



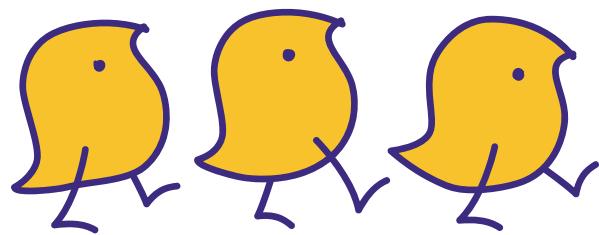
LangCon 2019



khaiii: Kakao Hangul Analyzer III

임재수 / 카카오

1. 형태소 분석기란?
2. 기존의 방법들
3. 세종 코퍼스
4. 문제의 정의
5. 모델 아키텍처
6. 정확도와 속도
7. 사용자 사전
8. 사용 및 참여



형태소 분석기란?

- 형태소와 어절
- 왜 형태소 분석을 해야 하나요?
- 형태소 분석의 모호성
- 한국어 형태소 분석의 어려움

형태소와 어절

• 형태소란?

일정한 의미가 있는 가장 작은 말의 단위로 발화체 내에서 따로 떼어낼 수 있는 것을 말한다. 즉, **더 분석하면 뜻이 없어지는 말의 단위**이다. 음소와 마찬가지로 형태소는 추상적인 실체이며 발화에서 다양한 형태로 실현될 수 있다.

-- 위키백과

- 어절이란?

한국어에서 문장을 구성하고 있는 도막도막의 성분으로서 발음의 기본이 되는 단위이다. 하나의 강세 음절이 홀로 또는 앞뒤에 하나 이상의 무강세 음절을 거느리고 나타나는 단위이다. 어절은 대개 띄어쓰기의 단위와 일치하며, 체언에 조사가 붙거나, 어미가 어간에 붙어서 이루어지기도 한다.

-- 위키백과

왜 형태소 분석을 해야 하나요?

Language
Conference
2019

- 영어의 경우,

- 원문

You're | my | sunshine, | my | only | sunshine.

- 단어(Word) 혹은 토큰(Token)

You | 're | my | sunshine | , | my | only | sunshine | .

왜 형태소 분석을 해야 하나요?

Language
Conference
2019

- 한국어의 경우,

- 원문

문재인대통령은 김정은국방위원장과 정상회담을 가졌다.

- 어절??

문재인대통령은 | 김정은국방위원장과 | 정상회담을 | 가졌다 | .

왜 형태소 분석을 해야 하나요?

Language
Conference
2019

- 한국어의 경우,

- 원문

문재인 대통령은 김정은 국방위원장과 정상회담을 가졌다.

- 형태소!!

문재인	대통령	은	김정은	국방	위원장	과
정상	회담	을	가지(다)	었	다	.

- 예문

“나는 오늘 하늘을 나는 새를 보았다.”

- 분석 결과

- 첫 번째 “나는”

나(대명사) + 는(조사)

- 두 번째 “나는”

날다(동사) + 는(관형형어미)

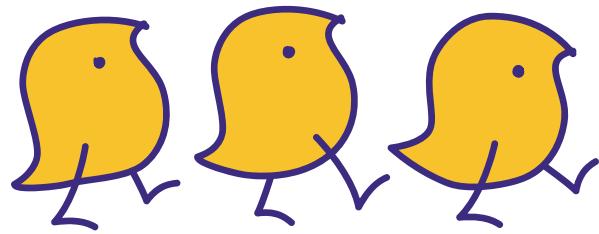
- 엄밀히 말하면,
 - 형태소 분석

“어절 내에서 다양하게 나뉘어지는 분석 후보들을,
모두 나열하여 분석하는 것”
 - 품사 부착

“모호한 형태소 분석 후보들 중에서,
문장 내에서 알맞은 형태소와 그 품사를 결정하는 것”

한국어 형태소 분석의 어려움

- 두 가지 중의성
 - Segmentation 중의성
 - 영어(X), 중국어(O), 일본어(O)
 - 품사 중의성
- 띄어쓰기
 - 사용자는 띄어쓰기 실수를 빈번하게 일으킴 → 믿기도 믿지 못하기도 힘듦
- 원형 복원
 - 쉬워 → 쉽다 + 어 vs 비워 → 비우다 + 어
- 복합 명사
 - 띄어 써도 붙여 써도 맞음 → 사용자마다 다름
 - 예: 대학생선교회



