 최민석[†], 김재균[†], 김재훈[†]
한국해양대학교[†], 한국전자통신연구원[‡]
young_ng@kmou.ac.kr, chkim@etri.re.kr, minah2018@kmou.ac.kr, homin@hanmail.net,
4169615@naver.com, ehdgus5136@naver.com, jgk20000@naver.com, jhoon@kmou.ac.kr

Building Korean Dependency Treebanks Reflected Chunking

Young Namgoong[O†], Chang-Hyun Kim[‡], Min-Ah Cheon[†], Ho-Min Park[†],
Ho Yoon[†], Min-Seok Choi[†], Jae-Kyun Kim[†], Jae-Hoon Kim[†]
Korea Maritime and Ocean University[†], Electronics and Telecommunications Research Institute[‡]

요 약

의존 구문 분석은 문장 구성 요소의 위치에 제약이 적고 생략에도 유연하게 대처할 수 있어 한국어 구문 분석에 적합하다. 하지만 의존 구문 분석을 수행할 때 지배소를 결정해야 할 노드 수가 많으면 계산의 복잡도가 올라가고, 각 노드의 지배소를 결정할 때 방향성 문제가 있어 구문 분석에 모호함을 더한다. 이때 지배소 후위 원칙을 엄격하게 적용할 경우 구문적 중심어와 의미적 중심어가 불일치하는 문제가 발생한다. 이러한 문제들을 해소하기 위해 구묶음을 수행한 문장으로 구문 분석을 수행할 수 있다. 따라서, 본 논문에서는 기존의 의존 구문 말뭉치를 말덩이 기반의 의존 구문 말뭉치로 변환하는 알고리즘을 기술하고, 이에 따라 구축한 말뭉치와 기존의 말뭉치를 정량적으로 비교한다.

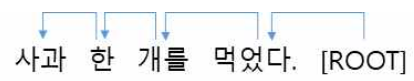
주제어: 말뭉치 변환, 구문분석, 부분 구문분석, 구묶음, 말덩이

1. 서론

구문 분석은 문장 구성성분들의 관계를 파악하는 과정을 말한다. 구문 분석을 통해 문장의 구조를 결정하고 이를 통해 문장의 의미적 중의성을 해소할 수 있다.

자연언어처리에서 구문 분석은 문장을 바라보는 관점에 따라 크게 구 구조 구문 분석(constituency parsing)과 의존 구문 분석(dependency parsing)으로 나뉜다. 구 구조 구문 분석은 문장을 구성 성분들의 결합으로 분석하는 방식이다. 반면 의존 구문 분석은 문장 성분 간의 지배소-의존소 관계를 파악함으로 문장의 구조를 분석하는 방식이다. 따라서 문장을 구성하는 요소의 위치에 제약이 적고 생략에도 유연하게 대처할 수 있어 한국어 구문 분석에 적합하다.

하지만 의존 구문 분석은 문장에서 지배소 및 의존 관계를 결정해야 할 노드 수가 많을수록 구문 분석기의 계산 복잡도는 높아지고 분석에 있어 모호함이 생긴다. 특히 지배소를 결정할 때 방향성 문제가 발생하는데, 기존의 구문 분석 방식대로 지배소 후위 원칙을 각 노드에 엄격하게 적용할 경우, 다음과 같이 구문적 중심어와 의미적 중심어가 불일치 하는 상황이 발생하게 된다.



(a) 예시 1

(b) 예시 2

그림 1. 의존 구문 분석 예시

그림 1-(a)에서 서술어 '먹었다'의 목적어는 의미적으로 '사과'이지만, 기존의 방식대로 의존 구문 분석을 하면 '개를'이 '먹었다'의 의존소가 된다. 그림 1-(b)의 예문 역시 문장 전체의 의미적 중심어는 용언 '하-'를 어간으로 갖는 '할'이지만, 기존의 방식대로 구문 분석을 하면 'ROOT'를 지배소로 갖는 노드는 문장의 가장 마지막에 있는 보조 용언 '있다.'가 된다.

