



인공지능

Artificial Intelligence

07 자연어 처리



1 자연어 처리와 특성

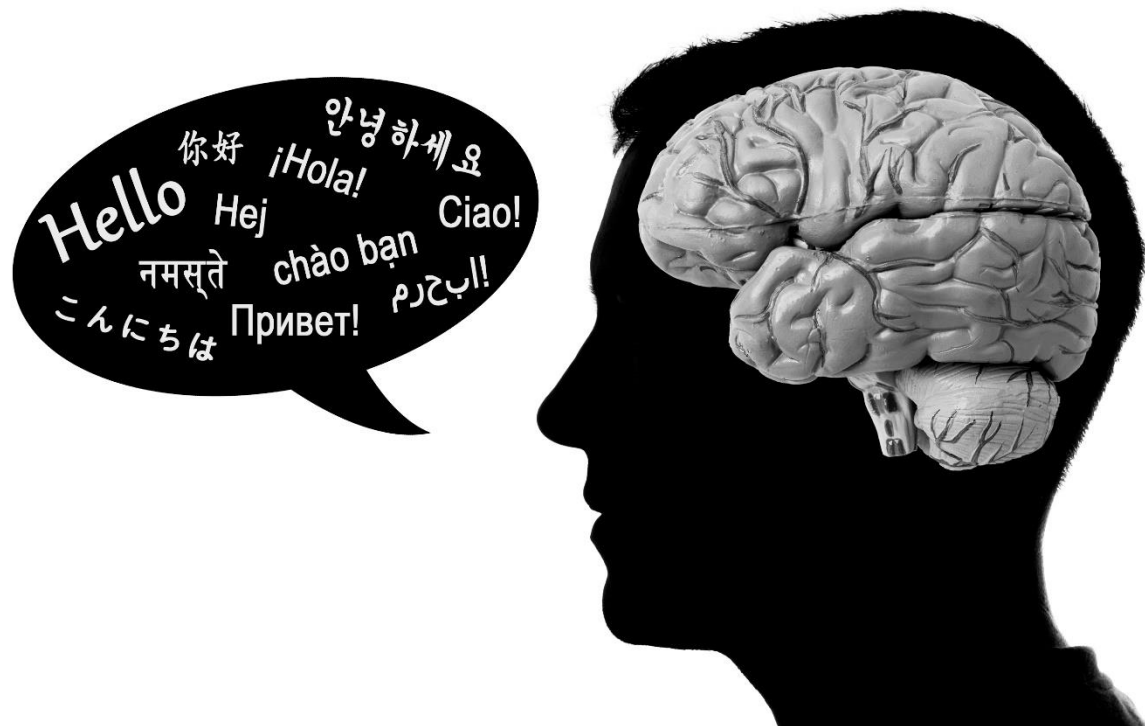
언어

- 인간이 사용하는 가장 자연스러운 정보전달 및 의사표현 수단
- 컴퓨터 인터페이스로서 필수적 요소
- **자연어** (自然語, natural language)
 - 특정 집단이 모국어로 사용하는 자연발생적인 언어
- **인공 언어**
 - 특정 목적을 위해 인위적으로 만들어진 언어
 - 자연언어에 비해 엄격한 문법구조
 - 형식 언어(formal language), 프로그래밍 언어, 에스페란토어



자연어의 특성

- 언어별 고유한 특성
- 띄어쓰기
 - 한글/영어 vs 일본어/중국어
- 활용(活用, conjugation) 변환의 정도
 - 한글: 동사, 형용사의 심한 활용
 - 영어: 동사의 비교적 규칙적인 활용



형태적 구조에 따른 언어 분류

고립어(孤立語, isolating language)

- 단어가 의미만 나타내고 형태의 변화가 없으며, 어순에 따라 문법적 관계 결정
- 중국어, 태국어, 베트남어

굴절어(屈折語, inflective language)

- 어근(語根, root)과 접사(接辭, 접두사/접미사)가 쉽게 분리되지 않는 형태
- 성, 수, 격, 시제, 법에 따라 단어가 변하는 언어
- 영어, 프랑스어, 라틴어

교착어(膠着語, agglutinating language)

- 어근에 문법형태소인 접사를 붙여 단어를 파생시키거나 문법적 관계를 나타내는 언어
- **한국어**, 일본어

포합어(抱合語, incorporative language)

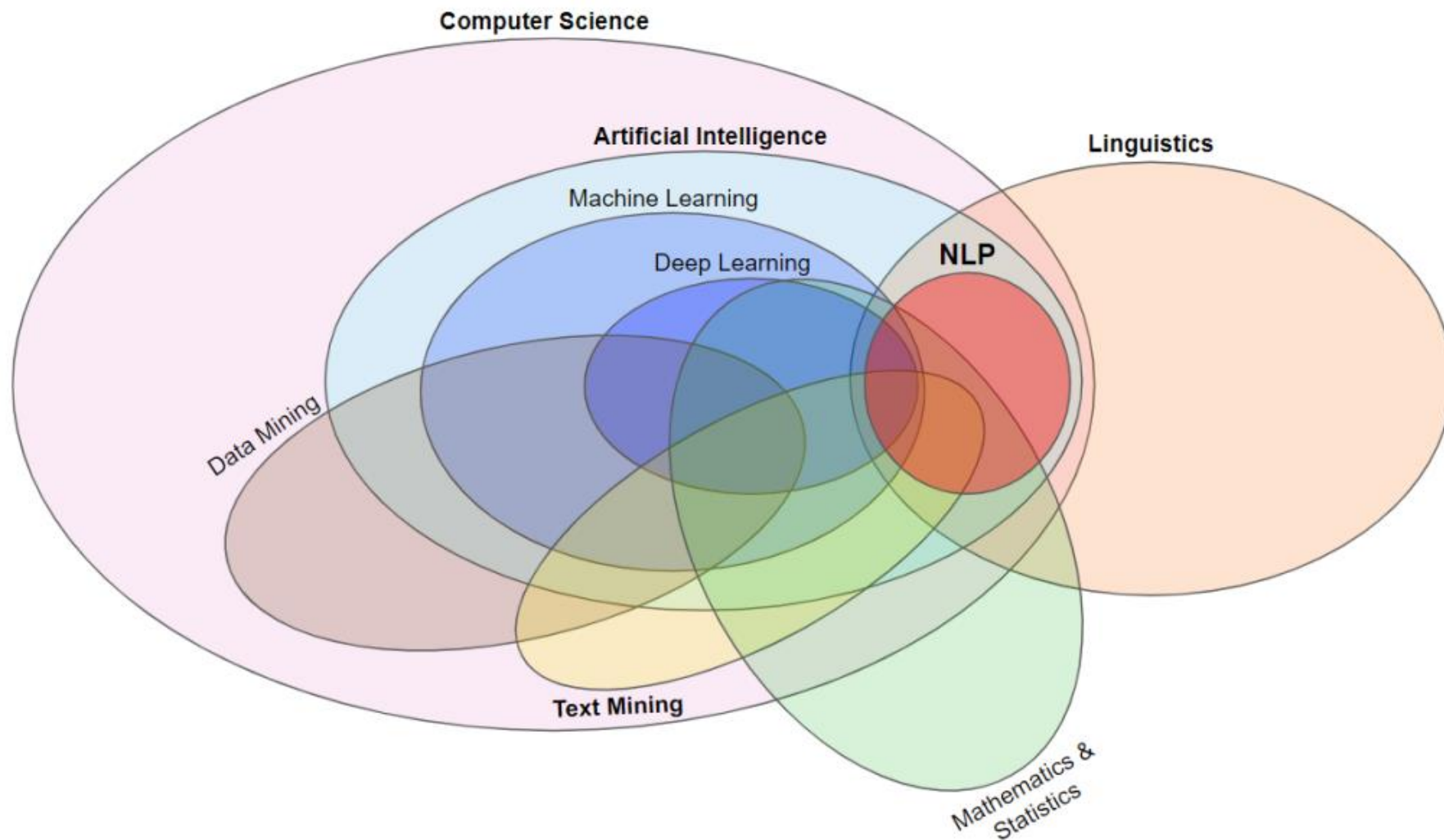
- 문장을 구성하는 요소가 서로 밀접하게 결합되어 마치 전체 문장이 하나의 단어를 이루는 것처럼 보이는 구조의 언어
- 에스키모어, 아메리카 인디언어, 호주 원주민어

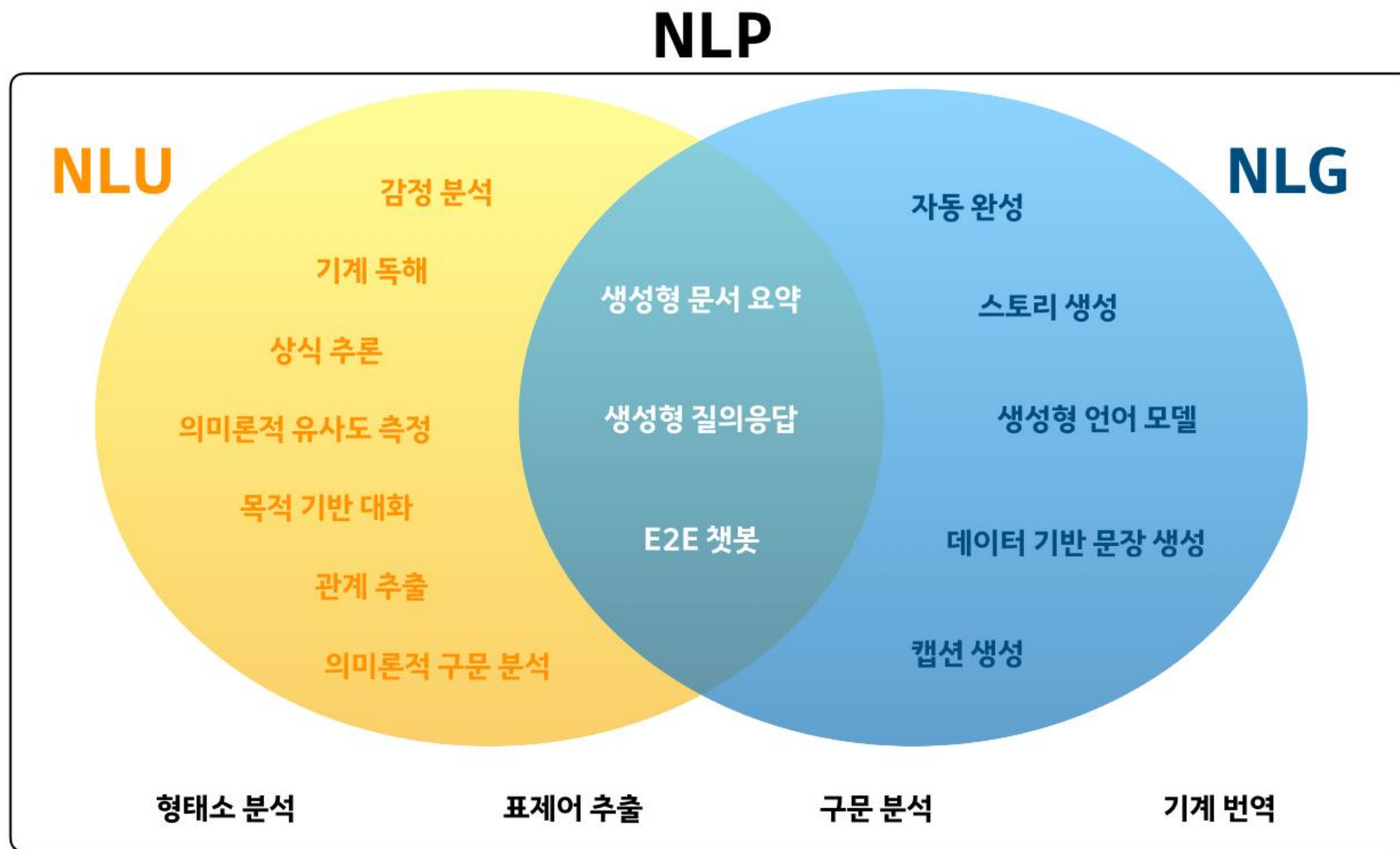
자연어 처리(Natural Language Processing)