- 규칙 기반
- 통계 기반

기존의 방법들

규칙 기반

• 규칙(Rule) 기반 방법

- “나는”

- 나(대명사) + 는(조사) → O
- 날다(동사) + 는(어미) → O

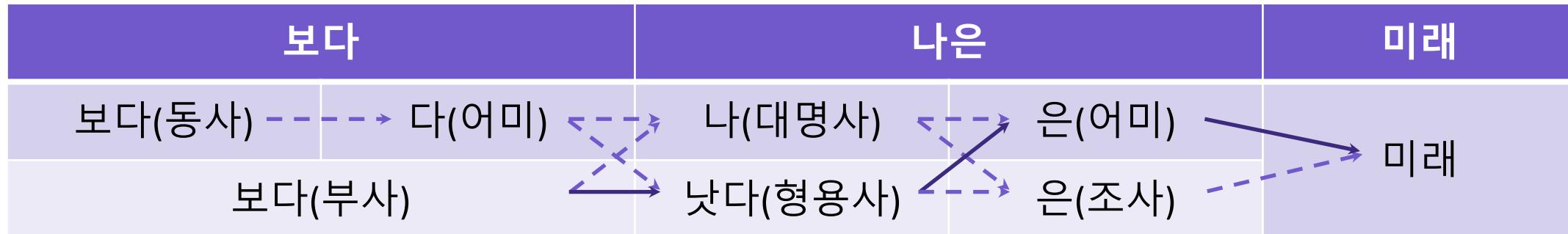
- 보조사 “는”은 받침이 없는 체언과 결합한다. → 규칙

- “나은”

- 나(대명사) + 은(조사) → X
- 낫다(형용사) + 은(어미) → O

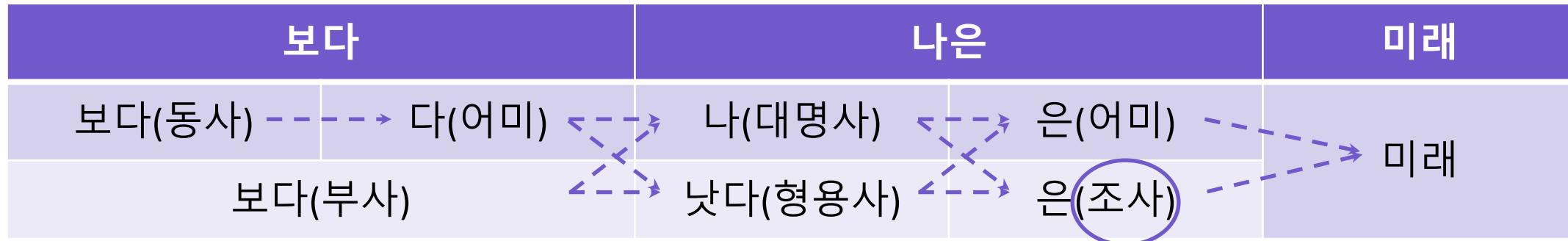
- 보조사 “은”은 받침이 있는 체언과 결합한다. → 규칙

- Hidden Markov Model(HMM)

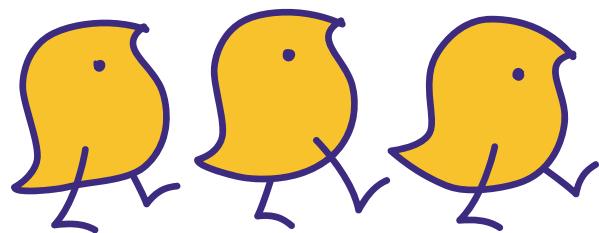


- 발생(emission) 확률
 - $P(\text{“나”} | \text{대명사})$, $P(\text{“은”} | \text{어미})$, $P(\text{“낫다”} | \text{형용사})$, $P(\text{“은”} | \text{조사})$
- 전이(transition) 확률
 - $P(\text{어미} | \text{대명사})$, $P(\text{조사} | \text{대명사})$, $P(\text{어미} | \text{형용사})$, $P(\text{조사} | \text{형용사})$
- Viterbi 알고리즘
 - 가능한 모든 경로 중 확률을 최대로 하는 경로를 찾는 동적 프로그래밍 알고리즘