이 같은 문제를 해결하기 위해 문장 내의 형태소들을 하나의 의미있는 구성 성분인 **말덩이**(chunk)로 구묶음한 뒤 구문 분석을 수행할 수 있다[1-3]. 말덩이란 인간이 한번에 받아들이는 언어의 단위로, 문법적·의미적으로 하나의 기능을 수행하며, 연속성, 비중첩성, 비재귀성이라는 특징을 가진다[4,5]. 또한, 형태소 분석된 문장에 대해 말덩이 단위를 인식하고 표지를 부여하는 과정을 구묶음(chunking)이라고 한다[1,6].

말덩이는 내용어 말덩이(content chunk)와 기능어 말덩이(function chunk)로 나누어지며, 하나의 내용어 말덩이 또는 하나의 내용어 말덩이와 하나 이상의 기능어 말덩이가 모여 하나의 문장 성분을 이룬다. 따라서 완전한 구묶음 이후의 문장은 그림 2와 같이 한국어의 7가지 **문장 성분**[1]으로 표현이 가능하다.

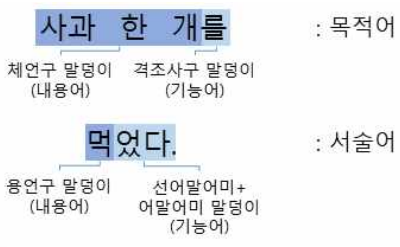
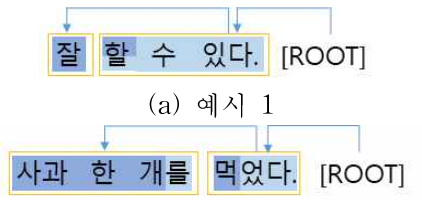


그림 2. 구묶음한 문장의 말덩이 및 문장 성분 예시

1) 주어, 서술어, 목적어, 보어, 관형어, 부사어, 독립어

이처럼 말덩이 단위로 구문 분석을 하게 되면 구문 분석이 문장 성분 단위로 이루어지므로 그림 3과 같이 구문 분석을 수행할 노드 수가 적어져 구문 분석기의 속도 및 정확도가 향상될 수 있다.



(a) 예시 1

(b) 예시 2

그림 3. 말덩이 단위로 수행한 의존 구문 분석 예시

또한, 기존의 방식과 달리 그림 3에서처럼 구문적 중심어와 의미적 중심어가 일치하므로, 한국어의 지배소 후위 원칙을 위배하지 않으면서 의미적으로도 어색하지 않은 구문 분석이 이루어질 수 있다. 이는 구문 분석 단계에서 구문 분석 이후에 이루어지는 의미 분석(semantic analysis)을 함께 수행할 수 있다는 면에서 의의가 있다. 따라서, 말덩이 기반의 구문 분석 말뭉치가 있으면 더욱 효과적인 구문 분석이 가능하다.

일반적으로 대량의 말뭉치를 구축하는 일에는 시간적, 비용적 측면에서 큰 노력이 필요하다. 따라서 본 논문에서는 **말덩이 기반의 의존 구문 말뭉치**를 구축하기 위해 기존의 의존 구문 말뭉치로부터 변환할 수 있는 알고리즘을 기술한다.

본 논문의 구성은 다음과 같다. 2장에서는 세종 구문 분석 말뭉치에서 의존 구문 말뭉치로의 변환 방안 및 한국어 의존 구문 말뭉치 구축에 관한 연구가 어떻게 진행되어 왔는지 서술한다. 3장에서는 기존의 의존 구문 말뭉치와 말덩이 기반의 의존 구문 말뭉치의 특성 및 차이를 소개한다. 이를 바탕으로 4장에서는 기존의 의존 구문 말뭉치를 말덩이 기반의 의존 구문 말뭉치로 변환하는 방법에 대해 기술한다. 5장에서는 변환된 말뭉치에 대한 분석을 하며, 6장에서는 결론 및 향후 연구에 대해 논한다.