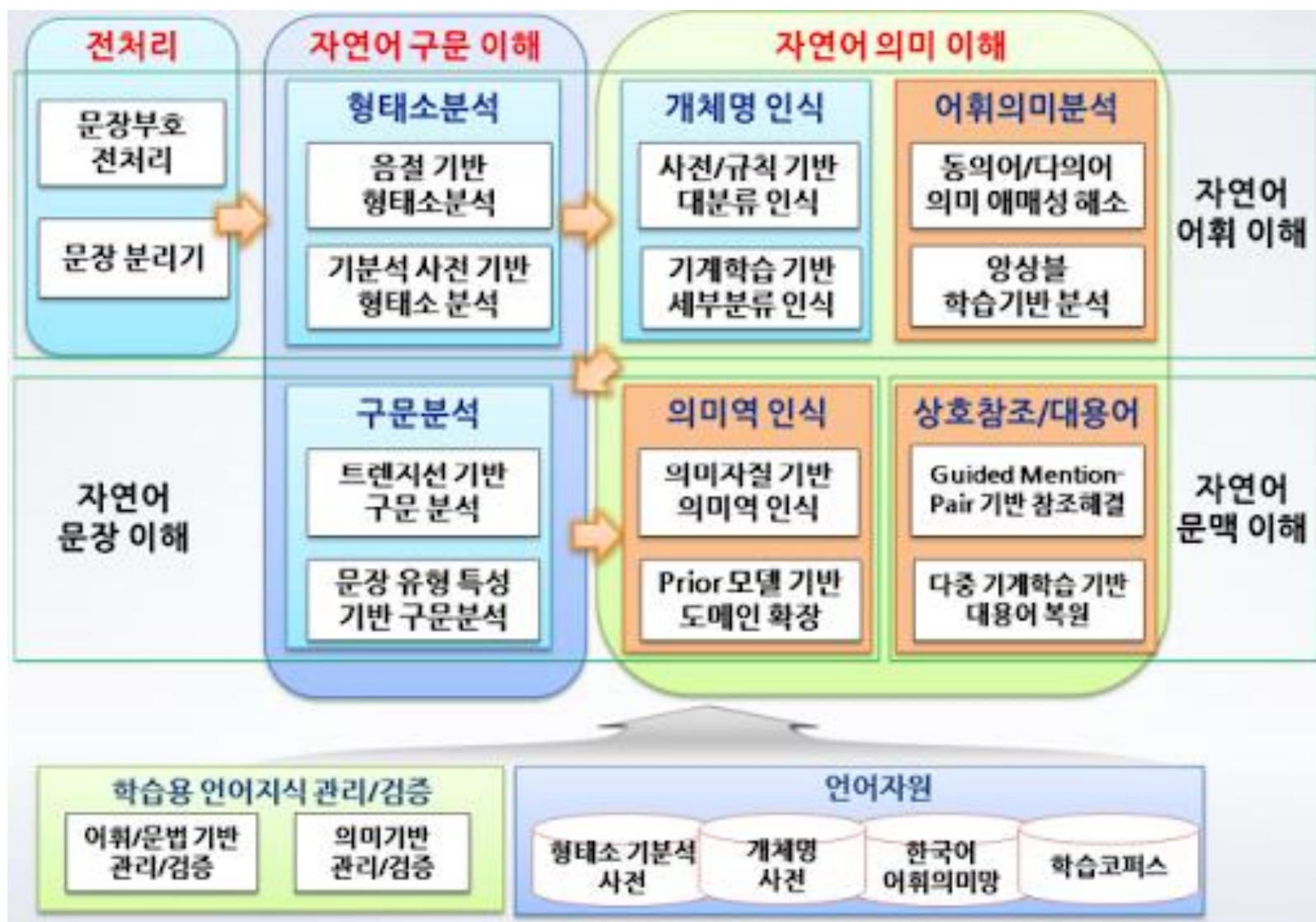
- 자연어는 일상 생활에서 사용하는 언어
- 자연어 처리는 자연어의 의미를 분석 처리하는 일
- 텍스트 분류, 감성 분석, 문서 요약, 번역, 질의 응답, 음성 인식, 챗봇, 콜센터, VOC 분석과 같은 응용



NLP 분류





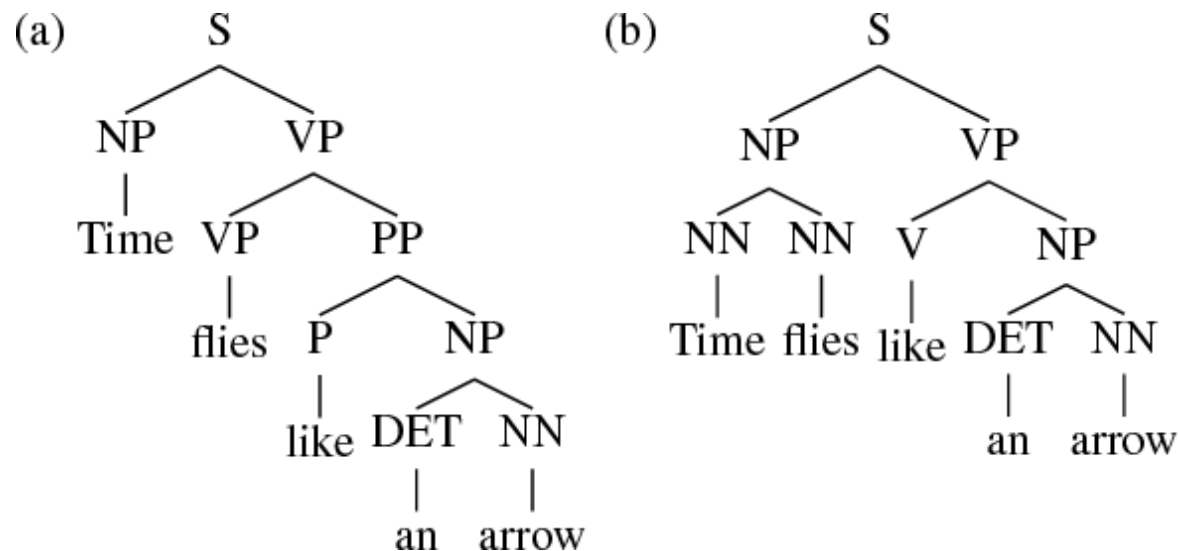


자연어 처리의 문제

- 동일한 표현이지만, 다양하게 해석될 수 있는 애매모호함(ambiguity)