- Conditional Random Fields(CRFs)



- 자질 함수(Feature Function)
 - 각 후보 형태소들에 대해 자질 함수를 적용하여 추출된 자질에 상태(품사) 가중치를 적용
 - 예를 들면,
 - “온” 형태소의 상태(품사) “조사”에 대해,
 - 이전 형태소가 받침이 있는지 여부(자질)에 따라 가중치를 부여
- 장점
 - 규칙 기반에서 적용했던 언어학적 지식을 자질로 표현할 수 있다!

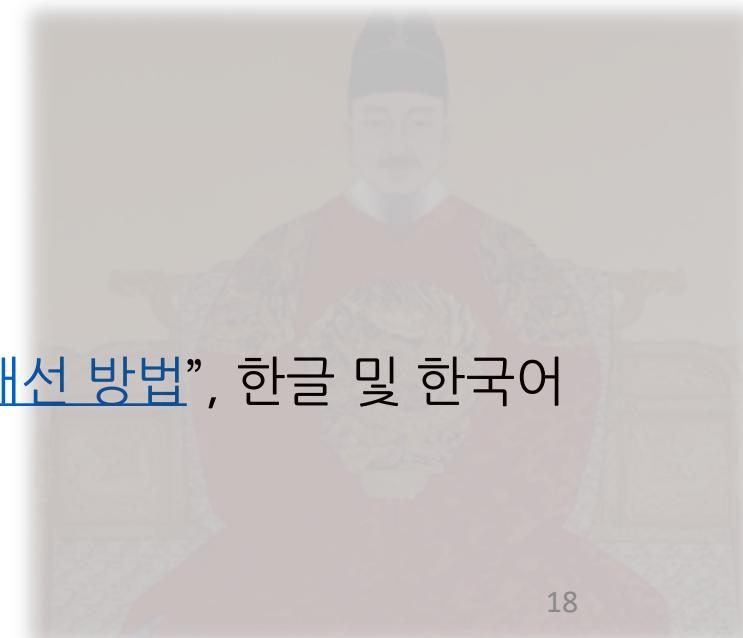


세종 코퍼스

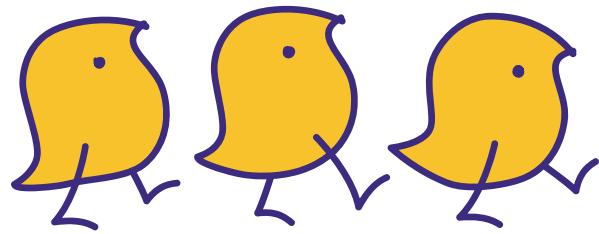
- 왜 세종 코퍼스인가요?
- 코퍼스의 오류와 수정
- 데이터 공개에 관하여

- 가장 큰 코퍼스
 - 천만 어절 이상
 - 1997년부터 2007년까지 10년간 구축
- 가장 많은 연구
 - 발표된 논문이 가장 많음
 - 오픈소스 프로젝트들이 가장 많이 사용
- 사실상 표준
 - 품사 집합
 - 코퍼스 구축 지침

- 세종 코퍼스의 오류
 - 방대한 양에 걸맞게 오류도 방대함
 - 연구자마다 수정한 각자의 수정본이 존재
 - 저작권의 이유로 수정본의 공개가 불가
- 문종 프로젝트
 - khaiii 학습을 위해 카카오 내부에서 진행한 프로젝트
 - 스크립트와 수작업에 의해 수정 (패치 ♦ 127만 건)
 - 약 4,500문장의 추가 코퍼스
 - 참고 논문
 - 한경은 외 2인, “[공개와 협업을 통한 세종 형태 분석 말뭉치 오류 개선 방법](#)”, 한글 및 한국어 정보처리 학술대회, 2017