2. 관련 연구

의존 구문 분석이 주목을 받으면서 기존의 구 구조를 의존 구조로 변환하는 연구가 진행되었다. 영어의 경우 구절마다 중심어 전파 규칙을 정하고, 이에 따라 의존 구조로 변환하는 방법을 사용한다[8]. 한국어 역시 중심어를 찾는 규칙을 이용해 구 구조의 Penn Korean Treebank, KAIST Treebank를 의존 구조로 변환하는 연구가 이루어 졌다[9].

UD(Universal Dependency)[7]의 관계 표지명을 한국어 의존 구문 분석에 적용하는 방법 및 한국어 UD 말뭉치 구축 방안도 활발히 연구되고 있다. UD는 2015년에 시작된 다양한 유형의 언어 간에 일관되게 통용될 수 있는 언어 표지를 개발하는 프로젝트이다. 우리나라에서도 기존에 구축된 세종 계획의 구 구조 기반 구문 분석 태그를 UD의 관계 태그로 변환하는 방안을 논의하고 있으며[10], 이를 기반으로 세종 구문 태그 부착 말뭉치를 UD 말뭉치로 변환하는 방법도 연구되었다[11]. 또한, 세종 구문 분석 말뭉치[12]를 의존 구문 말뭉치로 변환할 때 중심어를 설정하기 위한 중심어 전파 규칙에 관한 연구도 진행되었다[13]. 이때 기존의 방식대로 중심어 후위 원칙을 엄격하게 적용한 경우와 중심어 전위까지 허용하여 유연하게 적용한 경우의 말뭉치를 각각 구축하고, 유연한 중심어 후위 원칙에 따라 구축한 말뭉치도 구문 분석기의 성능을 크게 저하하지 않음을 시사했다. 본 논문에서는 [13]의 유연한 중심어 후위 원칙에 따라 구축한 말뭉치를 토대로 하여 말덩이 기반의 의존 구문 말뭉치로 변환하는 알고리즘을 기술하고자 한다.

3. 말덩이 기반 의존 구문 말뭉치 표기 방법

이 장에서는 변환 알고리즘을 설명하기에 앞서 한국어 의존 구조 말뭉치와 말덩이 기반 의존 구조 말뭉치의 형식 및 차이점을 다루고자 한다.

의존 구문 말뭉치를 구축할 때 UD의 CoNLL-U 형식[2]을 따르는 것이 보편화 되어있다. 구문 분석 말뭉치에 적용되는 UD의 CoNLL-U 형식은 그림 4와 같이 10개의 열로 이루어지며, 이 중 HEAD 열은 해당 토큰의 지배소 순번을, DEPREL 열은 지배소와의 의존 관계 표지를 기술한다.

[13]의 유연한 중심어 후위 원칙의 말뭉치는 CoNLL-U 형식을 따르며 그림 4와 같이 구성되어 있다. 세종 구문 분석 말뭉치와 마찬가지로, 한 행, 즉 구문 분석의 한 노드가 되는 토큰은 주로 어절 단위이며, 띄어쓰기가 있는 경우 문장 부호나 쌍이 있는 기호를 분리하여 하나의 토큰으로 취급하였다.

반면, 말덩이 기반의 의존 구조 말뭉치는 한 행이 내용어 말덩이와 기능어 말덩이로 이루어진 **문장 성분** 단위이다. 말덩이 기반의 의존 구조 말뭉치 역시 기본적으로 CoNLL-U 형식을 따르며 그림 5와 같이 이루어져 있다. 기존과 다른 점은 언어 형식을 기입하는 FORM 열에 말덩이의 특성을 반영하여 내용어 말덩이(conts)와 기능어 말덩이(func)로 나누어 표기하였다. 또한, 말덩이 표지를 기입하는 CHUNKTAG 열이 추가되었다. 지면 관계상 한국어에는 잘 사용되지 않는 FEATS, DEPS, MISC 열은 제외하고 기술하였다.

