

2016

국어 정보 처리 시스템 경진 대회 발표 자료집

일자 2016. 10. 7. (금)
장소 동아대학교 부민캠퍼스 국제관 4층 세미나실
주관 한국정보과학회 언어공학연구회
주최 국립국어원

대 회 일 정

■ 등 록			
13:00 ~ 13:30	등록 및 방명록 작성, 명찰 및 발표 자료집 배부		
■ 개 회 식			
13:30 ~ 13:40	개회식	사 회	차정원 교수
		개회사	박혁로 교수
		환영사	정호성 어문연구과장 (국립국어원)
■ 발표 1부 (지정 분야: 개체명 인식 시스템)		좌 장	최정도 학예연구사 (국립국어원)
13:40 ~ 14:00	서강 알팜 개체명 인식기 이건철, 임지수, 최호정, 권순재, 허윤석, 서정연 (서강대학교)		
14:00 ~ 14:20	단추(DANCHU) : NER 유홍연, 이호경, 김민철, 고영중 (동아대학교)		
14:20 ~ 14:40	Annie 임재수		
14:40 ~ 15:00	KAISER 함영균, 최동호, 최기선 (한국과학기술원)		
15:00 ~ 15:20	WordAngler 김보검, 강명윤, 민태홍, 이재성 (충북대학교)		
15:20 ~ 15:40	코너(KoNER) 신유현, 이상구 (서울대학교)		
■ 시연 및 BREAK TIME			
15:40 ~ 16:00	12팀 시스템 시연 및 BREAK TIME		
■ 발표 2부 (일반 분야)		좌 장	차정원 교수
16:00 ~ 16:05	단추(DANCHU) 안재현, 윤정민, 김혜민, 박용신, 배경만, 고영중 (동아대학교)		
16:05 ~ 16:10	한국어 세 줄 요약기 summ4riz3 설진석 (연세대학교)		
16:10 ~ 16:15	지도표시 및 음성 녹음/재생 기능을 가진 지역어 수집 및 검색 APP 김푸름, 박소현, 심용석, 오승록, 이상호, 이재람, 안동연 (전북대학교)		
16:15 ~ 16:20	문맥 기반 실시간 한국어 문장 교정 시스템 박영근, 최재성, 김재민, 이성동, 이현아 (금오공과대학교)		
16:20 ~ 16:25	ESPRESSO TOOL 2016 신창욱, 박태호, 박성재, 박다솔, 신영태, 차정원 (창원대학교)		
16:25 ~ 16:30	KOMORAN 3.0 신준수, 이승원 (SHINEWARE)		
■ 심사 회의			
16:30 ~ 18:00	심사 회의		
■ 시상식 및 폐회식			
18:00 ~ 18:30	시상식 및 폐회식	사 회	차정원 교수
		시 상	정호성 과장
		심사평	강승식 교수

※ 상기 일정은 일부 변동될 수도 있음.

- 지정분야/일반분야 팀이 경진대회 당일 불참하면 일정 조정이 됨.

2016

국어 정보 처리 시스템 경진 대회 발표 자료집

목 차

1. 서강 알짬 개체명 인식기	7
2. 단추(DANCHU) : NER	19
3. Annie	37
4. KAISER	51
5. WordAngler	57
6. 코너(KoNER)	75
7. 단추(DANCHU)	89
8. 한국어 세 줄 요약기_ summ4riz3	109
9. 지도 표시 및 음성 녹음/재생 기능을 가진 지역어 수집 및 검색 APP'뽕꼬!?'	117
10. 문맥 기반 실시간 한국어 문장 교정 시스템	133
11. ESPRESSO TOOL 2016	141
12. KOMORAN 3.0	149

국 립 국 어 원

서강 알짬 개체명 인식기

이건철, 임지수, 최호정, 권순재, 허윤석, 서정연

【서강대학교】

차 례

제 1 장 소프트웨어 소개	10
1.1 소프트웨어 명칭	10
1.2 소프트웨어 사용 환경	10
1.3 소프트웨어 특징	10
제 2 장 소프트웨어 설치 및 실행	12
2.1 소프트웨어 설치 방법	12
2.2 소프트웨어 파일 구조	12
2.2.1 주요 파일 설명	12
2.2.2 전체 구조	13
2.3 소프트웨어 실행 방법	13
제 3 장 소프트웨어 기능	14
3.1 프로그램 기능	14
3.2 프로그램 기능 제약	16
제 4 장 기타	18

제 1 장 소프트웨어 소개

1.1 소프트웨어 명칭

서강 알짬 개체명 인식기

: 알짬이란 여럿 가운데 가장 중요한 내용이라는 뜻의 순 우리말로, 문장 내에서 가장 중요한 내용을 분석하여 알려준다는 의미를 담고 있다.

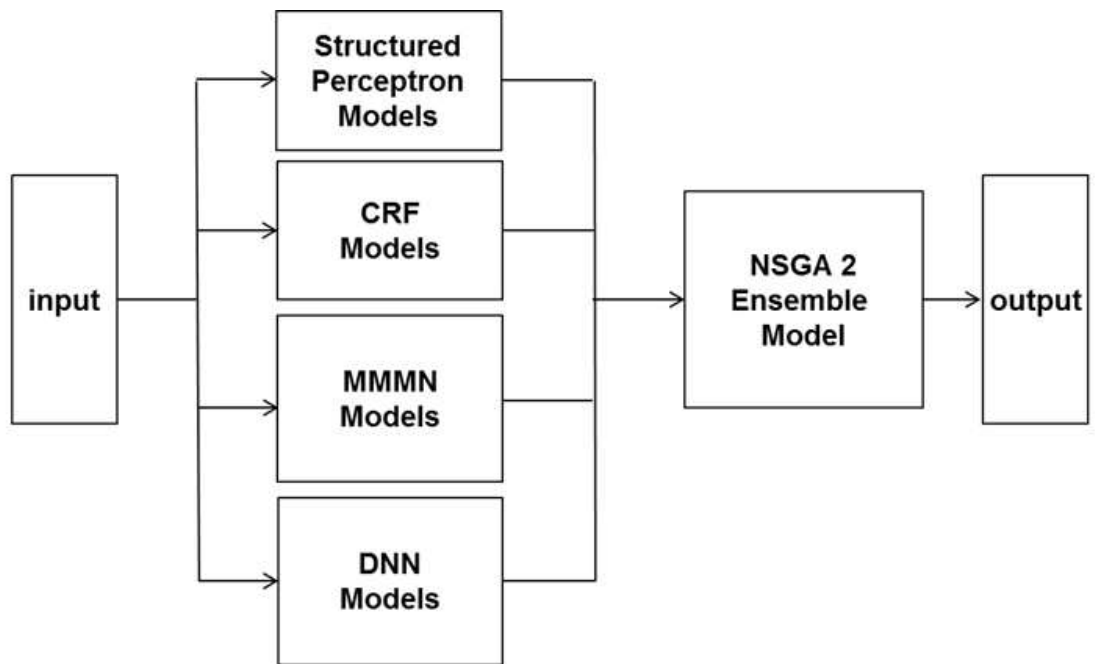
1.2 소프트웨어 사용 환경

- linux ubuntu 12.04.5 실시간 가용 RAM 64Gbyte 이상
- Python 2.7.3
- UTF-8

1.3 소프트웨어 특징

‘서강 알짬 개체명 인식기’는 서강대학교 자연어처리 연구실 여름방학 인턴쉽 프로젝트로 서강대학교 컴퓨터공학과 학부생과 석사생이 개발한 개체명 인식기이다. 개체명 인식기는 어떤 문장에서 인명, 지명, 기관명 같은 고유 명사개체명을 분석하여 자동으로 찾아주는 시스템이다. 개체명은 문장에서 주어 목적어와 같이 텍스트의 의도와 관련이 깊어, 이를 추출하는 것은 자연어 이해에서 매우 중요한 역할을 수행한다. 하지만 개체명은 신조어, 축약어, 방언(사회, 지역)과 같은 미등록어가 많고 같은 어휘가 문맥에 따라 다른 태그로 분류되기 때문에 개체명을 학습하기 힘들다.

본 시스템은 다양한 자질과 알고리즘으로 학습한 여러 학습기를 앙상블 러닝으로 결합하여 새로운 분류기를 구성하였다. 본 시스템은 8가지 자질과 5가지 알고리즘으로 10개의 다른 분류기를 구성하였다. 그리고 MOO 기반의 유전자 알고리즘으로 각 분류기의 가중치를 학습하여, 모델들의 가중합으로 최종 분류기를 구축하였다. 이런 앙상블 러닝을 사용하여, 각 학습기들은 상대적으로 분류 성능이 떨어지는 태그를 상호 보완하게 되어 전체 성능이 향상되었다.



제 2 장 소프트웨어 설치 및 실행

2.1 소프트웨어 설치 방법

1. cqasys_alzzam.tar.gz 파일 압축을 해제한다.

```
tar -zxvf cqasys_alzzam.tar.gz
```

2. module.sh을 실행하여 필요한 라이브러리를 설치한다.

```
chmod 755 module.sh  
./module.sh
```

2.2 소프트웨어 파일 구조

2.2.1 주요 파일 설명

① 실행 파일

cqasys_alzzam.py --- 소프트웨어 실행 파일

② 데이터 파일

- /DNER/deep_ner.py	--- LSTM-CRF 태거 클래스 파일
- /CRF/CRF.py	--- CRF 태거 클래스 파일
- /STRUCTURED/STRUCTURED.py	--- Structured Perceptron 태거 클래스 파일
- /SSVM/SSVM.py	--- MMMN 태거 클래스 파일
- */DATA/*.dat	--- 학습된 모델 object파일

③ 사전 파일

- */gazettee.txt	--- 개체명 사전
- */thres3.txt	--- 기분석 사전
- tained_weight	--- 학습된 가중치 정보

2.2.2 전체 구조

final/	
cqasys_alzzam.py 실행 스크립트
final_classifier.py 전체 분석기 코드
DNER/ LSTM-CRF 태거 관련 소스 및 모델
모음	
CRF/ CRF 태거 관련 소스 및 모델 모음
STRUCTURED/ Structed Perceptron 태거 및 모델 모음
SSVM/ MMMN 태거 및 모델 모음
DATA/ DBSCAN 클러스터 결과 및 모델 모음

2.3 소프트웨어 실행

1. cqasys_alzzam.py 스크립트를 실행한다.
(cqasys_alzzam.py -i [파일명])

```
intern16@nlprenu:~/final$ python cqasys_alzzam.py -i dev.json
```

2. result.json의 결과를 확인한다.

```
{
  "category": "",
  "category_weight": 0,
  "sentence": [
    {
      "id": 0,
      "resever_str": "",
      "text": "서호프와 과사노에게 연속 안타를 내주며",
      "morph": [
        { "id": 0, "lemma": "서호프", "type": "NNP", "position": 0},
        { "id": 1, "lemma": "와", "type": "JC", "position": 9},
        { "id": 2, "lemma": "과사노", "type": "NNP", "position": 13},
        { "id": 3, "lemma": "에게", "type": "JC", "position": 22},
        { "id": 4, "lemma": "연속", "type": "NNG", "position": 29},
        { "id": 5, "lemma": "안타", "type": "NNG", "position": 36},
        { "id": 6, "lemma": "를", "type": "JKO", "position": 42},
        { "id": 7, "lemma": "내주", "type": "VV", "position": 46},
        { "id": 8, "lemma": "며", "type": "EC", "position": 52}
      ],
      "word": [
        { "id": 0, "text": "서호프와", "type": "", "begin": 0, "end": 1},
        { "id": 1, "text": "과사노에게", "type": "", "begin": 2, "end": 3},
        { "id": 2, "text": "연속", "type": "", "begin": 4, "end": 4},
        { "id": 3, "text": "안타를", "type": "", "begin": 5, "end": 6},
        { "id": 4, "text": "내주며", "type": "", "begin": 7, "end": 8}
      ],
      "NE": [
        { "id": 0, "text": "서호프", "type": "PS", "begin": 0, "end": 0},
        { "id": 1, "text": "과사노", "type": "PS", "begin": 2, "end": 2}
      ],
      "chunk": [],
      "dependency": [],
      "SRL": [],
      "relation": [],
      "SA": []
    }
  ],
}
```

제 3 장 소프트웨어 기능

3.1 프로그램 기능

본 시스템은 입력에 대해 분석한 내용을 토대로 8개의 자질과 4개의 알고리즘으로 구성된 8개된 하이브리드 모델로서, 개체명을 분석하며 사용한 자질은 아래와 같다.

사용 자질	구분	예시
형태소 어휘 정보	어휘자질	서호프, 와, ...
형태소의 길이	어휘자질	서호프 -> 3
형태소 prefix, postfix 정보	어휘자질	한국정보과학회 -> 한,회, 한국, 학회 ...
형태소 품사 정보	통사자질	NNP, JC, ...
개체명 사전 정보	의미자질	한국정보과학회->OG, 서호프->PS
기분석 사전 정보	의미자질	학습 데이터에서 3회 이상 같은 개체명으로 분류되는 개체 집합
워드 임베딩	의미자질	단어의 vector representation 자질
워드 임베딩 클러스터	의미자질	Dbscan clustering result of Vector representation

1. 어휘 자질

가. 형태소 어휘정보

형태소의 어휘 정보이다. 해당 어휘와 앞·뒤 어휘(-2 ~ +2)를 선택적으로 자질로 사용한다.

나. 형태소 길이

다 .형태소 prefix, postfix 정보

해당 형태소의 접미 음절과 접두 음절 정보이다.

2. 통사 자질

가. 형태소 품사 정보

형태소의 품사 정보이다. 해당 형태소의 앞·뒤 품사(-2 ~ +2)를 선택적으로 자질로 사용한다.

3. 의미 자질

가. 개체명 사전 정보

대회 측에서 제공한 개체명 사전 정보

나. 기분석 사전 정보

학습 데이터에서 3회 이상 같은 태그가 부착되는 개체 집합

다. 워드 임베딩

대회 측에서 제공한 워드 임베딩 수행 결과

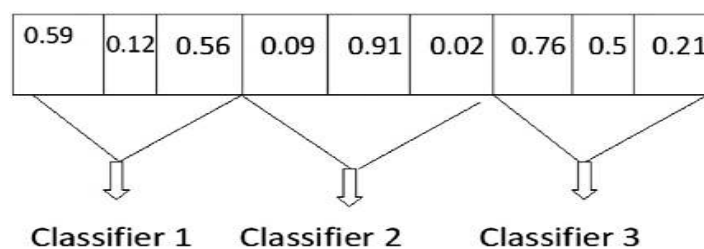
라. 워드 임베딩 클러스터

워드 임베딩 데이터를 DBSCAN 알고리즘으로 클러스터링 한 결과

본 시스템에서는 아래 4가지 알고리즘을 기반으로 다른 자질 및 파라미터로 학습한 분류기 6개를 사용한다.

1. Conditional Random Fields¹
2. Structured Perceptron²
3. Max Marginal Markov Networks³
5. Long Short-Term Memory with Conditional Random Fields⁴

또한, 위의 8개의 분류기들을 NSGA-II 알고리즘을 사용하여 Recall과 Precision을 Performance Index로 Multi Objective Optimize하여 각 분류기의 가중치를 계산한다. 마지막으로, 계산한 가중치를 사용하여 각 모델의 결과의 가중합으로 개체명을 인식한다.



1 <https://python-crfsuite.readthedocs.io/en/latest/>

2 <https://github.com/larsmans/seqlearn>

3 <https://pystruct.github.io/index.html>

4 <https://github.com/glample/tagger>

3.2 프로그램 기능 제약

- UTF-8 인코딩 환경에서 정상적인 동작을 보장함
- 입·출력 양식은 주최 측에서 제공한 형식

```
{
  "id" : 1,
  "text" : "맥스웰 방정식(Maxwell方程式, )은 전기와 자기의 발생, 전기장과 자기장, 전하 밀도와 전류 밀도의 형성을
  나타내는 4개의 편미분 방정식이다.",
  "morph" : [
    {"id" : 0, "lemma" : "맥스웰", "type" : "NNG", "position" : 20},
    {"id" : 1, "lemma" : "방정", "type" : "NNG", "position" : 30},
    {"id" : 2, "lemma" : "식", "type" : "NNG", "position" : 36},
    {"id" : 3, "lemma" : "(", "type" : "SS", "position" : 39},
    {"id" : 4, "lemma" : "Maxwell", "type" : "SL", "position" : 40},
    {"id" : 5, "lemma" : "方程式", "type" : "SH", "position" : 47},
    {"id" : 6, "lemma" : ",", "type" : "SP", "position" : 56},
    {"id" : 7, "lemma" : ")", "type" : "SS", "position" : 58},
    {"id" : 8, "lemma" : "은", "type" : "JX", "position" : 59},
    {"id" : 9, "lemma" : "전기", "type" : "NNG", "position" : 63},
    {"id" : 10, "lemma" : "와", "type" : "JC", "position" : 69},
    {"id" : 11, "lemma" : "자기", "type" : "NP", "position" : 73},
    {"id" : 12, "lemma" : "의", "type" : "JG", "position" : 79},
    {"id" : 13, "lemma" : "발생", "type" : "NNG", "position" : 83},
    {"id" : 14, "lemma" : ",", "type" : "SP", "position" : 89},
    {"id" : 15, "lemma" : "전기", "type" : "NNG", "position" : 91},
    {"id" : 16, "lemma" : "장", "type" : "XSN", "position" : 97},
    {"id" : 17, "lemma" : "과", "type" : "JC", "position" : 100},
    {"id" : 18, "lemma" : "자기", "type" : "NNG", "position" : 104},
    {"id" : 19, "lemma" : "장", "type" : "XSN", "position" : 110},
    {"id" : 20, "lemma" : ",", "type" : "SP", "position" : 113},
    {"id" : 21, "lemma" : "전하", "type" : "NNG", "position" : 115},
    {"id" : 22, "lemma" : "밀도", "type" : "NNG", "position" : 122},
    {"id" : 23, "lemma" : "와", "type" : "JC", "position" : 128},
    {"id" : 24, "lemma" : "전류", "type" : "NNG", "position" : 132},
    {"id" : 25, "lemma" : "밀도", "type" : "NNG", "position" : 139},
    {"id" : 26, "lemma" : "의", "type" : "JG", "position" : 145},
    {"id" : 27, "lemma" : "형성", "type" : "NNG", "position" : 149},
    {"id" : 28, "lemma" : "을", "type" : "JKO", "position" : 155},
    {"id" : 29, "lemma" : "나타내", "type" : "VV", "position" : 159},
    {"id" : 30, "lemma" : "는", "type" : "ETM", "position" : 168},
    {"id" : 31, "lemma" : "4", "type" : "SN", "position" : 172},
    {"id" : 32, "lemma" : "개", "type" : "NNB", "position" : 173},
    {"id" : 33, "lemma" : "의", "type" : "JG", "position" : 176},
    {"id" : 34, "lemma" : "편미분", "type" : "NNG", "position" : 180},
    {"id" : 35, "lemma" : "방정", "type" : "NNG", "position" : 190},
    {"id" : 36, "lemma" : "식", "type" : "NNG", "position" : 196},
    {"id" : 37, "lemma" : "이", "type" : "VCP", "position" : 199},
    {"id" : 38, "lemma" : "다", "type" : "EF", "position" : 202},
    {"id" : 39, "lemma" : ".", "type" : "SF", "position" : 205}
  ],
}
```



```

"word" : [
  {"id" : 0, "text" : "맥스웰", "type" : "", "begin" : 0, "end" : 0},
  {"id" : 1, "text" : "방정식(Maxwell方程式)", "type" : "", "begin" : 1, "end" : 6},
  {"id" : 2, "text" : "은", "type" : "", "begin" : 7, "end" : 8},
  {"id" : 3, "text" : "전기와", "type" : "", "begin" : 9, "end" : 10},
  {"id" : 4, "text" : "자기의", "type" : "", "begin" : 11, "end" : 12},
  {"id" : 5, "text" : "발생", "type" : "", "begin" : 13, "end" : 14},
  {"id" : 6, "text" : "전기장과", "type" : "", "begin" : 15, "end" : 17},
  {"id" : 7, "text" : "자기장", "type" : "", "begin" : 18, "end" : 20},
  {"id" : 8, "text" : "전하", "type" : "", "begin" : 21, "end" : 21},
  {"id" : 9, "text" : "밀도와", "type" : "", "begin" : 22, "end" : 23},
  {"id" : 10, "text" : "전류", "type" : "", "begin" : 24, "end" : 24},
  {"id" : 11, "text" : "밀도의", "type" : "", "begin" : 25, "end" : 26},
  {"id" : 12, "text" : "형성을", "type" : "", "begin" : 27, "end" : 28},
  {"id" : 13, "text" : "나타내는", "type" : "", "begin" : 29, "end" : 30},
  {"id" : 14, "text" : "4개의", "type" : "", "begin" : 31, "end" : 33},
  {"id" : 15, "text" : "편미분", "type" : "", "begin" : 34, "end" : 34},
  {"id" : 16, "text" : "방정식이다.", "type" : "", "begin" : 35, "end" : 39}
],
"NE" : [
  {"id" : 0, "text" : "맥스웰", "type" : "PS_NAME", "begin" : 0, "end" : 0},
  {"id" : 1, "text" : "Maxwell方程式", "type" : "OGG_BUSINESS", "begin" : 4, "end" : 5},
  {"id" : 2, "text" : "4개", "type" : "QT_COUNT", "begin" : 31, "end" : 32}
],
},

```

제 4 장 기타

학습기의 성능 및 결과는 아래와 같다(Dev set 기준). 각 학습기의 성능에 비해 단순 Voting으로도 성능이 상당히 향상되는 것을 볼 수 있다. 또한 Weighted Voting 결과는 ~로 이전 보다 매우 높은 성능을 보이는 것을 볼 수 있다.

다만, 이는 dev set에서의 결과이므로 training set에서 실험을 진행 할 것이다.

모델	성능
LSTM-CRFs_1	82.53
LSTM-CRFs_2	82.01
CRFs_1	82.11
MMMN_1	83.71
MMMN_2	81.97
Structed_Perceptron_1	83.42
Voting	85.96
Weighted Voting	88.xx

국립국어원

단추(DANCHU) : NER

유홍연, 이호경, 김민철, 고영중
【동아대학교】

차 례

제 1 장 소프트웨어 소개	22
1.1 소프트웨어 명칭	22
1.2 소프트웨어 사용 환경	22
1.3 소프트웨어 특징	22
제 2 장 소프트웨어 설치 및 실행	24
2.1 소프트웨어 설치 방법	24
2.2 소프트웨어 파일 구조	25
2.2.1 주요 파일 설명	25
2.2.2 전체 구조	26
2.3 소프트웨어 실행 방법	26
제 3 장 소프트웨어 기능	27
3.1 bi-LSTM-CRFs 모델	27
3.1.1 Recurrent Neural Networks(RNN)	27
3.1.2 Long Short-term Memory(LSTM)	28
3.1.3 bidirectional LSTM CRFs(bi-LSTM-CRFs)	29
3.2 단어 표상 확장	29
3.2.1 단어 임베딩 벡터	29
3.2.2 품사 자질 벡터	30
3.2.3 음절 기반 단어 임베딩 벡터	30
3.2.4 개체명 사전 자질 벡터	32
3.3 전체 구성도	33
제 4 장 실험	34
4.1 실험 환경	34
4.2 단어 임베딩 벡터 실험	34
4.3 품사 자질 벡터 실험	34
4.4 음절 기반 단어 임베딩 벡터 실험	35
4.5 개체명 사전 자질 벡터 실험	35
4.6 최종 실험 결과 비교	36
제 5 장 참고문헌	36

제 1 장 소프트웨어 소개

1.1 소프트웨어 명칭

- 단추(DANCHU) : NER

동아대학교 지능형시스템실험실의 한국어 언어 분석기 통합 시스템인 단추(DANCHU)의 두 번째 모델인 개체명 인식기를 말한다.

1.2 소프트웨어 사용 환경

- 운영체제 : Ubuntu 16.04 LTS 64bit
- CPU : i5-4690 3.50GHz
- 메모리 : 8GB
- 사용 언어 : python 2.7.12
- 라이브러리 : TensorFlow 0.10.0.rc0 이상
(※ TensorFlow 0.10.0 rc0 이하의 버전에서는 본 시스템이 동작 하지 않을 수 있습니다.)
- 실행 환경 : 터미널
- 텍스트 인코딩 : UTF-8

1.3 소프트웨어 특징

개체명(Named Entity)이란 문서 내에서 인명, 기관명, 지명, 시간, 날짜 등 고유한 의미를 가지는 단어를 말한다. 이러한 개체명을 문서로부터 추출하여 개체명의 종류를 결정하는 것이 개체명 인식(Named Entity Recognition)이다.

최근 개체명 인식 연구에서는 순차 레이블링(Sequence Labeling)에 특화된 Long Short-Term Memory(LSTM)기반의 bidirectional LSTM Conditional Random Fields(bi-LSTM-CRFs) 모델이 가장 우수한 성능을 보이고 있다. LSTM 기반의 모델은 각 단어를 잘 표현하는 단어 임베딩 벡터(word embedding vector)를 입력으로 받기 때문에, 단어 표상(word representation)에 의존적이다.

따라서 이러한 단어 표상을 잘 만들기 위한 많은 연구들이 진행되고 있다. 대표적인 방법으로는 대량의 말뭉치를 이용하여 사전 학습된(pretrained) 단어 임베딩 벡터를 활용하거나, 단어를 구성하고 있는 문자들의 임베딩 벡터로부터 단어 임베딩 벡터를 유도해내는 방법들이 연구 되고 있다.

본 시스템에서는 한국어 개체명 인식을 위하여 bi-LSTM-CRFs를 이용하고, 그 입력으로 사용되는 단어 표상을 확장하기 위해 사전 학습된 단어 임베딩 벡터, 품사 자질 벡터, 개체명 사전 자질 벡터, 그리고 음절 기반에서 확장된 단어 임베딩 벡터를 사용한다. 음절 단위에서 단어로 확장하기 위한 음절 임베딩 벡터 입력으로는 학습 데이터에서 나온 개체명 분포 정보를 벡터로 만들어 사용하였다.