- 원문의 저작권
 - 코퍼스의 annotation과는 별개로 원문인 뉴스 기사나 문학 작품 등은 저작권이 있는 콘텐츠
 - 저희가 제작한 패치♣도 원문을 유추할 수 있다는 이유로 공개가 불가
 - 스크립트 공개
 - 패치를 생성하거나 적용하는 스크립트
 - 프로그램에 의해 자동으로 수정이 가능한 몇 가지 오류를 수정하는 스크립트
 - 참고 링크
 - khaiii GitHub 저장소 내 위키 페이지 중 [문종 프로젝트](#) 문서
- ♣ 패치: 세종 코퍼스 원본과 오류를 수정한 수정본 사이에 다른 부분만을 추출한 것으로 원본과 패치를 이용해 수정본을 생성할 수 있음



문제의 정의

- RNN 방법과 문제점
- 음절과 형태소의 정렬
- 분류 문제로 정의

- Bi-LSTM with Attention

원문	심사숙고 했겠지만,
분석 결과	심사/NNG + 숙고/NNG 하/VX + 였/EP + 겠/EP + 지만/EC + ,/SP
입력(음절)	심 사 숙 고 <SPC> 했 겠 지 만 , <EOS>
출력(음절과 태그)	심/I-NNG 사/I-NNG 숙/B-NNG 고/I-NNG 하/I-VX 였/I-EP 겠/B-EP 지/I-EC 만/I-EC ,/I-SP <EOS>

- seq2seq 번역 모델과 유사
- 97% 이상의 높은 정확도

- 문제점

- 순차적 계산, 많은 파라미터 → 병렬 처리 불가, 많은 계산량 → 느림!
- 출력과 입력의 정렬이 힘듦 → 검색엔진에서 색인어의 위치 문제

음절과 형태소의 정렬

• 음절 기반 방법

- 심광섭, “[음절 단위의 한국어 품사 태깅에서 원형 복원](#)”, 소프트웨어 및 응용 제40권 제3호, 2013

음절(입력)	분석 결과	태그(출력)	태그 구분
심	심/I-NNG	I-NNG	단순
사	사/I-NNG	I-NNG	단순
숙	숙/B-NNG	B-NNG	단순
고	고/I-NNG	I-NNG	단순
했	하/I-VX, 였/I-EP	I-VX:I-EP:0	복합
겠	겠/B-EP	B-EP	단순
지	지/I-EC	I-EC	단순
만	만/I-EC	I-EC	단순
,	,/I-SP	I-SP	단순

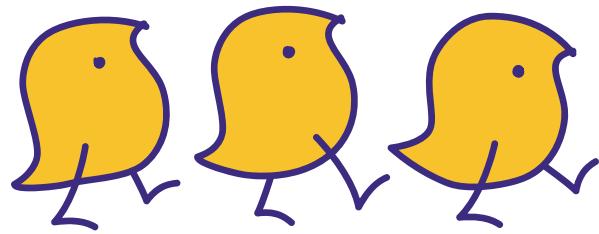
분류 문제로 정의

- 분류(Classification) 문제

- 각각의 입력 음절 대해, 출력 태그를 결정하는 문제
- 각 음절의 태그는 주변 정보(문맥)를 활용하여 독립적으로 판단
→ 병렬 처리가 가능

- 원형 복원

- 단순 태그 92개
- 복합 태그 400여 개
 - 입력 음절과 출력 태그를 key로 하는 원형 복원 사전
 - 예: “했/I-VX:I-EP:0” → “하/I-VX + 였/I-EP”