4. 변환 과정 및 알고리즘

이 장에서는 [13]의 말뭉치를 말덩이 기반의 의존 구문 말뭉치로 변환할 때 핵심이 되는 변환 과정에 관해 설명하고, 실제 말뭉치 변환에 사용한 알고리즘을 소개한다. 변환 과정을 직관적으로 설명하기 위해 그림 4와 그림 5 중 ID, FORM, XPOSTAG, CHUNKTAG, HEAD 열만을 간략히 표 1과 표 2로 나타내고 이를 예로 들어 변환 방법을 기술한다. 변환 과정을 설명하는 데 사용할 예문은 그림 4의 문장과 같다.

2) https://universaldependencies.org/format.html

```
#ORGSENT:프랑스의 세계적인 의상 디자이너 엠마누엘 웅가로가 실내 장식용 직물 디자이너로 나섰다.
ID      FORM      LEMMA     UPOSTAG XPOSTAG FEATS     HEAD    DEPREL  DEPS    MISC
1       프랑스의 프랑스 의PROPN   NNP+JKG _       4       nmod    _       _
2       세계적인 세계 적 이 ㄴ    ADJ     NNG+XSN+VCP+ETM _       4       acl     _       _
3       의상      의상      NOUN    NNG     _       4       nmod    _       _
4       디자이너 디자이너 NOUN    NNG     _       6       nmod    _       _
5       엠마누엘 엠마누엘 PROPN   NNP     _       6       nmod    _       _
6       웅가로가 웅가로 가PROPN   NNP+JKS _       11      nsubj   _       _
7       실내      실내      NOUN    NNG     _       8       nmod    _       _
8       장식용    장식 용   NOUN    NNG+XSN _       9       nmod    _       _
9       직물      직물      NOUN    NNG     _       10      nmod    _       _
10      디자이너로        디자이너 로       NOUN    NNG+JKB _       11      obl     _       _
11      나섰다.   나서 었 다 .      VERB    VV+EP+EF+SF     _       0       root    _       _
```

그림 4 . 기존 의존 구문 말뭉치

```
# text = 프랑스의 세계적인 의상 디자이너 엠마누엘 웅가로가 실내 장식용 직물 디자이너로 나섰다.
ID      FORM(conts)     FORM(func)      LEMMA   UPOSTAG XPOSTAG CHUNKTAG        HEADS   DEPREL
1       프랑스  의      프랑스 의PROPN  NNP+JKG NX+JMX  3       nmod
2       세계_적_이      ㄴ      세계 적 이 ㄴ   ADJ     NNG+XSN+VCP+ETM CX+ETX  3       acl
3       의상_디자이너_엠마누엘_웅가로  가      의상 디자이너 엠마누엘 웅가로 가       PROPN   NNG+NNG+NNP+NNP+JKS     NX+JKX  5       nsubj
4       실내_장식_용_직물_디자이너 로  실내 장식 용 직물 디자이너 로       NOUN    NNG+NNG+XSN+NNG+NNG+JKB NX+JKX  5       obl
5       나서    었_다_. 나서 었 다 .    VERB    VV+EP+EF+SF     PX+EPX+EFX+SYX  0       root
```

그림 5 . 말덩이 기반의 의존 구문 말뭉치

표 1 . 기존 의존 구문 말뭉치

ID	FORM	XPOSTAG	HEAD
1	프랑스의	NNP+JKG	4
2	세계적인	NNG+XSN+VCP+ETM	4
3	의상	NNG	4
4	디자이너	NNG	6
5	엠마누엘	NNP	6
6	웅가로가	NNP+JKS	11
7	실내	NNG	8
8	장식용	NNG+XSN	9
9	직물	NNG	10
10	디자이너로	NNG+JKB	11
11	나섰다.	VV+EP+EF+SF	0

표 2 . 말덩이 기반 의존 구문 말뭉치

ID	FORM		XPOS	CHUNK	HEAD
	cont	func			
1	프랑스	의	NNP+JKG	NX+JMX	3
2	세계 적 이	ㄴ	NNG+XSN +VCP+ETM	CX+ETX	3
3	의상 디자이너 엠마누엘 웅가로	가	NNG+NNG +NNP+NNP +JKS	NX+JKX	5
4	실내 장식 용 직물 디자이너	로	NNG+NNG +XSN+NNG +NNG+JKB	NX+JKX	5
5	나서	었 다 .	VV+EP +EF+SF	PX+EPX +EFX+SYX	0

4.1 변환 과정

의존 구문 말뭉치를 말덩이 기반의 말뭉치로 변환하는 전체 과정은 그림 6과 같다.