제 2 장 소프트웨어 설치 및 실행

2.1 소프트웨어 설치 방법

- TensorFlow 설치(CPU 사용 버전)

- pip Installation
- 우분투 터미널 창에서 아래와 같은 명령어 수행
- `sudo apt-get install python-pip python-dev`

예)

```
islab@islab-pc:~$
islab@islab-pc:~$
islab@islab-pc:~$ sudo apt-get install python-pip python-dev
```

- TensorFlow URL 확인(버전 확인 필수)

예)

```
https://storage.googleapis.com/tensorflow/linux/cpu/tensorflow-0.10.0-cp27-none-
linux_x86_64.whl
```

- `sudo pip install --upgrade $URL`

예)

```
islab@islab-pc:~$
islab@islab-pc:~$
islab@islab-pc:~$ sudo pip install --upgrade https://storage.googleapis.com/tenso
rflow/linux/cpu/tensorflow-0.10.0-cp27-none-linux_x86_64.whl
```

- 동작 확인
- python을 통해서 확인
- `import tensorflow as tf`을 통해서 확인

예)

```
islab@islab-pc:~$ python
Python 2.7.10 (default, Oct 14 2015, 16:09:02)
[GCC 5.2.1 20151010] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow as tf
```

- TensorFlow 업그레이드 (기존의 TensorFlow가 0.7.1 이상인 경우)
- `sudo pip install --upgrade $URL`

(※ 기존의 버전이 0.7.1 미만인 경우에는 `pip uninstall tensorflow`를 통해 삭제 후 원하는 버전의 TensorFlow 재설치)

예)

```
islab@islab-pc:~$
islab@islab-pc:~$
islab@islab-pc:~$ sudo pip uninstall tensorflow
```

2.2 소프트웨어 파일 구조

2.2.1 주요 파일 설명

- ① 실행 파일
 - 실행 파일 : klp2016_islab/bi_LSTM_CRF/test.pyc
- ② 데이터 파일
 - 워드 임베딩 :
klp2016_islab/word_embeddings/wikiCorpus_word2vector.hr
 - 음절 임베딩 :
klp2016_islab/word_embeddings/wikiCorpus_word2vector_uni_avg.hr
 - 음절 분포 임베딩 :
klp2016_islab/word_embeddings/BI_ne_norm_not_soft.txt
- ③ 사전 파일
 - 개체명 사전 폴더 : klp2016_islab/distribution2016/gazet/
- ④ 결과 파일
 - 결과 json 파일 : klp2016_islab/result/result.json

2.2.2 전체 구조

klp2016_islab/ 시스템폴더
 bi_LSTM_CRF/ 소스파일폴더
 cqasys_test.pyc 시스템 실행파일
 corpus/ 말뭉치 폴더
 distribution2016/ json관련폴더(경진대회 배포)
 result/ 최종결과폴더(result.json)
 word_embeddings/ 워드임베딩관련폴더

2.3 소프트웨어 실행

- 터미널에서 /klp2016_islab/bi_LSTM_CRF로 이동
- `python cqasys_test.pyc --test test.json_path` 명령어로 실행
`python test.pyc`로 시스템 동작
 -- test 옵션, 테스트 데이터 json의 경로
 test.json_path 경진대회에서 제공하는 테스트 데이터 json의 경로
- /klp2016_islab/bi-LSTM-CRF/result/result.json 으로 결과 확인

```

hongyeon@hy-ubuntu:~/Documents/study/klp2016_islab/bi_LSTM_CRF$ python cqasys_test.pyc --test ../test.json
Found 14546 unique words (121009 in total)
Loading pretrained embeddings from ../word_embeddings/wikiCorpus_word2vector.hr...
added 0 pretrained embeddings
Found 1370 unique characters
Loading pretrained embeddings from ../word_embeddings/wikiCorpus_word2vector_uni_avg.hr...
added 1 pretrained embeddings
Found 11 unique named entity tags
Found 46 unique named entity pos

```

제 3 장 소프트웨어 기능

이 장에서는 한국어 개체명 인식을 위해 본 시스템에서 사용한 학습 모델과 단어 표상 확장 방법에 대해 자세히 설명하고, 그에 따른 타당성을 입증하기 위해 제 4 장에서 실험 결과를 보인다.

3.1 bi-LSTM-CRFs 모델

3.1.1 Recurrent Neural Networks(RNN)

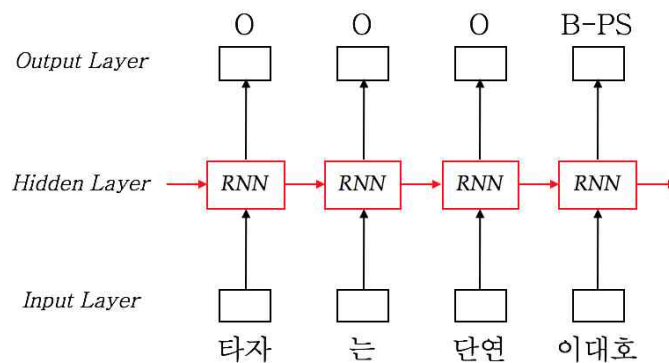


그림 18. Recurrent Neural Networks 모델

그림 1에서 보이는 것처럼 RNN은 단어 열(x_1, x_2, \dots, x_t)을 입력 계층에서 입력으로 받고, 그 입력은 은닉 계층(h_1, h_2, \dots, h_t)을 거쳐 입력을 잘 표현하는 벡터(y_1, y_2, \dots, y_t)로 출력 계층에서 출력된다. RNN 구조를 아래와 같은 식으로 정의할 수 있고, 아래 식에서 U, V, W 는 각 계층의 weight행렬이다.

$$h_t = \tanh(Ux_t + Vh_{t-1})$$

$$y_t = \text{softmax}(Wh_t)$$

RNN은 위의 수식과 그림 1에서 알 수 있듯이 이전 상태를 다음 상태로 계속 전달하는 모델이다. 따라서 이론상으로는 이전 상태를 기억하여 장기 의존성(long-range dependencies)을 다룰 수 있다. 하지만 실제로 위치상 멀리 있는 정보를 많이 잃어버리는 문제인 그래디언트 소멸 문제(Vanishing gradient problem)가 존재하기 때문에 장기 의존성을 유지할 수 없는 문제점이 있다.

3.1.2 Long Short-term Memory(LSTM)

LSTM은 RNN의 그래디언트 소멸 문제를 해결하여 장기 의존성을 잘 학습할 수 있는 특별한 모델이다. 이 문제를 해결하기 위하여 LSTM에서는 RNN의 은닉 계층 노드에 3개의 gate(input, output, forget)와 1개의 memory cell을 이용한다.

LSTM의 memory cell은 전체적인 상태를 기억하여 다음 상태로 전달하는 역할을 한다. forget gate를 이용하여 cell의 상태에서 어떤 정보를 제거할지 결정하고, input gate로 cell에서 어떤 정보를 갱신할지 결정한다. 마지막으로 output gate를 이용하여 cell의 어떤 정보를 전달할지 결정하여 장기 의존성을 유지한다. 본 시스템에서는 forget gate를 사용하지 않았다. LSTM 구조의 전체적인 수식은 다음과 같다.

- $i_t = \text{sigmoid}(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i)$
- $c_t = (1 - i_t) \odot c_{t-1} + i_t \odot \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c)$
- $o_t = \text{sigmoid}(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o)$
- $h_t = o_t \odot \tanh(c_t)$

위 식에서 i, c, o, h 는 각각 input gate, memory cell, output gate, hidden state이다. 그리고 \odot 는 element-wise product이며, W 는 weight 행렬을 b 는 bias를 나타낸다.

3.1.3 bidirectional LSTM CRFs(bi-LSTM-CRFs)

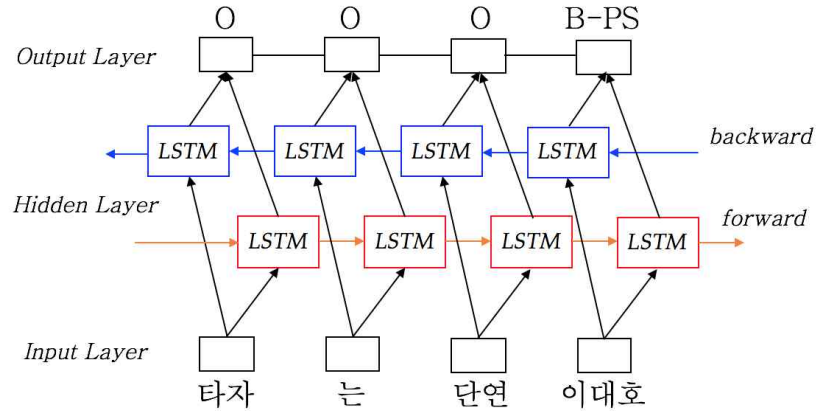


그림 19. bidirectional LSTM CRFs 모델

bi-LSTM-CRFs는 그림 2에서 보이는 것과 같이 LSTM에 입력 문자열을 양방향으로 받아서 각 단어 별로 은닉 계층의 결과를 얻고, 그 결과 간의 의존성을 추가한 모델이다. 본 시스템에서는 bi-LSTM-CRFs를 이용하여 한국어 개체명 인식을 한다.

3.2 단어 표상 확장

LSTM 기반의 모델은 각 단어의 임베딩 벡터를 입력으로 받기 때문에 단어 표상에 의존적이다. 따라서 그림 3과 같이 사전 학습된 단어 임베딩 벡터, 품사 자질 벡터, 음절 기반 단어 임베딩 벡터, 개체명 사전 자질 벡터를 사용하여 단어 표상을 확장한다.

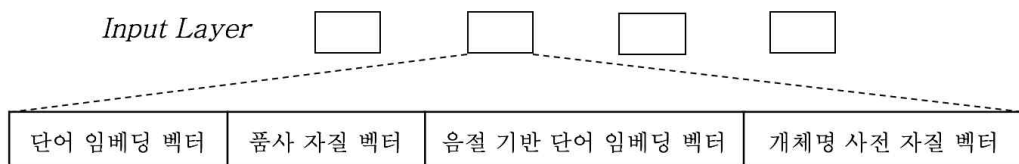


그림 20. 확장된 단어 표상

3.2.1 단어 임베딩 벡터

대부분의 개체명은 미등록어이기 때문에 학습 데이터에 나오지 않은 개체명을 잘 분류하는 것은 한계가 있다. 따라서 개체명 인식에서는 대량의 말뭉치에서 사전 학습된 단어 임베딩 벡터를 활용하여 단어의 집합을 확장하는 것이 중요하다. 본 시스템에서는 대회 주최 측에서 제공한 사전 학습된 단어 임베딩 벡터를 사용하였다. 제공된 사전 학습된 단어 임베딩 벡터의 차원은 50이며, 단어의 단위는 형태소와 품사태그가 결합된 형태이다. 예를 들면 아래와 같다.

- 단어 : “타자”, “는”, “이대호”
- 임베딩 단위 : “타자/NNG”, “는/JX”, “이대호/NNP”

3.2.2 품사 자질 벡터

개체명 인식에서는 품사 정보 또한 중요한 역할을 하기 때문에 품사 자질을 활용하는 것이 중요하다. 본 시스템에서는 현재 품사 정보를 표현하기 위하여 46차원으로 이루어진 one-hot 벡터를 사용하여 단어 표상을 확장하였다. 현재 품사가 NNP 일 때 one-hot 벡터의 예를 들면 표 1과 같다.

표 1. 품사 NNP에 대한 one-hot 벡터

NNG	NNP	NNB	VV	...	JKS	JKO	JKB	SN
0	1	0	0	...	0	0	0	0

3.2.3 음절 기반 단어 임베딩 벡터

음절 기반 단어 임베딩 벡터란 각 단어를 표현하기 위해 음절 단위의 임베딩 벡터를 기반으로 단어 단위의 임베딩 벡터로 확장한 벡터를 말한다. 단어는 음절의 시퀀스이기 때문에 음절 단위 임베딩 벡터의 확장은 단어를 표현하기에 적합하다. 본 시스템에서는 단어를 확장하기 위하여 그림 4에서 보이는 것과 같이 bidirectional LSTM을 이용하고, forward의 마지막 상태와 backward의 마지막 상태를 결합하여 단어 단위 임베딩 벡터로 사용하였다.

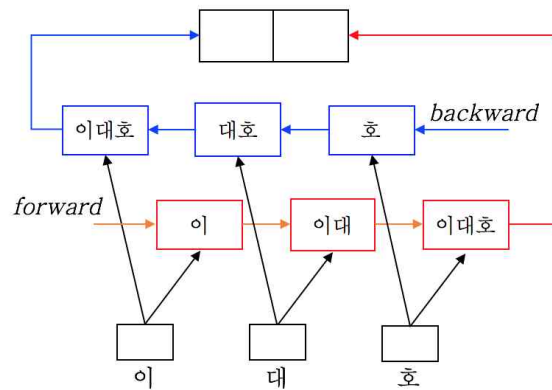


그림 21. 음절 기반 단어 임베딩 벡터를 위한 bi-LSTM

입력되는 음절 임베딩 벡터로는 음절별로 각 개체명 태그별 분포를 벡터로 만들어서 사용하였다. 그 결과로 만들어진 벡터는 11차원을 가진다(개체명 태그 5개 * BI정보 2개 + O 태그 1개 = 11). 예를 들어 “타자/O 는/O 단연/O 이대호/B-PS”문장에서 각 음절에 대응되는 음절 임베딩 벡터는 표 4과 같다. 표 4의 값들은 지면상 반올림 하여 표기하였다.

표 4 . 음절 단위 개체명 분포 벡터

	B					I					O
	PS	OG	LC	DT	TI	PS	OG	LC	DT	TI	
타	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.09	0.09	0.0
	9	9	9	9	9	9	9	9			9
자	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.09	0.09	0.0
	9	9	9	9	9	9	0	9			9
는	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.10	0.09	0.1
	9	9	9	9	9	9	9	9			0
단	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.09	0.09	0.0
	9	9	9	9	9	9	9	0			9
연	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.09	0.09	0.0
	0	9	9	9	9	9	9	9			9
이	0.1	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.08	0.08	0.0
	4	9	8	8	8	1	9	8			9
대	0.0	0.1	0.0	0.0	0.0	0.0	0.1	0.0	0.09	0.09	0.0
	9	1	9	9	9	9	0	9			9
호	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.09	0.09	0.0
	1	9	9	9	9	9	9	9			9

3.2.4 개체명 사전 자질 벡터

본 시스템에서는 개체명 사전 정보를 자질로 사용하여 단어 표상을 확장한다. 단어 표상을 확장하기 위한 개체명 사전은 대회 주최 측으로부터 전달 받은 사전을 사용하였고, 학습 데이터에서 출현한 개체명들을 추가로 사용하여 개체명 사전을 확장 하였다. 또한 개체명 사전 자질 벡터의 차원으로는 개체명 카테고리 수인 5차원을 사용 하였다. 개체명 사전 정보를 자질 벡터로 표현하기 위하여 현재 단어가 개체명 사전에 있으면 1 없으면 0으로 각 카테고리 별로 표현하였다. 개체명 사전이 표 5와 같을 때, 단어 "이대호"의 개체명 사전 자질 벡터의 예는 표 6과 같다.

표 5. 개체명 사전 예

개체명	개체명 태그
이민호	PS
이대호	PS
...	...
6개월	DT
6년간	DT
...	...

표 6. "이대호"의 사전 자질 벡터

PS	LC	OG	TI	DT
1	0	0	0	0

3.3 전체 구성도

본 시스템에서 제안하는 한국어 개체명 인식에 사용되는 bi-LSTM-CRFs의 전체 구성도는 그림 5와 같다.

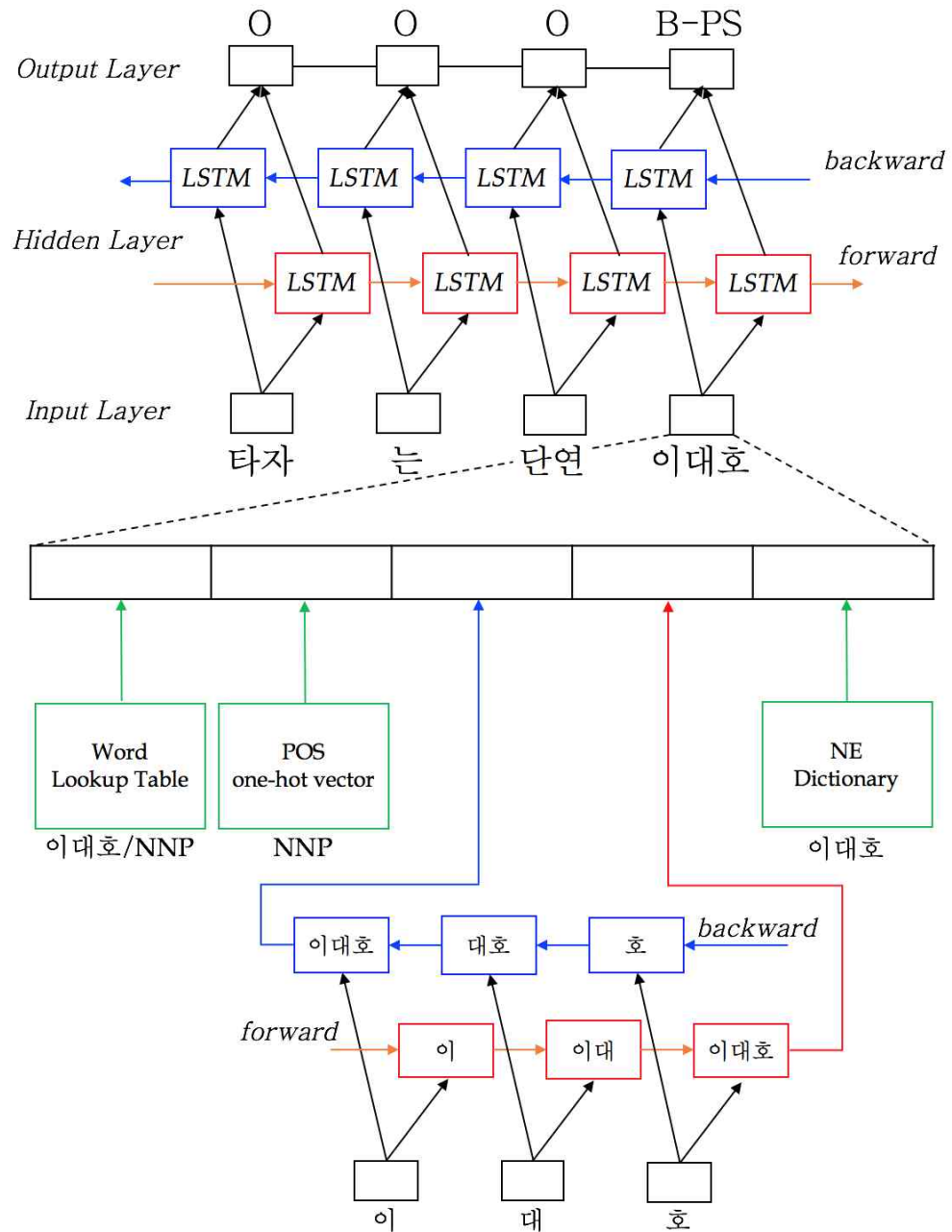


그림 22. 확장된 단어 표상을 이용한 전체 구성도

제 4 장 실험

4.1 실험 환경

한국어 개체명 인식에서 단어 표상 확장 방법의 성능 평가를 위해 bi-LSTM-CRFs를 TensorFlow로 구현하여 사용하였다. 또한 개체명 인식 모델 학습을 위해 국어경진대회 주최 측에서 배포한 학습 데이터 3,555 문장과 개발 데이터 501 문장을 사용하였다.

4.2 단어 임베딩 벡터 실험

단어 임베딩 벡터 실험에서는 단어의 차원을 50으로 하였고, 제공된 사전 학습된 단어 임베딩 벡터를 사용하였다. 단어 임베딩 벡터를 랜덤으로 초기화한 실험보다 사전 학습된 임베딩 벡터를 사용한 실험의 성능이 5%높았다.

표 7 . 단어 임베딩 벡터 실험 결과(F1)

	Dev
random	71.09
pretrain	76.09

4.3 품사 자질 벡터 실험

품사 자질 벡터 실험에서는 품사 자질 벡터를 one-hot 벡터로 추가하는 것이 좋은 자질이 될 수 있음을 보인다. 사전 학습된 단어 임베딩 벡터를 단독으로 사용한 실험 대비 품사 자질 벡터를 추가한 실험이 2.93% 증가하였다.

표 8 . 품사 자질 벡터 실험 결과(F1)

	Dev
pretrain	76.09
pretrain + pos	79.02

4.4 음절 기반 단어 임베딩 벡터 실험

음절 기반 단어 임베딩 벡터 실험에서는 음절 입력을 랜덤 벡터, 사전 학습된 음절 임베딩 벡터, 그리고 사전 학습된 음절 임베딩 벡터와 음절 단위 개체명 분포 임베딩 벡터를 결합한 벡터로 실험을 진행하였다. 랜덤 벡터와 사전 학습된 음절 임베딩 벡터는 50차원을 사용 하였고, 음절 단위 개체명 분포 임베딩 벡터는 11차원을 사용하였다. 사전 학습된 음절 임베딩 벡터로는 대회 주최 측으로부터 제공 받은 사전 학습된 형태소 단위 단어 임베딩 벡터 중 1음절인 경우를 음절 단위 임베딩 벡터로 보고 매칭 하여 사용하였다. 그 결과 랜덤으로 입력한 것 보다 사전 학습된 음절 임베딩 벡터와 음절 단위 개체명 분포 벡터를 결합한 성능이 0.82%향상하였고, 사전 학습 임베딩 벡터와 품사 자질 벡터만 사용한 실험보다 0.97%증가하였다.

표 9 . 음절 기반 단어 임베딩 벡터 실험 결과(F1)

	Dev
pretrain + pos	79.02
pretrain + pos + charLSTM(random)	79.17
pretrain + pos + charLSTM(pretrain)	79.44
pretrain + pos + charLSTM(pretrain + distribution)	79.99

4.5 개체명 사전 자질 벡터 실험

개체명 사전 자질 벡터 실험에서는 개체명 사전의 정보를 활용하여 자질 벡터로 추가하는 것이 좋은 자질임을 보인다. 개체명 사전 자질 벡터를 추가한 결과 음절 기반 단어 임베딩 벡터 실험보다 2.76%증가한 82.75%로 가장 높은 성능을 보였다.

표 10 . 개체명 사전 자질 벡터 실험 결과(F1)

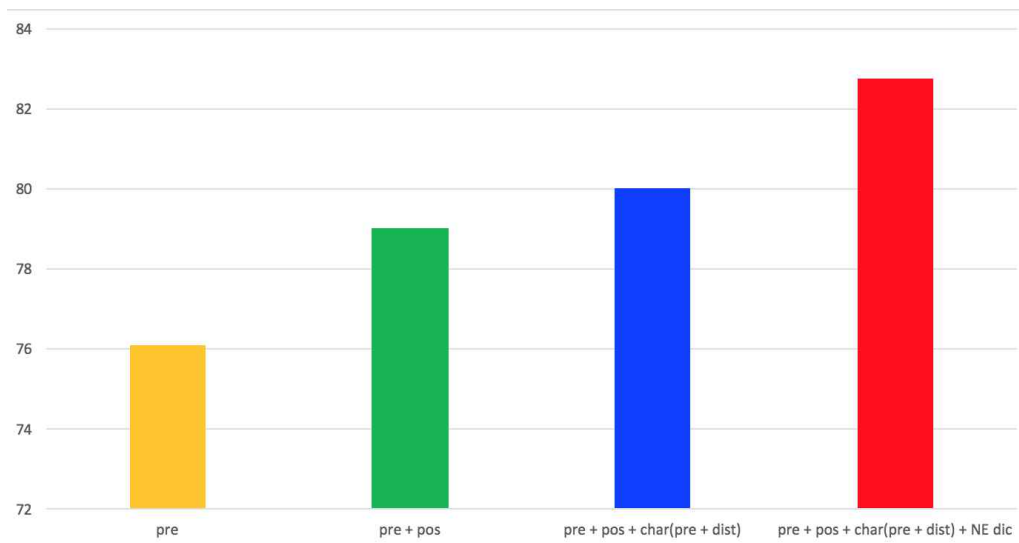
	Dev
pretrain + pos + charLSTM(pretrain + distribution)	79.99
pretrain + pos + charLSTM(pretrain + distribution) + NE dictionary	82.75

4.6 최종 실험 결과 비교

최종 실험 결과로는 총 네 종류의 실험 중 가장 높은 성능을 모아서 비교한다. 결과적으로 사전 학습된 단어 임베딩 벡터만을 사용한 실험 대비 품사 자질 벡터와 음절 기반 단어 임베딩 벡터, 그리고 개체명 사전 자질 벡터를 추가한 실험의 성능이 6.66% 증가 하였다.

표 11 . 최종 실험 결과 비교(F1)

	Dev
pretrain	76.09
pretrain + pos	79.02
pretrain + pos + charLSTM(pretrain + distribution)	79.99
pretrain + pos + charLSTM(pretrain + distribution) + NE dictionary	82.75



제 5 장 참고문헌

1. 유홍연, 고영중, "품사 임베딩과 음절 단위 개체명 분포 기반의 Bidirectional LSTM CRFs를 이용한 개체명 인식", 2016 HCLT 심사 발표 예정

국립국어원

Annie

임재수
【개인 참가】

차 례

제 1 장 소프트웨어 소개	40
1.1 소프트웨어 명칭	40
1.2 소프트웨어 사용 환경	40
1.3 소프트웨어 특징	41
제 2 장 소프트웨어 설치 및 실행	42
2.1 소프트웨어 설치 방법	42
2.1.1 다운로드에 의한 방법	42
2.1.2 git clone에 의한 방법	42
2.2 소프트웨어 파일 구조	42
2.2.1 주요 파일 설명	42
2.2.2 전체 구조	43
2.3 소프트웨어 실행 방법	44
제 3 장 소프트웨어 기능	46
3.1 프로그램 기능	46
3.1.1 성능 평가	46
3.1.2 모델 학습	46
3.2 프로그램 기능 제약	47
제 4 장 기타	48
4.1 CRF 자질	48
4.2 SVM 자질	48
4.3 성능 평가	48

제 1 장 소프트웨어 소개

1.1 소프트웨어 명칭

- Annie 는개체명을뜻하는 Named Entity, 줄여서 NE라고 부르는 발음 유사성을 이용한 일종의 유희적 이름입니다.
- (제 옆자리에 앉아 있는 사람의 이름이라고 절대 밝힐 수 없습니다.)
- 본 소프트웨어를 개발하게 된 동기는 오픈소스 시대에 걸맞게 기존의 소프트웨어를 조합하여 빠르게 아이디어를 실현하는 것이 가능한 지를 실험해 보는 것이었습니다.

1.2 소프트웨어 사용 환경

- 지원 OS: Linux, Mac OS
- 권장 메모리: 4GB 이상
- 사용 언어: Python
- 실행 환경: Python 2.6 혹은 2.7
- 텍스트 인코딩: UTF-8
- 의존 패키지
 - CRFsuite
 - author: Naoaki Okazaki
 - title: CRFsuite: a fast implementation of Conditional Random Fields (CRFs)
 - year: 2007
 - url: <http://www.chokkan.org/software/crfsuite/>
 - libsvm
 - author: Chang, Chih-Chung and Lin, Chih-Jen
 - title: LIBSVM: A library for support vector machines
 - journal: ACM Transactions on Intelligent Systems and Technology
 - volume: 2
 - issue: 3
 - pages: 27:1--27:27
 - year: 2011
 - url: <http://www.csie.ntu.edu.tw/~cjlin/libsvm>
 - scikit-learn
 - title: Machine Learning in Python
 - url: <http://scikit-learn.org>

1.3 소프트웨어 특징

- 본 소프트웨어는 순수하게 Python 언어로 구현되어 있어 구현이 간단하고 이해하기 쉬워 수정이 용이합니다.
- Python 프로그램의 소스코드는 PEP8¹스타일 가이드를 준수합니다.
- 또한 처음 개발을 시작할 때부터 GitHub에 소스 및 개발 과정(이슈)이 공개되어 누구나 재현해 볼 수 있습니다.