- 윈도우와 문맥
- CNN 모델 아키텍처

모델 아키텍처

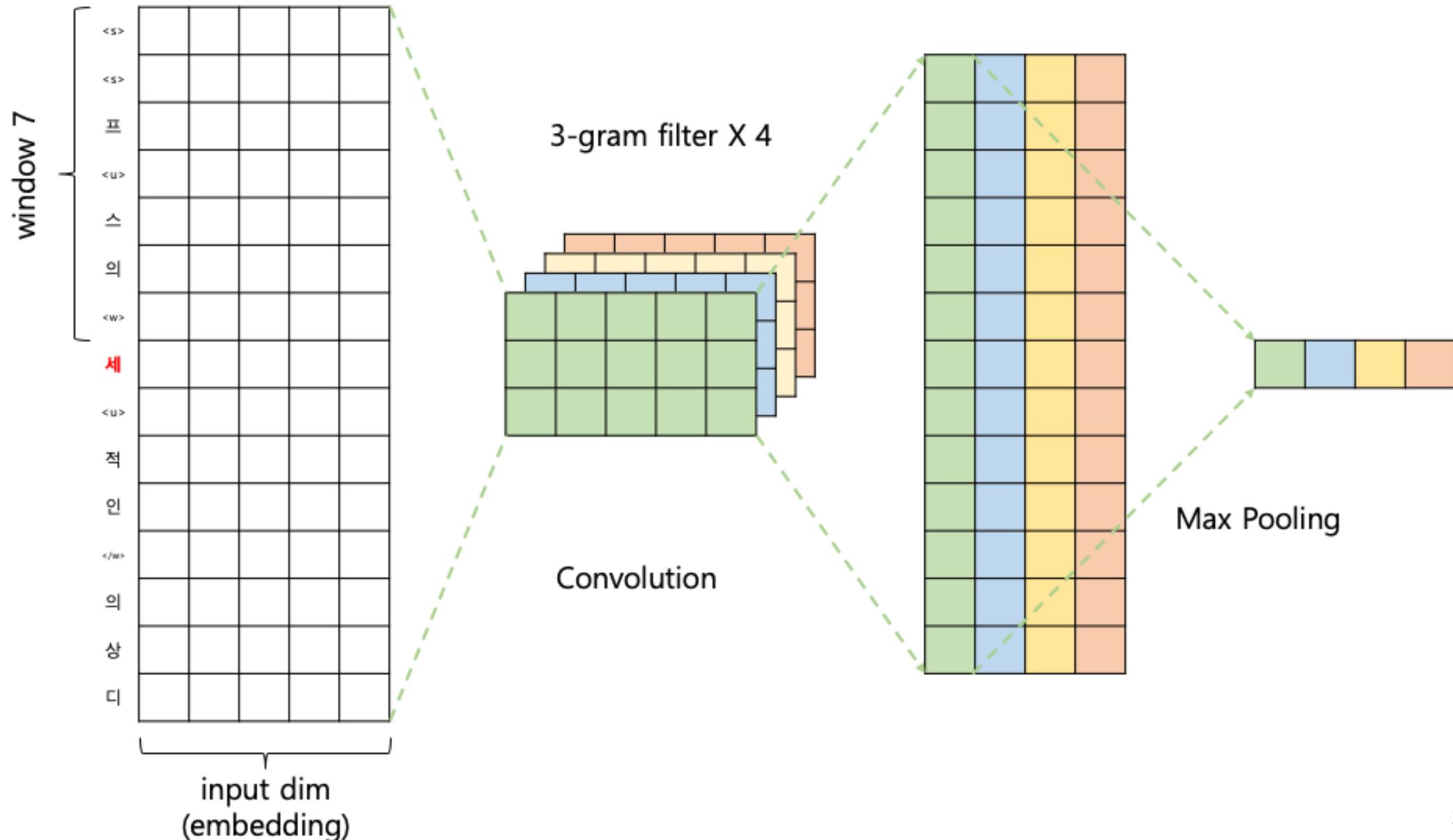
- “프랑스의 세계적인 의상 디자이너 엠마누엘 ...”

-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7
<s>	<s>	프	<u>	스	의	<w>	<u>세</u>	<u>	적	인	</w>	의	상	디