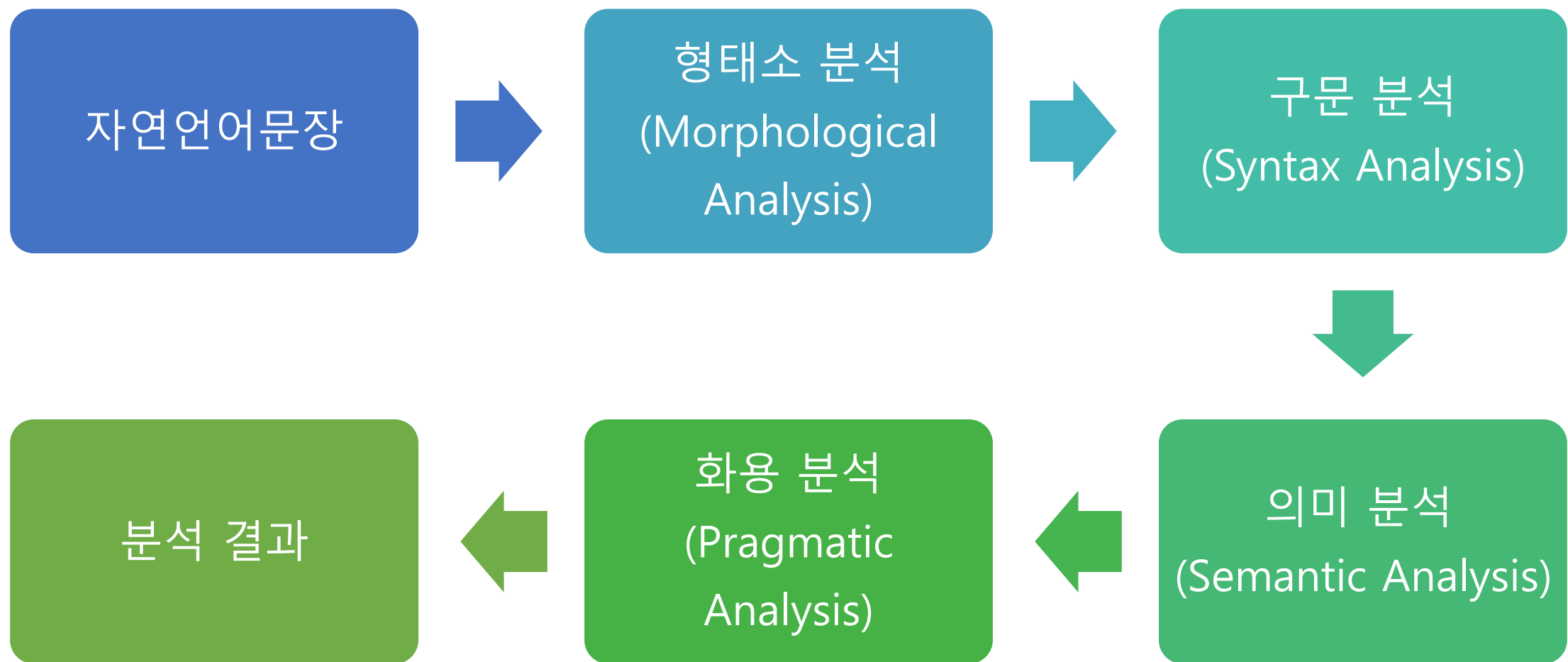


- 동일한 구문이지만, 2가지 이상의 구조로 분석될 수 있는 애매모호함



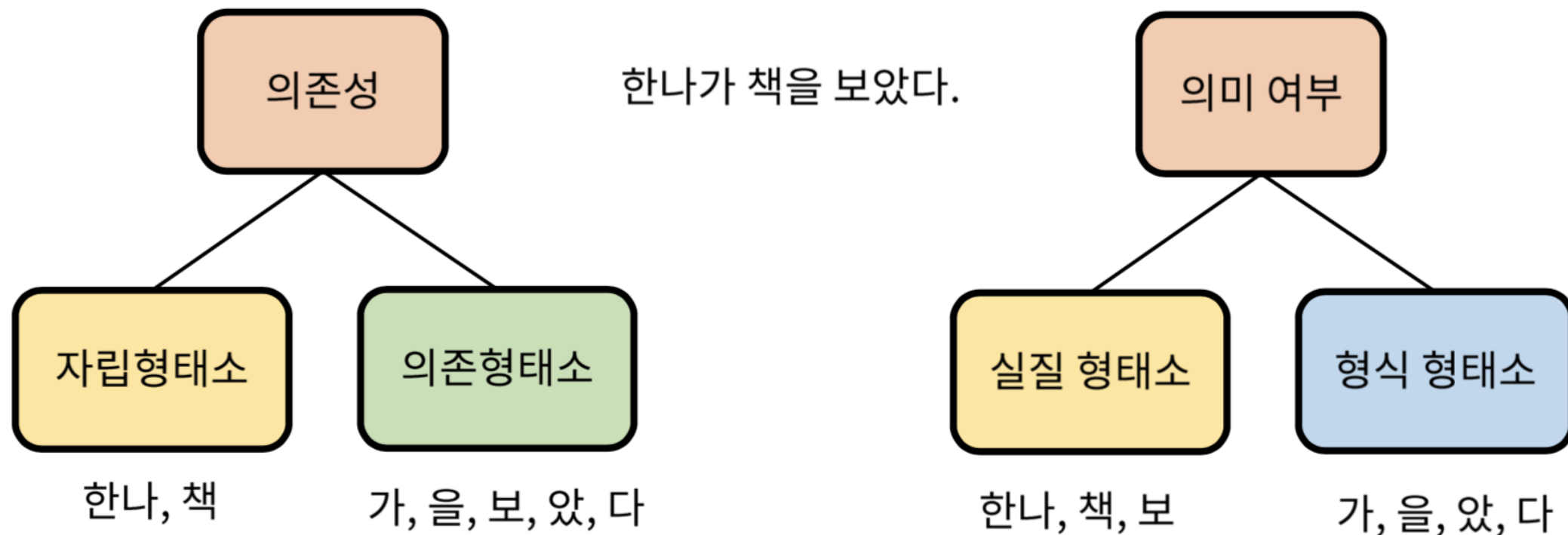
- S: Sentence
- NP(명사구), NN(명사), VP(용언구), PP(후치사구), P(서술형용사), DET(한정사)

자연어 처리 분석 단계



형태소 분석(morphological analysis)

- 입력된 문자열을 분석하여 형태소 단위로 분해하는 과정

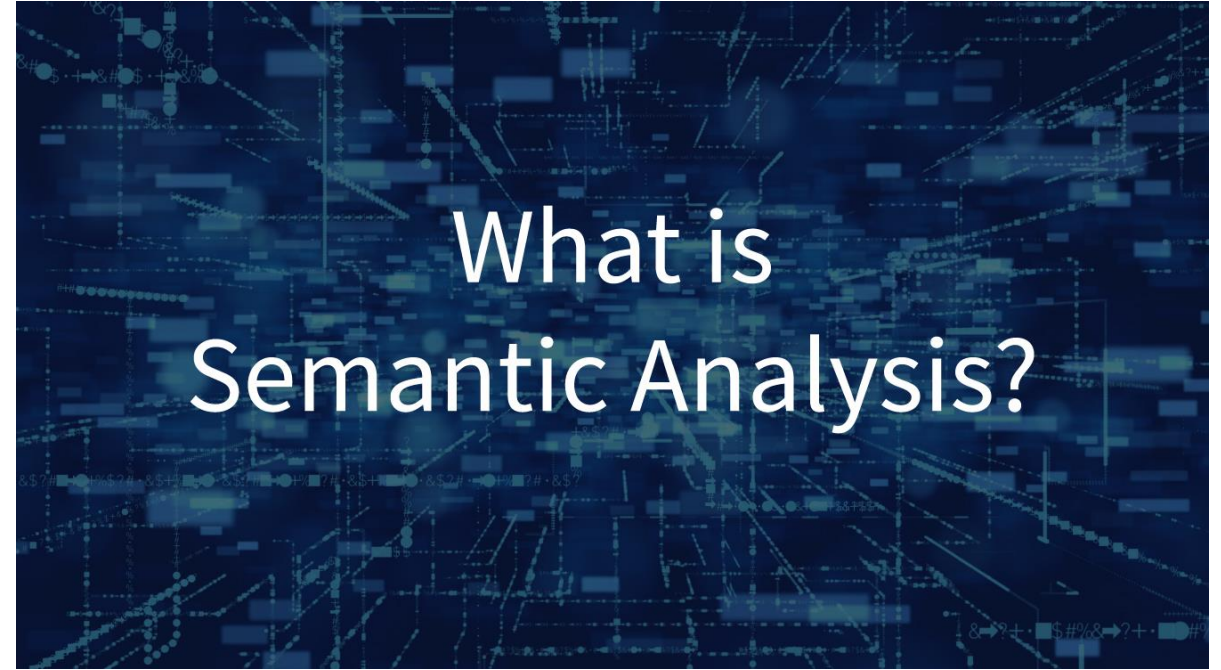


구문 분석(syntactic analysis)

- 구문(syntax)에 따라 문장이 가지는 **구문 구조**를 분석하여, 문장을 구성하는 **문자열**들이 문장에서 어떤 **역할**을 하는지 **결정**
 - 구문(syntax): 문장이나 구절을 만드는 규칙
 - '나는 책을 읽는다' = 나는(**주어**) + 책을(**목적어**) + 읽는다(**서술어**)
- 구문 분석 결과는 **파스 트리**(parse tree) 형태로 표현
- **파싱**(parsing)
 - 문장을 구문 분석하여 파스 트리를 만들어 내는 작업

의미 분석(semantic analysis)

- 구문 분석의 결과를 해석하여 **문장의 의미**를 결정하는 것



화용 분석(pragmatic analysis, 話用/語用 分析)

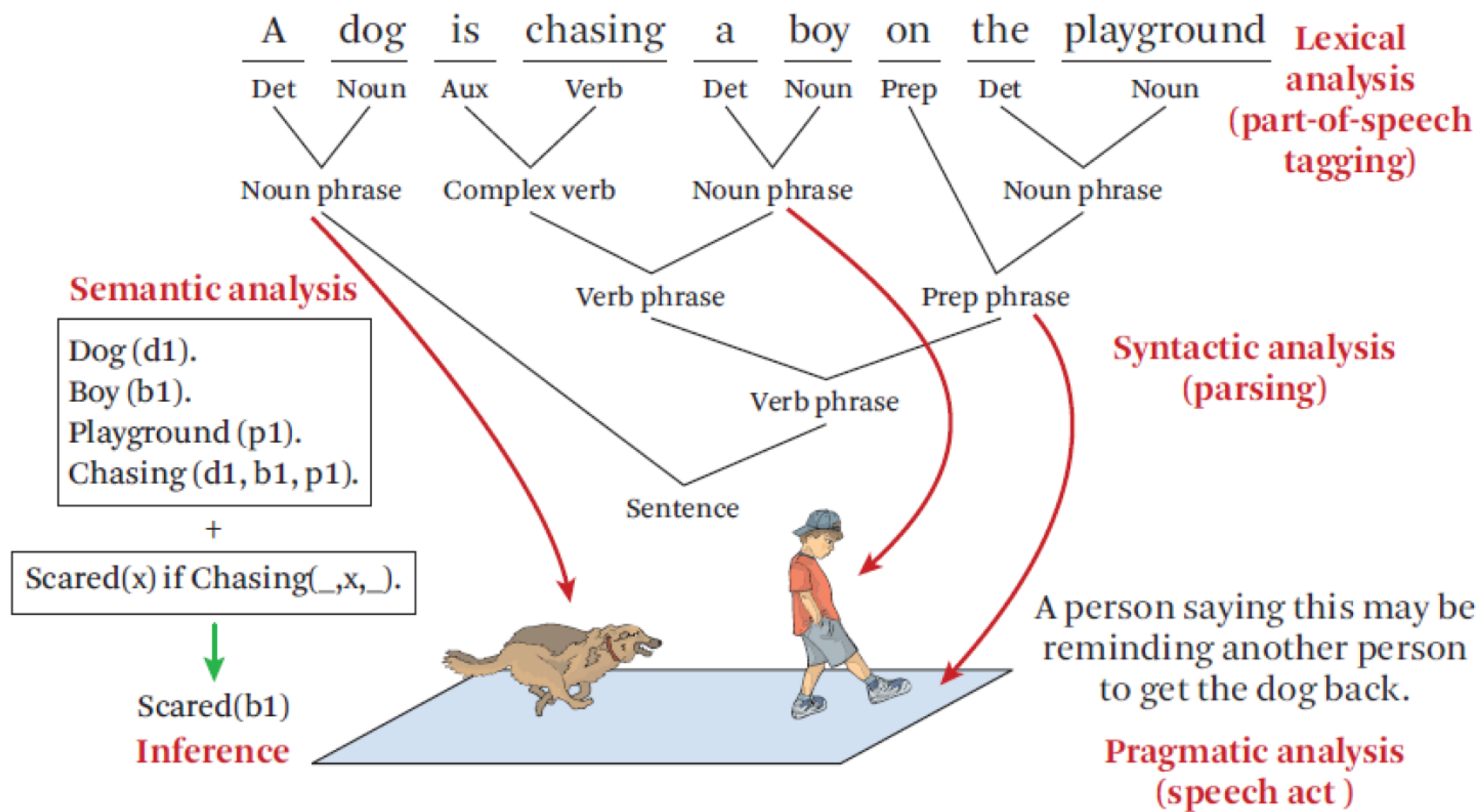
- 실제 **상황적 맥락** 즉, 말하는 이와 듣는 이의 관계, 시간과 장소, 주제를 **고려**하여 문장이 **실세계**와 가지는 **연관** 관계를 분석하는 것

가게 손님 : '우유 있어요?'

점원: (손님이 우유를 사려고 한다!) '우유는 오른쪽 통로에 있어요'



자연어 처리 단계



자연어 처리 단계

Part of speech:

NP NNP RB VBD IN NNP NNP , CC PRP VBZ RB VBG PRP IN PRP .
Mrs. Clinton previously worked for Mr. Obama, but she is now distancing herself from him .

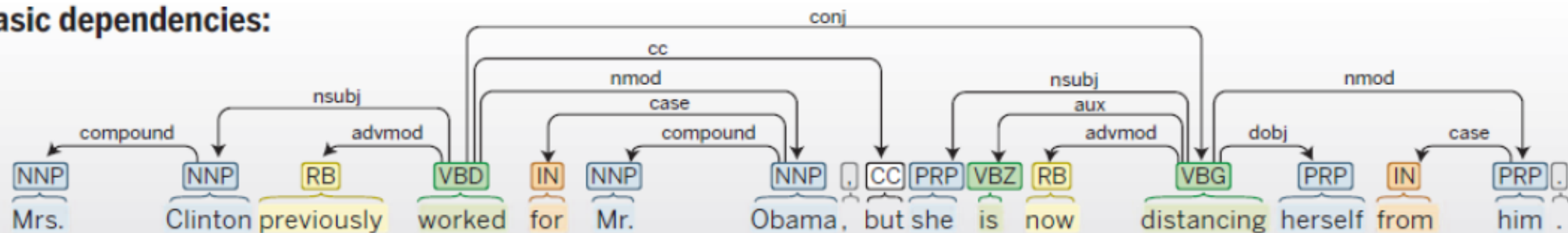
Named entity recognition:

Person Date Person Date
Mrs. Clinton previously worked for Mr. Obama, but she is now distancing herself from him.