¹ <https://www.python.org/dev/peps/pep-0008/>

제 2 장 소프트웨어 설치 및 실행

2.1 소프트웨어 설치 방법

2.1.1 다운로드에 의한 방법

- 배포 페이지¹에서 원하는 버전을 다운로드받습니다.

```
$ wget https://github.com/krikkit/annie/archive/0.5.tar.gz
```

- 압축을 풀면 `annie-$VERSION` 디렉토리가 생성됩니다. (문서를 작성하는 시점의 최신 버전은 0.5입니다.)

```
$ tar -xzf 0.5.tar.gz
```

2.1.2 git clone에 의한 방법

- 최신 소스코드를 설치하고자 한다면 다음과 같이 ``git clone`` 명령을 수행합니다.

```
$ git clone https://github.com/krikkit/annie.git
```

2.2 소프트웨어 파일 구조

2.2.1 주요 파일 설명

- JSON 포맷의 입력 파일을 받아들이고 역시 JSON 포맷으로 개체명을 인식하는 일련의 과정을 한번에 진행하는 스크립트와, 이 스크립트의 실행에 필요한 디렉토리는 다음과 같습니다.
 - `cqasys_annie.bash`: 일괄 실행 스크립트
 - `bin/`: 각종 단계별 실행 스크립트/바이너리 프로그램
 - `json2feat.py`: JSON 포맷으로부터 CRF 자질을 추출하는 프로그램
 - `crfsuite.Linux`: CRFsuite 태깅 및 학습을 위한 실행 바이너리

¹ <https://github.com/krikkit/annie/releases>

- iob2json.py: IBO2 태깅된 결과로부터 JSON 포맷을 생성하는 프로그램
- tag_ps.py: 3음절 인명을 추가적으로 태깅하는 프로그램
- env/: Python 의존 패키지
 - venv_annie.tar.gz: Python virtualenv 환경 (Cent OS 6.6 x64)
- lib/: 실행 스크립트가 참조하는 라이브러리
- rsc/: 실행에 필요한 리소스
 - crf.model.gz: CRF 모델
 - ◆ 최초 실행 시 crf.model 파일로 압축 해제
 - gazette.annie: train 코퍼스와 기 배포된 gazette를 병합해 구축한 개체명 사전
 - nusvc_model.pkl: SVM 모델 파일
 - word2vec.pkl.gz: 기 배포된 wikiCorpus_word2vector.hr 파일을 가공한 파일
 - ◆ 최초 실행 시 word2vec.pkl 파일로 압축 해제

2.2.2 전체 구조

```

annie/
├─ bin/
│   ├── baseline.py
│   ├── build_gazette.py
│   ├── crfsuite.Darwin
│   ├── crfsuite.Linux
│   ├── eval.py
│   ├── index_word2vec.py
│   ├── iob2json.py
│   ├── json2feat.py
│   └─ tag_ps.py
├─ env/
│   ├── requirements.txt
│   └─ venv_annie.tar.gz
├─ lib/
│   ├── feature.py
│   ├── gazette.py
│   └─ getopts.py

```

```
| └─ sentence.py
|   └─ word2vec.py
└─ notebook/
    │ └─ NNP+NNG_2+3.ipynb
    │ └─ README.md
    │ └─ ps_classifier.ipynb
    │ └─ ps_histogram.ipynb
└─ rsc/
    │ └─ crf.model.gz
    │ └─ gazette.annie
    │ └─ nusvc_model.pkl
    │ └─ word2vec.pkl.gz
└─ README.md
└─ cqasys_annie.bash
└─ user_manual.md
```

2.3 소프트웨어 실행 방법

- 아래와 같이 스크립트를 실행합니다.

```
$ ./cqasys_annie.bash -i dev.json
0) setting environments
1) convert JSON input to CRF feature
2) tag with CRF model
3) convert IOB2 to JSON
WARNING:root:I- category is different from B-
WARNING:root:I- category is different from B-
WARNING:root:I- category is different from B-
4) tag PS NEs
```

- 출력 파일을 따로 지정하지 않으면 `result.json` 파일이 출력됩니다.
- `cqasys_annie.bash` 스크립트의 사용법은 아래와 같습니다.

```
$ ./cqasys_annie.bash --help
Usage: cqasys_annie.bash [options]
Options:
  -h, --help      show this help message and exit
  -i FILE          input file
  --rsc-dir=DIR    resource dir <default: ./rsc>
  --output=FILE    output file <default: result.json>
```

- 스크립트 내부적으로 다음 절차에 따라 진행됩니다.
 1. python 의존 패키지(가상 환경)를 `/tmp/venv_annie`에 압축을 풉니다. (최초에 한번만 수행됩니다.)
 2. JSON 입력 파일로부터 CRF 자질을 추출합니다.
 3. CRFsuite 실행 파일과 CRF 모델을 이용하여 개체명 인식을 수행합니다.
 4. 개체명을 태깅한 IOB2 출력과 입력 JSON 파일을 이용해 출력 형태를 JSON 포맷으로 만듭니다.
 5. scikit-learn 패키지와 SVM 모델을 이용해 3음절 인명을 추가로 태깅합니다.

제 3 장 소프트웨어 기능

3.1 프로그램 기능

- 이하 Python 프로그램을 개별적으로 실행하려면 먼저 다음과 같이 환경 변수를 지정해 라이브러리의 위치를 설정해 줍니다.

```
$ export PYTHONPATH=`pwd`/lib
```

3.1.1 성능 평가

- 시스템에 의해 자동으로 태깅한 JSON 파일과 정답 JSON 파일을 비교하여 다음과 같이 precision/recall을 측정할 수 있습니다.

```
$ bin/eval.py -g dev.json --input=result.json
```

3.1.2 모델 학습

- 기 배포된 `gazette` 파일과 `train.json` 파일을 병합해 새로운 사전 파일을 생성합니다.

```
$ bin/build_gazette.py -g  gazette -c train.json  
--output=./rsc/gazette.annie
```

- `train.json` 파일을 이용해 CRF 모델 학습을 위한 자질 포맷으로 출력합니다.

```
$ bin/json2feat.py -g  ./rsc/gazette.annie --input=train.json  
--output=train.crffeat
```

- 자질로 표현된 파일을 이용해 CRF 모델을 학습합니다.

```
$ bin/crfsuite.`uname`  learn -m ./rsc/crf.model train.crffeat
```

- 기 배포된 word2vec 파일 `wikiCorpus_word2vector.hr`를 인덱싱 합니다.

```
$ bin/index_word2vec.py -o ./rsc/wod2vec.pkl
--input=wikiCorpus_word2vector.hr
```

- **scikit-learn** 사용을 위해 아래 패키지를 설치합니다.
 - scikit-learn: v0.17.1
 - numpy: v1.6.1
 - scipy: v0.9.0
- **Python** 패키지 설치 프로그램인 `pip`를 이용해 아래와 같이 한번에 설치 합니다.

```
$ pip install -r env/requirements.txt
```

- 혹은 `virtualenv`가 설치되어 있는 Cent OS 6.6 x86 리눅스 시스템에서는 아래와 같이 압축을 풀고 가상 환경에 진입합니다.

```
$ tar -C /tmp -xzf /path/to/annie/env/venv_annie.tar.gz
$ source /tmp/venv_annie/bin/activate
```

- **scikit-learn** 패키지가 설치되었다면, 마지막으로 `notebook/ps_classifier.ipynb` 파일을 Jupyter Notebook¹을 이용해 실행합니다. `rsc/` 디렉토리 아래 SVM 모델 파일 `nusvc_model.pkl`이 생성됩니다.

3.2 프로그램 기능 제약

- 본 프로그램은 Cent OS 6.6 x64 리눅스 환경에 맞춰져 있습니다.

¹ <http://jupyter.org/>

제 4 장 기타

4.1 CRF 자질

- CRF 학습을 위해 -2 ~ +2 위치의 5개 형태소를 기반으로 아래의 자질을 사용했습니다.
 - lemma, lemma bigram
 - pos tag, tag bigram, tag trigram
 - gazette 사전 매칭 IOB2 태그 및 그 bigram, trigram
 - 1, 2음절 prefix/suffix
 - 형태소의 길이는 단독이 아니라 아래의 조합만 사용했습니다.
 - gazette 사전 매칭 + 길이
 - 1, 2음절 prefix/suffix + 길이
 - lexical form(pattern)
 - 한글 -> '가', 한자 -> '漢', 영문 -> 'A', 숫자 -> '0', 기호 -> '.'
 - begin/end of sentence
 - begin/middle/end of word(어절)
 - 이전 어절의 마지막 형태소, 다음 어절의 첫 형태소 및 그 결합

4.2 SVM 자질

- CRF 학습 후 인명의 재현율이 다소 떨어지는 데 착안하여 3음절 NNP 품사를 갖는 단일 형태소에 대해 별도의 인명 분류기를 학습했습니다.
- SVM 학습을 위해 -2 ~ +2 위치의 기 배포된 형태소 embedding 벡터 (50 x 5 = 250차원)를 사용했습니다.

4.3 성능 평가

- `train.json` 파일을 이용해 학습한 후 `dev.json` 파일에 측정한 최종 성능은 아래와 같습니다.

```

===== DT =====
# of NEs in gold standard:  316
# of NEs in test file      : 304
# of NEs in both(matched):  276
Precision: 0.9079
Recall:    0.8734
F1-score:  0.8903
  
```



```
===== LC =====
# of NEs in gold standard:  238
# of NEs in test file      : 285
# of NEs in both(matched):  216
Precision: 0.7579
Recall:    0.9076
F1-score:  0.8260

===== OG =====
# of NEs in gold standard:  412
# of NEs in test file      : 327
# of NEs in both(matched):  285
Precision: 0.8716
Recall:    0.6917
F1-score:  0.7713

===== PS =====
# of NEs in gold standard:  581
# of NEs in test file      : 502
# of NEs in both(matched):  463
Precision: 0.9223
Recall:    0.7969
F1-score:  0.8550

===== TI =====
# of NEs in gold standard:  42
# of NEs in test file      : 42
# of NEs in both(matched):  38
Precision: 0.9048
Recall:    0.9048
F1-score:  0.9048

===== TOTAL =====
# of NEs in gold standard:  1589
# of NEs in test file      : 1460
# of NEs in both(matched):  1278
Precision: 0.8753
Recall:    0.8043
F1-score:  0.8383
```


국립국어원

KAISER

함영균, 최동호, 최기선

【한국과학기술원】

차 례

제 1 장 소프트웨어 소개	54
1.1 소프트웨어 명칭	54
1.2 소프트웨어 사용 환경	54
1.3 소프트웨어 특징	54
제 2 장 소프트웨어 설치 및 실행	55
2.1 소프트웨어 설치 방법	55
2.2 소프트웨어 파일 구조	55
2.2.1 주요 파일 설명	55
2.2.2 전체 구조	55
2.3 소프트웨어 실행 방법	56
제 3 장 소프트웨어 기능	56
3.1 프로그램 기능	56
3.2 프로그램 기능 제약	56

제 1 장 소프트웨어 소개

1.1 소프트웨어 명칭

KAISER (KAIST Sensitive NER tagger)

1.2 소프트웨어 사용 환경

- Ubuntu 14.04 이상
- 2GB ram
- Python
- 실행 환경: 리눅스 서버 환경
- 텍스트 인코딩: utf-8

1.3 소프트웨어 특징

현존하는 개체명 인식 시스템들은 전문가들의 수작업 특질과 지식에 의해 어노테이션으로 구축된 작은 규모의 지도학습 데이터를 학습데이터로 사용한다. 이러한 개체명 인식기의 성능은 학습데이터의 양, 언어적 특질, 그리고 개체명 사전에 의해서 결정되는 것으로 알려져 있다. 영어의 경우에는 오랜 시간 다양한 학술대회 등을 통하여 양질의 학습데이터가 공개되어 있으며, 또한 영어의 언어적 특질(대문자로 시작, 관사 등) 및 개체명 사전에 의해 상당히 많은 성능향상이 이루어지고 있다.

한국어의 경우에는 영어와 다른 언어적 특질을 갖고 있어 위와 같은 방법론을 적용하기 어려우며, 다른 언어권에 비해 공개되어 있는 학습데이터 및 개체명 사전이 부족한 것이 사실이다.

본 KAISER 시스템은 이러한 문제를 해결하기 위해 다음과 같은 방법을 사용하였다.

첫째, 개체명 학습 데이터에 부착된 개체명 태그 이외에도, 유의어들에 대한 동일한 개념태그를 부착하였다. 이를 위해, 한국어 워드임베딩 단어들을 클러스터링 하여, 유의어의 경우 동일한 아이디 값을 갖도록 하였고, 이를 기계학습의 특질로 사용하였다.

둘째, 개체명 사전에 대한 문자열 매칭이 아닌, 워드임베딩 벡터간의 유사도 계산을 통해 개체명 사전의 적용범위를 확장시켰다.

이를 통해, 적은 리소스를 사용하는 한국어 개체명 인식 시스템의 성능을 전반적으로 향상시킬 수 있을 것으로 기대한다.

제 2 장 소프트웨어 설치 및 실행

2.1 소프트웨어 설치 방법

1. KAISER.tar.gz 파일의 압축을 푼다.
2. 폴더 내에 있는 CRF++-0.58.tar.gz 파일의 압축을 푼다. 경로를 지정하는 경우, 문제가 생길 수 있다.
3. 압축을 풀고 생긴 CRF++-0.58폴더로 이동해서 다음을 입력한다:
 - a. % ./configure
 - b. % make
 - c. % su
 - d. # make install

2.2 소프트웨어 파일 구조

2.2.1 주요 파일 설명

- ① 실행 파일
KAISER.sh --- shell script based 실행 파일.
사용법: ./KAISER.sh ./training_data/dev.txt ./output.txt
와 같이 ./KAISER.sh [인풋] [아웃풋] 의 형식으로 사용 가능.
- ② 데이터 파일
./lib/ --- 클러스터변환 파일, 학습 모델, word2vector 데이터 등.
- ③ 사전 파일
./lib/gazette --- 주어진 사전 파일

2.2.2 전체 구조

KAISER/	
word2vec/.....	word2vec 파일 폴더
src/.....	소스파일 폴더
lib/.....	resources 폴더
training_data/.....	sample files(optional)
CRF++-0.58.tar.gz.....	CRF++ 프로그램 압축파일

2.3 소프트웨어 실행

KAISER.sh --- shell script based 실행 파일.

사용법: ./KAISER.sh ./training_data/dev.txt ./output.txt

위와 같이 ./KAISER.sh [인풋] [아웃풋] 의 형식으로 사용 가능.

제 3 장 소프트웨어 기능

3.1 프로그램 기능

- 개체명 클러스터링 기능

본 시스템은 nominal-feature CRF 기계학습을 사용하였기에, 단어의 워드 임베딩의 실제 벡터값을 입력으로 사용하지 않았다. 그러나 유의어들에 대해 동일한 nominal-feature를 제공하기 위해 워드 임베딩 데이터를 클러스터링하여 고유값을 태깅하였고, 해당 모듈은 소프트웨어에 포함되어 있다.

- 개체명 유의어 탐색 기능

본 시스템은 사용하는 개체명 사전의 양이 적다는 점에 착안, 개체명 사전에 대한 문자열 매칭 방법이 아닌, 단어와 개체명 사전 단어와의 워드 임베딩의 코사인 유사도 방식을 적용하였다.

3.2 프로그램 기능 제약

- 입력 형식:
[word] [tag]
- 개체명 사전의 형식:
[word] [tag]

국립국어원

WordAngler

김보겸, 강명윤, 민태홍, 이재성

【충북대학교】

차 례

제 1 장 소프트웨어 소개	60
1.1 소프트웨어 명칭	60
1.2 소프트웨어 사용 환경	60
1.3 소프트웨어 특징	60
제 2 장 소프트웨어 설치 및 실행	62
2.1 소프트웨어 설치 방법	60
2.2 소프트웨어 파일 구조	60
2.2.1 주요 파일 설명	60
2.2.2 전체 구조	63
2.3 소프트웨어 실행 방법	64
제 3 장 소프트웨어 기능	65
3.1 프로그램 기능	65
3.2 프로그램 기능 제약	67
제 4 장 기타	68
4.1 시스템 성능	68
4.2 프로그램 수행 과정	69

제 1 장 소프트웨어 소개

1.1 소프트웨어 명칭

본 소프트웨어는 '2016 국어정보처리 시스템 경진대회'의 지정분야인 '개체명 인식 시스템' 개발에 참가하는 소프트웨어로, 기계학습 모델인 Conditional Random Fields (CRFs)를 사용하여 개체명을 자동으로 추출하는 프로그램이다. 프로그램명인 'WordAngler'는 단어를 의미하는 'word'와 낚시꾼을 의미하는 단어인 'angler'의 합성어로, '주어진 문장에서 적절한 개체명 단어들을 찾아 낚아 올리는 프로그램'이라는 의미로 사용하였다.

1.2 소프트웨어 사용 환경

지원 OS	Windows(7, 8, 10), Linux ubuntu
권장 메모리	1GB
사용 언어	C, C++
실행 환경	g++ 4.8.1, visual studio 2012, 2013
텍스트 인코딩	UTF-8
open library	jsoncpp 4.8, CRFsuite 0.12

<표 1> 소프트웨어 사용 환경

1.3 소프트웨어 특징

본 소프트웨어는 CRFsuite 오픈 소스 라이브러리를 사용하여 구축된 기계학습 기반의 개체명 인식/추출 프로그램이다. C++ 클래스 기반으로 작성하여, 프로그램의 확장이 매우 용이하며, 현재는 경진대회의 시스템 제약사항으로 인해 사용하고 있지는 않지만, 사용자 사전을 활용할 수 있도록 구조를 설계하여, 향후 모듈의 성능을 쉽게 개선할 수 있도록 하였다. 학습 데이터로 사용 가능한 자질 값은 형태소 어휘 및 품사, 어절 정보와 word2vec을 사용한 단어 벡터로 매우 제한이 되어 있어, 기계학습을 활용하기가 쉽지 않지만, 본 프로그램에서는 word2vec의 단

어 벡터를 n 개의 클래스로 분류하여 자질로 사용하거나, 단어 벡터의 유사도 점수를 활용하여, 유사 어휘 정보를 자질로 사용하는 등, 모듈의 성능을 높일 수 있는 유용한 자질 값을 새롭게 개발하여 사용하였다. 만약 의존구문분석, WordNet 등의 어휘 자원을 활용한다면 더욱 성능을 높일 수 있을 것으로 기대한다. 자세한 기능 및 수행 과정은 3장과 4장에서 단계별로 설명한다.

제 2 장 소프트웨어 설치 및 실행

2.1 소프트웨어 설치 방법

- 1) klpexpo_lake.tar.gz 파일 압축 해제
 > tar -zxvf klpexpo_lake.tar.gz
- 2) klpexpo_lake/compile/ 폴더에서 make 수행(실행파일 컴파일)
- 3) klpexpo_lake/test/ 폴더에서 Test.sh 수행 (실행)

2.2 소프트웨어 파일 구조

2.2.1 주요 파일 설명

- ① 실행 파일 – klpexpo_lake/test/bin/
 - LearnData2CoNLL.run: 학습데이터(train.json)를 CoNLL 포맷으로 변경
 - CoNLL2CRFsuite.run: 생성된 CoNLL 데이터와 CRF에서 사용할 자질 값 설정 파일(FeatureTemplate.txt)을 읽어, CRFsuite의 학습 데이터 생성
 - crfsuite: CRFsuite 학습을 위한 프로그램
 - Kmeans4WordVectors.run: 배포된 word2vec 데이터 파일 (wikiCorpus_word2vector.hr)을 k-means 알고리즘을 사용하여 n개의 class로 분류하는 프로그램
 - cqasys_WordAngler.run : 수행 파일(dev.json)을 입력받아 개체명을 찾은 후 출력하는 메인 프로그램
- ② 데이터 파일 – klpexpo_lake/test/LearnData
 - FeatureTemplate.txt: CRFsuite에서 사용할 자질 값 설정 파일
 - train.300.crfsuite.model: 미리 학습된 CRFsuite 모델
 - wikiCorpus_word2vector.300.class: 배포된 word2vec 파일을 300개의 클래스로 분류한 파일
- ③ 사전 파일 – klpexpo_lake/test/Data
 - gazette: 배포된 사전 파일
- ④ 라이브러리 – klpexpo_lake/compile/
 - libjson_linux-gcc-4.8_libmt.a: jsoncpp 오픈 라이브러리
 - libcrfsuite.a, liblbfgs.a, libcqdb.a: CRFsuite 오픈 라이브러리와 학습 및

수행에 필요한 오픈 알고리즘

2.2.2 전체 구조

• 실행 프로그램 컴파일

klpexpo_lake/compile/

Common/ 입출력 구조체 헤더 및 jsoncpp, crfsuite 오픈 라이브러리 헤더

* jsoncpp/ ... jsoncpp 라이브러리 헤더

* CRFsuite/ ... crfsuite 라이브러리 헤더

DataConstructor/ .. CoNLL 및 CRFsuite 학습 데이터 생성 클래스

DataLoader/ 입력 자료 및 학습 데이터 로딩 클래스

DataExtractor/ 사전 검색 및 WordAngler 및 crfsuite 메인 클래스

* WordAngler.h/cpp 프로그램 메인 클래스

* WordAngler_DicSearch.h/cpp ... 사전 검색 클래스

* CRFsuiteMain.h/cpp CRFsuite 메인 클래스

* libjson_linux-gcc-4.8_libmt.a: jsoncpp 오픈 라이브러리

* libcqdb.a, libcrfsuite.a liblbfgs.a: crfsuite 및 학습/수행에 필요한 오픈 라이브러리

* cqasys_WordAngler.cpp – 프로그램 메인 wrapper

• 프로그램 실행

klpexpo_lake/test/

bin/ 학습 및 수행 프로그램

Data/ 배포된 학습 및 수행 데이터, 사전파일

LearnData/ 모델 학습 데이터

execute/ 프로그램 수행이 이루어질 공간

* Learn.sh 학습 프로그램 실행 쉘

* Test.sh 수행 프로그램 실행 쉘

2.3 소프트웨어 실행

- **cqsys_WordAngler.run** (프로그램 옵션)

- i 분석_대상_입력_파일명
- o 분석_결과_출력_파일명
- rsc 학습_데이터_경로
- crf CRFsuite_학습_모델_파일명
- w2v word2vec를_n개_클래스로_분류한_파일명
- dic 사전사용여부(0: 사용안함, 1: 사용(default))

- 수행 예제

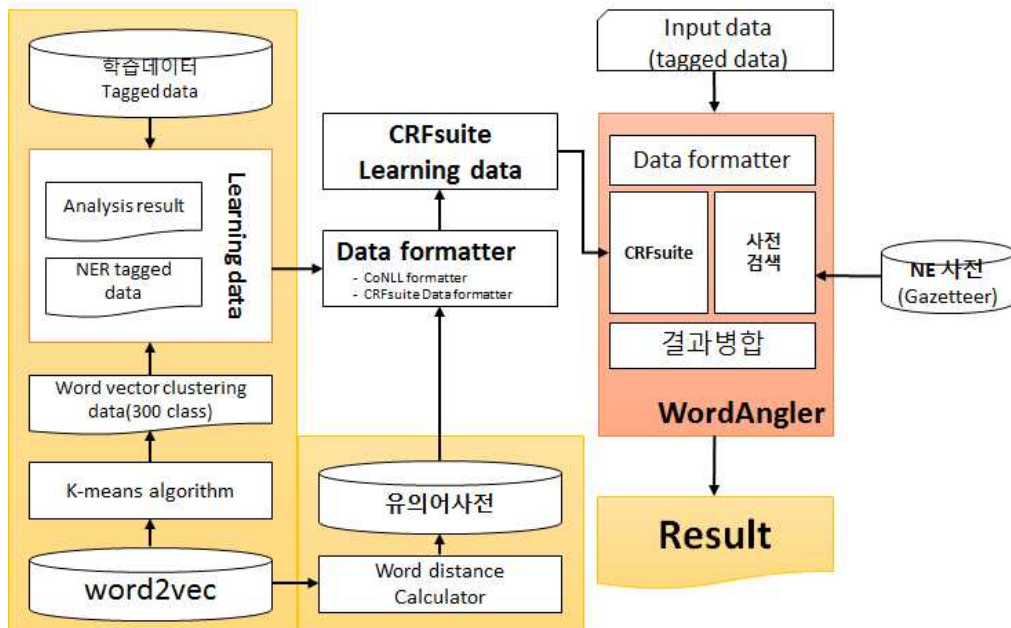
```
> cqsys_WordAngler.run -i dev.json -o result.json -rsc ./ -crf  
train.300.crfsuite.model -w2v wikiCorpus_word2vector.300.class -dic 1
```

참고: -crf , -w2v, -dic 옵션은 default 설정이 되어 있음.

제 3 장 소프트웨어 기능

3.1 프로그램 기능

본 프로그램은 오픈 소스로 공개된 기계학습 모델 중 사용이 쉽고 간편하며, 성능적으로 우수함이 검증된 CRF(Conditional Random Fields) 모델을 사용하였다. CRF는 자연어처리 뿐만 아니라 다양한 분야에서 활용이 되고 있는 대표적인 모델로서, 레이블 부착 확률을 여러 자질들의 조건에 의해 확률을 계산하기 때문에 필요한 모든 자질을 사용하여 구현이 가능하며, 수정 및 변경이 매우 용이하다는 장점이 있다. CRF를 지원하는 오픈 소스로는 MALLET, SCARF, CRF++, CRFsuite 등이 있으며, 이 중 CRFsuite는 여러 CRF 기반 오픈 소스 중 속도가 가장 빠름이 증명되었으며, 사용이 쉽고, 자질 조건의 수정이 편리하며, 다양한 학습 알고리즘을 지원하고 있어, 본 프로그램은 이를 사용하였다. 이외에도 본 대회에서 제공된 개체명 사전인 'gazetteer' 사전과, word2vec를 사용한 단어 벡터를 k-means 알고리즘을 사용하여 n개의 클래스로 분류한 뒤 자질 값으로 사용하고, 단어 벡터간의 유사도 점수를 계산하여, 각각의 형태소들의 유사 어휘 사전을 새롭게 구축하여 사용하는 등, 다양한 방법을 적용하였다. 전체 시스템의 구성은 다음과 같다.



<그림 1> WordAngler 시스템 개요도

CRF 모델에서 사용하는 자질 값(Feature) 은 다음과 같다.

자질값(Feature)	범위(scope)
형태소 어휘(lemma)	[-2,2]
형태소 태그(tag)	[-2,2]
형태소 어휘 / 형태소 품사	[-2,2]
어절 띄어쓰기(word-delimiter)	[-1,1]
<i>word vector clustering number</i>	[-2,2]
형태소 유사 어휘(lemma)	[-1,1]
형태소 태그 유사 어휘(tag)	[-1,1]

<표 2> 개체명 인식에 사용한 자질 값 목록 및 어휘 범위

자질 값 목록 중 'word vector clustering number'는 학습 데이터로 제공된 형태소들의 'word vector(wikiCorpus_word2vector.hr)'를 k-means 알고리즘을 사용하여, n개의 클래스로 분류한 뒤 사용한 클래스 번호이다(예를 들어, '대한민국/NNP' 형태소 벡터의 클래스 번호는 '265'이다).