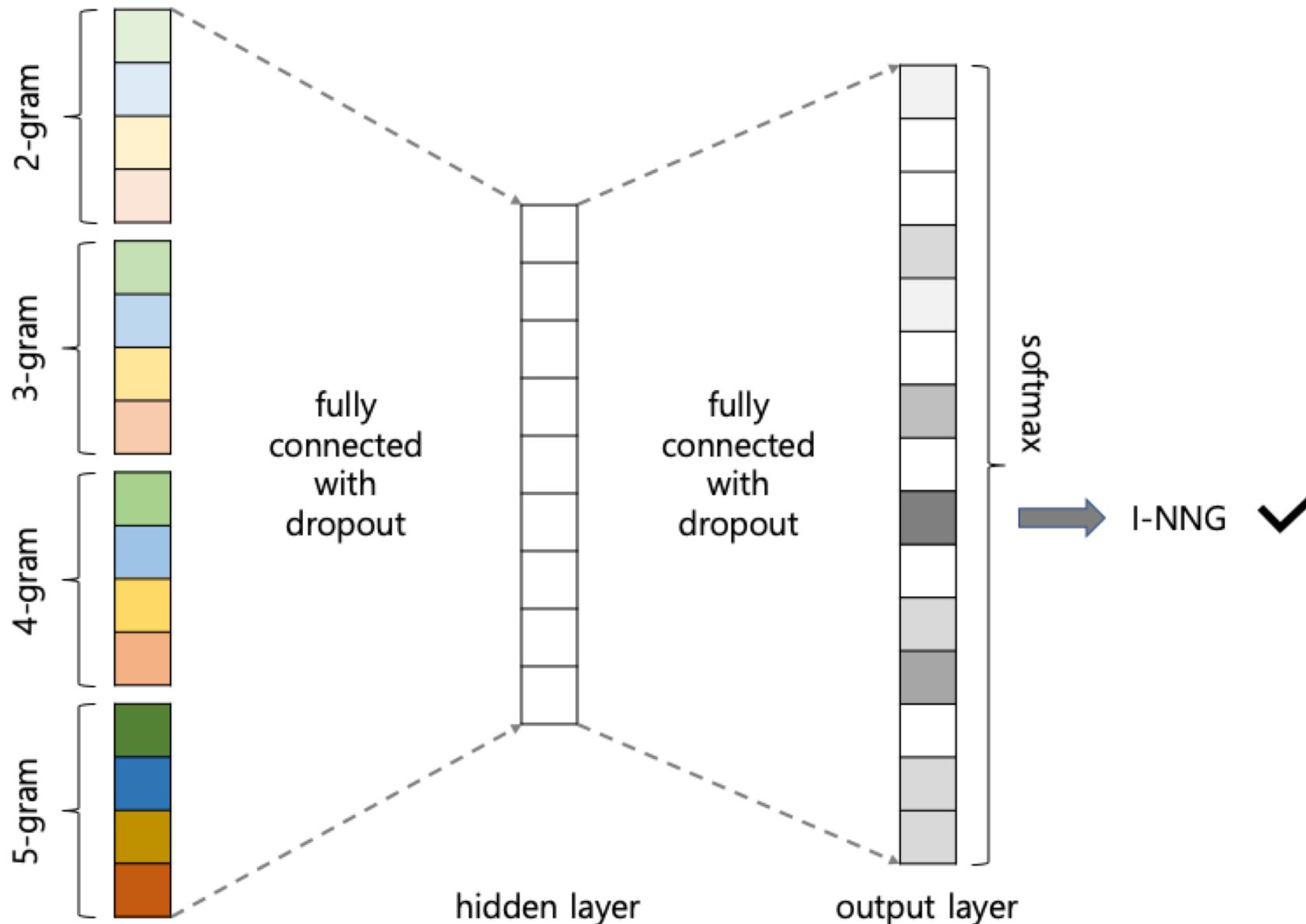
- 가상 음절

가상 음절	의미
<u>	Out of Vocabulary 음절
<w>	왼쪽 어절 경계 (공백)
</w>	오른쪽 어절 경계 (공백)
<s>	왼쪽 패딩 (임베딩, 학습)
</s>	오른쪽 패딩 (임베딩, 학습)

CNN 모델 아키텍처



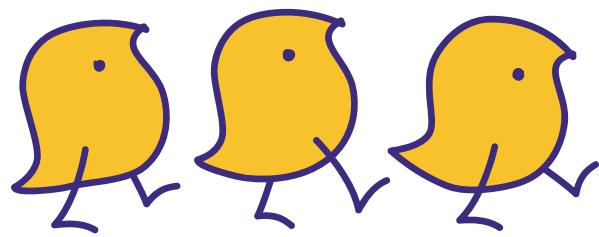
CNN 모델 아키텍처



CNN 모델 아키텍처

- 참고 논문

- Yoon Kim, “[Convolutional Neural Networks for Sentence Classification](#)”, EMNLP, 2014
- 아래와 같이 대입하면 됩니다.
 - 논문의 단어 → 음절
 - 논문의 문장 → 문맥