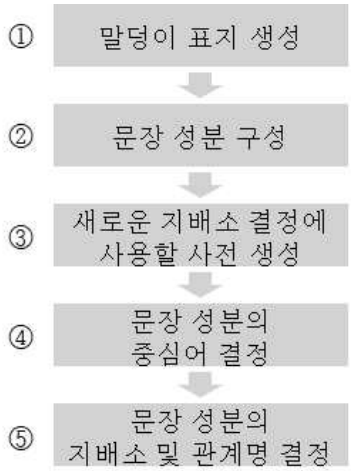


그림 6 . 말뭉치 변환 전체 과정

① 말덩이 표지 생성

말덩이 기반의 의존 구문 말뭉치로 변환하기 위해서는 먼저 문장에 대해 구묶음이 수행되어야 한다. 본 논문에서는 [14]의 순차 태깅 모델을 이용하여 변환하려는 세종 말뭉치에 대해 구묶음을 수행하였다. 예문에 대해 구묶음을 수행하면 그림 7과 같이 예측된 말덩이 열을 얻을 수 있다.

② 문장 성분 구성

새로운 말뭉치의 한 행을 이루는 단위를 구성한다. 말덩이 기반 의존 구문 말뭉치에서 한 행은 문장 성분으로 이루어지며, 이는 내용어 말덩이와 기능어 말덩이로 이루어진다. 따라서 ①에서 얻은 말덩이 표지를 통해 간단히 입력 문장을 문장 성분 단위로 표현할 수 있다.

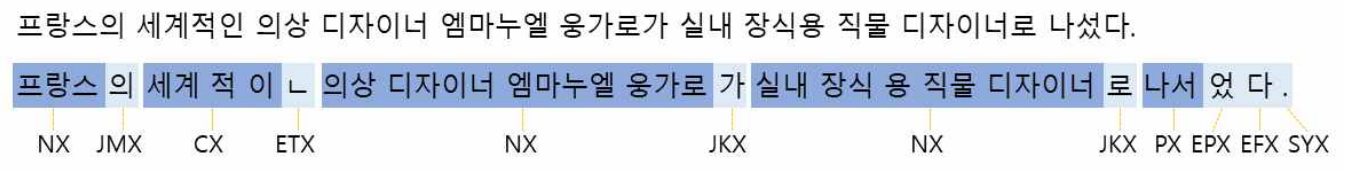


그림 7 . 구묶음 수행 결과. 위에서부터 차례로 원문, 형태소 단위의 문장, 말덩이 표지이다.

③ 사전 생성

문장 성분 단위로 재구성된 노드는 기존과는 다른 ID를 갖게 된다. 따라서 기존의 ID와 새로운 ID의 관계 정보를 저장할 필요가 있다. ②의 결과를 이용해 표 3과 같이 ID 관계 정보를 가진 사전을 만들 수 있다. 또한, 표 4와 같이 사전의 값(value)과 키(key)를 반전시킨 역사전을 만들면, 기존의 ID 중 문장 성분의 중심어에 해당하는 토큰를 선정했을 때, 해당 토큰의 HEAD 정보를 새로운 ID에 전파할 수 있다.