자질로 사용한 유의어 사전은 학습 데이터로 제공된 word2vec 데이터에 포함된 각각의 형태소들 간의 유사도 점수를(cosign similarity) 모두 계산하여, 각 형태소별 가장 유사한 형태소를 하나의 pair 쌍으로 묶어 생성하였다. 이렇게 생성된 유의어 사전은 학습 데이터의 부족 문제를 일부 해결할 수 있는 가능성을 제공한다. 예를 들면, 학습 데이터에 '전남 드래곤즈/OG'가 없는 경우, 생성된 유의어 사전을 보면 '전남'은 '포항'으로, '드래곤즈'는 '스틸러스'로 가장 유사한 어휘가 생성되어 있어, '포항 스틸러스' 정보를 사용하여 개체명을 추출할 수 있다.

CRF 모델의 수행이 끝나고 나면, 학습 데이터로 제공된 사전('gazette')을 사용하여 사전에 등록되어 있는 개체명을 추출한다. 개체명 사전을 검색하는 방법은 형태소 단위로 총 4개의 형태소를 최대 개체명 단위로 보고, 문장의 시작 형태소 위치부터, 1~4개의 형태소를 차례로 합치면서 사전을 검색하고, 사전에서 찾은 개체명은 긴 이름 우선순위로 추출하였다. 사전 검색 알고리즘은 다음과 같다.

```

cnt_morp = count(morphemes in sentence)
for i = 0 to cnt_morp-1
    word.clear();
    for j = i to i+4 && j < cnt_morp
        word += morphemes[j]->lemma
        if (search in dictionary(word)) then ne_candidate = (word, ne_tag)
    if (!ne_candidate.empty()) then result <- ne_candidate

```

<표 3> 개체명 사전 검색 알고리즘

사전 검색은 해시 테이블 알고리즘을 사용하였으며, 사용자 사전을 사용할 수 있도록 프로그램 구조를 구성하였으나, 현재에는 제공된 시스템 사전만을 사용하여 정보를 추출한다.

CRF와 사전 검색이 끝나고 나면, 두 개의 출력 결과 중 어휘의 범위가 서로 교차 혹은 일치하는 결과가 존재한다. 이러한 경우에는 개체명의 길이가 보다 긴 이름 우선순위로 정답을 출력하며, 어휘의 범위가 같을 경우에는 사전 등록 개체명 우선순위로 정답을 출력하였다.

본 프로그램의 결과를 살펴보면, 아직 해결해야 할 문제들이 몇 가지 존재한다. 예를 들면, 동일한 어휘에 대한 중의성 태그 문제(예> 서울 - 지명으로 사용되었을 경우 LC(장소), 축구팀 이름으로 사용되었을 경우 OG(기관명)), 미등록어에 의한 추출 오류(예> '존 웨인'의 경우 PS(사람이름)으로 태깅되어야 하나, 이름인 '웨인'만 태깅하는 경우), 복합 명사의 인식 오류(예> 'LA 애인절스/OG'를 'LA/LC' + '애인절스/OG'로 분석하는 경우) 등이다. 현재는 이와 관련된 문제를 해결하기 위해 학습 데이터로 제공된 단어 벡터의 유사도를 사용하여 해결하는 모듈을 개발하고 있으나, 아직 모델이 추가되지는 못하였다. 이는 추후 시스템을 개선해 나가며 기능을 추가할 예정이다.

3.2 프로그램 기능 제약

- 입출력 양식: 경진대회에서 배포한 형태소 분석이 완료된 JSON 형식의 파일
- 필수 라이브러리: jsoncpp, crfsuite
- 입출력 파일 인코딩: UTF-8

제 4 장 기타

4.1 시스템 성능

시스템의 성능 측정은 테스트용으로 제공된 자료인 'dev.json' 파일을 대상으로 수행하였으며, 학습 데이터는 제공된 자료 중 'train.json'과 'wikiCorpus_word2vector.hr'를 사용하였다. 정답의 일치여부는 어휘 범위(boundary)의 완전 일치와 정답 개체명 태그의 일치, 두 가지 조건을 모두 만족하는 경우를 정답으로 평가하였으며, 평가 지수로는 재현율(recall)과 정확률(precision)과 F1-score로 계산하였다. 평가 모델은 CRF 모델만 사용한 '모델 1'과 CRF와 사전을 함께 사용한 '모델 2'이다.

$$\text{재현율}(\text{recall}) = \frac{\text{cnt}(\text{정답과 일치하는 결과})}{\text{cnt}(\text{정답 개체명})} \quad \text{식(1)}$$

$$\text{정확률}(\text{precision}) = \frac{\text{cnt}(\text{정답과 일치하는 결과})}{\text{cnt}(\text{시스템 출력 결과})} \quad \text{식(2)}$$

$$\text{F1-score} = \frac{2 * \text{recall} * \text{precision}}{\text{recall} + \text{precision}} \quad \text{식(3)}$$

	모델 1	모델 2
recall	0.726	0.829
precision	0.796	0.734
F1-score	0.759	0.778

<표 4> 시스템 평가 결과

사전을 함께 사용한 '모델 2'가 CRF만 사용한 '모델 1'보다 F1-score 기준으로 약 2%point 성능이 높았다. 이는 학습 데이터에 존재하지 않는 개체명이 사전에 등록된 경우가 많기 때문에, 전체적인 재현율을 약 10% point 높일 수 있었다. 단, 정확률은 약 6%point의 성능이 떨어진다. 이는 'LA 애인절스/OG'와 같이 복합 명사로 구성된 경우, 각 형태소를 하나의 개체명을 인식할 수 있기 때문으로 보인다(예를 들면, 'LA/LC', '애인절스/OG'). 자세한 세부 오류의 유형은 4.2절에 정리하였으며, 현재는 각 오류 유형별 해결을 위한 모듈 개발을 진행 중에 있다.

4.2 프로그램 수행 과정

4.2.1 학습 프로그램 수행 과정(Learn.sh)

(1) Kmeans4WordVectors.run

- word2vec 학습 자료('wikiCorpus_word2vector.hr') Clustering
- CRFsuite 모델의 자질 값으로 사용할 단어 벡터의 클러스터 번호 생성.
30, 50, 100, 150, 300, 500 개의 클러스터로 실험한 결과, 가장 성능이 좋았던 300개의 클러스터 생성
- 학습에 사용할 자료로 미리 생성해 놓음
- 출력 결과

</s>	210
NUM/SN	254
./SF	242
의/JKG	47
하/XSV	181
다/EF	265
,/SP	202
에/JKB	133
ㄴ/ETM	163

<그림 2> 단어 벡터의 클러스터링 결과

(2) LearnData2CoNLL.run

- 학습 데이터('train.json')과 word vector clustering 자료를 사용하여 CoNLL 출력 포맷으로 변경
- 출력 예

AP	SL	B	216	Direct	SL	B-OG
통신	NNG	I	35	무선	NNG	I-OG
은	JX	I	269	의	JKG	-
NUM	SN	B	254	년	NNB	B-DT
일	NNB	I	238	월	NNB	I-DT
(SS	I	202)	SS	-
이하	NNG	I	48	미만	NNG	-
한국	NNP	B	143	일본	NNP	B-LC
시간	NNG	I	254	시간	NNB	-
)	SS	I	202	(SS	-
올라주원	NNP	B	1000	UNK	NNP	B-PS
,	SP	I	202	그리고	MAJ	-
유잉	NNP	B	22	Howard	SL	B-PS
을	JKO	I	142	려는	ETM	-
비롯	XR	B	297	위시	NNG	-
하	XSV	I	181	았	EP	-
아	EC	I	142	아서	EC	-
애드리언	NNP	B	1000	UNK	NNP	B-PS
댄틀리	NNP	B	1000	UNK	NNP	I-PS
,	SP	I	202	그리고	MAJ	-
팻	NNP	B	19	에드	NNP	B-PS
라일리	NNP	B	230	Tom	SL	I-PS
감독	NNG	B	54	수석코치	NNG	-
,	SP	I	202	그리고	MAJ	-
캐시	NNG	B	90	APIs	SL	B-PS
러시	NNG	B	86	아워	NNG	I-PS

그림 3. 학습 데이터로 사용할 CoNLL 형식 데이터

(3) CoNLL2CRFsuite.run

- 생성된 CoNLL 학습 데이터를 미리 작성한 자질 값 설정 파일 ('FeatureTemplate.txt')을 사용하여 CRFsuite의 학습 데이터 생성
- 자질 값 설정 파일 및 출력 예

CRFsuiteFeatureTemplate m	
<pre> #word - 0 w [-2,0] w [-1,0] w [0,0] w [1,0] w [2,0] w [-1,0]/[0,0] w [0,0]/[1,0] #tag - 1 p [-2,1] p [-1,1] p [0,1] p [1,1] p [2,1] p [-1,1]/[0,1] p [0,1]/[1,1] #lextag - 0,1 wp [-2,0]/[-2,1] wp [-1,0]/[-1,1] wp [0,0]/[0,1] wp [1,0]/[1,1] wp [2,0]/[2,1] </pre>	<pre> B-OG w[0]=AP w[1]=통신 w[2]=은 w[0 1]=AP 통신 p[0]=SL p[1]= I-OG w[-1]=AP w[0]=통신 w[1]=은 w[2]=NUM w[-1]= O w[-2]=AP w[-1]=통신 w[0]=은 w[1]=NUM w[2]= B-DT w[-2]=통신 w[-1]=은 w[0]=NUM w[1]=일 w[2]=(w[-1]= I-DT w[-2]=은 w[-1]=NUM w[0]=일 w[1]=(w[2]=이하 w[-1]= O w[-2]=NUM w[-1]=일 w[0]=(w[1]=이하 w[2]=한국 O w[-2]=일 w[-1]=(w[0]=이하 w[1]=한국 w[2]=시간 B-LC w[-2]=(w[-1]=이하 w[0]=한국 w[1]=시간 w[2]= O w[-2]=이하 w[-1]=한국 w[0]=시간 w[1]=) w[2]= O w[-2]=한국 w[-1]=시간 w[0]=) w[1]=올라주원 w[2]= B-PS w[-2]=시간 w[-1]=) w[0]=올라주원 w[1]=, w[2]=유영 O w[-2]=) w[-1]=올라주원 w[0]=, w[1]=유영 w[2]=을 w[-1]= B-PS w[-2]=올라주원 w[-1]=, w[0]=유영 w[1]=을 w[2]=비롯 O w[-2]=, w[-1]=유영 w[0]=을 w[1]=비롯 w[2]=하 w[-1]= O w[-2]=유영 w[-1]=을 w[0]=비롯 w[1]=하 w[2]=아 w[-1]= O w[-2]=을 w[-1]=비롯 w[0]=하 w[1]=아 w[2]=애드리언 w[-1]= O w[-2]=비롯 w[-1]=하 w[0]=아 w[1]=애드리언 w[2]=댄틀리 B-PS w[-2]=하 w[-1]=아 w[0]=애드리언 w[1]=댄틀리 w[2]=, w[-1]= I-PS w[-2]=아 w[-1]=애드리언 w[0]=댄틀리 w[1]=, w[2]=맷 w[-1]= O w[-2]=애드리언 w[-1]=댄틀리 w[0]=, w[1]=맷 w[2]=라일리 B-PS w[-2]=댄틀리 w[-1]=, w[0]=맷 w[1]=라일리 w[2]=감독 I-PS w[-2]=, w[-1]=맷 w[0]=라일리 w[1]=감독 w[2]=, w[-1]= O w[-2]=맷 w[-1]=라일리 w[0]=감독 w[1]=, w[2]=캐시 O w[-2]=라일리 w[-1]=감독 w[0]=, w[1]=캐시 w[2]= B-PS w[-2]=감독 w[-1]=, w[0]=캐시 w[1]=러시 w[2]= I-PS w[-2]=, w[-1]=캐시 w[0]=러시 w[1]=감독 w[2]= O w[-2]=캐시 w[-1]=러시 w[0]=감독 w[1]=, w[2]= O w[-2]=러시 w[-1]=감독 w[0]=, w[1]=TV w[2]=해설가 </pre>

그림 5. CRFsuite 학습 자료

그림 4. 자질값 설정 파일

(4) crfsuite learn

- 오픈 소스인 'crfsuite'를 사용하여 CRFsuite 모델 생성
- 생성된 모델 파일은 binary 파일이다.
- crfsuite 수행 과정

```

CRFSuite 0.12 Copyright (c) 2007-2011 Naoki Okazaki

Start time of the training: 2016-09-22T08:02:29Z

Reading the data set(s)
[1] train.300.crfsuite.txt
0....1....2....3....4....5....6....7....8....9....10
Number of instances: 3555
Seconds required: 1.710

Statistics the data set(s)
Number of data sets (groups): 1
Number of instances: 3554
Number of items: 121002
Number of attributes: 249874
Number of labels: 11

Feature generation
type: CRF1d
feature.minfreq: 0.000000
feature.possible_states: 0
feature.possible_transitions: 0
0....1....2....3....4....5....6....7....8....9....10
Number of features: 294933
Seconds required: 0.630

L-BFGS optimization
c1: 0.000000
c2: 1.000000
num_memories: 6
max_iterations: 2147483647
epsilon: 0.000010
stop: 10
delta: 0.000010
linesearch: MoreThuente
linesearch.max_iterations: 20

***** Iteration #1 *****
Loss: 136297.872587
Feature norm: 1.000000
Error norm: 120433.159115

```

그림 6. CRFsuite 학습 수행 과정

4.2.2 실행 프로그램 수행(Test.sh)

- cqasys_WordAngler.run
- 테스트 대상 자료인 'dev.json'과 학습 프로그램의 결과들을 읽은 후 최종 결과 출력
- 결과 출력 폼은 제공된 입력 자료인 'dev.json'과 동일하며, 개체명 결과인 'NE'의 내용만 모듈의 출력 결과로 바뀌어 출력된다.
- 최종 출력 폼('result.json')

```
{
  "category": "",
  "category_weight": 0.000000,
  "sentence": [
    {
      "id": 0,
      "resever_str": "",
      "text": "서호프와 파사노에게 연속 안타를 내주며",
      "morp": [
        { "id": 0, "lemma": "서호프", "type": "NNP", "position": 0},
        { "id": 1, "lemma": "와", "type": "JC", "position": 9},
        { "id": 2, "lemma": "파사노", "type": "NNP", "position": 13},
        { "id": 3, "lemma": "에게", "type": "JC", "position": 22},
        { "id": 4, "lemma": "연속", "type": "NNG", "position": 29},
        { "id": 5, "lemma": "안타", "type": "NNG", "position": 36},
        { "id": 6, "lemma": "를", "type": "JKO", "position": 42},
        { "id": 7, "lemma": "내주", "type": "VV", "position": 46},
        { "id": 8, "lemma": "며", "type": "EC", "position": 52}
      ],
      "word": [
        { "id": 0, "text": "서호프와", "type": "", "begin": 0, "end": 1},
        { "id": 1, "text": "파사노에게", "type": "", "begin": 2, "end": 3},
        { "id": 2, "text": "연속", "type": "", "begin": 4, "end": 4},
        { "id": 3, "text": "안타를", "type": "", "begin": 5, "end": 6},
        { "id": 4, "text": "내주며", "type": "", "begin": 7, "end": 8}
      ],
      "NE": [
        { "id": 0, "text": "서호프", "type": "PS", "begin": 0, "end": 0},
        { "id": 1, "text": "파사노", "type": "PS", "begin": 2, "end": 2}
      ]
    }
  ],
  ...
}
```

그림 7. WordAngler 최종 출력 결과 예

국 립 국 어 원

코너(KoNER)

신유현, 이상구

【서울대학교】

차 례

제 1 장 소프트웨어 소개	78
1.1 소프트웨어 명칭	78
1.2 소프트웨어 사용 환경	78
1.3 소프트웨어 특징	79
제 2 장 소프트웨어 설치 및 실행	81
2.1 소프트웨어 설치 방법	81
2.2 소프트웨어 파일 구조	81
2.2.1 주요 파일 설명	81
2.2.2 전체 구조	81
2.3 소프트웨어 실행 방법	81
제 3 장 소프트웨어 기능	83
3.1 프로그램 기능	83
3.2 프로그램 기능 제약	85
제 4 장 기타	86

제 1 장 소프트웨어 소개

1.1 소프트웨어 명칭

1.1.1 명칭 의미 및 설명

코너(KoNER)는 Korean Named Entity Recognition의 약자이다. 영어 단어 코너(Corner)와의 발음상의 유사성을 고려한 명칭이다. 영어 단어 코너(Corner)는 1) 특별한 목적을 위해 구성하기 위한 한정된 일부 장소 및 2) '모퉁이, 구석'이라는 뜻을 지닌다.

코너(KoNER)는 한국어 개체명 인식이라는 특별한 목적을 위해 구현된 프로그램이라는 뜻이 있다. 또한, "모퉁이(코너)를 돌면 새로운 행운이 기다리고 있을 거야"라는 구절이 있듯이, 코너(KoNER)를 통해서, 한국어 개체명 인식의 성능 향상에 도움이 되고자 하는 뜻을 담고 있다.

1.1.2 이유 및 개발 동기

개체명 인식은 질의 응답 시스템, 정보 검색 시스템, 기계 번역 시스템 등 다양한 분야에서 사용되는 기술이다. 하지만 한국어 개체명 인식의 경우, 다른 연구가 활발한 언어와 비교하였을 때 공개된 말뭉치가 부족하다는 문제점이 있다. 그리하여 개체명 인식 문제 해결에 어떻게 접근해야 할까 고민하고 있는 와중에 경진 대회 소식을 접하게 되었다. 이러한 말뭉치를 기반으로 한국어 개체명 인식이라는 분야에 조금의 도움이 되고 싶어 개발하게 되었다. 한국어 개체명 인식이라는 특정 도메인에서의 높은 정확도는 것이 목적이다.

1.2 소프트웨어 사용 환경

- 지원 OS : Linux (Ubuntu 14.04.4 LTS)
- 권장 메모리 : RAM 32GB 이상
- 사용 언어 : python
- 실행 환경 : python 2.7.x
- 필요 라이브러리 : Numpy, Theano
- 텍스트 인코딩 : utf-8

1.3 소프트웨어 특징

1.3.1 특징

① 딥 러닝 기법을 이용하여 사전 지식의 최소화

딥 러닝의 경우, 워드 임베딩을 입력 값으로 사용하여 문제를 해결한다. 이때 워드 임베딩을 만드는 과정에서 원본 텍스트(raw text)만을 이용하므로, 사전 지식의 영향력이 줄어들 수 있다. 또한 딥 러닝 구조 중 하나인 RNN (Recurrent Neural Network)를 이용하여, 사전 지식 없이도 높은 정확도를 낼 수 있다.

② 사람이 글을 읽는 방법 모사

딥 러닝 모델 중 하나인 Bidirectional LSTM을 이용하여, 전향(forward) 시에는 왼쪽에서 오른쪽으로 글을 읽는 사람의 모습을 모사할 수 있다. 후향(backward) 시에는 사람이 왼쪽에서 오른쪽으로 글을 읽으면서, 다음에 읽은 단어(오른쪽 단어)가 이전에 읽은 단어(왼쪽 단어)의 판단에 영향을 미치는 것을 고려할 수 있다.

따라서, 결국에는 왼쪽에서 오른쪽으로 글을 읽으며 이전 정보들을 저장하고 있고 새롭게 접하는 단어를 이용하여 전체적인 맥락을 파악하는 것을 양방향 LSTM(Bi-directional LSTM)을 이용하여 모사하였다.

③ 단어의 형태적(morphological) 특징을 고려

입력 값으로 형태소 및 형태소의 자음, 모음을 이용(ex. ㅎ, ㄷ, ㅁ, ㅌ, ㅈ, ㅊ)한다. 따라서 새로운 단어가 등장하더라도 형태적 유사함을 토대로 기존 단어와의 연관성을 지을 수 있다. 또한, 복수형 여부, 접두사, 접미사 등을 이용하여 단어를 의미적, 문법적으로 이해할 수 있게 된다.

④ 문법적 특징을 고려

알고리즘의 입력 값에 품사(Part-of-speech)를 이용하여, Bi-directional LSTM을 학습한다. Bi-directional LSTM이 각 품사 간의 의존(dependency) 관계를 학습할 수 있게 되므로, 이를 통해 단어의 문법적 역할을 동시에 학습할 수 있다. 예를 들어, 명사와 명사가 연속되서 나올 경우에는 개체명일 확률이 높다는 것을 학습하게 된다.

⑤ 기구축사전의 활용 방안 극대화

기구축사전의 값을 BIOES 인코딩 방법을 이용하여 입력 값으로 사용한다. 예를 들어, "LA"라는 단어는 독립적으로 존재할 경우에는 지명(LC)를 의미하지만, "LA 레이커스"의 경우에는 기관명(OG)을 의미하게 된다. 이렇게 사전에 존재하는 LA에 대한 정보를 "S-LC, B-OG"로 나타내어 문장의 문맥에 따라서 다른 의미를 지니는 단어임을 학습하도록 한다.

cf) BIOES 인코딩 : B(Begin), I(Inside), O(outside), E(End), S(Single)의 5가지 태그를 이용하여 인코딩 하는 방법으로, BIO 인코딩의 확장판이다.

1.3.2 장점

- ① RNN의 한 종류인 Bidirectional LSTM을 사용함으로써, 텍스트 및 특징(feature) 간의 의존(dependency) 정보를 충분히 반영할 수 있다. 각 의존 관계 학습 시에 형태소, 자음/모음, 품사, 기구축 정보(lexicon)이 동시에 고려된다.
- ② 형태적 특징을 이용함으로써, 새로운 단어가 들어와도 이전 단어와의 연관성을 극대화 시킬 수 있다. 또한 "어제"와 "어저께"처럼 유사한 단어가 들어오는 경우에도 두 단어의 연관성을 고려할 수 있다.
- ③ 사람이 개체명을 인식하는 방법을 학습시킬 수 있다. 왼쪽에서 오른쪽으로 읽어나가며 이해하는 것을 모사한다. 이때 형태소 단위의 입력 값을 받음으로써 최소한의 의미로부터 뜻을 파악한다. 해당 형태소의 다른 형태소와의 형태적 유사성을 이용하여 해당 단어가 무엇인지 유추할 수 있다.
- ④ 특징 추출(feature engineering)을 최소화할 수 있다. 경진대회에서 제공한 것 외에 다른 사전 지식(prior knowledge) 없이 학습하였다.

1.3.3 기대 효과

기구축 사전(lexicon)이 완벽하지 않더라도 미등록어에 대한 대응이 가능하고, 기존 도메인에 대한 선수 지식을 이용한 특징 추출(feature engineering) 과정을 줄일 수 있다. 이는 신조어 및 다양한 문법적 지식을 요구하는 한국어에 특히나 적합한 방법일 것으로 기대된다.

선수 지식(prior knowledge)이 부족한 경우에도 구축된 말뭉치와 형태소 임베딩만을 이용하여 한국어 개체명 인식의 정확도 향상을 할 수 있을 것으로 기대된다. 사람의 개입 및 노력을 최소화하였음에도 높은 성능을 보일 것으로 기대된다.

제 2 장 소프트웨어 설치 및 실행

2.1 소프트웨어 설치 방법

1. KoNER.tar 파일 압축 해제


```
>> tar -xvf KoNER.tar
>> cd KoNER
```
2. 라이브러리 설치


```
>> pip install numpy
>> pip install theano
```
3. chmod 777 cqasys_KoNER.sh

2.2 소프트웨어 파일 구조

2.2.1 주요 파일 설명

① 실행 파일

tagger.py --- 실행 파일

② 데이터 파일

data/[input_name] --- 입력 파일

data/[output_name] --- 출력 파일

③ 사전 파일

data/gazette --- 기구축 사전(lexicon) 파일

data/wikiCorpus_word2vector.hr --- 워드 임베딩 파일

2.2.2 전체 구조

KoNER/

data/ 입력 파일, 출력 파일, 워드임베딩, 사전(lexicon) 폴더

model/ 학습된 모델 폴더

cqasys_KoNER.sh 실행 스크립트

tagger.py 실행 파일

json2base_type.py json을 세로 형식으로 바꿔주는 파일
 json2base.py 경진대회 제공 json을 세로 형식으로 바꿔주는 파일
 base2json.py 세로 형식의 파일을 json으로 바꿔주는 파일
 *.pyc tagger.py에서 사용하는 파일들

2.3 소프트웨어 실행

소프트웨어 실행을 위해서는 아래와 같은 터미널에 아래와 같은 명령어를 입력한다.

```
>> ./cqasys_KoNER.sh [input_file] [output_file]
```

```
ex) ./cqasys_KoNER.sh "./data/dev.json" "./data/result.json"
```

해당 프로그램은 총 3가지 입력 값을 받는다.

- input_file : 입력 데이터 파일이 존재하는 경로 및 이름
ex) "./data/dev.json"
- output_file : 원하는 출력 데이터 파일의 경로 및 이름
ex) "./data/result.json"

[cqasys_KoNER.sh 내부 실행 명령어]

```
#!/bin/bash
```

```
if [ $# -eq 0 ]; then
```

```
    echo "./cqasys_KoNER [input_file] [output_file]"
```

```
    exit 1;
```

```
fi
```

```
python ./json2base_type.py $1 > $1.txt
```

```
python ./tagger.py --input $1.txt --output $2.txt
```

```
python ./base2json.py --input $1 --output $2 --tags $2.txt
```

제 3 장 소프트웨어 기능

3.1 프로그램 기능

제출한 프로그램의 기능:

한국어 개체명 인식

해당 프로그램에서의 개체명은 총 5가지로 기관명(OG), 인명(PS), 지명(LC), 숫자(DT), 시간(TI)을 의미한다. 해당 프로그램은 json 형식의 입력 파일로부터, 형태소 및 품사 태깅을 추출한다. 형태소와 품사를 이용하여 해당 형태소의 개체명 여부를 판별하는 프로그램이다.

임의의 문장에 대한 형태소와 품사를 입력 값으로 받아 하나의 의미 단위를 이루는 형태소들의 시퀀스를 찾아 해당 시퀀스의 개체명 카테고리를 분류해주는 프로그램이다.

주요 방법론 :

해당 프로그램은 입력 값으로 json 형식의 파일을 받는다. json 형식의 파일은 아래 그림 1과 같이 변환되어 입력 값으로 사용된다.