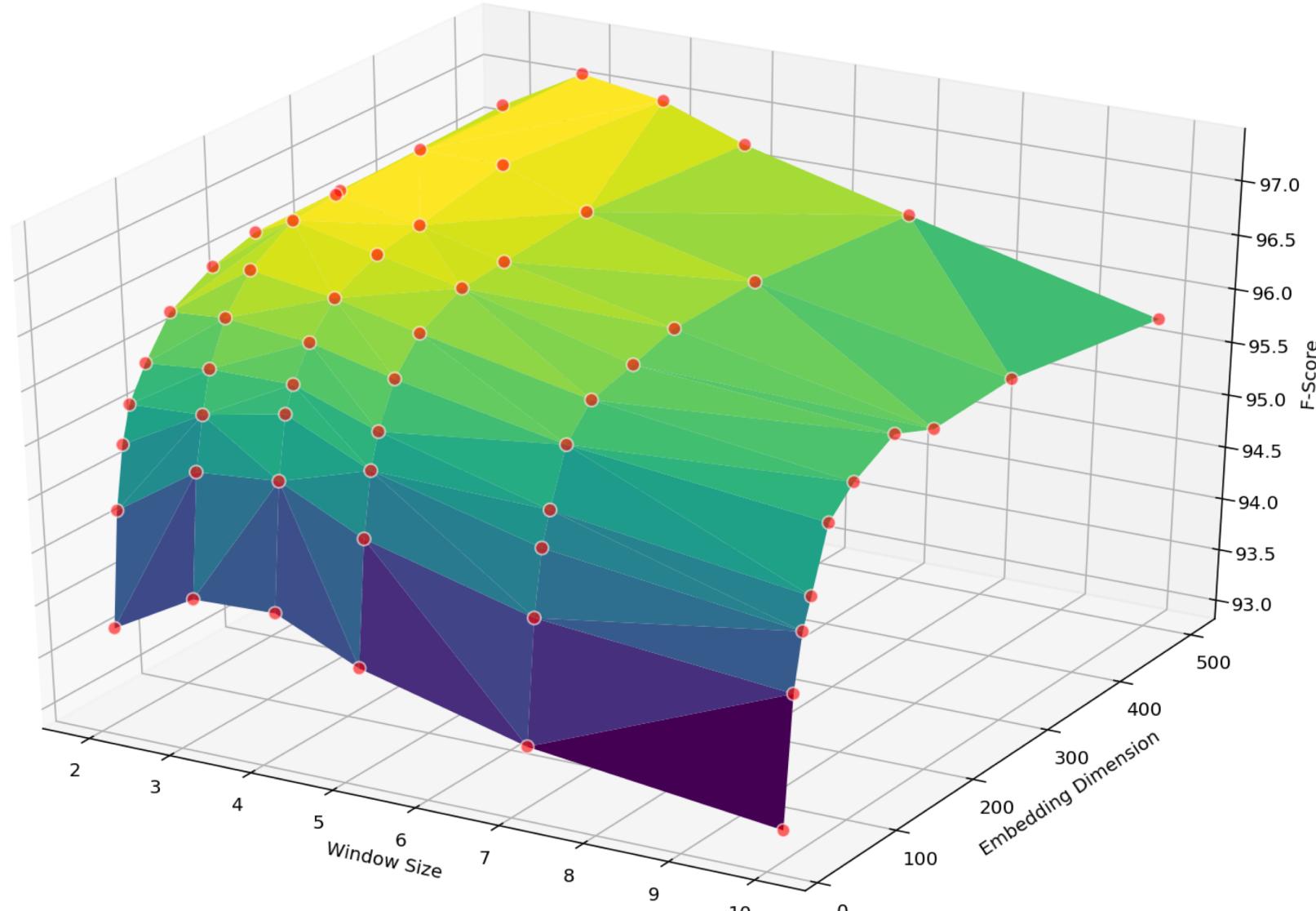


정확도와 속도

- 윈도우, 임베딩 크기에 따른 성능
- Base 모델과 Large 모델

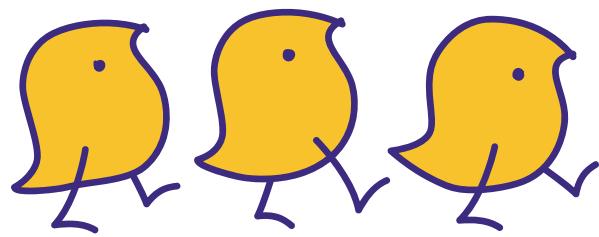
- 윈도우
 - 2, 3, 4, 5, 7, 10
- 임베딩
 - 20, 30, 40, 50, 70, 100, 150, 200, 300, 500
- Metric
 - F-Score: 정확도와 재현률의 조화평균

윈도우, 임베딩 크기에 따른 성능



Base 모델과 Large 모델

- Base 모델
 - 윈도우: 3, 임베딩: 30
 - 95% 이상이면서 가장 작은 모델
- Large 모델
 - 윈도우: 3, 임베딩 150
 - 97% 이상이면서 비교적 작은 모델
- 속도
 - 1만 문장 분석 속도 비교
 - Base: 10.5초 vs Large: 78.8초



- 기분석 사전
- 오분석 패치

사용자 사전

- 특징
 - 단일 어절에 대해 문맥에 상관없이 일괄적인 분석 결과를 갖는 경우에 사용
 - CNN 모델 적용 전에 동작
 - 적용된 경우 CNN 모델을 적용하지 않으므로 속도도 빨라짐
- 기술 예

입력 어절	분석 결과
이더리움*	이더리움/NNP

 - “이더리움”으로 시작하는 모든 어절에 적용
- 참고 링크
 - khaïii GitHub 저장소 내 위키 페이지 중 [기분석 사전](#) 문서

- 특징

- 여러 어절에 걸쳐서 충분한 문맥과 함께 오분석을 바로잡아야 할 경우에 사용
- CNN 모델 적용 후에 동작
- 속도는 거의 영향이 없음

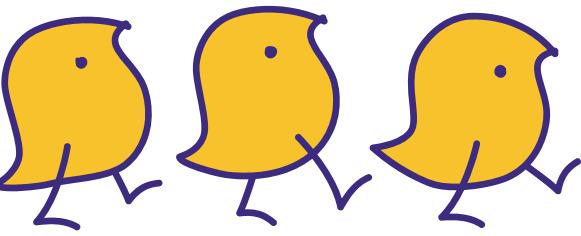
- 기술 예

입력 텍스트	오분석 결과	정분석 결과