표 3 . ID 사전

new_ID (문장성분)	old_ID (토큰)
1	1
2	2
3	3, 4, 5, 6
4	7, 8, 9, 10
5	11

표 4 . 역 ID 사전

old_ID (토큰)	new_ID (문장성분)
1	1
2	2
3	3
4	3
5	3
6	3
7	4
8	4
9	4
10	4
11	5

예를 들어, 표 1의 1번 토큰('프랑스의')이 변환된 말뭉치에서 어떤 HEAD를 가지게 되는지 기술하면 다음과 같다. 표 1에서 1번 ID에 해당하는 HEAD는 4번이다. 4번은 표 4의 역 ID 사전에서 3번 ID를 값(value)으로 가진다. 즉, HEAD에 해당하는 토큰('디자이너')이 변환된 말뭉치에서 3번 ID의 문장 성분에 있음을 알 수 있다. 따라서, 최종적으로 변환된 말뭉치에서 우리가 찾고자 하는 토큰('프랑스의')이 속해있는 문장 성분의 HEAD 즉, 지배소는 3번 토큰으로 결정할 수 있다.

④ 문장 성분 내의 중심어 결정

말뭉치 변환 과정에서 핵심이 되는 단계이다. ③에서 예로 든 것과 같이 변환 전과 후가 같은 토큰으로 이루어진 경우, 해당 토큰이 곧 문장 성분의 중심어가 되므로 중심어를 결정할 필요가 없다. 하지만 일반적으로 변환된 말뭉치에서 한 문장 성분은 표 2의 3번 ID처럼 기존 말뭉치의 여러 토큰에 해당하는 경우가 대부분이다. 이럴 경우, 이 중 중심어가 되는 토큰을 선정하여 ③의 예시와 같은 과정을 거치면 해당 문장 성분의 최종 HEAD를 결정할 수 있다.

문장 성분 내의 중심어를 선정하는 규칙은 명료하다. 문장 성분을 이루고 있는 내용어 토큰들 중 그 HEAD가 해당 토큰들 내에 없는 것이 중심어가 된다. 즉, 문장 성분 내에서 의존소를 가지지만 지배소는 가지고 있지 않은 토큰이 해당 문장 성분의 중심어가 된다.

이를 표 2의 3번 문장 성분을 예로 들어 설명하면 다음과 같다. 3번 문장 성분을 이루는 토큰들은 기존 말뭉치에서 [3, 4, 5, 6]의 ID에 해당하며 각각 [4, 6, 6, 11]을 HEAD로 가졌다. 이 HEAD 번호 중 4와 6은 해당 토큰에 이미 존재하지만, 11은 그렇지 않다. 즉, 4와 6을 HEAD로 가지는 [3, 4, 5]번은 이 토큰 리스트 내에서 지배소를 가지지만, 11번을 HEAD로 가지는 6번 토큰은 해당 문장 성분 내에서 다른 토큰들의 수식만 받을 뿐 지배소를 가지고 있지 않다. 따라서 6번 토큰을 이 문장 성분의 중심어로 선정하여 ③의 예시와 같은 과정을 거치게 되면 해당 문장 성분의 새로운 HEAD도 결정할 수 있게 된다. 이는 말덩이로 이루어진 문장 성분이라는 단위가 의미적으로는 물론 구문적으로도 한 덩어리를 이루기에 가능한 방법이다.

⑤ 문장 성분의 지배소 및 관계명 결정

이상의 과정에서 역 ID 사전을 통해 말덩이 기반으로 변환된 의존 구문 말뭉치의 HEAD 즉, 지배소를 결정하였으며, 문장 성분을 이루는 토큰이 여러 개일 경우 중심어를 선정한 뒤 해당 문장 성분의 지배소를 결정하였다. 말뭉치를 구성하는 항목은 HEAD 외에도 UPOSTAG, DEPREL 등이 있다. 이러한 정보들은 각 문장 성분의 중심어가 결정되었을 때, 해당 토큰이 원래 갖고 있던 요소들을 따르면 된다. 즉, 표 2의 1번 문장 성분('프랑스의')은 기존 의존 구문 말뭉치의 UPOSTAG인 PROPN과 DEPREL인 nmod를 그대로 갖게 된다. 표 2의 3번 문장 성분의 경우, 중심어에 해당하는 토큰('웅가로')의 기존 UPOSTAG인 PROPN과 DEPREL인 nsubj을 갖게 된다.