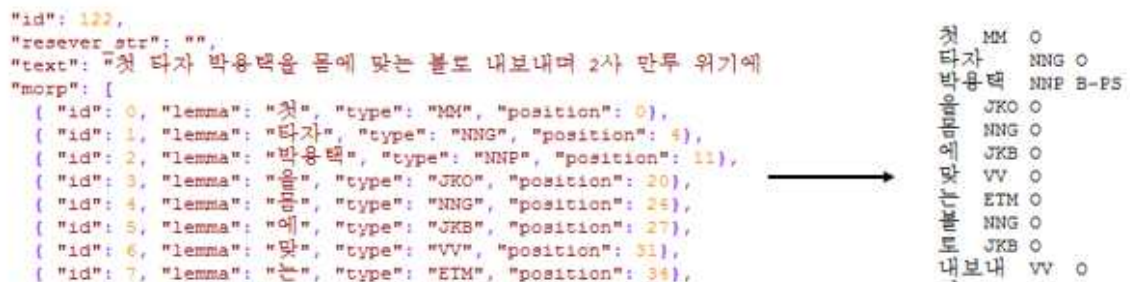


그림 35 json 형식의 입력 파일(왼쪽)의 형식 변환

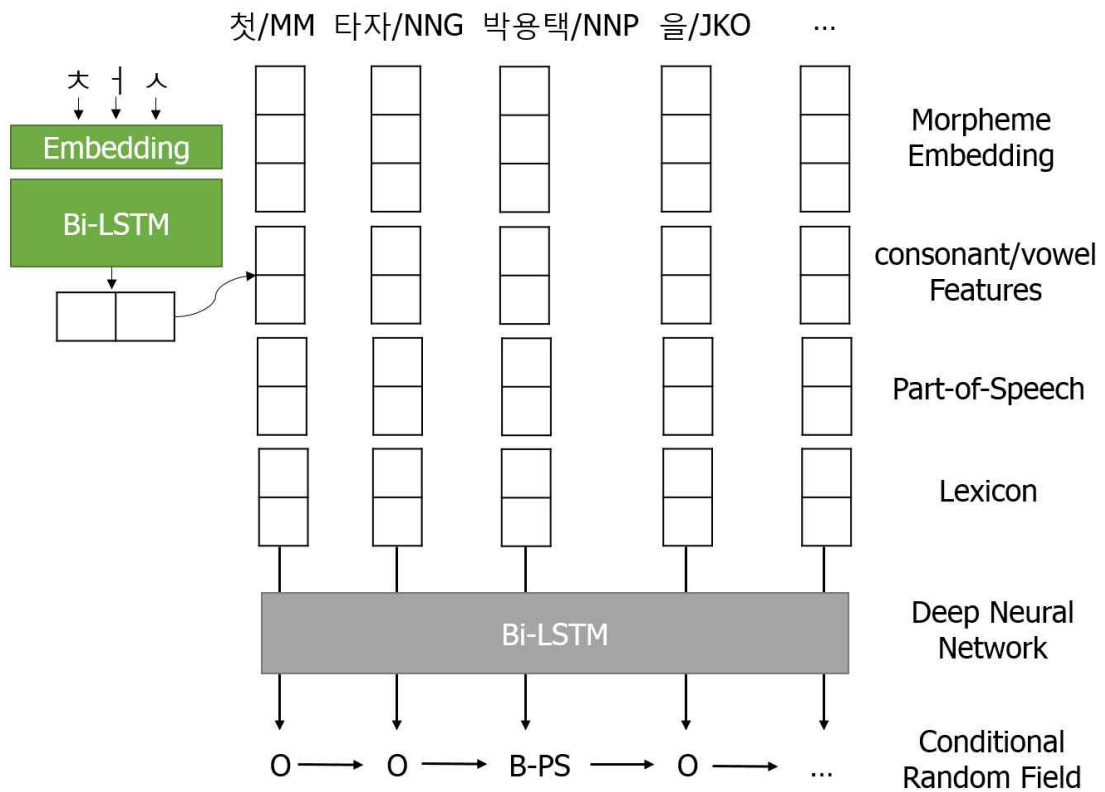


그림 36 개체명 인식 알고리즘의 전체 구조

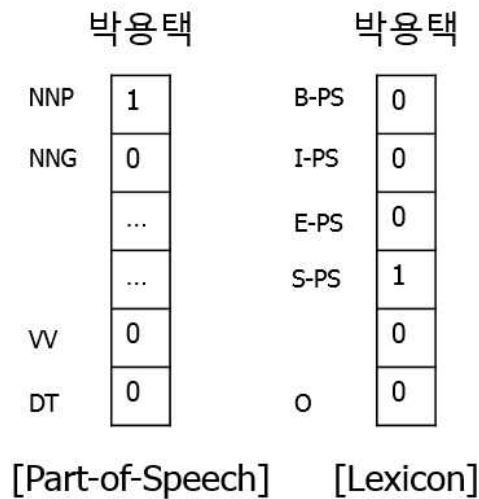


그림 37 품사 및 사전의 벡터 표현 예시

먼저, 알고리즘의 입력 값과 출력 값을 살펴보겠다. 그림 2에서는 변환된 입력 파일의 첫 번째 열인 형태소와 두 번째 열인 품사를 입력 값으로 사용하고, 세 번째 열인 개체명 태그를 출력 값으로 이용하여 학습한다. 즉, 입력 값은 "첫/MM, 타자/NNG, 박용택/NNP, 을/JKO, ..."이고, 출력 값은 "O, O, B-PS, O, ..."이다. (이때, 출력 값에 사용되는 인코딩(encoding) 기법에는

BIOES 방법을 사용하여 학습하였다.)

그림 2와 같이 알고리즘에서는 해당 입력 값(ex. 첫/MM, 타자/NNG)에 대해 4가지 관점으로 벡터 표현을 생성한다. 첫 번째는 형태소 임베딩을 이용하여 입력 값을 표현하는 것이다. 이는 pre-train된 워드 임베딩 값을 lookup table에서 가져온다. 두 번째는 자음/모음 특징을 추출하여 입력 값을 표현하는 것이다. 이러한 자음/모음 특징 추출에는 Bidirectional LSTM이 사용된다. 즉, "첫/MM"의 경우에는, "ㄷ, ㅌ, ㅍ"을 입력 값으로 하는 Bidirectional LSTM layer를 추가하였다. 세 번째는 품사를 이용하여 입력 값을 표현하는 것이다. 그림 3의 왼쪽과 같이 품사 전체 개수를 길이로 하는 원 핫(one-hot) 벡터를 이용하였다. 네 번째는 사전(lexicon)을 이용하여 입력 값을 표현하는 것이다. (이때 사전은 제공된 gazette를 이용하였다.) 그림 3의 오른쪽과 같이 사전 내에서 해당 단어가 어떻게 쓰였는지에 따라 정해진 레이블에 0 또는 1의 값을 갖는다. 예를 들어, "박용택 PS", "LA 레이커스 OG", "LA LC" 세 가지 경우가 있다고 하자. 이 경우에 "박용택"은 "박용택 PS"에서만 쓰였으므로 "S-PS"에 대응된다. "LA"의 경우에는 "B-OG, S-LC"에 대응되며, "레이커스"는 "E-OG"에 대응된다.

3.2 프로그램 기능 제약

입출력 양식 제한 :

프로그램의 입출력 양식은 해당 경진 대회 제공 json 파일 양식을 따른다.

json2base_type.py 파일을 이용하여, json을 세로 형식으로 바꿔 알고리즘의 입력 값으로 사용하였다. 해당 파일은 [input_file].txt (ex. input_file이 dev.json인 경우 dev.json.txt)으로 저장된다.

알고리즘의 출력 파일 또한 세로 형식으로 출력된다. 해당 파일은 [output_file].txt (ex. output파일이 result.json인 경우, result.json.txt)으로 저장된다. 따라서 base2json.py를 이용하여 세로 형식을 json 형식으로 변환한다. 해당 base2json.py는 train.json, dev.json을 입력 값으로 받은 경우에는 result.json 형식을 오류 없이 만든다.

제공된 두 파일(train.json, dev.json)에 대해서는 잘 작동하나, 새로운 파일에 대해서 세로형식의 파일을 json으로 변환하는 과정에서 오류가 생길 수 있다.

입력 파일 (json 형식) : dev.json

출력 파일 (json 형식) : result.json

알고리즘 입력 파일 (세로 형식) : dev.json.txt

알고리즘 출력 파일 (세로 형식) : result.json.txt

제 4 장 기타

train.json으로 학습한 모델을 이용하여 구한 dev.json에 대한 F-Score 결과이다. 학습은 총 50번의 에폭(epoch)을 통해 이루어졌다.

501개의 문장으로 이루어진 dev.json의 경우에는 개체명 태깅에 평균적으로 90초 내외가 소요되었다.

성능 측정에는 conll 평가 스크립트를 이용하였다. 즉, "B-PS", "[B-PS, I]" 등의 개체 청크(chunk) 단위로 성능을 측정하였다.

	특징 (Feature)	F-Score
1	형태소 임베딩	79.69
2	형태소 임베딩+자음/모음 특징	80.66
3	형태소 임베딩+자음/모음 특징+품사	83.09
4	형태소 임베딩+자음/모음 특징+품사+사전	85.71

표 45 특징에 따른 F-Score 값

표 1에서 볼 수 있듯이 가장 성능이 좋은 모델은 F-Score 85.71로 4가지 벡터 표현 방법을 모두 사용한 모델이었다. 표 2는 가장 좋은 성능을 보인 4번째 모델에 대한 각 카테고리 별 성능을 나타내고 있다. 5가지의 개체명 중 사람(PS)에 해당하는 F-Score가 0.899로 제일 높고, 기관(OG)에 해당하는 F-Score가 0.797로 제일 낮다.

	정밀도 (Precision)	재현율 (Recall)	F-Score
DT	0.894	0.880	0.887
LC	0.793	0.853	0.822
OG	0.824	0.772	0.797
PS	0.915	0.885	0.899
TI	0.872	0.810	0.840

표 46 4번째 모델의 각 개체명 카테고리 별 정밀도, 재현율, F-Score

실행 커맨드 모음:

```
>> tar -xvf KoNER.tar
>> cd KoNER
>> pip install numpy
>> pip install theano
>> chmod 777 cqasys_KoNER.sh
>> ./cqasys_KoNER.sh "./data/dev.json" "./data/result.json"
```

결과 파일 경로 : ./data/result.json

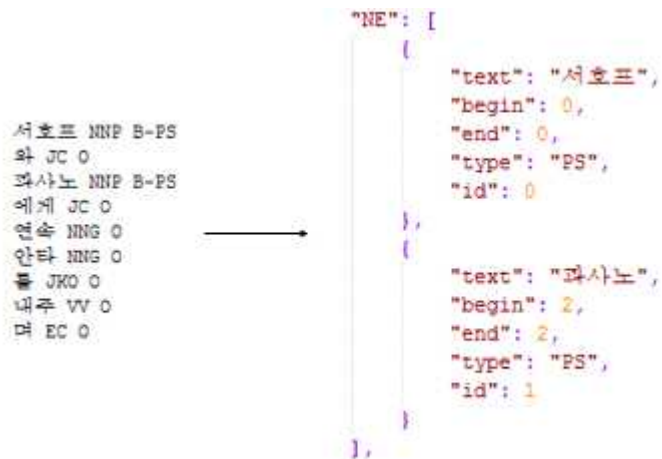


그림 38 세로 형식의 알고리즘 결과 파일(왼쪽),
json 형식의 결과 파일 (오른쪽)

국 립 국 어 원

단추(DANCHU)

안재현, 윤정민, 김혜민, 박용신, 배경만, 고영중
【동아대학교】

차 례

제 1 장 소프트웨어 소개	92
1.1 단추(DANCHU)	92
1.2 소프트웨어 사용 환경	93
1.3 소프트웨어 특징	93
1.3.1 언어분석을 위한 학습모델	93
1.3.2 음절 임베딩 벡터	95
1.3.3 웹 페이지와 디자인	97
제 2 장 소프트웨어 설치 및 실행	98
2.1 소프트웨어 설치 방법	98
2.2 소프트웨어 실행 방법	98
2.3 단추 언어분석 통합시스템	99
제 3 장 소프트웨어 기능	102
3.1 프로그램 기능	102
3.1.1 단추 형태소 분석기	102
3.1.2 단추 개체명 인식기	103
3.1.3 단추 의존 구문분석기	104
3.1.4 단추 의미역 결정기	105
3.2 프로그램 기능 제약	106
제 4 장 기타	106
4.1 시스템 성능	106
4.2 참고 문헌	108

제 1 장 소프트웨어 소개

1.1 단추(DANCHU)

자연어 처리는 인간이 발화하는 언어현상을 기계적으로 분석하여 기계가 이해할 수 있는 형태로 만드는 것을 의미한다. 자연어 처리에는 언어분석과 대화모델 등 많은 연구 분야가 있으며, 언어 분석은 자연어에 대한 형태소 분석(Part Of Speech), 개체명 인식(Named Entity Recognition), 의존 구문분석(Dependency Parser), 의미역 결정(Semantic Role Labeling) 등 자연어 처리의 기초가 되는 연구를 말한다. 본 대회에 제안하는 단추(DANCHU : Dong-A Natural language analysis tools for Communication with HUmAn)는 동아대학교 지능형 시스템 연구실에서 구현한 언어 분석 통합시스템이다. 단추(DANCHU)라는 이름은 우리나라 옛말의 “첫 단추를 잘 꿰어야 한다.”라는 말에서 착안한 이름이다. 첫 단추가 잘 꿰어져야 다음 단추가 잘 꿰어지듯 자연어 응용 분야에서 언어분석은 첫 단추와 같고, 정확한 언어분석은 많은 자연어 응용분야에서 좋은 토대가 된다.

언어분석의 기존 연구에서는 규칙기반과 통계기반, 사전기반을 많이 사용하여 연구를 했다. 그러나 규칙과 사전을 생성하기 위해서는 사람이 수작업으로 생성하기 때문에 많은 노력과 비용이 필요하며, 언어가 점차 확장됨에 따라 사전과 규칙이 많아지는 문제점이 있다. 최근에는 기계학습(Machine Learning)과 심층학습/딥러닝(Deep Learning)을 이용하여 이와 같은 문제점을 개선하였다. 본 시스템은 순차 레이블링(Sequential Labeling)에 높은 성능을 보이고 있는 Bidirectional Long Short Term Memory Conditional Random Fields(bi-LSTM-CRFs)과 단어 표상(word embedding), 음절 기반 임베딩 벡터를 이용하여 높은 성능을 기록하였다.

본 시스템은 웹페이지로 제작되었으며, 문장을 입력하면 언어분석을 진행하며 형태소분석기, 개체명인식기, 의존 구문분석기, 의미역 결정 모델을 순차적으로 실행하여 4가지의 결과를 사용자에게 보여준다.

1.2 소프트웨어 사용 환경

- 운영체제 : Ubuntu 15.10 x64
- 사용 언어 : Python, C++, Java, JSP, Java Script, HTML
- 라이브러리 : Tensorflow¹, Numpy²

1.3 소프트웨어 특징

1.3.1 언어분석을 위한 학습모델

단추(DANCHU) 통합시스템은 언어분석 통합시스템으로써 입력된 문장의 형태소 분석, 개체명 인식, 의존 구문구조 분석, 의미역 결정 등을 실행한다. 각 언어분석 시스템은 모두 시퀀스 레이블링의 문제로 기존의 CRFs, Structure-SVM 보다 성능 면에서 개선된 bi-LSTM-CRFs를 사용하여 학습하였다. bi-LSTM-CRFs는 동아대학교 지능형시스템 연구실에서 자체 개발된 소스로 Tensor Flow로 구현하였다. 아래 (1) ~ (3)은 RNN, LSTM, bi-LSTM-CRFs에 대한 설명이다.

(1) Recurrent Neural Networks(RNN)

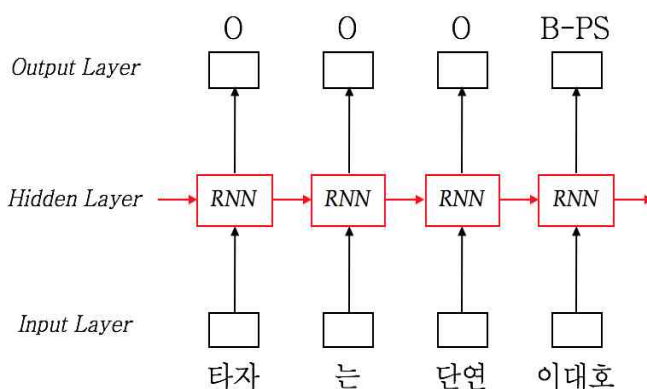


그림 40. Recurrent Neural Networks 모델

그림 1에서 보이는 것처럼 RNN은 단어 열(x_1, x_2, \dots, x_t)을 입력 계층에서 입력으로 받고, 그 입력은 은닉 계층(h_1, h_2, \dots, h_t)을 거쳐 입력을 잘 표현하는 벡터(y_1, y_2, \dots, y_t)가 출력 계층으로 출력된다. RNN 구조를 아래와 같은 식으로 정의할 수 있고, 아래 식에서 U, V, W 는 각 계층의 weight행렬 이다.

1 <https://www.tensorflow.org/>

2 <http://www.numpy.org/>

$$\begin{aligned} h_t &= \tanh(Ux_t + Vh_{t-1}) \\ y_t &= \text{softmax}(Wh_t) \end{aligned}$$

RNN은 위의 수식과 그림 1에서 알 수 있듯이 이전 상태를 다음 상태로 계속 전달하는 모델이다. 따라서 이론상으로는 이전 상태를 기억하여 장기 의존성(long-range dependencies)을 다룰 수 있다. 하지만 실제로 위치상 멀리 있는 정보를 많이 잃어버리는 그래디언트 소멸 문제(Vanishing gradient problem)가 존재하기 때문에 장기 의존성을 유지할 수 없는 문제점이 있다.

(2) Long Short-term Memory(LSTM)

LSTM은 RNN의 그래디언트 소멸 문제를 해결하여 장기 의존성을 잘 학습할 수 있는 특별한 모델이다. 이 문제를 해결하기 위하여 LSTM에서는 RNN의 은닉 계층 노드에 3개의 gate(input, output, forget)와 1개의 memory cell을 이용한다. LSTM의 memory cell은 전체적인 상태를 기억하여 다음 상태로 전달하는 역할을 한다. forget gate를 이용하여 cell의 상태에서 어떤 정보를 제거할지 결정하고, input gate로 cell에서 어떤 정보를 갱신할지 결정한다. 마지막으로 output gate를 이용하여 cell의 어떤 정보를 전달할지 결정하여 장기 의존성을 유지한다. LSTM 구조의 전체적인 수식은 다음과 같다.

$$\begin{aligned} i_t &= \text{sigmoid}(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \\ c_t &= (1 - i_t) \odot c_{t-1} + i_t \odot \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\ o_t &= \text{sigmoid}(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \quad h_t = o_t \odot \tanh(c_t) \end{aligned}$$

위 식에서 i, c, o, h 는 각각 input gate, memory cell, output gate, hidden state이다. 그리고 \odot 는 element-wise product이며, W 는 weight 행렬을 b 는 bias를 나타낸다.

(3) bidirectional LSTM CRFs(bi-LSTM-CRFs)

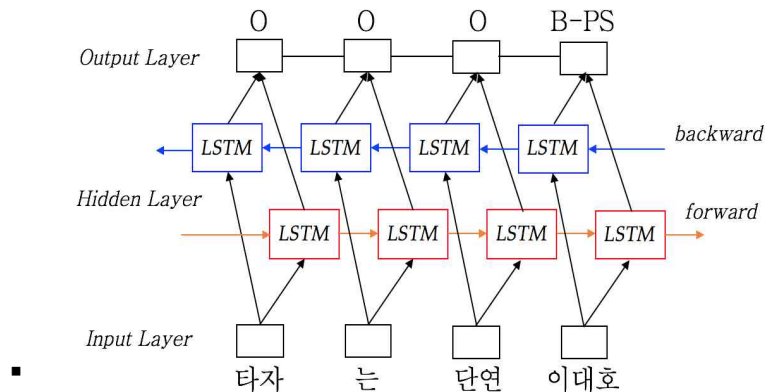


그림 41. bidirectional LSTM CRFs 모델

bi-LSTM-CRFs는 그림 2에서 보이는 것과 같이 LSTM에 입력 문자열을 양방향으로 받아서 각 단어 별로 은닉 계층의 결과를 얻고, 그 결과 간의 의존성을 추가한 모델이다.

1.3.2 음절 임베딩 벡터

bi-LSTM-CRFs를 사용하여 학습하기 위해선 어절 혹은 음절, 형태소 단위의 단어 표상(word embedding)이 필요하다. 단어표상이란 말뭉치의 단어를 벡터로 표현한 것으로 벡터의 값은 말뭉치에서 단어가 가지는 의미나 역할을 잘 표현해주는 값 이어야 한다. 이와 같이 단어의 의미와 맥락을 고려하여 단어를 벡터로 표현한 것을 단어 표상이라고 한다.

본 시스템에서 사용한 기본적인 단어 표상은 아래 표 1과 같다.

표 49 사용한 기본적인 단어 표상

모듈	사용한 임베딩
형태소 분석기	음절 임베딩 벡터
개체명 인식기	형태소 임베딩 벡터
의존 구문분석기	어절 임베딩 벡터
의미역 결정	어절 임베딩 벡터

본 시스템에서는 표 1과 같이 기본적인 단어표상을 사용하였고 개체명 인식과 의존 구문분석, 의미역 결정 모듈에서는 음절 기반 임베딩 벡터를 추가로 사용하였다.

음절 기반 단어 임베딩이란 각 단어를 표현하기 위해 음절 단위의 임베딩 벡터를 기반으로 하여 단어 단위의 임베딩 벡터로 확장한 벡터를 말한다. 단어는 음절의 시퀀스이기 때문에 음절 단위의 임베딩 벡터의 확장은

단어를 표현하기에 적합하다. 그림 3에서 보이는 것과 같이 bidirectional LSTM을 이용하고, forward의 마지막 상태와 backward의 마지막 상태를 결합하여 단어 단위 임베딩 벡터로 사용하였다.

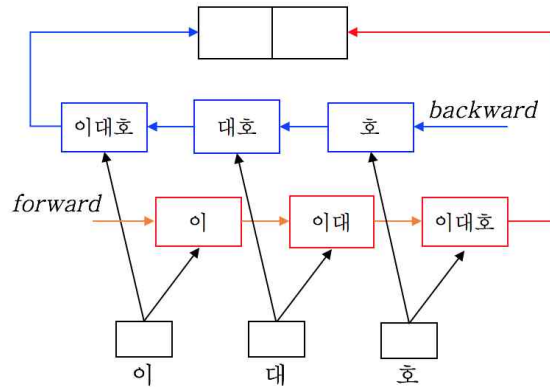


그림 42. 음절 기반 단어 임베딩 벡터를 위한 bi-LSTM

예를 들어 개체명 인식에 음절 기반 단어 임베딩 벡터를 적용하기 위해서 입력되는 음절 임베딩 벡터로는 음절별로 각 개체명 태그별 분포를 벡터로 만들어서 사용하였다. 그 결과로 만들어진 벡터는 11차원을 가진다(개체명 태그 5개 * B정보 2개 + O 태그 1개 = 11). 최종적으로 모델에 입력 될 때는 분포 벡터에 softmax를 이용하여 확률로 변환 후 사용한다. 예를 들어 "타자/O 는/O 단연/O 이대호/B-PS"문장에서 각 음절에 대응되는 음절 임베딩 벡터는 표 2와 같다. 표 2의 값들은 지면상 반올림 하여 표기하였다.

표 50 . 음절 단위 개체명 분포 벡터

	B					I					O
	PS	OG	LC	DT	TI	PS	OG	LC	DT	TI	
타	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.09	0.09	0.0
	9	9	9	9	9	9	9	9			9
자	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.09	0.09	0.0
	9	9	9	9	9	9	0	9			9
는	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.10	0.09	0.1
	9	9	9	9	9	9	9	9			0
단	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.09	0.09	0.0
	9	9	9	9	9	9	9	0			9
연	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.09	0.09	0.0
	0	9	9	9	9	9	9	9			9
이	0.1	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.08	0.08	0.0
	4	9	8	8	8	1	9	8			9
대	0.0	0.1	0.0	0.0	0.0	0.0	0.1	0.0	0.09	0.09	0.0
	9	1	9	9	9	9	0	9			9
호	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.09	0.09	0.0
	1	9	9	9	9	9	9	9			9

1.3.3 웹사이트 디자인과 기능

(1) 웹 디자인

본 단추 언어분석 통합시스템은 동아대학교 웹서버에서 언어분석을 실행한다. 그리하여 어떠한 환경에서도 웹에 접근을 할 수 있다면 어디에서든 언어분석의 결과를 볼 수 있다. 아래 그림 4는 단추 언어분석 통합시스템에 접속한 사진이다.



그림 43 단추 언어분석 통합시스템 메인화면

POS(형태소 분석) 버튼과 NER(개체명 인식) 버튼, DP(의존 구문분석) 버튼, SRL(의미역 결정) 버튼, ALL(모든 언어분석을 수행) 버튼으로 구성 되어 있다. 기능적인 면에서 각각의 언어분석은 높은 성능을 보이고 있으며, 사용자가 보기 편한 디자인으로, 누구나 쉽게 언어분석을 수행할 수 있도록 구성하였다. 간단한 조작을 통해 사용자에게 편리성을 제공하며, 높은 성능을 통하여 사용자에게 신뢰성을 높일 수 있다.

제 2 장 소프트웨어 설치 및 실행

2.1 소프트웨어 설치 방법

본 단추 언어분석 통합시스템은 동아대학교 지능형시스템 연구실에 있는 웹서버에 구축되어 있어 특별한 설치 없이 소프트웨어를 실행할 수 있습니다.

단추 언어 분석 통합시스템에 접속하기 위해선 인터넷 웹페이지에 아래의 URL을 입력하면 본 시스템의 웹서버에 접속 할 수 있다.

표 51 단추 언어분석 통합시스템 웹서버 URL

단추 언어분석 통합시스템 URL
http://168.115.119.125:8086/Danchu/index.jsp

2.2 소프트웨어 실행 방법

본 단추 언어분석 통합시스템은 웹페이지로 구축 되어 웹페이지 IP를 이용하여 접속 하면 된다. 입력창에 분석을 하고 싶은 문장을 입력하고 분석버튼(POS, NER, DP, SRL, ALL)을 누르면 해당 모듈이 돌아 분석결과를 출력하게 된다.

2.2.1 웹브라우저 접속

단추(DANCHU) 언어분석 통합시스템은 웹페이지로 제작되었다. 따라서 어디에서든 웹브라우저로 단추 언어분석 통합시스템에 접근할 수 있다. 먼저 아래 그림 4와 같이 크롬 웹브라우저를 실행한다.

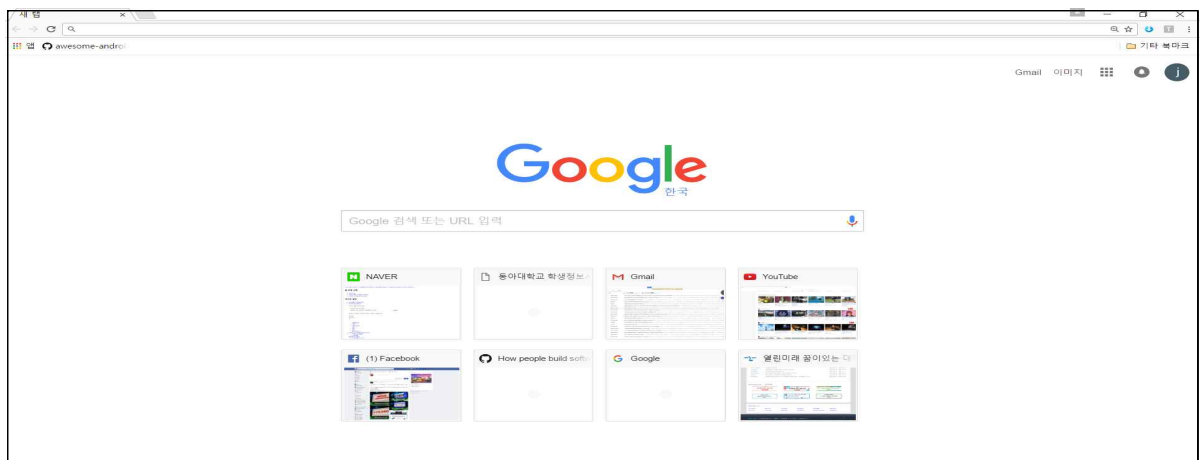


그림 44 웹브라우저 실행

2.2.2 IP접속

동아대학교 지능형 시스템 연구실에서 제공하는 IP주소를 주소창에 입력하여 접속한다.