Co-reference:

Mention Ment M Mention M
Mrs. Clinton previously worked for Mr. Obama, but she is now distancing herself from him.

Basic dependencies:





2 한국어 문법

형태론

(形態論, morphology)

- 단어의 형태 변화와 그 구성에 관련된 규칙을 다루는 문법 분야

통사론

(統辭論, syntax)

- 어떤 순서로 단어를 배열해야 문장이 되는가에 대한 문장 구성, 구성요소 간의 관계 및 기능을 다루는 분야

음운론

(音韻論, phonology)

- 음운의 성질, 소리와 소리가 만나면 생기는 변화를 다루는 분야

형태론(形態論, morphology)

- 형태소들을 결합하여 어절을 형성하는 체계와 규칙, 낱말의 특징을 다룸
- 음절(音節, syllable)
 - 한 번에 소리 낼 수 있는 소리 단위
 - 문장을 소리 나는 대로 쓸 때 글자 한 자에 대응하는 것
 - 예. '산에 눈이 내린다' ⇒ [사네누니내린다]
 - [사], [네], [누], [니], [내], [린], [다]
- 어절(語節)
 - 띄어쓰기가 되어 있는 말의 덩어리
 - 예. [산에], [눈이], [내린다]
- 어근(語根, root)
 - 단어형성(單語形成, word-formation)에서 그 단어의 뜻을 나타내는 최소 의미 단위

형태론(形態論, morphology)

■ 접사(接辭, affix)

- 단어를 형성할 때 어근에 붙어 그 뜻을 제한하는 주변 부분
 - 어근 앞에 오는 **접두사**, 어근 뒤에 오는 **접미사**

■ 어간(語幹, stem)

- 동사, 형용사와 같은 용언이 활용할 때 변하지 않는 부분

■ 어미(語尾, ending)

- 용언을 활용할 때 어간에 붙어서 변화하는 부분
 - '치솟다' = '치'(접두사), '치솟'(어근) + '다'(어미)

■ **어말 어미**(語末 語尾)

- 단어의 끝자리에 들어가는 어미
- 하나의 어간에 어말어미는 하나

■ **선어말 어미**(先語末 語尾)

- 어말 어미의 앞자리에 들어가는 어미
- 선어말어미는 여러 개 결합 가능

하 시 었 겠 사오 ㅅ 니 ㄴ 다

(어근) (존경법) (과거) (추측) (공손) (합쇼체) (직설법) (원칙법) (어말어미)

(선어말 어미)

형태소(形態素, morpheme)

- 의미를 가지는 요소로서 더 이상 분해할 수 없는 가장 작은 문법 단위
- 분해하면 뜻을 잃어버리게 되는 언어의 최소 단위
- **자립여부에 따른 분류**
 - **자립형태소**: 자립적이어서 자신이 무슨 뜻인지 알 수 있게 하는 것
 - **의존형태소**: 자기가 가진 의미를 나타내려면 다른 말에 의존해서 합해져야 하는 것
- **실질적 의미 포함 여부에 따른 분류**
 - **어휘형태소 (실질형태소)**: 개별적인 뜻이 사전에 나와 있는 형태소
 - **문법형태소 (형식형태소)**: 문법적 기능을 나타내는 형태

‘할 수 없지’

하-(의존형태소, 어근)/-ㄴ(의존형태소, 관형사적 접사)/수(자립형태소, 의존 명사)/
없(의존형태소, 어근)/-지(의존형태소, 어말 굴절 접사)

품사(品詞, part-of-speech; POS)

- 단어를 **문법적 성질**의 공통성에 따라 몇 갈래로 묶어 놓은 것
- **역할**에 따른 분류
 - 체언, 용언, 수식언, 관계언, 독립언
- **의미**에 따른 분류 (9품사)
 - 명사, 대명사, 수사, 동사, 형용사, 관형사, 부사, 조사, 감탄사
 - 체언(體言): 명사, 대명사, 수사
 - 용언(用言): 동사, 형용사
 - 수식언(修飾言): 관형사, 부사
 - 관계언(關係言): 조사
 - 독립언(獨立言): 감탄사

품사(品詞, part-of-speech; POS)

- **명사**(名詞, noun)
 - 물건이나 장소, 사건, 추상적 개념과 같은 이름을 나타내는 것
- **대명사**(代名詞, pronoun)
 - 명사를 대신하는 지시하는 단어
- **수사**(數詞, numeral)
 - 수량이나 순서를 나타내는 단어
- **동사**(動詞, verb)
 - 사물의 움직임을 나타내는 말
 - 활용 변환을 하고 어간과 어미로 구성
- **형용사**(形容詞, adjective)
 - 사물의 성질이나 상태를 나타내는 말
 - 활용 변환을 하고 어간과 어미로 구성
- **관형사**(冠形詞, determiner)
 - 체언 앞에서 그 체언을 꾸며 주는 말
- **부사**(副詞, adverb)
 - 용언을 꾸며주는 말
- **감탄사**(感歎詞, interjection)
 - 화자의 부름, 놀람, 느낌, 대답을 나타내는 독립언
- **조사**(助詞, postposition)
 - 체언에 붙어 체언에 일정한 자격을 부여하는 말

통사론(統辭論, syntax)

■ 문장(文章, sentence)

- 말하는 사람의 생각을 온전하게 나타내는 말의 묶음 중에서 가장 작은 단위
- 기본적으로 주어와 서술어로 구성
- 단문(短文), 복문(複文)

■ 절(節, clause)

- 복문에서 주어와 서술어 관계인 부분

■ 구(句, phrase)

- 절은 아니지만 어절이 몇 개 뭉쳐서 의미적 기능을 하는 것

■ 문장성분(文章成分)

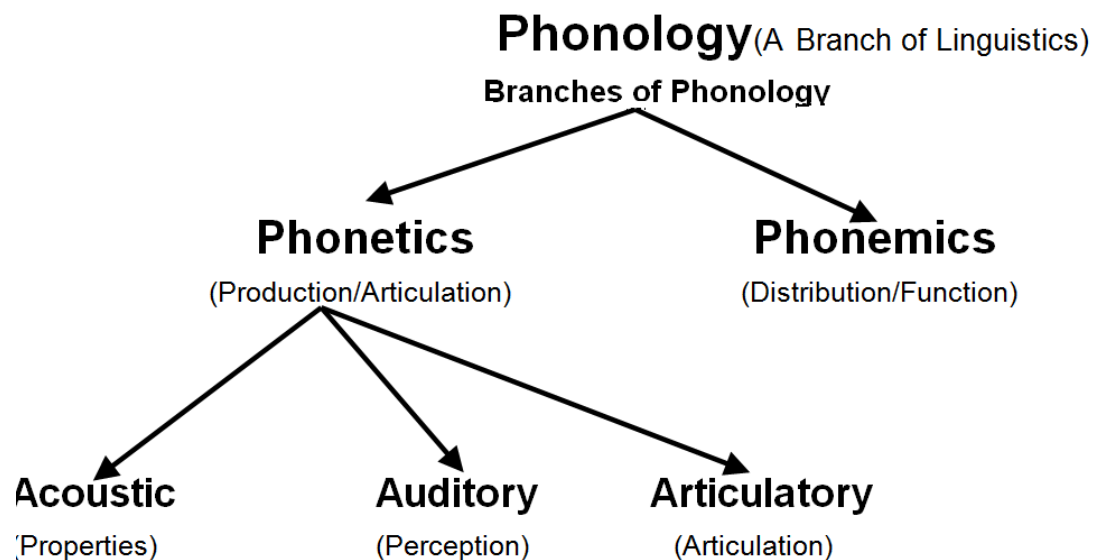
- 어절을 역할에 따라 주어, 목적어, 서술어, 보어, 관형어, 부사어, 독립어로 분류한 것

심청이는 빨간 연꽃을 매우 좋아했다.
주어 관형어 목적어 부사어 서술어

그래서 심청이는 왕비가 되었다.
독립어 주어 보어 서술어

음운론 (音韻論, phonology)

- 음운의 성질, 소리와 소리가 만나면 생기는 변화를 다루는 분야
- 음성인식 분야의 중요한 기반 이론
- 음운(音韻)
 - 음소(音素, phoneme): 자음, 모음
 - 운소(韻素, prosodeme): 소리의 길이, 높낮이, 세기
- 음운현상
 - 음절의 끝소리 규칙, 자음동화, 구개음화, 모음조화
 - 움라우트(' | '역행동화), 원순모음화, 전설모음화
 - 음운 축약·탈락, 사잇소리 현상, 된소리되기 등





3 형식 문법

형식 문법과 언어

- **형식 문법**(formal grammar)

- 특정 형식 언어에 속하는 **스트링**(string, 상황에 따라 문자열 또는 문장)을 생성하는 데 사용되는 유한개의 **생성 규칙**(production rule)들의 집합

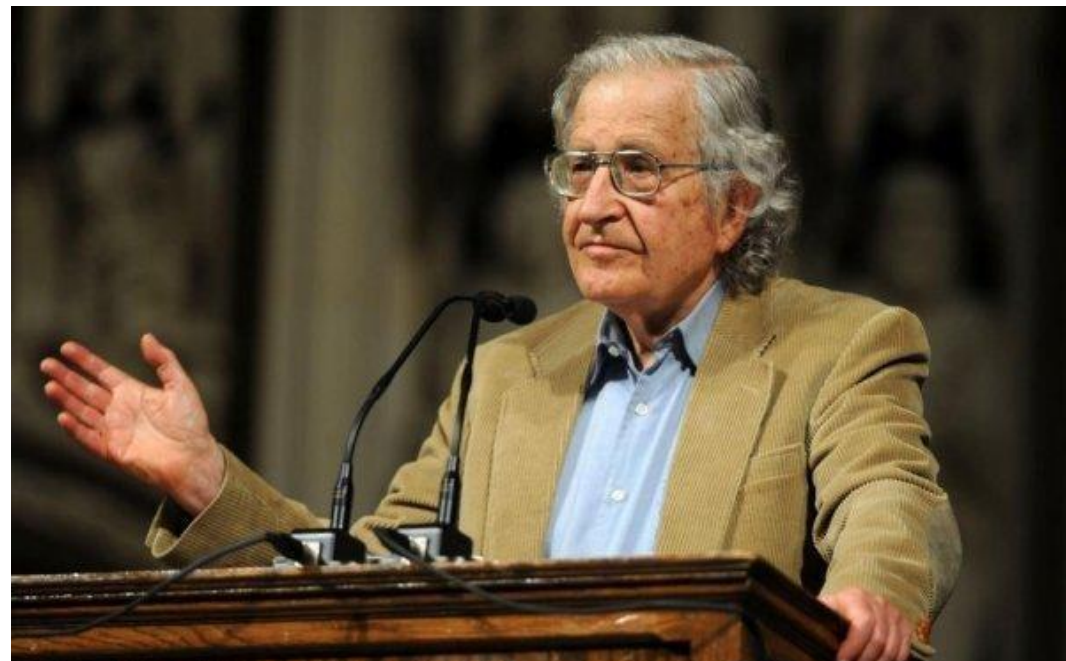
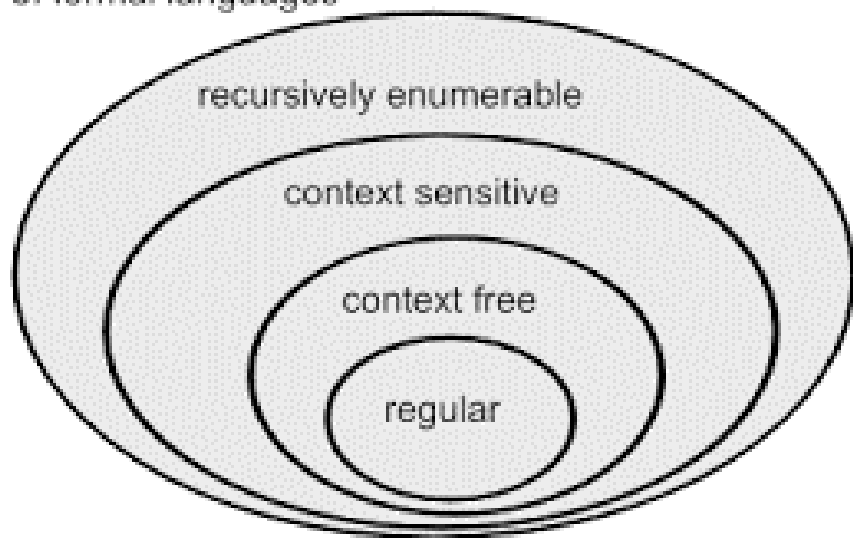
- **형식 언어**(formal language)