이러한 과정을 통해 그림 4의 의존 구문 말뭉치를 변환하게 되면 그림 5과 같은 말덩이 기반의 의존 구문 말뭉치를 구축할 수 있다.

4.2 변환 알고리즘

그림 8은 이상에서 설명한 바와 같이 의존 구문 말뭉치를 말덩이 기반의 말뭉치로 변환하는 알고리즘을 기술한 것이다.

```
def To_Chunk_Dependency_Corpus(dependency_corpus):
    # 문장 성분 단위로 분리
    toConst = To_Constituent(dependency_corpus)

    # look-up 사전 생성
    idDict = ID_Dictionary(toConst) # {new_ID: old_ID}
    idDict_reversed = ID_Dictionary_Reversed(idDict)
                                    # {old_ID: new_ID}

    # 문장 성분 내의 중심어 선정
    for old_id in idDict[new_ID]:
        # 토큰의 head에 해당하는 id가
        # 문장 성분 내에 없으면 이 토큰을
        # 해당 문장 성분의 중심어(content)로 선정
        if not old_id.HEAD in idDict[new_ID]:
            content_list = Add_to_Content_List(old_id)

    # 선정한 중심어를 토대로 역 ID 사전을 이용하여
    # 문장 성분의 최종 지배소 및 관계명 결정
    for old_id in content_list:
        new_head = idDict_reversed[old_ID]
        new_relation = old_id.DEPREL
        new_upostag = old_id.UPOSTAG
```

그림 8 . 말덩이 기반 의존 구문 말뭉치 변환 알고리즘

5. 말덩이 기반 의존 구문 말뭉치의 분석 (Chunked Dependency Corpus)

대량의 말덩이 기반 의존 구문 말뭉치를 구축하기 위해, 4.2절에서 기술한 알고리즘을 기존의 의존 구문 말뭉치[13]에 적용하였다. 기존의 말뭉치는 세종 구문 분석 말뭉치를 [13]의 의존 구문 구조 변환 도구를 이용하여 생성한 것으로 총 62,345문장으로 이루어져 있다. 이를 기반으로 본 논문의 변환 알고리즘을 적용하여 말덩이 기반 의존 구문 말뭉치를 생성한 뒤 띄어쓰기 및 형태소 분석 오류가 전파된 13,053문장을 제외하고 정제된 문장들을 모은 결과, 생성된 말뭉치는 총 49,292문장이었다. 표 5는 기존의 말뭉치와 말덩이 기반 의존 구문 말뭉치를 정량적으로 비교한 내용이다. 동등한 비교를 위해 기존의 말뭉치 중 정제된 말뭉치만을 집계에 이용했다. 비교 항목 중 '전체 행 수'는 구문 분석의 한 단위인 노드의 개수와 일치하며, '기본단위 / 행'의 기본단위는 기존 말뭉치에서는 형태소를 가리키며 변환된 말뭉치에서는 말덩이를 가리킨다.

표 5 . 의존 구문 말뭉치 비교

	기존 의존 구조 말뭉치	**말덩이** 기반 의존 구조 말뭉치
문장 수	49,292	49,292
표지 종류 수	45	18
형태소 수	1,054,859	1,054,859
말덩이 수	•	804,854
행 단위	토큰 (token)	문장 성분 (constituent)
전체 행 수	526,378	377,301
의존 관계 태그 수	50	50

6. 결론 및 향후 연구

본 논문에서는 기존의 구문 분석 말뭉치를 말덩이 기반의 말뭉치로 변환하는 알고리즘을 기술하였다. 말덩이는 한번에 받아들여지는 언어의 의미적 단위로, 한국어 구문 분석의 문제점을 완화하기 위해 도입되었다. 한국어 구문 분석은 많은 노드 수로 인해 계산의 복잡도가 높고, 지배소를 결정할 때 방향성 문제가 발생한다는 특징이 있다. 말덩이는 형태소들을 문법적, 의미적으로 하나의 기능을 하는 단위로 나타내 주기 때문에 노드 수를 감소시킬 수 있고, 이로 인해 구문적 중심어와 의미적 중심어가 일치하게 되어 방향성 문제도 해소할 수 있다. 하지만 대량의 말덩이 기반 의존 구문 말뭉치를 구축하는 데에는 시간적, 비용적 노력이 많이 들어가므로, 본 연구에서는 기존의 의존 구문 말뭉치를 말덩이 기반으로 변환하는 알고리즘을 소개하고 이를 적용하여 말뭉치를 구축하였다.