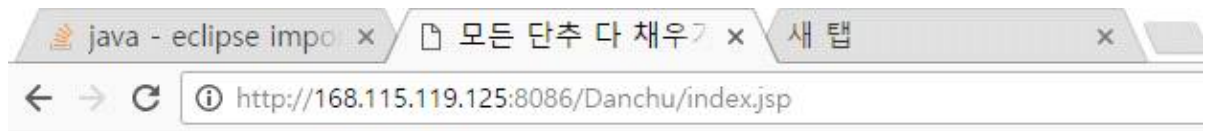


그림 45 IP주소

2.3 단추 언어분석 통합시스템

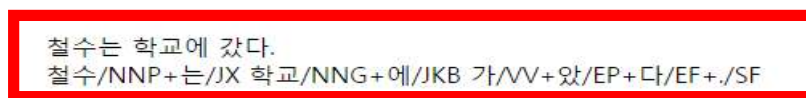
2.3.1 메인페이지



2.3.2 입력창



2.3.3 형태소 분석 결과



2.3.4 개체명 인식 결과

보라돌이와 두비는 12월 25일부터 아침 8시에 KBS에서 만나보실 수 있습니다.

보라돌이 : OG

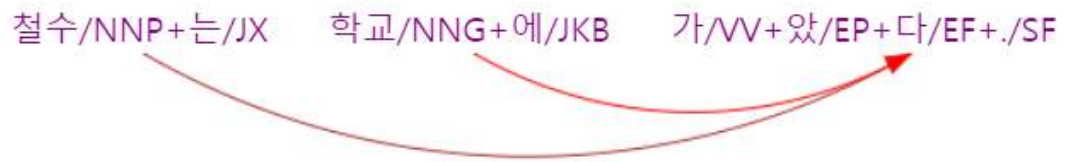
두비 : PS

12월25일부터 : DT

아침8시 : TI

KBS : OG

2.3.5 의존 구문 분석 결과



1 3 철수/NNP+는/JX NP_SBJ
 2 3 학교/NNG+에/JKB NP_AJT
 3 0 가/VV+았/EP+다/EF+./SF VP

2.3.6 의미역 결정 결과

POS NER DP SRL

갔다.		
가.01	철수는_ARG1	학교에_ARG2

3 장 소프트웨어 기능

3.1 프로그램 기능

단추(DANCHU) 언어분석 통합시스템은 입력된 한국어 문장을 형태소 분석, 개체명 인식, 의존 구문분석, 의미역 결정 등 4가지의 분석결과를 제공한다. 4가지 모듈 모두 bi-LSTM-CRFs와 음절 기반 임베딩 벡터를 사용하여 성능 면에서 개선되었다. 각 모듈에 대한 설명은 아래와 같다.

3.1.1 형태소 분석

형태소 분석기는 음절 단위로 품사를 부착하는 모델을 제안한다. 품사 부착을 위해 bi-LSTM-CRFs와 음절 기반 임베딩 벡터를 사용하였다. 먼저 bi-LSTM-CRFs은 forward 단계에서 현재 입력에 대한 상태층의 정보가 뒤의 상태에 영향을 주며, backward 단계에서 뒤에 상태가 앞의 상태에 영향을 주어 학습이 되기 때문에 다른 순차 레이블링을 위한 기계학습과 달리 작은 수의 자질만으로도 좋은 결과를 얻을 수 있다. bi-LSTM-CRFs은 음절단위로 입력이 결정되며, 각 음절에 대한 음절 표상을 나타내는 벡터를 구성하여 입력으로 사용된다. 시스템에서는 이를 위해 대용량 원시 말뭉치를 기반으로 음절 표상을 나타내는 64차원의 벡터를 생성하여 입력으로 사용하였다. bi-LSTM-CRFs의 성능을 향상시키기 위해 시스템에서는 각 음절이 전체 학습 말뭉치에서 출현한 품사 태그의 분포를 벡터로 구성하여 bi-LSTM-CRFs의 입력을 확장하였다. 각 음절은 학습 말뭉치에서 여러 형태소에 포함될 수 있으며, 여러 형태소는 다양한 품사를 가질 수 있다. 학습 말뭉치에서 음절이 포함된 형태소의 품사 빈도수를 계산한 후 softmax를 통해서 확률을 각 차원의 값으로 사용하였다. 시스템에서 제안하는 음절 단위 형태소 분석기는 bi-LSTM-CRFs으로 음절 단위 품사 태그를 결정한 후 기존의 방법과 같이 기분석 사전과 원형 복원 사전을 적용하여 최종적으로 형태소 분석된 결과를 보여준다.

3.1.2 개체명 인식

한국어 개체명 인식을 위해 bi-LSTM-CRFs를 이용하고, 입력으로 사용되는 단어 표상을 확장하기 위해 사전 학습된 단어 임베딩 벡터, 품사 임베딩 벡터, 그리고 음절 기반에서 확장된 단어 임베딩 벡터를 사용한다. 단어 표상 확장을 위해 첫 번째로 사전 학습된 단어 임베딩 벡터가 사용된다. 대부분의 개체명은 미등록어이기 때문에 학습 데이터에 나오지 않은 개체명을 잘 분류하는 것에는 한계가 있다. 따라서 대량의 말뭉치를 이용하여 단어 임베딩 벡터를 사전 학습하고, 단어 집합을 확장시킨다. 두 번째로는 사전 학습된 품사 임베딩 벡터를 사용하여 단어 표상을 확장한다. 개체명 인식에서는 품사의 시퀀스 또한 중요하기 때문에 품사를 잘 표현하는 임베딩 벡터가 중요하다. 품사 임베딩 벡터를 사전 학습하기 위해 대량의 말뭉치를 형태소 분석을 하고, 단어를 삭제한 뒤 품사만 학습하여 사용한다. 세 번째는 음절 기반 임베딩 벡터로부터 단어 기반 임베딩 벡터로 유도하여 단어 표상을 확장하는 방법이다. 임베딩 벡터 확장을 위한 입력인 음절 임베딩 벡터로는 각 음절별로 학습 코퍼스에 나온 개체명의 분포를 벡터로 만들어 사용하였다. 아래 그림 4는 개체명 인식 과정을 그림으로 도식화한 것이다.

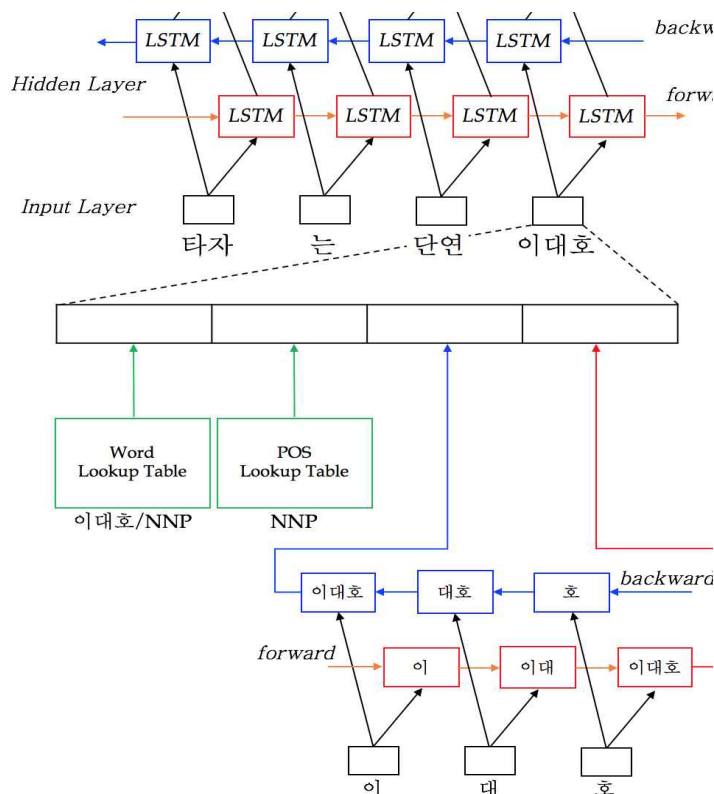


그림 53. 단어, 품사, 음절 임베딩 결합한 전체 구성도

3.1.3 의존 구문분석

국내에서 한국어 의존구문분석에 관한 연구가 활발히 진행되고 있지만 의존 관계만을 결과로 제시하고 의존 관계명을 제공하지 않는 경우가 많았다. 제안하는 의존 구문 분석기는 의존 파서와 의존 관계명을 부착하는 모델을 제안한다. 의존 파서는 전이 기반 방식을 사용하였으며 신경망을 통해 학습하여 의존관계의 유무를 판단하였다. 의존 관계명 부착은 의존경로(Dependency Path)와 음절의 의존 관계명 분포를 반영하는 음절 임베딩을 이용한 의존 관계명 부착모델을 제안한다. 아래 그림 5와 같이 의존 경로를 추출할 수 있다.

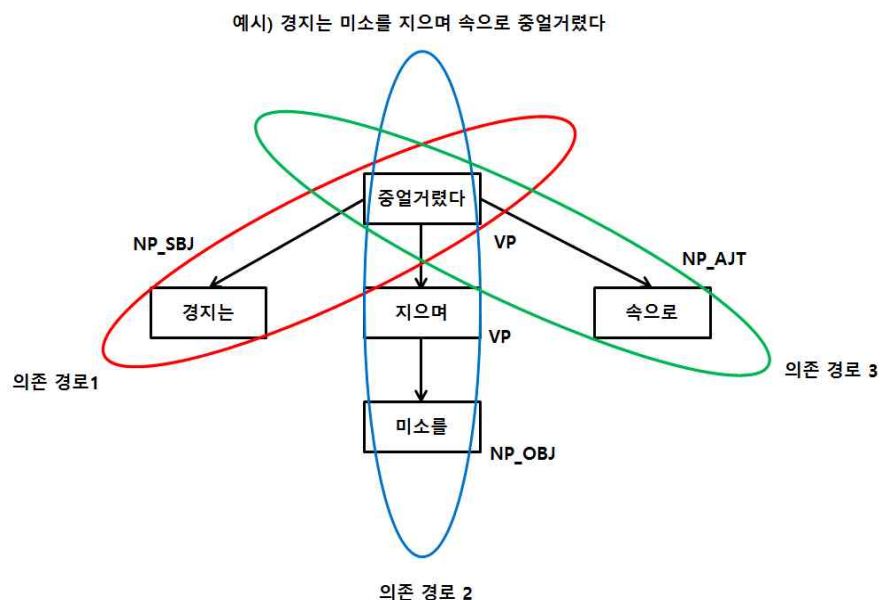


그림 54 의존 경로 예시

문장에서 나올 수 있는 최적의 입력 열인 의존 경로(Dependency Path)를 순차 레이블링에서 좋은 성능을 나타내고 있는 bi-LSTM-CRFs의 입력 값으로 사용하여 의존 관계명을 결정한다. 의존 경로의 경우 원래 문장으로부터 의존관계를 중복하여 추출하기 때문에 원래 문장으로 복구하는 작업이 필요하며, 성능평가는 복구를 한 문장으로부터 성능을 측정하였다. 제안된 기법은 의존 경로에 따라 어절 및 음절 단어표상(word embedding)만을 사용하여 최적의 입력 열을 이용하여 순차적으로 부착하였으며, 음절 분포 임베딩을 사용하여 성능 면에서 개선하였다.

3.1.4 의미역 결정

최근 의미역 결정 연구는 의미역 말뭉치와 기계학습 알고리즘을 이용한 연구가 주를 이루고 있다. 시스템에서는 순차적 레이블링 영역에서 좋은 성능을 보이고 있는 딥 러닝 기반의 Bidirectional LSTM-CRFs 기반으로 음절의 의미역 태그 분포를 고려한 의미역 결정 모델을 제안한다. bi-LSTM-CRFs는 어절단위로 입력이 결정되며, 각 어절에 대한 어절 표상을 나타내는 벡터를 구성하여 입력으로 사용된다. 시스템에서는 이를 위해 대용량 원시 말뭉치를 기반으로 어절 표상을 나타내는 64차원의 벡터를 생성하고, 여기에 어절에 포함된 형태소의 품사와 현재 어절과 의미적으로 연관된 서술어에 대한 정보가 반영된 벡터를 결합하여 입력으로 사용한다. bi-LSTM-CRFs의 성능을 향상시키기 위해 시스템에서는 각 어절에 포함된 음절이 전체 학습 말뭉치에서 출현한 의미역 태그의 분포를 반영하여 어절의 벡터를 구성하여 bi-LSTM-CRFs의 입력을 확장하였다. 각 음절은 학습 말뭉치에서 여러 서술어의 의미역에 포함될 수 있으며, 여러 의미역을 나타내는 다양한 품사를 가질 수 있다. 학습 말뭉치에서 음절이 포함된 의미역의 의미역 태그 빈도수를 계산한 후 softmax를 통해서 확률을 각 차원의 값으로 사용하였다. 각 음절별로 생성된 의미역 태그의 분포 벡터는 bi-LSTM 기반의 음절 의미역 태그 분포가 반영된 어절 벡터를 생성하는 모델의 입력으로 사용되며, 어절에 포함된 모든 음절에 대해 forward와 backward 단계를 거쳐 생성된 벡터를 생성하는데 이용된다. 생성된 음절의 의미역 분포 정보가 반영된 어절에 대한 벡터는 어절단위로 SRL을 진행하는 bi-LSTM-CRFs 모델의 입력 벡터에 결합하여 확장함으로써 개선된 의미역 결정 모델을 제안한다. 아래 그림 6은 단추 의미역 결정모델을 그림으로 도식화 한 것이다.

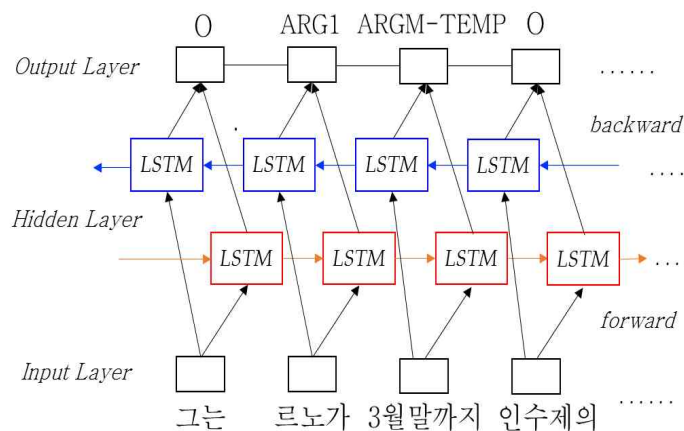


그림 55. bi-LSTM-CRFs

3.2 프로그램 기능 제약

- 입력 문서

- 한 행이 하나의 문장인 형식의 말뭉치
- 특수문자에 대한 분석은 아스키 코드표에 존재하는 특수기호만 사용

제 4 장 기타

4.1 시스템 성능

4.1.1 형태소 분석 성능

[1]에서 제안한 음절의 품사 분포 벡터를 이용한 bi-LSTM-CRFs 기반의 음절 품사 태깅 방법을 적용하였을 때 표 4와 같이 CRF 기반의 방법에 비해 7.65% 향상된 92.93%의 음절 단위 품사 태깅 성능을 보였으며, 표 5에서 기분석 사전, 불규칙 변환 사전을 적용한 후 원형복원 했을 때 CRF보다 3.01% 향상된 97.09%의 성능을 보였다.

표 52 bi-LSTM-CRFs를 이용한 음절 단위 품사태깅 성능(%)

Model	40만 어절
CRF (음절+어절자질)	85.28
bi-LSTM-CRFs (랜덤+음절의 품사 태그분포)	92.93 (+7.65)

표 53 기분석 사전과 원형복원을 적용한 성능 비교(%)

Model	40만 어절
CRF(음절+어절자질)+기분석사전+원형복원	94.08
bi-LSTM-CRFs (랜덤+음절의 품사태그분포)+기분석사전+원형복원	97.09 (+3.01)

4.1.2 개체명 인식

[2]에서 제안한 한국어 개체명 인식을 위해 bi-LSTM-CRFs에 입력으로 들어가는 단어 표상을 확장하는 방법을 이용하였다. 단어 표상을 확장하기 위하여 사전 학습된 단어 임베딩 벡터, 사전 학습된 품사 임베딩 벡터, 그리고 음절 기반 단어 임베딩 벡터를 사용하였다. 그 결과 품사 임베딩 벡터와 음절 기반 단어 임베딩 벡터를 추가한 모델이 사전 학습된 단어 임베딩 벡터만을 사용한 모델에 비해 4.93% 증가한 높은 성능을 얻을 수 있었다.

표 54 bi-LSTM-CRFs를 이용한 개체명 인식 성능(%)

Model	Micro-F1
형태소 단어 임베딩	74.59
형태소 단어 임베딩 + 형태소 품사 임베딩	77.58(+2.99)
형태소 단어 임베딩 + 형태소 품사 임베딩 + 음절의 개체명 태그분포	79.52(+4.93)

4.1.3 의존구문 분석

[3]에서 제안한 음절 기반 임베딩 벡터와 bi-LSTM-CRFs를 이용하여 의존 관계명을 부착하였고 의존 구조 파싱은 전이기반 방식과 퍼셉트론을 이용하여 의존 구조를 분석하였다.

표 55 전이 기반과 퍼셉트론을 이용한 의존 파싱 실험 성능(%)

Model	Micro-F1
전이기반 의존구조 파싱	85.61

표 56 bi-LSTM-CRFs를 이용한 의존 경로를 적용한 의존 관계명 부착 모델

Model	Micro-F1
bi-LSTM-CRFs (어절 임베딩 + 품사 빈도벡터)	95.16%
bi-LSTM-CRFs (어절 임베딩 + 품사 빈도벡터 + 음절 임베딩벡터)	96.01%

4.1.4 의미역 결정

[4]에서 제안한 음절의 의미역 분포 벡터를 이용한 bi-LSTM-CRFs 기반의 의미역 결정 방법을 적용하였을 때 표5와 같이 기존모델에 비해 2.41% 향상된 66.13%의 의미역 결정 성능을 보였다.

표 57 음절의 의미역 태그 분포를 이용한 실험 결과(%)

Model	Micro-F1
bi-LSTM-CRFs 모델 + 서술어 앞의 어절만 의미역 결정	63.72
bi-LSTM-CRFs 모델 + 서술어 앞의 어절만 의미역 결정 + 음절의 의미역 태그 분포	66.13(+2.41)

4.2 참고 문헌

- [1] 김혜민, 윤정민, 안재현, 배경만, 고영중, "품사 분포와 Bidirectional LSTM-CRFs를 이용한 음절 단위 형태소 분석기", 2016 HCLT 심사 발표 예정
- [2] 유홍연, 고영중, "품사 임베딩과 음절 단위 개체명 분포 기반의 Bidirectional LSTM CRFs를 이용한 개체명 인식", 2016 HCLT 심사 발표 예정
- [3] 안재현, 이호경, 고영중, "의존 경로와 음절단위 의존 관계명 분포 기반의 Bidirectional LSTM CRFs를 이용한 한국어 의존 관계명 레이블링", 2016 HCLT 심사 발표 예정
- [4] 윤정민, 배경만, 고영중, "음절의 의미역 태그 분포를 이용한 Bidirectional LSTM-CRFs 기반의 한국어 의미역 결정", 2016 HCLT 심사 발표 예정

국립국어원

한국어 세 줄 요약기 summ4riz3

설진석

【연세대학교】

차 례

제 1 장 소프트웨어 소개	112
1.1 소프트웨어 명칭	112
1.2 소프트웨어 사용 환경	112
1.3 소프트웨어 특징	112
제 2 장 소프트웨어 설치 및 실행	113
2.1 소프트웨어 설치 방법	113
2.2 소프트웨어 파일 구조	113
2.2.1 주요 파일 설명	113
2.2.2 전체 구조	114
2.3 소프트웨어 실행 방법	114
제 3 장 소프트웨어 기능	115
3.1 프로그램 기능	115
3.2 프로그램 기능 제약	115
제 4 장 기타	116

제 1 장 소프트웨어 소개

1.1 소프트웨어 명칭

한국어 세 줄 요약기(summ4riz3)는 한국어로 된 문서를 자동으로 세 줄 요약 해주는 프로그램입니다. 자동으로 문서를 요약해서 정리해주는 프로그램이 있다면 편리할 것이라 생각했고, 직접적으로 만들게 된 계기는 한국 인터넷 뉴스의 제목이 내용을 적절하게 표현하고 있지 못하다고 판단해서 요약본을 보는 것이 괜찮지 않을까 싶었기 때문입니다.

1.2 소프트웨어 사용 환경

- 지원 OS: 파이썬 2.7.x를 지원하는 모든 OS
- 권장 메모리: 0.5GB 이상
- 사용 언어: 파이썬 2.7
- 실행 환경: 파이썬을 사용할 수 있는 모든 환경
- 텍스트 인코딩: UTF-8

별도로 파이썬 웹 프레임워크 django를 활용하여 온라인 서비스화 할 수 있는 버전도 만들었으나 데모 페이지를 게시할 서버가 없었고, 이에 대해선 django로 웹사이트화 했다는 부분 외에는 추가되는 점이 없어서 별도로 기술하진 않겠습니다.

1.3 소프트웨어 특징

본 소프트웨어는 주어진 문서에서 가장 중요하다고 여겨지는 문장을 선정하여 보여줍니다. 문서를 자동으로 요약 해준다는 것은 시간이 부족한 현대인에게 있어서 유용한 프로그램이 될 것이며, 기사, 위키피디아 문서부터 논문까지 모든 종류의 문서에 적용될 수 있다는 장점이 있습니다.

제 2 장 소프트웨어 설치 및 실행

2.1 소프트웨어 설치 방법

본 소프트웨어에서 사용 중인 외부 파이썬 라이브러리가 있기 때문에 virtualenv와 같은 가상 환경 위에서 사용하는 것이 적절합니다. 필요한 라이브러리는 'requirements.txt'에 기술되어 있습니다.

리눅스로 예를 들면

0. 제출한 파일 중 'standalone.zip'의 압축 해제된 디렉토리를 기준으로

1. `pip install virtualenv`

2. `virtualenv venv`

3. `source venv/bin/activate`

4. `pip install -r requirements.txt`

5. `python example.py`

로 'example.txt' 내에 있는 문서를 요약 할 수 있습니다.

2.2 소프트웨어 파일 구조

2.2.1 주요 파일 설명

standalone 버전 기준으로 작성합니다.

① 실행 파일

summ4riz3.py — 라이브러리로서 패키지 되어있는 파일

example.py — 위 패키지를 사용하는 예제

② 데이터 파일

example.txt — example.py에서 예제로써 이용 될 문서 파일

③ 사전 파일

model/word3vec — Word2Vec 알고리즘으로 학습된 한국어 단어 임베딩 모델 파일

model/word3vec.syn0.npy — 위 word3vec 파일의 부가 데이터 파일

model/word3vec.syn1neg.npy — 위 word3vec 파일의 부가 데이터 파일

2.2.2 전체 구조

```
standalone/  
  model/ ..... Word2Vec 단어 임베딩 모델  
  example.txt..... 사용 예제에 사용되는 예시 문서  
  example.py..... 사용 예제  
  summ4riz3.py..... 본 소프트웨어
```

2.3 소프트웨어 실행

파이썬 프롬프트 혹은 IDLE에서 `import summ4riz3` 를 한 후에, 만약 변수 `text`에 문서가 스트링 형식으로 저장되어있다고 하면 `summ4riz3.summarize(text)`가 가장 중요하다고 판단되는 문장 3개를 출현 순으로, 리스트 식으로 반환합니다.

다른 방식으로, `from summ4riz3 import TextRank`
`TextRank(text).summarize(count=4)`

와 같이 요약할 문장의 수를 조절할 수도 있습니다.

실제로 사용되는 파이썬 코드에서도 비슷한 방식으로 사용할 수 있습니다.

그 사용 예시는 `example.py`와 같으며, 이는 터미널에서 `python example.py`로 실행해 볼 수 있습니다.

제 3 장 소프트웨어 기능

3.1 프로그램 기능

본 프로그램은 주어진 문서를 문장 단위로 나누고, 문장들을 TextRank 알고리즘을 통해서 랭킹을 결정 해서 상위 n개를 뽑아내는 방식으로 문서를 요약합니다. TextRank 알고리즘은 PageRank 알고리즘의 변형이며, 여기선 각 노드(이 경우엔 문장) 간 주고받는 어떤 값이 중요하게 작용합니다. 본 소프트웨어는 그 '어떤 값'을 한국어에 잘 맞도록 새로운 알고리즘을 사용하였습니다. 구체적으로는 주어진 두 문장의 명사와 동사가 얼마나 겹치는지를 jaccard coefficient로 계산하고, 각 문장을 구성하는 명사와 동사들의 Word2Vec 단어 임베딩 벡터들의 합을 구해서 cosine similarity를 계산하는 방식으로 이뤄집니다. 각각으로 PageRank를 적용해서 중요도를 구하고, 두 중요도를 섞어서 최종 순위를 매기게 됩니다. 명사와 동사 구분에는 한국어 형태소 분석기 라이브러리를 사용하여 명사는 어근만, 동사는 원형을 사용하도록 했습니다. Word2Vec은 나무위키(<http://namu.wiki>)와 몇 달간 크롤링을 통해 수집한 인터넷 뉴스 기사들을 사용하여 학습했고 (약 25만 종류의 단어), 학습에 필요한 파라미터는 다양한 실험을 거쳐서 최적의 값으로 사용되었습니다.

3.2 프로그램 기능 제약

Word2Vec 학습 모델이 상대적으로 크기가 크기 때문에 메모리를 많이 소모하며, 만약 이 부분을 제거하고 jaccard 모델만 사용하게 되면 메모리 문제를 해결 할 수 있습니다. Word2Vec 학습 모델은 out-of-vocabulary 단어를 처리 할 수 없으며, 따라서 본 소프트웨어는 한국어만을 대상으로 요약해 줍니다. 다만 이 역시도 해당 부분을 제거하고 외국어의 품사 태깅 함수(예: nltk)를 추가하면 어떤 언어에서도 작동하는 보편적인 알고리즘이 됩니다.

제 4 장 기타

본 소프트웨어의 프로토타입인 summariz3는 온라인 데모가 존재합니다.

<https://summariz3.herokuapp.com>

이는 본 소프트웨어보다 성능이 떨어지지만 잠시 인터넷 커뮤니티 사이트들에서 인기를 조금 얻어서 언론에서 소개된 적이 있습니다.

http://sports.khan.co.kr/news/sk_index.html?art_id=201606221136003&sec_id=562901

<http://www.insight.co.kr/newsRead.php?ArtNo=65634>

국립국어원

지도 표시 및 음성 녹음/재생 기능을 가진 지역어 수집 및 검색 APP '뽕꼬!?'