- 특정 **형식 문법에 따라 생성될 수 있는 모든 스트링의 집합**
- 문법으로부터 스트링들을 생산해 내는 **생성 문법**(generative grammar)
- 문자열이 특정 언어에 포함되는지를 판단하는 **해석 문법**(analytic grammar)

형식 문법 종류

- 노엄 촘스키(Avram Noam Chomsky)의 분류
 - 정규 문법(Type-3 문법)
 - 문맥 자유 문법(Type-2 문법),
 - 문맥 의존 문법(Type-1 문법)
 - 무제약 문법(Type-0 문법)

Chomsky hierarchy of formal languages



형식 언어 문법

■ 형식언어 문법 G

$$G = (N, \Sigma, P, S)$$

- N : 비단말 기호 (nonterminal symbol)
- Σ : 단말 기호
- P : 생성 규칙의 집합
- S : 시작 기호

정규 문법(regular grammar, Type-3 grammar)

▪ 우선형(right linear) 문법

$$A \rightarrow a$$

$$A \rightarrow aB$$

$$A \rightarrow \epsilon$$

$$A \in N \cup \{S\}$$

$$a \in \Sigma,$$

$$\epsilon$$

▪ 좌선형(left linear) 문법

$$A \rightarrow a$$

$$A \rightarrow Ba$$

$$A \rightarrow \epsilon$$

정규 언어(regular language)

- 정규 문법을 사용하여 생성되는 언어
- 정규식(regular expression)으로 표현 가능
- 유한 상태 기계(finite state machine)로 정규식 인식 가능

메타문자	의미	사용례
.	어떤 문자나 하나	.at hat, cat과 같이 at로 끝나는 3문자 문자열
[]	대괄호[] 문자 하나	[aeiou] a,e,i,o,u중의 한 글자
		[A-Z] 영문자 대문자 중 하나 [A-Za-z0-9] 영문자, 숫자 중 하나
[^]	대괄호 안의 문자를 제외한 문자	[^hc]at hat, cat을 제외한 .at
^	문자열의 첫번째 위치, 행(line) 기반 도구에서 행의 첫번째 위치	^hat hat으로 시작하는 행 또는 문자열
\$	문자열의 마지막 위치, 행 기반도구에서 개행 문자 이전 위치	cat\$ cat으로 끝나는 행 또는 문자열
*	이전 문자를 0회 이상 반복	ba* ba, baa, baaa와 같은 표현 [xy]* x와 y를 사용해서 만들 수 있는 모든 문자열
+	이전 문자를 1회 이상 반복	ba+ baa, baaa 등 표현
?	이전 문자를 0 또는 1회 반복	ba? b 또는 ba
	양쪽에 있는 것 중에서 하나 선택	cat dog cat 또는 dog
₩. ₩?	점(.) 하나, 물음표(?) 하나	
₩b ₩B	접두사	₩bun₩B un으로 시작하는 단어

문맥 자유 문법(context-free grammar, Type-2 grammar)

- 프로그래밍 언어의 문법
- 컴파일러는 기본적으로 문맥 자유 문법을 다룸
- 정규 언어는 문맥 자유 언어의 부분 집합

$$A \rightarrow w$$

$$A \in N \cup \{S\}$$

$$w \in (N \cup \Sigma)^* - \{\epsilon\}$$

문맥 의존 문법(context-sensitive grammar, Type-1 grammar)

- 생성규칙의 LHS(left hand side)는 시작부분과 끝부분 포함

$$\alpha A \beta \rightarrow \alpha \gamma \beta$$

$$A \in N$$

$$\alpha, \beta \in (N \cup \Sigma)^*$$

$$\gamma \in (N \cup \Sigma)^+$$

- 비단말 기호 A 를 기호 양 옆의 상황에 따라, 즉 문맥에 따라 다르게 대치 가능
- 문맥 자유 언어는 문맥 의존 언어에 포함

무제약 문법(unrestricted grammar, Type-0 grammar)

- 가장 일반적인 형식 문법

$$\alpha \rightarrow \beta$$

$$\alpha \in (N \cup S \cup \Sigma)^+$$

$$\beta \in (N \cup S \cup \Sigma)^*$$

- 생성 규칙을 정의하는데 제약이 없음
- 튜링 머신**(Turing machine)은 무제약 문법의 문장 인식 가능



4 형태소 분석과 품사 태깅

형태소 분석

- 자연어 처리의 가장 기본적인 분석 작업
- **단어**(한국어의 경우, 어절)에 대하여, **형태소**들을 인식
- 불규칙한 활용이나 축약, 탈락 현상이 일어난 경우에 원형 복원
- **어휘 분석**(lexical analysis)이라고도 함
- 여러 **사전**(辭典, dictionary) **정보** 이용
 - 문법형태소 사전, 어휘형태소 사전, 전문 용어 사전, 사용자 정의 사전, 기분석 사전
- **규칙**(rule) 기반 분석
- 언어별 난이도 차이
 - 영어, 프랑스어: 쉬움
 - 한국어, 일본어, 아랍어: 어려움

형태소 분석 - 영어

- 규칙변환 및 불규칙 변환 명사와 동사

- 규칙

- Walk, walks, walking, walked, walked
- Table, tables

- 불규칙

- Eat, eats, eating, ate, eaten
- Catch, catches, catching, caught, caught
- Cut, cuts, cutting, cut, cut
- Goose, geese

- 형태소 분석의 예

- Cats cat +N +PL
- Cat cat +N +SG
- Cities city +N +PL
- Merging merge +V +Present-participle
- Caught catch +V +Past-participle

N : noun(명사)

PL: plural (복수)

SG: singular (단수)

V: 동사

Present-participle (현재분사)

Past-participle (과거분사)

형태소 분석 - 한글

- 교착어의 특성 때문에 영어에 비해 복잡

- **중의성**(ambiguity) 포함

- 예 : '나는'

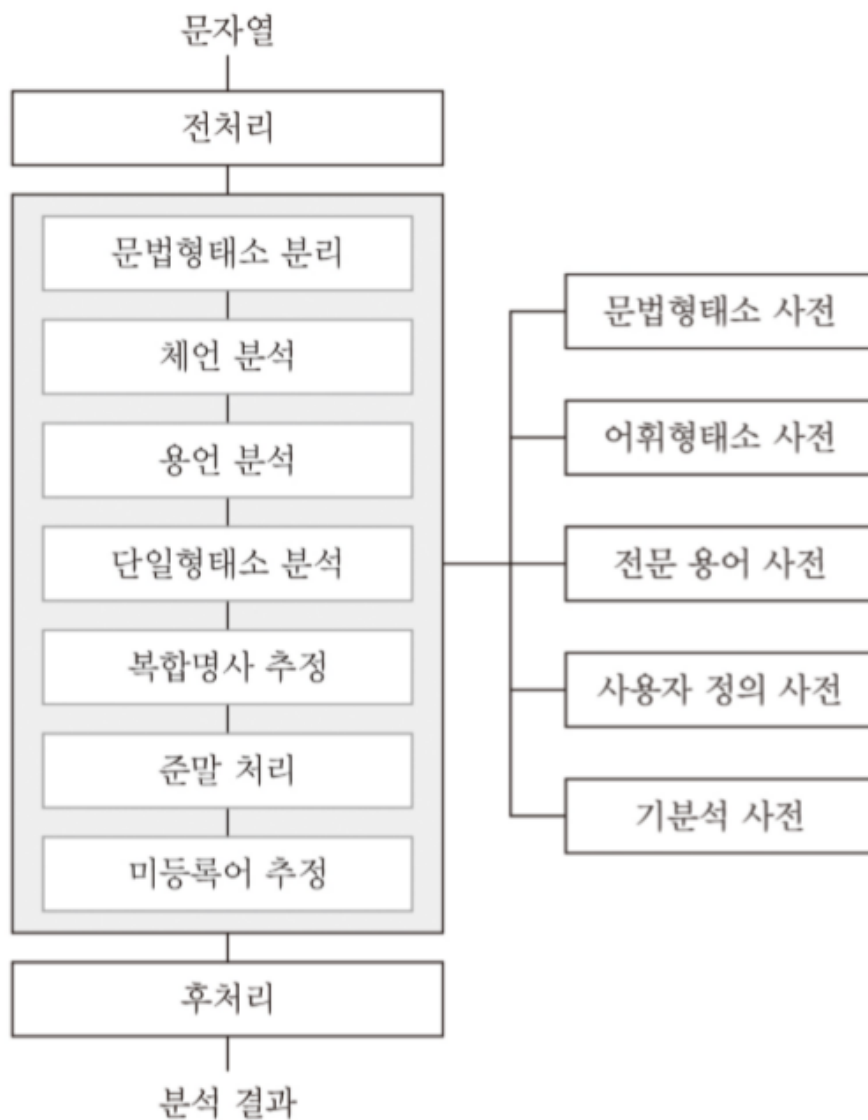
- 나(대명사, **I**) + 는 (조사),
- 나(동사, **produce**) + 는 (어미)
- 날(동사, **fly**) + 는 (어미)

- 접두사, 접미사 처리

- 형태소 결합의 예 : **친구에게서였었다라고**

친구(명사) + 에게(조사) + 서(조사) + 이(서술격조사) + 었(과거시제어미) + 었(회상어미) + 다(어말어미) + 라고(인용격조사)