향후 본 연구에서 구축한 말덩이 기반의 의존 구문 말뭉치를 이용하여 한국어 의존 구문 분석을 수행하고, 기존의 말뭉치를 이용했을 때보다 성능에 있어 어떤 변화를 가져오는지 비교하는 연구를 지속해 나갈 예정이다.

감사의 글

이 논문은 2019년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원(R7119-16-1001, 지식증강형 실시간 동시통역 원천기술 개발)과 2017년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(NRF-2017M3C4A7068187, 한국어 정보처리 원천 기술 연구 개발)

참고문헌

[1] S. Abney, "Parsing by chunks", Principle-based parsing, eds. Berwick, R. Abney, S. and Tenny, C., Kluwer Academic Publishers, 1991.

[2] S. Abney, "Part-of-speech and partial parsing" Corpus-Based methods in language and Speech Processing, eds. Young, S and Bloothooft, G.,

Kluwer Academic Publishers, pp. 118-173, 1996.
[3] 김재훈, "부분 구문분석 방법론", 정보처리학회지, 제7권, 제6호, pp. 83-96, 2000.
[4] 김재훈, "한국어 부분 구문분석의 단위와 그 표지", 한국해양대학교 컴퓨터공학과, 기술문서, KMU-NLP-TR-2000-006, 2000.
[5] 남궁영, 김창현, 천민아, 박호민, 윤호, 최민석, 김재훈, "구문 분석을 위한 한국어 말덩이 정의", 제30회 한글 및 한국어 정보처리 학술대회 논문집, pp. 409-412, 2018.
[6] 박의규, 나동열, "한국어 구문분석을 위한 구묶음 기반 의존명사 처리", 인지과학, 제17권, 제2호, pp. 11-138, 2006.
[7] M. C. de Marneffe, T. Dozat, N. Silveira, K. Haverinen, F. Ginter, J. Nivre, C. D. Manning, "Universal Stanford Dependencies: A cross-linguistic typology", Proceedings of the Language Resources and Evaluation Conference, vol. 14, pp. 4585-4592, 2014.
[8] R. Johansson and P. Nugues, "Extended Constituent-to-Dependency Conversion for English", Proceedings of the 16th Nordic Conference of Computational Linguistics, pp. 105-112, 2007.
[9] J. Chun, N, Han, J. D. Hwang, J. D. Choi, "Building Universal Dependency Treebanks in Korean", Proceedings of the 11th International Conference on Language Resources and Evaluation, pp. 2194-2202, 2018.
[10] 이찬영, 김진웅, 김한샘, "Universal Dependency 관계 태그셋의 한국어 적용", 제30회 한글 및 한국어 정보처리 학술대회 논문집, pp. 334-339, 2018
[11] 원혜진, 류법모, "세종구구조말뭉치 기반 Universal Dependency 말뭉치 반자동 생성 연구", pp. 545-547, 2019.
[12] 홍윤표, "21세기 세종 계획 사업 성과 및 과제", 새국어생활, 제19권, 제1호, pp. 5-33, 2009.
[13] 최용성, 이공주, "한국어 구절 구문 코퍼스의 의존 구문 구조 트리로의 변환에서 중심어 전파 규칙", 제30회 한글 및 한국어 정보처리 학술대회 논문집, pp. 514-519, 2018.
[14] 남궁영, 김창현, 천민아, 박호민, 윤호, 최민석, 김재균, 김재훈, "BI-LSTM/CRF 모델을 이용한 한국어 구묶음", 한국정보과학회 학술발표논문집, pp. 631-633, 2019.