이 다른 것	이/JKS + _ + 다/VA + 른/MM + _ + 것/NNB	이/JKS + _ + 다르/VA + 른/ETM + _ + 것/NNB

- CNN 모델의 결과가 “오분석 결과”와 같을 경우 “정분석 결과”로 수정

- 참고 링크

- khaiii GitHub 저장소 내 위키 페이지 중 [오분석 패치](#) 문서



사용 및 참여

- 설치 방법
- Python 코드 예제
- 오픈소스 참여

- Python 설치 방법

```
git clone https://github.com/kakao/khaiii  
cd khaiii  
mkdir build  
cd build  
cmake ..  
make package_python  
cd package_python  
pip install .
```

- 자세한 방법은

- khaiii GitHub 저장소 내 위키 페이지 중 [빌드 및 설치](#) 문서

Python 코드 예제

- 코드

```
from khaiii import KhaiiiApi
api = KhaiiiApi()
for word in api.analyze('안녕, 세상.'):
    print(word)
```

- 출력

```
안녕,   안녕/IC + ,/SP
세상.   세상/NNG + ./SF
```

오픈소스 참여

- 슬랙
 - 주소: <https://khaiii.slack.com>
 - 가입 요청: <https://join-khaiii.herokuapp.com>
 - develop 채널에서 같이 개발하실 분을 기다리고 있습니다!
- 개발자를 위한 가이드
 - khaiii GitHub 저장소 내 [위키 페이지](#) 참고



LangCon 2019



감사/NNG + 하/XSV + 봐니다/EF + ./SF