한국어 형태소 분석 과정



형태소 분석의 활용

■ 한국어

- 맞춤법이나 띄어쓰기 교정
- 단어의 품사 결정 정보 제공
- 단어의 의미 추정 정보 제공
- 검색엔진 색인(index) 구성

■ 영어

- 철자교정 : refelence
- 하이픈 위치 결정(hyphenation) : refer-ence
- 품사(Part-of-Speech, POS) 분석 : google, googler
- 음성합성(text-to-speech, TTS)
 - hothouse (/t/, /θ/)
- 단어의 의미 추정
 - 'Twas brillig and the slithy toves...
- 기계번역
- 색인(indexing) : 검색 엔진

형태소 분석기



Natural Language Analysis
with Python NLTK

spaCy

sentencepiece

Khایی(Kakao Hangul Analyzer iii)



KoNLPy

Mecab (메케브)

한나눔

GILab at KAIST

Kkma (꼬꼬마)

Okt(Twitter)

Komoran (코모란)

영어 Tokenizing 라이브러리

■ NLTK

- 파이썬에서 영어 텍스트 전처리 작업을 하는데 많 쓰이는 라이브러리
- 50여 개가 넘는 말뭉치 리소를 활용해 영어 텍스트를 분석할 수 있게 제공
- 직관적으로 함수를 쉽게 사용할 수 있게 구성돼 있어 빠르게 텍스트 전처리 가능
- 단어 단위 토큰나이징과 문장 단위 토큰나이징을 하는 모듈이 따로 있으며 'a', 'the' 같은 관사나 'is'와 같이 자주 의미는 별로 없지만 자주 등장하는 단어인 불용어들을 모아 불용어 사전을 구성하고 있어 따로 불용어를 정의할 필요없이 바로 사용가능

■ Spacy

- NLTK와 같은 오픈소스 라이브러리
- 주로 상업용 목적으로 만들어졌다는 점이 NLTK와 다르며, 영어를 포함한 8개 언어에 대한 자연어 전처리 모듈을 제공하고, 빠른 속도로 전처리 가능
- 원하는 언어에 대한 전처리를 한 번에 해결할 수 있다는 장점이 있으며, 특히 딥러닝 언어 모델의 개발도 지원하고 있어 매력적임
- NLTK와 다르게 단어 단위, 문장 단위 토큰나이징을 한가지 모듈을 통해 처리

한글 Tokenizing 라이브러리

- KoNLPy(지도학습 기법으로 학습)
 - 한글 자연어 처리를 쉽고 간결하게 처리할 수 있도록 만들어진 오픈소스 라이브러리
 - 국내에 이미 만들어져 사용되고 있는 여러 형태소 분석기를 사용할 수 있게 허용
 - 형태소 분석으로 형태소 단위의 토큰나이징을 가능하게 할뿐만 아니라 구문 분석을 가능하게 해서 언어 분석을 하는 데 유용한 도구
- 한글 텍스트의 경우에는 형태소 단위 토큰나이징이 필요
 - Hannanum (한나눔)
 - Kkma (꼬꼬마)
 - Komoran (코모란)
 - Mecab (메케브)
 - Okt(Twitter)

한나눔 형태소 분석기, Hannanum (KAIST)

Hannanum Class

`class konlpy.tag._hannanum.Hannanum(jvmpath=None)`

Wrapper for [JHannanum](#).

JHannanum is a morphological analyzer and POS tagger written in Java, and developed by the [Semantic Web Research Center \(SWRC\)](#) at KAIST since 1999.

```
>>> from konlpy.tag import Hannanum
>>> hannanum = Hannanum()
>>> print(hannanum.analyze(u'롯데마트의 흑마늘 양념 치킨이 논란이 되고 있다.'))
[[('롯데마트', 'ncn'), ('의', 'jcm')], [('롯데마트의', 'ncn')], [('롯데마트', '
>>> print(hannanum.morphs(u'롯데마트의 흑마늘 양념 치킨이 논란이 되고 있다.'))
['롯데마트', '의', '흑마늘', '양념', '치킨', '이', '논란', '이', '되', '고', '있
>>> print(hannanum.nouns(u'다람쥐 현 쳃바퀴에 타고파'))
['다람쥐', '쳃바퀴', '타고파']
>>> print(hannanum.pos(u'웃으면 더 행복합니다!'))
[('웃', 'P'), ('으면', 'E'), ('더', 'M'), ('행복', 'N'), ('하', 'X'), ('바니다',
```

매개 변수: `jvmpath` – The path of the JVM passed to `init_jvm()`.

Class initializer

`analyze(phrase)`

Phrase analyzer.

This analyzer returns various morphological candidates for each token. It consists of two parts: 1) Dictionary search (chart), 2) Unclassified term segmentation.

`morphs(phrase)`

Parse phrase to morphemes.

`nouns(phrase)`

Noun extractor.

`pos(phrase, ntags=9, flatten=True)`

POS tagger.

This tagger is HMM based, and calculates the probability of tags.

매개 변수: • `ntags` – The number of tags. It can be either 9 or 22.
• `flatten` – If False, preserves eojeols.

Methods

꼬꼬마 형태소 분석기, Kkma (서울대학교)

Kkma Class

```
class konlpy.tag._kkma.Kkma(jvmpath=None, max_heap_size=1024)
```

Wrapper for [Kkma](#).

Kkma is a morphological analyzer and natural language processing system written in Java, developed by the Intelligent Data Systems (IDS) Laboratory at [SNU](#).

```
>>> from konlpy.tag import Kkma
>>> kkma = Kkma()
>>> print(kkma.morphs(u'공부를 하면할수록 모르는게 많다는 것을 알게 됩니다.'))
['공부', '를', '하', '면', '하', 'ㄹ수록', '모르', '는', '것', '이', '많', '다는']
>>> print(kkma.nouns(u'대학에서 DB, 통계학, 이산수학 등을 배웠지만...'))
['대학', '통계학', '이산', '이산수학', '수학', '등']
>>> print(kkma.pos(u'다 까먹어버렸네요?ㅋㅋ'))
[('다', 'MAG'), ('까먹', 'VV'), ('어', 'ECD'), ('버리', 'VXV'), ('었', 'EPT'),
]
>>> print(kkma.sentences(u'그래도 계속 공부합니다. 재밌으니까!'))
['그래도 계속 공부합니다.', '재밌으니까!']
```

경고:

There are reports that `Kkma()` is weak for long strings with no spaces between words. See issue [#73](#) for details.

매개 변수: • **jvmpath** -- The path of the JVM passed to [init_jvm\(\)](#).
수: • **max_heap_size** -- Maximum memory usage limitation (Megabyte)
[init_jvm\(\)](#).

Class initializer

`morphs(phrase)`

Parse phrase to morphemes.

`nouns(phrase)`

Noun extractor.

`pos(phrase, flatten=True, join=False)`

POS tagger.

매개 변수: • **flatten** -- If False, preserves eojeols.
• **join** -- If True, returns joined sets of morph and tag.

`sentences(phrase)`

Sentence detection.

Methods

코모란 형태소 분석기, Komoran

Komoran Class

```
class konlpy.tag._komoran.Komoran(jvmpath=None, userdic=None,
modelpath=None, max_heap_size=1024)
    Wrapper for KOMORAN.
```

KOMORAN is a relatively new open source Korean morphological analyzer written in Java, developed by [Shineware](#), since 2013.

```
>>> cat /tmp/dic.txt # Place a file in a location of your choice
코모란      NNP
오픈소스     NNG
바람과 함께 사라지다      NNP
>>> from konlpy.tag import Komoran
>>> komoran = Komoran(userdic='/tmp/dic.txt')
>>> print(komoran.morphs(u'우왕 코모란도 오픈소스가 되었어요'))
['우왕', '코모란', '도', '오픈소스', '가', '되', '었', '어요']
>>> print(komoran.nouns(u'오픈소스에 관심 많은 멋진 개발자님들!'))
['오픈소스', '관심', '개발자']
>>> print(komoran.pos(u'혹시 바람과 함께 사라지다 봤어?'))
[('혹시', 'MAG'), ('바람과 함께 사라지다', 'NNP'), ('보', 'VV'), ('었', 'EP'), (
```

매개변수:

- **jvmpath** -- The path of the JVM passed to `init_jvm()`.
- **userdic** -- The path to the user dictionary. This enables the user to enter custom tokens or phrases, that are mandatorily assigned to tagged as a particular POS. Each line of the dictionary file should consist of a token or phrase, followed by a POS tag, which are delimited with a `<tab>` character. An example of the file format is as follows:

```
바람과 함께 사라지다 NNG
바람과 함께      NNP
자연어 NNG
```

If a particular POS is not assigned for a token or phrase, it will be tagged as NNP.

- **modelpath** -- The path to the Komoran HMM model.
 - **max_heap_size** -- Maximum memory usage limitation (Megabyte)
- `init_jvm()`

`morphs(phrase)`

Parse phrase to morphemes.

`nouns(phrase)`

Noun extractor.

`pos(phrase, flatten=True, join=False)`

POS tagger.

- 매개변수:
- **flatten** -- If False, preserves eojeols.
 - **join** -- If True, returns joined sets of morph and tag.

Class initializer

Methods

은전한뿔 형태소 분석기, Mecab

Mecab Class

경고:

Mecab() is not supported on Windows.

```
class konlpy.tag._mecab.Mecab(dicpath='/usr/local/lib/mecab/dic/mecab-ko-dic')
    Wrapper for MeCab-ko morphological analyzer.
```

[MeCab](#), originally a Japanese morphological analyzer and POS tagger developed by the Graduate School of Informatics in Kyoto University, was modified to MeCab-ko by the [Eunjeon Project](#) to adapt to the Korean language.

In order to use MeCab-ko within KoNLPy, follow the directions in optional-installations.

```
>>> # MeCab installation needed
>>> from konlpy.tag import Mecab
>>> mecab = Mecab()
>>> print(mecab.morphs(u'영등포구청역에 있는 맛집 좀 알려주세요.'))
['영등포구', '청역', '에', '있', '는', '맛집', '좀', '알려', '주', '세요', '.']
>>> print(mecab.nouns(u'우리나라에는 무릎 치료를 잘하는 정형외과가 없는가!'))
['우리', '나라', '무릎', '치료', '정형외과']
>>> print(mecab.pos(u'자연주의 쇼핑물은 어떤 곳인가?'))
[('자연', 'NNG'), ('주', 'NNG'), ('의', 'JKG'), ('쇼핑물', 'NNG'), ('은', 'JX')]
```

매개 변수: **dicpath** -- The path of the MeCab-ko dictionary.

morphs(*phrase*)
Parse phrase to morphemes.

nouns(*phrase*)
Noun extractor.

pos(*phrase*, *flatten*=True, *join*=False)
POS tagger.

매개 변수: • **flatten** -- If False, preserves eojjeols.
• **join** -- If True, returns joined sets of morph and tag.

morphs(*phrase*)

Parse phrase to morphemes.

nouns(*phrase*)

Noun extractor.

pos(*phrase*, *flatten*=True, *join*=False)

POS tagger.

매개 변수: • **flatten** -- If False, preserves eojjeols.
• **join** -- If True, returns joined sets of morph and tag.

Class initializer

Methods



Okt(Twitter) 형태소 분석기

Okt Class

경고:

Twitter() has changed to Okt() since v0.5.0.

```
class konlpy.tag._Okt.Okt(jvmpath=None, max_heap_size=1024)
```

Wrapper for [Open Korean Text](#).