김푸름, 박소현, 심용석, 오승록, 이상호, 이재람, 안동언

【전북대학교】

차 례

제 1 장 소프트웨어 소개	120
1.1 소프트웨어 명칭	120
1.2 소프트웨어 사용 환경	120
1.3 소프트웨어 특징	121
제 2 장 소프트웨어 설치 및 실행	123
2.1 소프트웨어 설치 방법	123
2.2 소프트웨어 파일 구조	125
2.2.1 주요 파일 설명	125
2.2.2 전체 구조	125
2.3 소프트웨어 실행 방법	126
2.3.1 소프트웨어 실행	126
2.3.2 검색기능	127
2.3.3 지역어 지도보기 기능(지도)	128
2.3.4 지역어 정보입력 기능(녹음)	129
제 3 장 소프트웨어 기능	130
3.1 프로그램 기능	130
3.2.1 지역어 수집(기능1)	130
3.2.2 지역어 검색(기능2)	130
3.2.1 지역어 지도 표시(기능3)	130
3.2.1 음성 녹음 및 재생(기능4)	130
3.2 프로그램 기능 제약	130
제 4 장 기타	131
4.1 추후 방향	131
4.2 유지 보수	131

제 1 장 소프트웨어 소개

1.1 소프트웨어 명칭

1.1.1 명칭

지도 표시 및 음성 녹음/재생 기능을 가진 지역어 수집 및 검색APP '뭐꼬!?'이다. 기존 텍스트 형식에 머물러 있던 국어학 연구 범위를 음성자료 및 위치정보 까지도 병행하여 사용자가 편리하게 사용 할 수 있는 단계로 확대하여 누구나 쉽게 접근할 수 있는 APP이다.

1.1.2 설명

이 소프트웨어는 연구자와 학생들이 현재까지 수집한 대규모 지역어 데이터를 충분히 활용한다. 기존의 지역어 수집 APP은 해당 지역어의 정보만을 얻을 수 있으나 '뭐꼬!?'는 정보와 위치기반서비스, 녹음 기능을 탑재하여 실시간으로 지역어에 대한 정보를 입력할 수 있으며, 원하는 지역어를 검색하여 지도에 표시하여 보여줄 수 있다. 이러한 지역어 지도는 기존의 APP들과 차별화된 기능으로써 해당 지역어가 사용되는 위치를 정확하게 알 수 있으며, 더 나아가 여러개의 지역어에 대한 구분까지 할 수 있는 기능이다.

1.1.3 개발 동기

지역어는 현재 뉴스, 드라마, 영화 등 다양한 대중매체에서 공개적으로 사용되고 있다. 많은 사람들의 지역어에 대한 관심이 늘어나고 있다. 이에 '뭐꼬!?'는 한국인들에게 올바른 지역어의 사용과 명맥을 이어나갈 수 있도록 도움을 주고자 한다. 또한 지역어에 대한 데이터관리, 지역어 변화에 대한 정보 등 전문가들도 사용하기에 용이한 APP을 만들고자 하였다.

1.2 소프트웨어 사용 환경

- 지원os : Android 4.0.3 플랫폼 이상 스마트폰
- 권장 메모리 : 16MB 이상
- 사용 언어 : JAVA, PHP
- 개발 환경 :
 - 서버 : APM(Apache, PHP, Mysql), JSON(통신)
 - 클라이언트 : Android Studio, JDK(1.8.x), 안드로이드 기반 스마트폰
- 텍스트 인코딩 : UTF-8

1.3 소프트웨어 특징

1.3.1 특징

① 지역어 수집

- 1) 국립국어원에서 배포한 지역어 자료 CD의 데이터를 기반으로 제작
- 2) 해당 지역어의 녹음 파일이 없거나 추가를 원할 시에 사용자나 관리자가 직접 녹음 파일을 저장하고 수정할 수 있다.

② 지역어 검색

- 1) 사용자가 지역어를 검색 시, 해당 지역어를 서버에서 찾고 불러옴
- 2) 검색할 때 음성 파일 유무와 지역어가 통용되는 위치 정보를 확인할 수 있다.

③ 지역어 지도 표시

- 1) 사용자의 현재 위치 정보를 지역어와 함께 저장할 수 있다.
- 2) 지역어와 함께 저장된 위치정보를 지도에 표시할 수 있다. (마커)

④ 음성 녹음 및 재생

- 1) 국어정보원에서 배포한 지역어 자료 CD에 저장되어 있는 기존 음성파일을 불러온다.
- 2) 현재 제공된 APP으로 지역어 녹음, 재생, 저장 및 수정이 가능하고, 해당 지역어의 위치를 표시할 수 있다.

1.3.2 장점

① 전문가

- 1) 기존에 지역어를 수집하는 방식에서 스마트폰을 이용하여 쉽게 데이터를 수집 및 검색 할 수 있다.
- 2) 단어별로 구분할 수 있는 기능을 활용하여 지역어가 어떠한 환경요건으로 인해 다르게 사용되었는지를 쉽게 구분하여 볼 수 있다.
- 3) 연구 중 알지 못했던 다양한 지역어를 사용자가 입력함으로써 직접 답사를 하지 않아도 연구를 할 수 있다.

② 학생

- 1) 국어책에서 쉽게 접할 수 없는 지역어와 거주 지역이 아닌 지역에 대한 언어를 쉽게 접하며 언어 영역에 대한 지식의 폭이 넓어질 수 있다.
- 2) 관련 학과의 학생들은 이 어플리케이션을 통해 자료조사를 보다 쉽게 할 수 있다.

③ 일반인

- 1) 타 지역에 여행을 가는 경우, 지역어에 대한 부담감 없이 여행할 수 있다.

1.3.3 기대효과

- 1) 다양한 기능과 사용자 편의성을 가진 스마트폰 APP 언어 지도 시스템을 보급한다.
- 2) 전국적 규모의 언어 정보를 시각적으로 이해할 수 있도록 제시한다.
- 3) 언어 연구 등 각계 전문가가 다양한 조사 자료를 활용할 수 있는 기반을 마련한다.
- 4) 학생, 일반인들도 쉽게 지역어에 접근할 수 있는 기반을 마련한다.
- 5) 국립국어원의 조사 자료 활용도를 높이고, 앞으로 있을 자료 조사에 효율적으로 대비하는 도구로 사용한다.
- 6) 스마트폰만 있으면 언제 어디서나 지역어를 확인할 수 있다.
- 7) 기존에 조사된 각종 언어 자료를 통합할 수 있는 도구를 제공한다.
- 8) 구어와 방언 자료를 교육에 활용할 수 있도록 시청각 매체를 제공한다.
- 9) 다양한 계층에서 한국어의 실상을 잘 파악할 수 있도록 한다.
- 10) 무형의 자산인 언어 자료를 보존하고 활용하는 데 기초 자료로 이용한다.

제 2 장 소프트웨어 설치 및 실행

2.1 소프트웨어 설치 방법

- 현재는 APK파일을 다운받아서 직접 핸드폰에 넣어야 하지만 추후 Google play store에 등록하여 쉽게 다운 및 설치 가능하게 변경 예정

1. app-debug.apk 파일을 안드로이드 핸드폰에 옮긴다.
2. 핸드폰의 파일관리자로 들어가 app-debug.apk 파일을 실행시킨다.

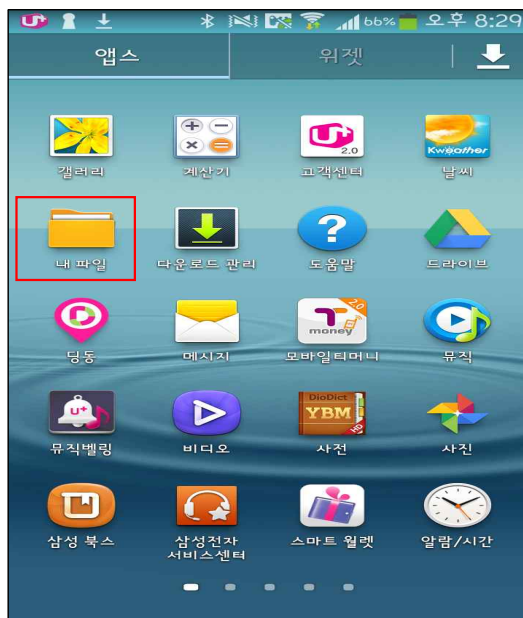


그림 2.1) 파일관리자 열기

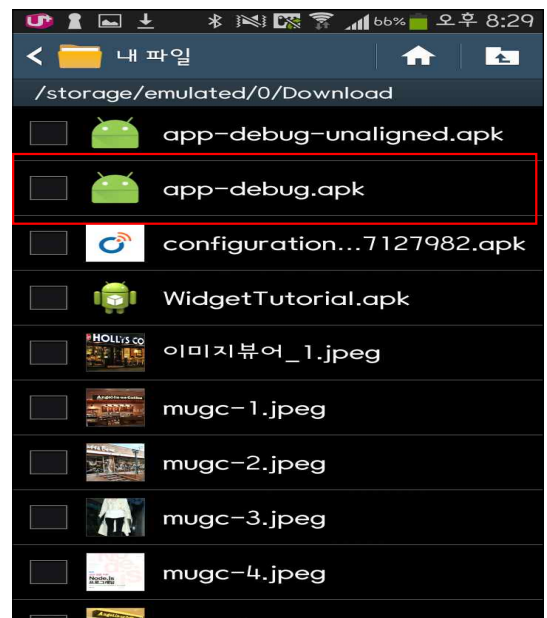
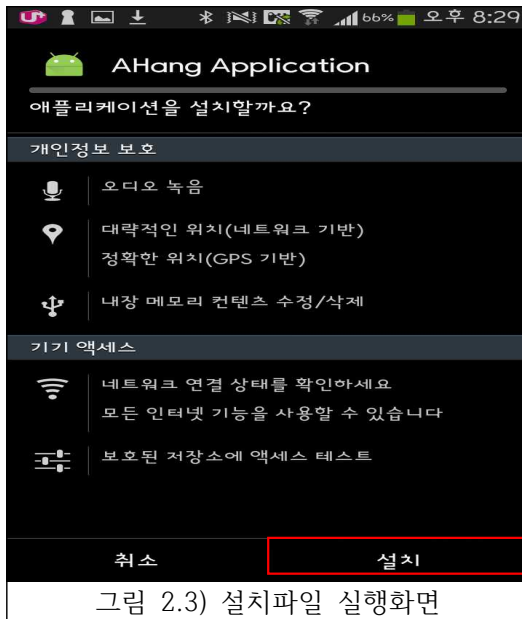


그림 2.2) app-debug.apk 파일 실행

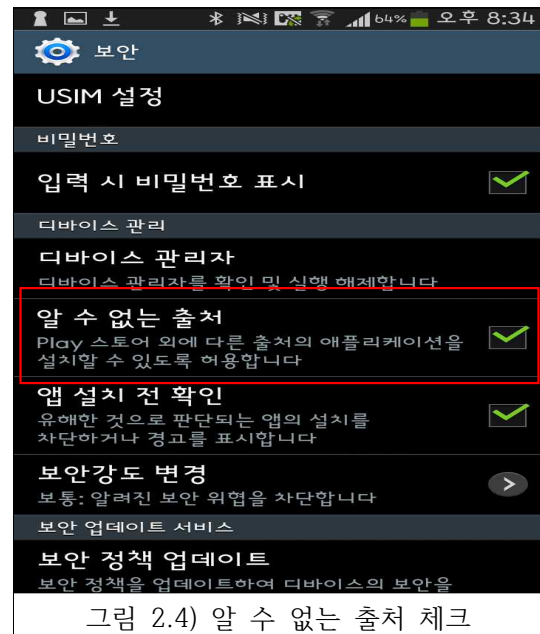
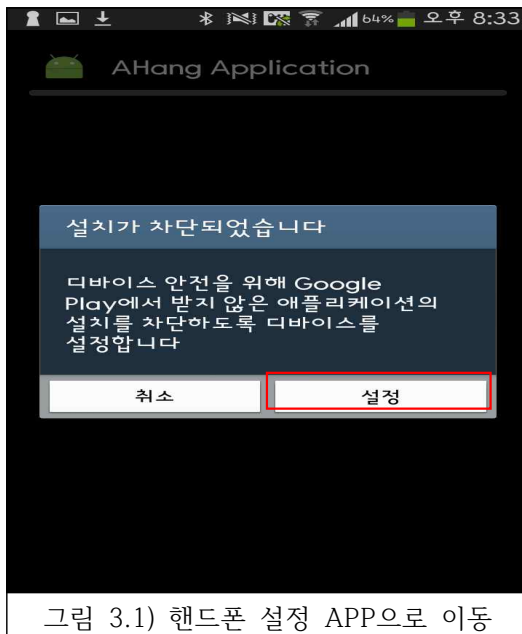


3. 알 수 없는 출처의 APP으로 인해 설치가 차단 되었을 때 방법

- 설치가 차단되었을 경우는 스마트폰 기본 설정에서 Play store에서 다운받지 않은 APP에 대해서 설치가 불가능하게 설정되어 있는 경우다.

이때

그림 3.1)과 같이 바로 설정창으로 넘어갈 수 있는 화면이 제공된다면 설정버튼을 클릭해서 수정해 주면 된다.



2.2 소프트웨어 파일 구조

2.2.1 주요 파일 설명

① 실행 파일

1. app-debug.apk 파일

② 데이터 파일

1. 모든데이터는 서버 및 데이터베이스에 저장된다.
2. 서버(녹음파일 저장)
 - record 폴더 : 사용자가 녹음한 파일이 .mp4파일로 저장되어 진다.
3. 데이터베이스(지역어 정보 저장)
 - record 테이블 : 지역어에 대한 정보(표준어, 주소, 녹음파일 위치 등)가 저장된다.

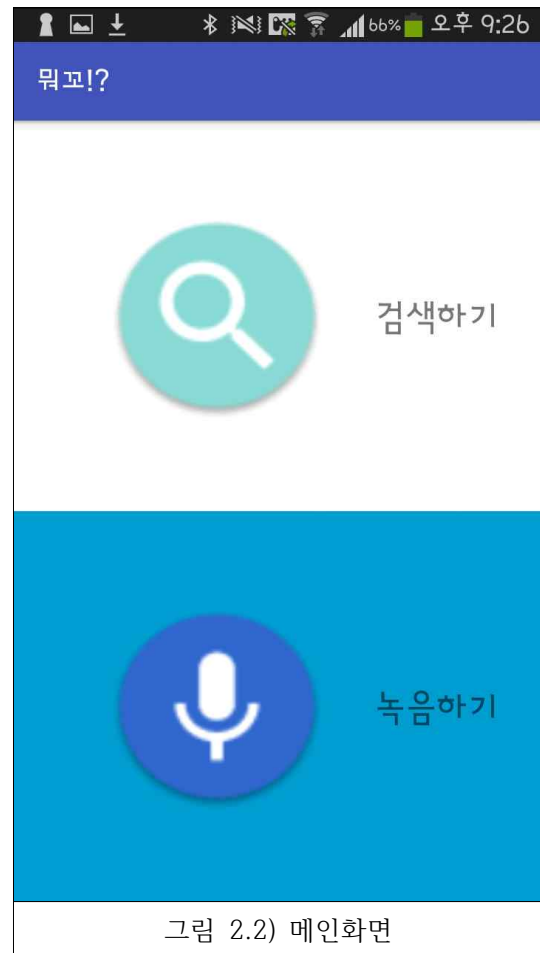
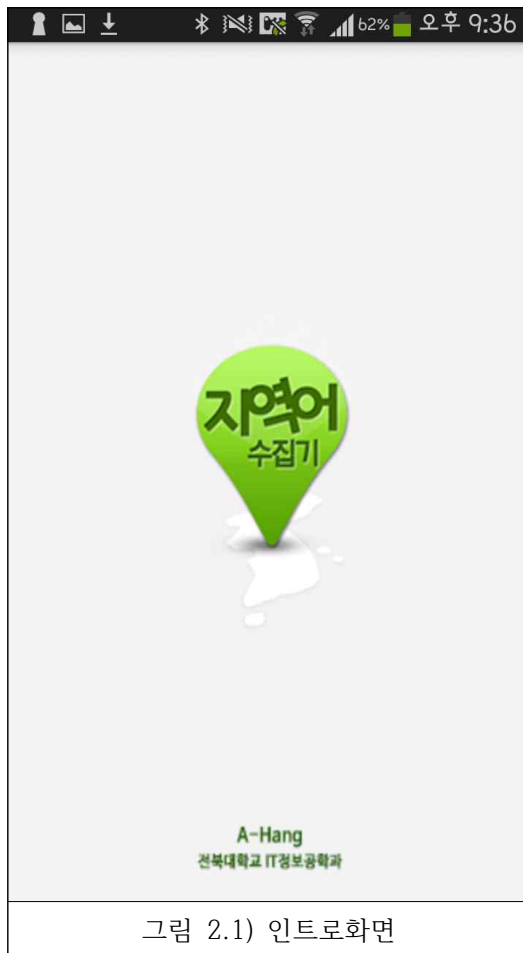
2.2.2 전체 구조

Project/
src/ 소스파일 리소스파일 폴더
main/ 소스파일 폴더
res/ 레이아웃 파일 폴더
drawble/..... APP이미지 파일 폴더
string/..... 언어 리소스 폴더
AndroidManifest.xmlAPP권한 설정 파일

2.3 소프트웨어 실행

- 어플리케이션 다운로드 방법은 2.1 방식과 동일하다.

2.3.1 소프트웨어 실행(인트로화면 및 메인화면)



2.3.2 검색기능

검색기능 초기화면(그림 3.1)을 기준으로 사용자가 검색하고 싶은 표준어를 입력(그림3.2)하고 검색버튼을 누른다.

해당 표준어에 대응하는 지역어 리스트가 화면에 출력된다. 이때 오른쪽에 색상이 표시가 되는데 이 색상은 지역어 지도에 보여질 색상이다. 초기 색상은 모두동일하게 설정되어 있으며(그림 3.3), 색상랜덤 버튼을 클릭하였을 경우 색상이 변경되어 보여진다(그림 3.4)

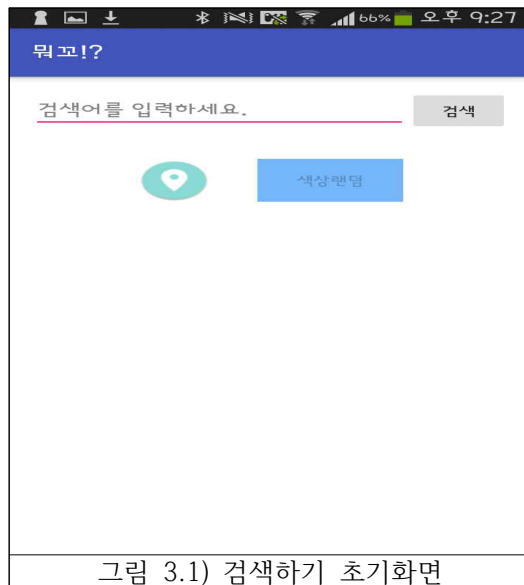


그림 3.1) 검색하기 초기화면

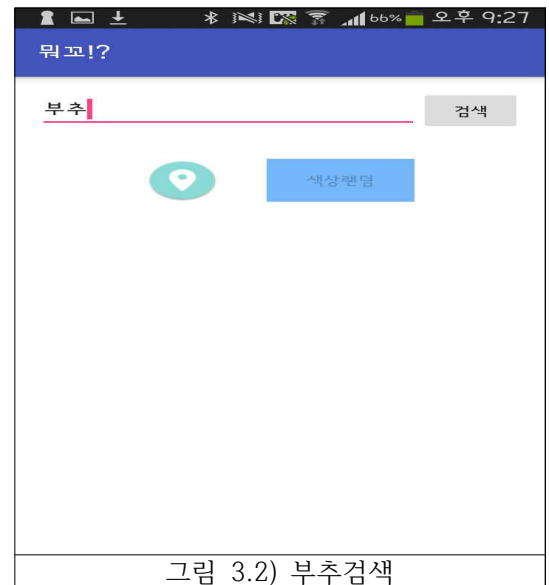


그림 3.2) 부추검색

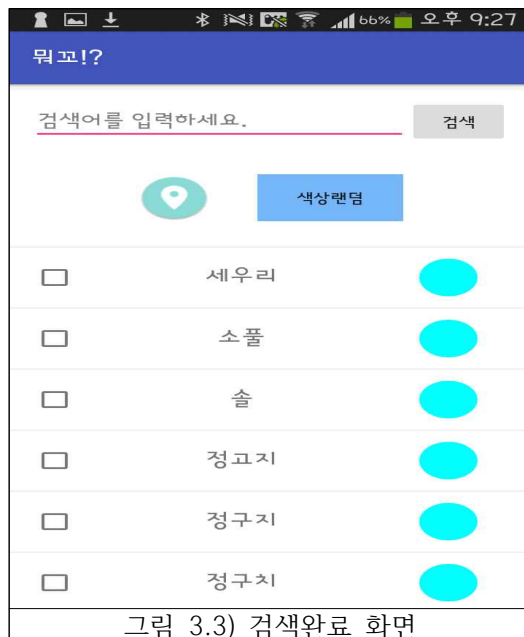


그림 3.3) 검색완료 화면

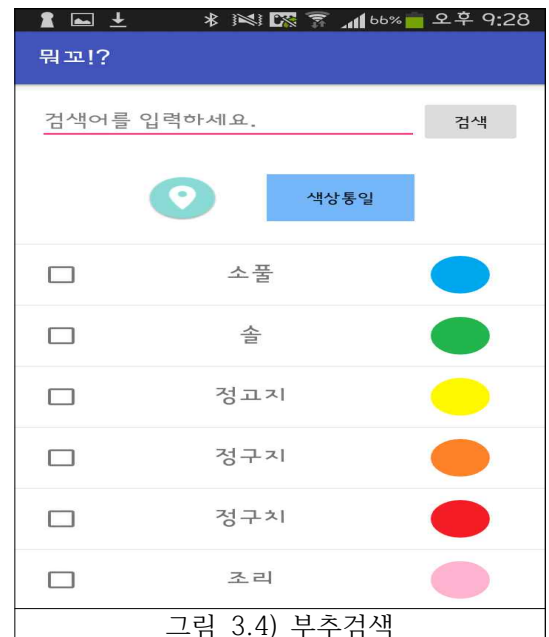


그림 3.4) 부추검색

2.3.3 지역어 지도보기 기능(지도)

검색된 지역어가 어느 지역에서 사용되어 지는지 간략하게 확인하고 싶으면 검색된 리스트를 클릭하면 사용되는 지역을 확인할 수 있다.(그림 4.1) 만약 사용되는 지역을 좀 더 세부적으로 지도화면으로 보고 싶다면 지도에 보여질 지역어를 체크한다.(그림 4.2) 그 후 색상통일 버튼 좌측에 있는 마커버튼을 클릭하게되면 체크되어진 지역어가 지도에 보여지게 된다.(그림 4.3) 또한 지도에 표시된 지역어에 대해서 세부적인 정보 및 음성파일을 듣고 싶다면 표시된 마커를 클릭하면 세부정보 화면이 뜨게 된다.(그림 4.4) 여기서 마이크 표시의 버튼을 클릭하면 음성파일을 들을 수 있다.

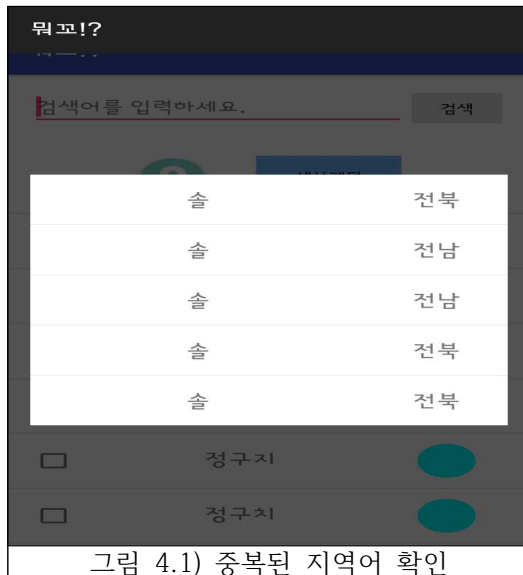


그림 4.1) 중복된 지역어 확인

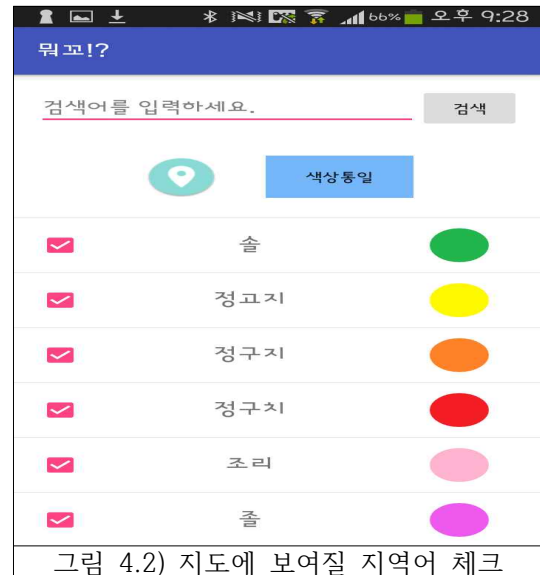


그림 4.2) 지도에 보여질 지역어 체크

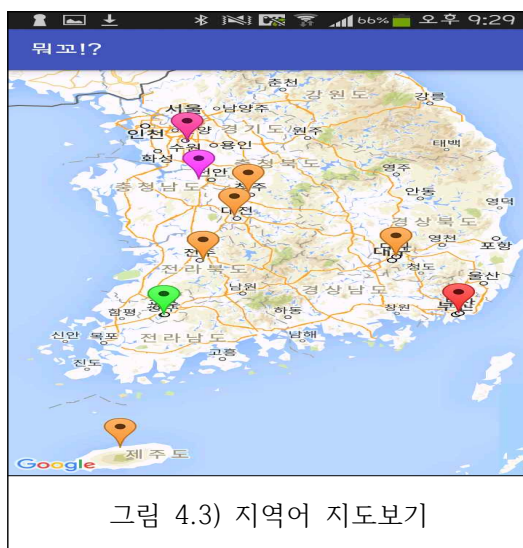


그림 4.3) 지역어 지도보기

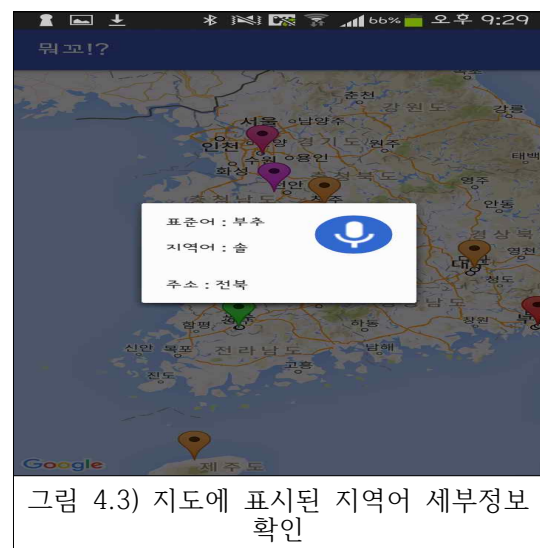


그림 4.3) 지도에 표시된 지역어 세부정보 확인

2.3.4 지역어 정보입력 기능(녹음)

메인화면(그림 5.1)에서 녹음하기 기능으로 들어가면 녹음화면(그림 5.2)으로 넘어가게 된다. 이 화면에서는 지역어에 대한 녹음파일을 생성 할 수 있다. 가운데 녹음버튼을 클릭하게 되었을 시 녹음중이라는 표시로 버튼이 빨간색으로 변하게 된다.(그림 5.3) 녹음중 현재 녹음파일의 시간이 가운데 표시가 되며, 녹음을 완료하기 위해서는 맨 우측 네모 버튼을 클릭하면 녹음이 중지된다. 녹음한 파일을 재생하여 이상이 없는지 확인하기 위해서는 맨 좌측 재생 버튼을 클릭하면 녹음한 파일을 들어볼 수 있다.