Open Korean Text is an open source Korean tokenizer written in Scala, developed by Will Hohyon Ryu.

```
>>> from konlpy.tag import Okt
>>> okt = Okt()
>>> print(okt.morphs(u'단독입찰보다 복수입찰의 경우'))
['단독', '입찰', '보다', '복수', '입찰', '의', '경우']
>>> print(okt.nouns(u'유일하게 항공기 체계 종합개발 경험을 갖고 있는 KAI는'))
['항공기', '체계', '종합', '개발', '경험']
>>> print(okt.phrases(u'날카로운 분석과 신뢰감 있는 진행으로'))
['날카로운 분석', '날카로운 분석과 신뢰감', '날카로운 분석과 신뢰감 있는 진행', '날카로운 분석과 신뢰감 있는 진행으로']
>>> print(okt.pos(u'이것도 되나욐ㅋㅋ'))
[('이', 'Determiner'), ('것', 'Noun'), ('도', 'Josa'), ('되나욐', 'Noun'), ('ㅋㅋ', 'Noun')]
>>> print(okt.pos(u'이것도 되나욐ㅋㅋ', norm=True))
[('이', 'Determiner'), ('것', 'Noun'), ('도', 'Josa'), ('되나요', 'Verb'), ('ㅋㅋ', 'Noun')]
>>> print(okt.pos(u'이것도 되나욐ㅋㅋ', norm=True, stem=True))
[('이', 'Determiner'), ('것', 'Noun'), ('도', 'Josa'), ('되다', 'Verb'), ('ㅋㅋ', 'Noun')]
```

매개 변수: • **jvmpath** -- The path of the JVM passed to [init_jvm\(\)](#).
• **max_heap_size** -- Maximum memory usage limitation (Megabyte) [init_jvm\(\)](#).

morphs(phrase, norm=False, stem=False)

Parse phrase to morphemes.

normalize(phrase)

nouns(phrase)

Noun extractor.

phrases(phrase)

Phrase extractor.

pos(phrase, norm=False, stem=False, join=False)

POS tagger. In contrast to other classes in this subpackage, this POS tagger doesn't have a *flatten* option, but has *norm* and *stem* options. Check the parameter list below.

매개 변수: • **norm** -- If True, normalize tokens.
• **stem** -- If True, stem tokens.
• **join** -- If True, returns joined sets of morph and tag.

Class initializer

Methods

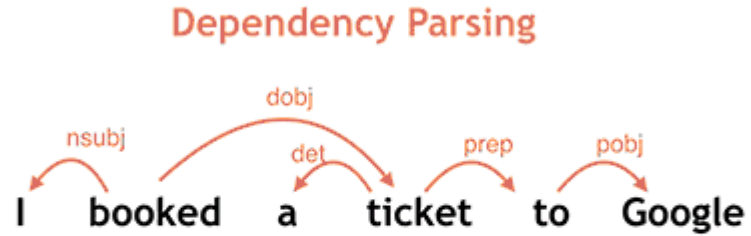
- Khaiii(Kakao Hangeul Analyzer iii)는 kakao가 2018년 말 공개한 오픈소스 한국어 형태소 분석기
- 국립국어원이 구축한 세종 코퍼스를 이용해 CNN 모델을 적용해 학습
- Khaiii의 아키텍처는 입력 문장을 문자 단위로 읽어 들인 뒤 convolution filter가 이 문자들을 슬라이딩해 가면서 정보를 추출
- 출력 노드에서는 이렇게 모은 정보들을 종합해 형태소의 경계와 품사 태그를 예측
- 카카오 측 설명에 따르면 모델을 C++로 구현해 GPU 없이도 형태소 분석이 가능하며 실행 속도 역시 빠름

- 형태소 분석, 품사 판별 등을 지원하는 파이썬 기반 한국어 자연어 처리 패키지
- 데이터 패턴을 스스로 학습하는 비지도 학습 접근법을 지향하기 때문에 하나의 문장 혹은 문서에서보다는 어느 정도 규모가 있으면서 동질적인 문서 집합(homogeneous documents)에서 잘 작동
- soynlp 형태소 분석기는 데이터의 통계량을 확인해 만든 단어 점수 표로 작동
- 단어 점수는 크게 응집확률(Cohesion Probability) 과 브랜칭 엔트로피(Branching Entropy)를 활용
- 구체적으로는 주어진 문자열이 유기적으로 연결돼 함께 자주 나타나고(응집 확률이 높을 때), 그 단어 앞뒤로 다양한 조사, 어미 혹은 다른 단어가 등장하는 경우(브랜칭 엔트로피가 높을 때) 해당 문자열을 형태소로 취급
- 예를 들어, 주어진 Corpus에서 '꿀잼'이라는 단어가 연결돼 자주 나타났다면 '꿀잼'을 형태소라고 봄(응집 확률이 높음)
- 한편 '꿀잼' 앞에 '영화', '정말', '너무' 등 문자열이, 뒤에 'ㅋㅋ', 'ㅎㅎ', '!!' 등 패턴이 다양하게 나타났다면 이 역시 '꿀잼'을 형태소로 취급 (브랜칭 엔트로피가 높음)

구글 센텐스피스(sentencepiece)

- 구글에서 공개한 비지도 학습기반 형태소 분석 패키지
- 1994년 제안된 바이트 페어 인코딩(BPE : Byte Pair Encoding)기법 등을 지원
- pip 설치를 통해 파이썬 콘솔에서도 사용 가능

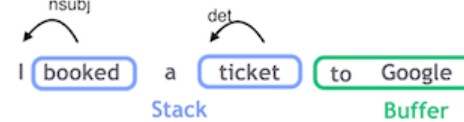
Google Natural Language API (Entities, Sentiment, Syntax)



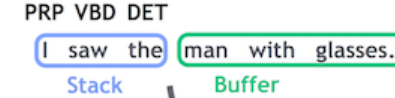
Input/TransitionState (C++)

Sentence w/ partial annotations

Parsing:

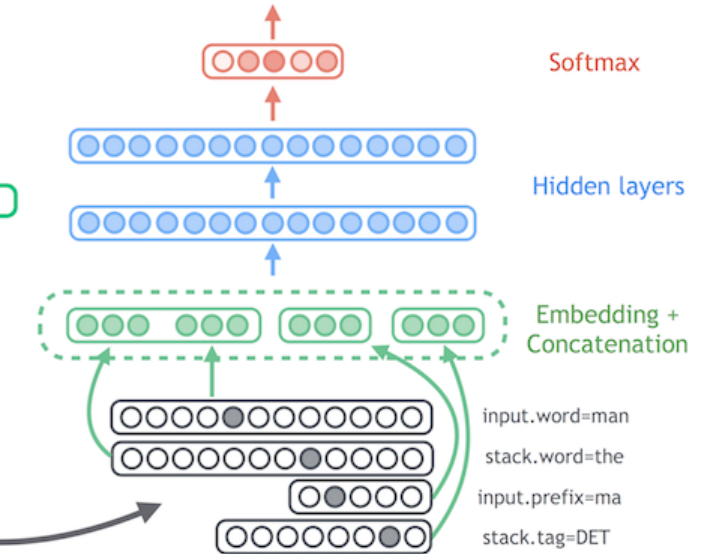


Tagging:



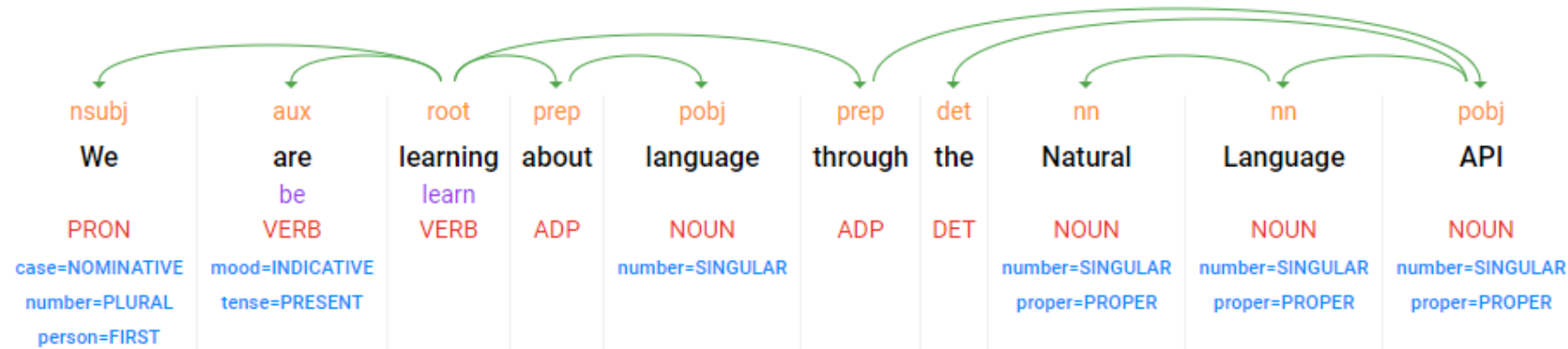
Sparse feature extraction

Network (TensorFlow)



Feed-Forward SyntaxNet Architecture (Overview)

☒ Dependency
 ☒ Parse Label
 ☒ Part of Speech
 ☒ Lemma
 ☒ Morphology



품사 태깅(POS tagging)

- 문장의 각 단어에 품사의 범주 및 문법적 기능에 대한 태그(tag, 꼬리표)를 붙이는 것
- **말뭉치(corpus)**: 언어 연구를 위해 컴퓨터가 읽을 수 있는 형태로 텍스트를 모아 놓은 것
- 주석 처리된 말뭉치를 이용한 품사 태깅 방법

대한민국은 무척 아름다운 나라이다.

'대한민국'_N '은'_J '무척'_M '아름답'_P '은'_E '나라'_N '이'_J '다' E '._'_S

AT0: 관사
NN1: 단수명사
VVB: 동사기본형
PRP: 전치사

A man see a woman with a telescope on the hill.

A_AT0 man_NN1 see_VVB a_AT0 woman_NN1 with_PRP a_AT0 telescope_NN1
on_PRP the_AT0 hill_NN1.

품사 태깅 기법

▪ 규칙 기반 방법

- 문장 구성의 원리, 문맥적인 정보와 형태소 정보 등을 이용하여 수작업으로 만든 규칙 사용
- 일관성있는 규칙을 많이 만들어내기는 어렵지만, 만들어진 규칙은 비교적 쉽게 사용 가능

▪ 기계학습 기반 방법

- 품사 태깅이 되어있는 말뭉치를 학습 데이터로 사용하여 분류기를 학습하여 사용
- 통계적 기계학습 알고리즘: 은닉 마르코프 모델(HMM), 최대 엔트로피 마르코프 모델(MEMM), 조건부 랜덤 필드(CRF) 등
- SVM
- 딥러닝 신경망

개체명 인식(named entity recognition)

- 개체명(Named Entity)란 인명(PS), 지명(LC), 날짜(DT), 시간(TI), 기관(OG) 등 문서 내에서 특정한 의미를 가진 단어 또는 어구를 뜻함
- 개체명 인식(Named Entity Recognition)이란 텍스트 내에서 개체명을 찾아 해당하는 의미의 범주를 결정하는 것을 뜻함
- 정보검색, 정보추출, 질의응답에서 중요한 역할
- “자베르 경감이 장발장과 1832년 파리에서 마주쳤다”

[자베르]_{인명} 경감이 [장발장]_{인명}과 [1832]_{날짜}년 [파리]_{지명}에서 마주쳤다

개체명 인식(named entity recognition)

In fact, the **Chinese** NORP market has the **three** CARDINAL most influential names of the retail and tech space – **Alibaba** GPE, **Baidu** ORG, and **Tencent** PERSON (collectively touted as **BAT** ORG), and is betting big in the global **AI** GPE in retail industry space. The **three** CARDINAL giants which are claimed to have a cut-throat competition with the **U.S.** GPE (in terms of resources and capital) are positioning themselves to become the 'future **AI** PERSON platforms'. The trio is also expanding in other **Asian** NORP countries and investing heavily in the **U.S.** GPE based **AI** GPE startups to leverage the power of **AI** GPE. Backed by such powerful initiatives and presence of these conglomerates, the market in APAC AI is forecast to be the fastest-growing **one** CARDINAL, with an anticipated **CAGR** PERSON of **45%** PERCENT over **2018 - 2024** DATE.