녹음이 완료되었다면 다음 버튼을 클릭하여 녹음파일에 대한 지역어 정보를 입력할 수 있는 화면(그림 5.4)으로 넘어간다. 이 화면에서는 현재 나의 위치가 자동으로 검색되어 지도에 표시되며 해당 주소가 가운데 보여진다. 그리고 녹음파일에 대한 표준어 및 지역어를 입력하고 등록 버튼을 누르면 지역어에 대해 녹음파일, 표준어, 지역어, 상세주소 등이 데이터베이스에 저장된다.

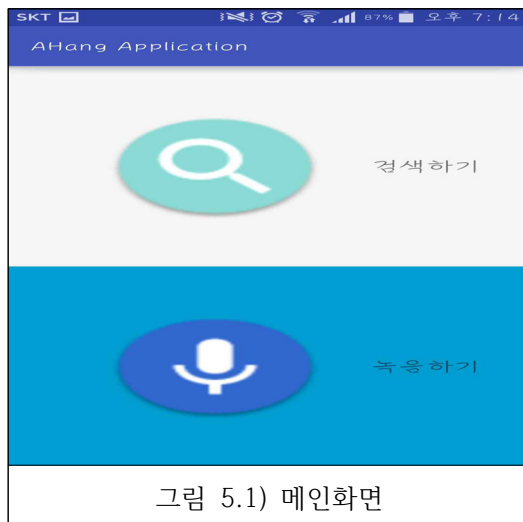


그림 5.1) 메인화면



그림 5.2) 녹음 초기화면



그림 5.3) 녹음중 화면

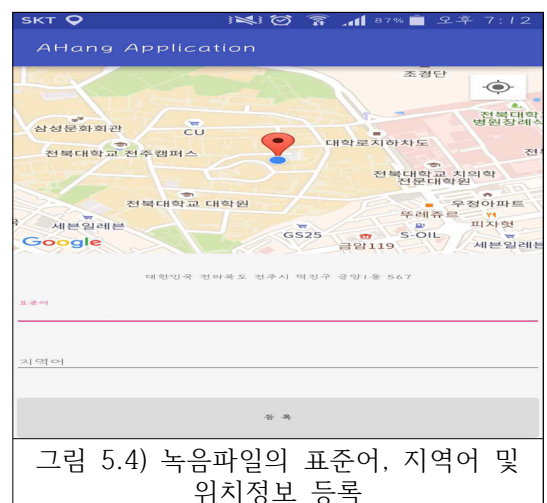


그림 5.4) 녹음파일의 표준어, 지역어 및 위치정보 등록

제 3 장 소프트웨어 기능

3.1 프로그램 기능

3.1.1 지역어 수집(기능1)

- 1) 스마트폰의 녹음 기능을 이용하여 지역어를 녹음한다.
- 2) 스마트폰의 GPS기능을 이용하여 지역어를 사용하는 위치정보를 입력한다.
- 3) 지역어에 상대적인 표준어를 함께 입력한다.
- 4) 지역어를 제공한 사람에 대한 정보를 입력할 수 있다.

3.1.2 지역어 검색(기능2)

- 1) 정보를 얻고자 하는 지역어 또는 표준어를 입력하여 검색한다.
- 2) 검색된 단어와 같은 의미를 가진 모든 지역어가 리스트로 나타난다.
- 3) 해당 단어 리스트에서 단어별 색상을 선택해 분류할 수 있다.

3.1.3 지역어 지도 표시(기능3)

- 1) 우리나라 지도에 데이터가 분류된 색상으로 나타난다.(마커)
(이 때, 선택되지 않은 데이터는 기본색으로 제공한다.)
- 2) 지도상에서 데이터(마커)를 클릭 했을 때, 상세 정보를 볼 수 있는 화면이 제공된다.

3.1.4 음성 녹음 및 재생(기능4)

- 1) 지역어 수집 시 음성 녹음 및 녹음된 음성을 확인 할 수 있다.
- 2) 지역어 지도의 상세정보화면에서 데이터 음성 파일을 재생 할 수 있다.

3.2 프로그램 기능 제약

- 1) 현 APP은 안드로이드 OS에서만 가능하다.
- 2) 스마트폰 종류에 따라 화면 구성이 달라질 수 있다.
- 3) 사용자가 녹음, 위치 정보에 대한 권한을 APP에 주지 않으면 사용 불가하다.
(안드로이드 버전 6.0(마쉬멜로우) 이상에만 해당)

제 4 장 기타

4.1 추후 방향

- 사용자들의 의견을 반영하여 UI/UX 개선
- 외국인에게도 이용 가능한 외국어 지원
- 아이폰(iOS) 에서도 사용 가능한 APP 개발
- 지역어 지도의 세분화 개발(기호표시)

4.2 유지 보수

- 관리자는 출시 후 정기적으로 사용자들의 의견을 수렴하여 만족도 높은 APP으로 개선시킨다.
- 지역어 연구원들과의 교류를 통해 더 나은 서비스로 발전시킨다.
- 불필요하거나 중복된 정보를 주기적으로 확인하여 제거한다.

국립국어원

문맥 기반 실시간 한국어 문장 교정 시스템

박영근, 최재성, 김재민, 이성동, 이현아

【금오공과대학교】

차 례

제 1 장 소프트웨어 소개	136
1.1 소프트웨어 명칭	136
1.2 소프트웨어 사용 환경	136
1.3 소프트웨어 특징	136
제 2 장 소프트웨어 설치 및 실행	137
2.1 소프트웨어 설치 방법	137
2.2 소프트웨어 파일 구조	137
2.2.1 주요 파일 설명	137
2.2.2 전체 구조	138
2.3 소프트웨어 실행 방법	138
제 3 장 소프트웨어 기능	140
3.1 프로그램 기능	140
3.2 프로그램 기능 제약	140
제 4 장 기타	140

제 2 장 소프트웨어 설치 및 실행

2.1 소프트웨어 설치 방법

압축 파일을 원하는 위치에 압축 해제함

2.2 소프트웨어 파일 구조

2.2.1 주요 파일 설명

① 실행 파일

SentenceAnalysor.exe --- 소프트웨어 실행 파일

② 모듈 파일

dll/IMEChecker.exe --- 한영 변환 여부를 확인하고 연동

dll/vcruntime.dll --- IMEChecker.exe 를 구동하기 위함

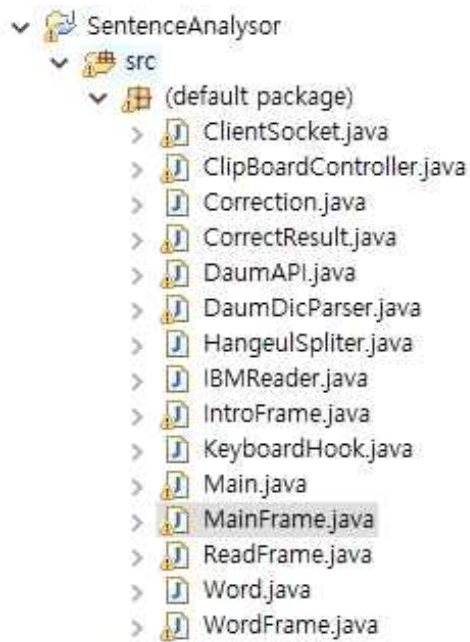
dll/ucrtbased.dll --- IMEChecker.exe 를 구동하기 위함

dll/mfc140ud.dll --- IMEChecker.exe 를 구동하기 위함

③ 인터페이스 파일

img/ --- 프로그램 내 인터페이스에 사용할 이미지 파일

2.2.2 전체 구조



<프로그램 전체 구조>

2.3 소프트웨어 실행

SentenceAnalysor.exe 파일을 실행한다. 다른 프로그램, 예를 들어 메모장, 한글, 인터넷 브라우저 등 글을 입력할 수 있는 환경에서 한글 입력을 시작한다.

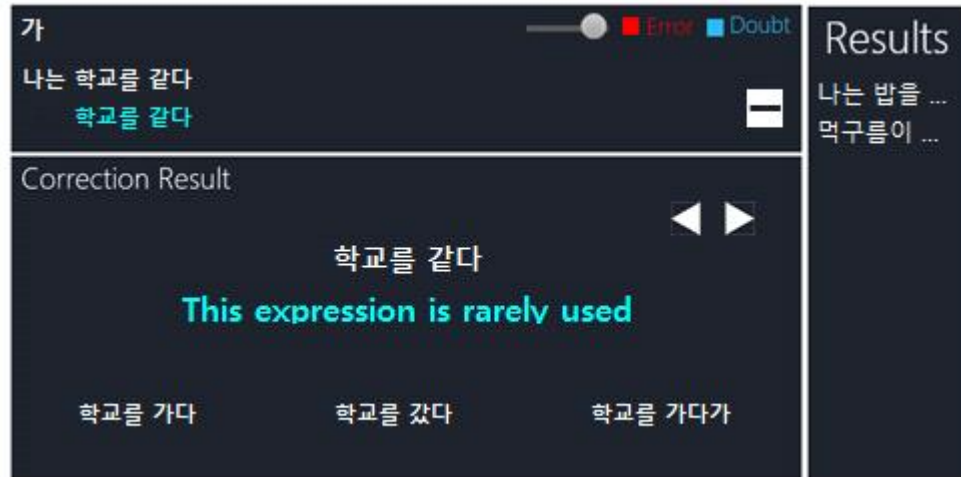
프로그램은 타 프로세스의 사용자 입력과 연동되어 사용자의 입력을 출력한다. 2 어절이 생길 경우, 즉 공백이 두 번 발생할 경우 분석 결과를 출력한다.



<기본 출력 화면>

기본 출력 화면에서는 최종 교정 문구를 보여주며, 다른 추천 어절은 보여주지 않는다. 또한 위의 슬라이드 바로 투명도를 조절할 수 있다. +를

눌러 상세 출력 화면으로 전환할 경우 보다 상세한 분석 결과를 확인할 수 있다.



< 상세 출력 화면 >

상세 출력화면에서는 자신이 입력한 문구에 대한 분석 내용과, 유사 어절 추천을 확인할 수 있다. 어절 단위의 교정을 볼 수 있기 때문에, ◁ 와 ▷ 버튼으로 다른 어절 분석으로 넘어갈 수 있다. 올바른 분석에 대한 결과는 출력하지 않는다.

추천 어절은 유사도 (추천도) 가 높은 순으로 왼쪽부터 출력한다. 사용자는 입력이나 추천 어절 및 단어에 대해 검색을 할 수 있다. 드래그 후 우클릭한 뒤 복사 또는 영한사전 검색을 할 수 있으며, 영한사전은 다음 영한사전 결과를 파싱하여 출력한다.

F2 또는 Enter 키를 입력할 경우 해당 문장은 종결되었다고 본다. 그 후 새로운 문장을 입력하게 되면 기존 분석 결과는 Results 에 저장되고 새로운 문장을 입력할 수 있다.

과거 분석 결과를 확인하고 싶다면, Results 의 항목을 클릭하면 된다. 원한다면 다시 크기 조절 버튼을 눌러 기본 출력 화면으로 전환할 수 있다.

제 3 장 소프트웨어 기능

3.1 프로그램 기능

프로그램은 사용자의 입력을 받고, DB의 말뭉치 문맥 정보를 이용하여 적절한 어절을 추천 및 교정해줄 수 있다. 또한 시스템이 추천해준 어절을 복사할 수 있고, 영한사전을 검색할 수도 있다.

분석한 어절은 기록되어 사용자가 원하는 경우 조회 또는 삭제가 가능하다. 또한 서버에서 사용자가 보낸 기록들을 관리하여, 이미 분석한 문장에 대해서는 기존 분석을 출력해줄 수 있기 때문에 프로그램이 지속적으로 사용될수록 속도 향상을 기대할 수 있다.

3.2 프로그램 기능 제약

1. 타 프로그램과 동시 사용을 전제하므로, 타 프로그램 없이 직접 사용 불가
2. 서버가 연동되어야 분석이 가능하므로, 하기 경우에는 사용 불가
 - 서버 또는 클라이언트의 인터넷 미연결
 - 서버가 작동하지 않을 경우
3. 키보드 후킹을 사용하므로 은행 등 보안을 요구하는 매체와 동시 사용 불가

제 4 장 기타

본 프로그램의 기반 어문 자료는 국립 국어원에서 제공하는 말뭉치와 각종 언론의 기사를 수집하였다. 또한 사용자의 입력 특성을 고려하여, 기초 한국어 교재와 미숙자의 작문 사례를 참조한 예문도 추가하였다.

국립국어원

ESPRESSO TOOL 2016

신창욱, 박태호, 박성재, 박다솔, 신영태, 차정원

【창원대학교】

차 례

제 1 장 소프트웨어 소개	144
1.1 Espresso Tool	144
1.2 소프트웨어 사용 환경	144
1.3 소프트웨어 특징	144
제 2 장 소프트웨어 설치 및 실행	145
2.1 소프트웨어 설치 방법	145
2.1.1 라이브러리 설치	145
2.1.2 컴파일	145
2.2 소프트웨어 파일 구조	145
2.2.1 주요 파일 설명	145
2.2.2 전체 구조	146
2.3 소프트웨어 실행 방법	146
제 3 장 소프트웨어 기능	147
3.1 프로그램 기능	147
3.2 프로그램 기능 제약	147
제 4 장 기타	147

제 2 장 소프트웨어 설치 및 실행

2.1 소프트웨어 설치 방법

2.1.1 라이브러리 설치

Espresso Tool을 실행하기 위해서는 CRF++와 Boost 라이브러리의 설치가 필요하다. CRF++와 Boost는 압축파일로 프로젝트 내에 포함되어 있다. 각각 bootstrap.sh를 실행시켜 설치할 수 있다. 두 라이브러리는 ./lib/에 설치된다.

```
$ sh bootstrap_boost.sh
$ sh bootstrap_crf.sh
```

혹 아래와 같은 오류가 뜬다면,
\$bash bootstrap_boost.sh
을 이용해 설치하기 바란다.

```
bootstrap_boost.sh: 5: bootstrap_boost.sh: source: not found
mv: `b2'를 설명할 수 없음: 그런 파일이나 디렉터리가 없습니다.
mv: `bjam'를 설명할 수 없음: 그런 파일이나 디렉터리가 없습니다.
```

2.1.2 컴파일

본 도구는 다음과 같은 방법으로 컴파일한다. 컴파일 위치는 도구가 설치된 최상위 폴더에서 진행한다.

```
$ make
```

2.2 소프트웨어 파일 구조

2.2.1 주요 파일 설명

① 실행 파일

Espresso --- 소프트웨어 실행 파일

② 사전 파일

dic/ --- 형태소태그 부착을 위한 데이터파일

NeDic/ --- 개체명 사전 (개체명 사전, 개체명 정보, 일반명사 사전)

③ 모델 파일

model/ --- 개체명, 구문분석, SRL모델, 워드벡터사전, 구문분석사전

④ 라이브러리 파일

lib/ --- 형태소 분석기 라이브러리, CRF++, boost 라이브러리

⑤ 코드 및 헤더파일

src/ : 프로젝트 코드

include/ : 헤더 파일

2.2.2 전체 구조

[프로젝트폴더]

dic/ : 형태소태그 부착을 위한 데이터파일 폴더

lib/ : 형태소태거 라이브러리, CRF 라이브러리, boost 라이브러리 폴더

model/ : 개체명, 구문분석, SRL모델, 워드벡터사전, 구문분석사전 폴더

NeDic/ : 개체명 사전 폴더

src/ : 프로젝트 코드 폴더

include/ : 헤더 파일 폴더

Makefile

2.3 소프트웨어 실행 방법

본 도구는 컴파일 후 생성된 “Espresso”파일로 실행한다. [입력파일]을 결정하는 -i 옵션과 [출력파일]을 결정하는 -o 옵션을 사용하여 다음과 같이 실행한다. 입력파일은 확장자까지 정확하게 적어준다. 실행 파일은 컴파일이 완료되면 도구가 설치된 최상위 폴더 위치에 생성된다.

```
$ ./Espresso -i [입력파일] -o [출력파일] -j
```

```
[예] $ ./Espresso -i input.txt -o json.txt -j
```

출력파일은 JSON형식의 결과 파일이 출력된다. [-j] 옵션 없이 실행시키면, CoNLL 2008¹형식의 결과 파일이 출력된다. ([예] \$./Espresso -i input.txt -o json.txt)

1 <http://catalog.ldc.upenn.edu/LDC2006T03>

제 3 장 소프트웨어 기능

3.1 프로그램 기능

Espresso Tool은 하나의 도구로 여러 가지 자연어처리 기능을 수행하는 도구로 현재 형태소 분석 및 품사 태깅, 구문 분석, 개체명 인식 및 분류, 의미역 인식 및 분류, 상호참조 해결의 기능을 수행한다. 한 행이 하나의 문장인 말뭉치를 입력으로 넣고 실행하면 각각의 모델의 결과가 출력 옵션에 따라 JSON 또는 CoNLL 2008형식으로 출력된다. 출력된 문서는 입력된 문서의 문장을 분석하여 형태소 품사 태그, 구문 태그, 개체명 태그, 의미역 논항 태그, 상호참조 정보가 레이블링 되어 있다. (단, 상호참조 정보는 JSON출력 형식을 지정한 경우에만 출력.)

3.2 프로그램 기능 제약

본 도구는 UTF-8 인코딩 문서만이 분석 가능하게 설정되어 있다. 컴파일이 가능하다면 대상 인코딩을 EUC-KR로 변경할 수 있다. 본 도구는 관리자 권한 없이 설치 및 실행이 가능하도록 하기 위하여, CRF++ 라이브러리와 Boost 라이브러리가 ./lib/에 각각 설치되어 있다고 가정한다. 따라서 라이브러리가 이미 기기에 설치되어 있다고 해도 Makefile의 경로를 수정하지 않는 한 링크하지 않는다.

제 4 장 기타

4.1 처리 속도 : 초당 100문장(3.00Ghz)

4.2 메모리 사용 : 1,500MB

4.3 시스템 성능

- 형태소 분석 및 형태소 품사 태그 부착 : 96%
- 구문 분석 : 81%
- 개체명 인식 및 분류 : 89% (sport domain)
- 의미역 인식 및 분류 : 65%
- 상호참조 해결(생략 포함 시) : 60%

국립국어원

KOMORAN 3.0

신준수, 이승원

【SHINEWARE】

차 례

제 1 장 소프트웨어 소개	152
1.1 KOMORAN 3.0	152
1.2 소프트웨어 사용 환경	152
1.3 소프트웨어 특징	152
제 2 장 소프트웨어 설치 및 실행	153
2.1 소프트웨어 설치 방법	153
2.2 소프트웨어 파일 구조	153
2.3 소프트웨어 실행 방법	154
2.3.1 학습 방법	154
2.3.2 분석 방법	154
제 3 장 소프트웨어 기능	155
3.1 프로그램 기능	155
3.2 프로그램 기능 제약	155
제 4 장 기타	155
4.1 처리속도	155
4.2 메모리 사용	156
4.3 분석 기능	156

제 1 장 소프트웨어 소개

1.1 KOMORAN 3.0

KOMORAN 3.0은 shineware에서 개발한 한국어 형태소 분석기로서 Korean Morphological Analyzer라는 뜻을 담고 있다. 자연어처리의 기본 엔진이 되는 형태소 분석기를 Apache license2.0 의 오픈 소스로 공개함에 따라 형태소 분석기를 필요로 하는 다양한 분야에서 폭넓게 사용할 수 있도록 하는데 목적이 있다.

1.2 소프트웨어 사용 환경

- OS : JVM이 지원가능한 모든 UTF-8 기반의 운영체제 지원
- Memory : 256 MB 이상
- 저장공간 : 10MB 이상
- 기타 : git 클라이언트 필요 (<https://git-scm.com/>)

1.3 소프트웨어 특징

KOMORAN 3.0은 github에 apache license 2.0으로 공개되어 있으며 이전 KOMORAN 버전들이 갖고 있는 높은 분석 성능, 쉬운 사전 관리 등의 장점을 그대로 계승하였다.

또한 속도 개선을 위해서 기존에 사용하던 자료구조인 TRIE를 Aho-corasick TRIE로 변경함에 따라 분석 속도가 2배 이상 빨라졌다.

기존에는 제공되지 않았던 학습 모듈이 제공됨에 따라 사용자가 직접 KOMORAN 3.0을 학습 시킬 수 있음은 물론이고 사용하고자 하는 도메인 (쇼핑몰, 뉴스, 블로그 등)에 적합하게 튜닝도 가능하다.

검색엔진에서의 사용성을 높이기 위해서 입력된 텍스트에 대한 형태소 분석 결과 뿐만 아니라 문장 내 형태소의 위치 정보를 제공함에 따라 검색 엔진에서 다양한 방법으로 활용 될 수 있다는 장점이 있다.

제 2 장 소프트웨어 설치 및 실행

2.1 소프트웨어 설치 방법

KOMORAN 3.0은 실행가능한 .jar 형태가 제공되며 별도의 설치를 필요로 하지 않는다.

KOMORAN 3.0의 소스코드는 github에 gradle 프로젝트 형태로 공개되어 있으며 git 클라이언트를 통해 clone 받은 후 gradle 명령어를 이용하여 eclipse 프로젝트를 생성 할 수 있다.

단, 설치 대상 PC에 인터넷이 연결되어 있어야 한다.

```
git clone https://github.com/shin285/KOMORAN.git
cd KOMORAN
./gradlew eclipse
```

eclipse 프로젝트 생성 후에는 소스 레벨에서 사용자가 원하는 형태로 실행이 가능하다.

제공되는 실행가능한 .jar는 kr.co.shineware.nlp.komoran.run 패키지의 KomoranConsoleRunner 클래스를 eclipse의 Runnable Jar 명령어를 통해 export 시킨 것이다.

2.2 소프트웨어 파일 구조

2.2.1 주요 파일 설명

- 실행 파일

gnrsys_komoran.jar

- 형태소 분석용 데이터 파일

models_full/ : 형태소 분석을 위한 데이터 폴더

user_data/fwd.user : 기분석 사전

user_data/dic.user : 사용자 사전

• 학습용 데이터 파일

학습용 데이터 파일은 사용자가 직접 편집 가능

corpus_build/dic.irregular : 학습 시 사용되는 불규칙 사전

corpus_build/dic.word : 학습 시 사용되는 단어 사전

corpus_build/grammar.in : 학습 시 사용되는 문법 사전

user_data/wiki.titles : 학습 시 선택적으로 사용되는 외부 사전 (option)

• 기타 파일

in.txt : 형태소 분석기 실행 테스트를 위한 샘플 입력 파일

2.3 소프트웨어 실행 방법

2.3.1 학습 방법

제공된 gnrsys_komoran.jar

파일을 통해 학습 데이터로부터 학습 모델을 생성할 수 있다.

학습 시에는 불규칙 사전(dic.irregular), 단어 사전(dic.word), 문법 사전(grammar.in)이 포함된 폴더명(corpus_build)과 학습 결과가 저장될 폴더명(model_test)이 필요하며 선택적으로 위키피디아 타이틀, 지명, 인명 등과 같은 외부 사전(user_data/wiki.titles)을 학습시킬 수 있다.

```
java -jar gnrsys_komoran.jar -train &학습 데이터가 포함된 폴더(corpus_build)& -model  
&학습 모델이 저장될 폴더명(model_test)& [-externalDic &학습 시 추가시킬 사전"]
```

학습이 완료되면 model_test 폴더가 생성되고 그 안에 형태소 분석에 필요한 파일들(학습 모델 파일)이 생성된 것을 확인할 수 있다.

2.3.2 분석 방법

제공된 gnrsys_komoran.jar

파일과 학습 모델을 통해 입력 파일에 대한 형태소 분석 결과를 제공한다.

분석 시에는 학습 모델 파일들이 포함된 폴더명(models_full. 단, 2.3.1을 수행하였다면 model_test도 가능)과 분석 대상 파일(in.txt), 분석 결과가 저장될 파일명(out.txt)를 필요로 한다.

분석 시 선택적으로 사용자 사전(user_data/dic.user)과 기본 분석 사전(user_data/fwd.user)을 사용할 수 있다.

```
java -jar gnrsys_komoran.jar -model &학습 모델이 포함된 폴더명(models_full)&
-in &분석 대상 파일(in.txt)& -out &분석 결과 파일(out.txt)" [-userDic &사용자
사전 파일(user_data/dic.user)&] [-fwd &기분석 사전 파일(user_data/fwd.user)&]
```

분석이 완료되면 분석 결과가 out.txt에 저장된다.

제 3 장 소프트웨어 기능

3.1 프로그램 기능

KOMORAN 3.0은 사전 관리 및 학습 데이터 구축을 통해 두 어절 이상의 어휘를 하나의 품사로 처리가 가능하다. 예를 들어 사용자 사전에 “바람과 함께 사라지다”를 NNP 품사로 등록한 경우에는 “바람과 함께 사라지다는 평점이 좋던데?”라는 입력 문장이 들어오면 “바람과 함께 사라지다/NNP + 는/JX + 평점/NNG + 이/JKS + 좋/VA + 던데/EF + ?/EF”와 같이 분석된다.

사용자 사전이 아닌 학습 데이터에 multi-word로 구성된 어휘가 있다면 사용자 사전과같이 지속적으로 같은 결과를 내는 것이 아닌 문맥에 따라 적합한 결과를 낼 수 있다.

3.2 프로그램 기능 제약

gnrsys_komoran.jar는 UTF-8 환경만 지원
eclipse 프로젝트 import 시 JDK 1.8 이상 필요

제 4 장 기타

4.1 처리 속도

초당 1.0MB 분석
초당 약 5만 어절 분석
초당 약 1만 문장 분석

4.2 메모리 사용

학습 데이터에 따라 상이 (권장 128MB 이상)

4.3 분석 성능

어절 단위 정확률 93.60%

형태소 단위 정확률 96.12%

조직 위원장

차 정 원 창원대학교

조직 위원

김 경 선 다이퀘스트

이 청 재 SKT

정 호 성 국립국어원

나 승 훈 전북대학교

정 상 근 SKT

최 정 도 국립국어원

심사 위원장

강 승 식 국민대학교

심사 위원

강 현 규 건국대학교

김 선 영 국립국어원

이 청 재 SKT

차 정 원 창원대학교

김 경 선 다이퀘스트

나 승 훈 전북대학교

정 상 근 SKT

최 정 도 국립국어원

2016 국어 정보 처리 시스템 경진 대회 - 개체명 인식 시스템 -

발 행 인 송 철 의

발 행 처 국립국어원

서울특별시 강서구 금남화로 154

등록번호 국립국어원 2016-01-05

인 쇄 일 2016년 10월 5일

발 행 일 2016년 10월 7일

주 관 한국정보과학회 언어공학연구회

서울시 서초구 방배3동 984-1 머리재빌딩 401호

주 최 국립국어원