To further elaborate on the geographical trends, **North America** LOC has procured **more than 50%** PERCENT of the global share in **2017** DATE and has been leading the regional landscape of **AI** GPE in the retail market. The **U.S.** GPE has a significant credit in the regional trends with **over 65%** PERCENT of investments (including M&As, private equity, and venture capital) in artificial intelligence technology. Additionally, the region is a huge hub for startups in tandem with the presence of tech titans, such as **Google** ORG, **IBM** ORG, and **Microsoft** ORG.

개체명 인식 기법

▪ 규칙 기반의 방법

- 개체명을 인식하기 위해 사전들과 규칙들 이용
- 사전의 종류
 - **개체명 사전**: 인명, 지명, 조직명 등의 개체명에 대한 직접적인 정보
 - **결합 단어 사전**: 개체명과 함께 붙어서 나타나는 단어 정보
- 개체명 인식 규칙
 - **단어구성 규칙**: 개체명의 후보가 되는 단어 자체의 정보 이용
 - **문맥 규칙**: 문장에서 개체명의 후보가 되는 단어의 주변 정보 이용

▪ 기계학습 기반의 방법

- 단어의 품사 정보, 문자 유형, 주변 단어 정보, 사전 정보 등의 특징들을 이용하여 개체명을 부류로 간주하여 분류기 개발



5 구문 분석

구문 분석(syntactic analysis)

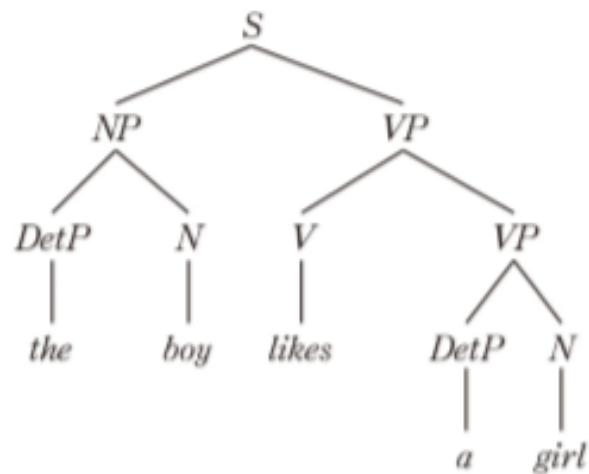
- 구문에 따라 주어진 문장에서 **단어들의 역할**을 파악하여 문장을 **계층적인 트리 구조**로 변환하는 작업
- **구문(syntax)**
 - 문장이나 구절을 만드는 규칙
 - 생성 규칙을 사용한 언어 정의 예
- **구문 분석 접근 방법**
 - 규칙 기반 구문 분석
 - 기계학습 기반 구문 분석

규칙 기반 구문 분석

- 문맥 자유 문법 형태의 구구조 문법(句構造 文法)을 이용하여 문장 분석
- 구구조 문법
 - 문장의 문법을 $N, V, Adj, DetP$ 등의 품사 기호와, $NP, VP, AdjP$ 등의 구(句) 기호를 사용하여 문맥 자유 문법 형태로 기술

- 1) $S \rightarrow NP VP$
- 2) $VP \rightarrow V NP$
- 3) $NP \rightarrow DetP N \mid AdjP NP$
- 4) $AdjP \rightarrow Adj \mid Adv AdjP$
- 5) $N \rightarrow boy \mid girl$
- 6) $V \rightarrow sees \mid likes$
- 7) $Adj \rightarrow big \mid small$
- 8) $Adv \rightarrow very$
- 9) $DetP \rightarrow a \mid the$

The boy likes a girl.



(S (NP (DetP the) (N boy))
(VP (V likes) (VP (DetP a) (N girl))))

파싱(parsing)

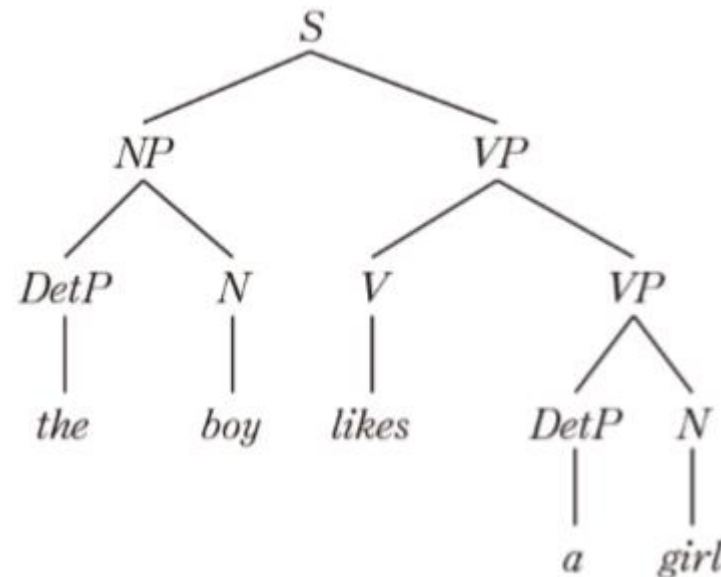
- **파서**(parser) 또는 **구문 분석기**(syntactic analyzer)

- 파싱(parsing)을 수행하는 프로그램

- **파싱 기법 분류**

- 트리 구조의 생성 진행 방향에 따라
- **하향식**(top-down) 파싱
- **상향식**(bottom-up) 파싱

The boy likes a girl.



하향식(top-down) 파싱

- 문장 시작 기호 S 에서 시작
- 생성 규칙을 반복 적용하여 주어진 문장을 얻는 과정을 통해 구문의 구조 파악

The boy likes a girl.

1) $S \rightarrow NP VP$

$S \rightarrow NP VP$ (규칙 1 적용)

2) $VP \rightarrow V NP$

$\rightarrow DetP N VP$ (규칙 3 적용)

3) $NP \rightarrow DetP N \mid AdjP NP$

$\rightarrow the N VP$ (규칙 9 적용)

4) $AdjP \rightarrow Adj \mid Adv AdjP$

$\rightarrow the boy VP$ (규칙 5 적용)

5) $N \rightarrow boy \mid girl$

$\rightarrow the boy V NP$ (규칙 2 적용)

6) $V \rightarrow sees \mid likes$

$\rightarrow the boy likes NP$ (규칙 6 적용)

7) $Adj \rightarrow big \mid small$

$\rightarrow the boy likes DetP N$ (규칙 3 적용)

8) $Adv \rightarrow very$

$\rightarrow the boy likes a N$ (규칙 9 적용)

9) $DetP \rightarrow a \mid the$

$\rightarrow the boy likes a girl$ (규칙 5 적용)

상향식(bottom-up) 파싱

- 문장에서 시작하여 문장 시작 기호 S 방향으로 파싱 트리를 생성
- 생성 규칙의 오른쪽(RHS, right hand side)에 대응하는 부분을 해당 규칙의 왼쪽 부분(LHS)으로 변화하는 과정 반복

The boy likes a girl.

- | | |
|---|---|
| 1) $S \rightarrow NP VP$ | <i>The boy likes a girl.</i> $\rightarrow$ <i>DetP boy likes a girl</i> (규칙 3 적용) |
| 2) $VP \rightarrow V NP$ | $\rightarrow$ <i>DetP N likes a girl</i> (규칙 5 적용) |
| 3) $NP \rightarrow DetP N \mid AdjP NP$ | $\rightarrow$ <i>NP likes a girl</i> (규칙 3 적용) |
| 4) $AdjP \rightarrow Adj \mid Adv AdjP$ | $\rightarrow$ <i>NP V a girl</i> (규칙 6 적용) |
| 5) $N \rightarrow boy \mid girl$ | $\rightarrow$ <i>NP V DetP girl</i> (규칙 3 적용) |
| 6) $V \rightarrow sees \mid likes$ | $\rightarrow$ <i>NP V DetP N</i> (규칙 5 적용) |
| 7) $Adj \rightarrow big \mid small$ | $\rightarrow$ <i>NP VNP</i> (규칙 3 적용) |
| 8) $Adv \rightarrow very$ | $\rightarrow$ <i>NP VP</i> (규칙 2 적용) |
| 9) $DetP \rightarrow a \mid the$ | $\rightarrow S$ (규칙 1 적용) |

구문 분석의 어려움

- 하나의 문장이 여러 구문 구조로 해석될 수 있는 **중의성**(重義性)
- **구조적 중의성**(structural ambiguity)
 - 하나의 문장이 다수의 구조로 해석될 수 있는 성질
- **어휘적 중의성**(lexical ambiguity)
 - 하나의 단어가 여러 품사로 사용될 때 발생하는 다수 구조로 해석될 수 있는 성질

구조적 중의성(structural ambiguity)

Tom saw Jerry in the park.

$S \rightarrow NP VP$

$NP \rightarrow N \mid DetP N \mid NP PP$

$VP \rightarrow V NP \mid VP PP$

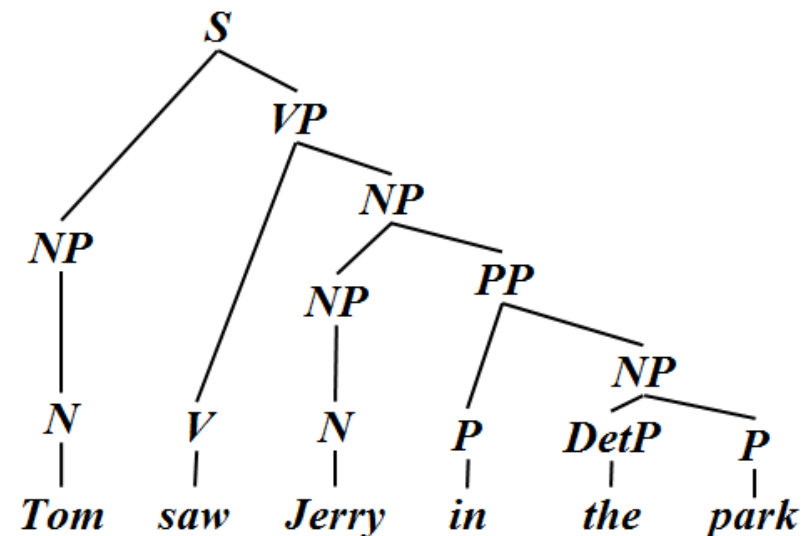
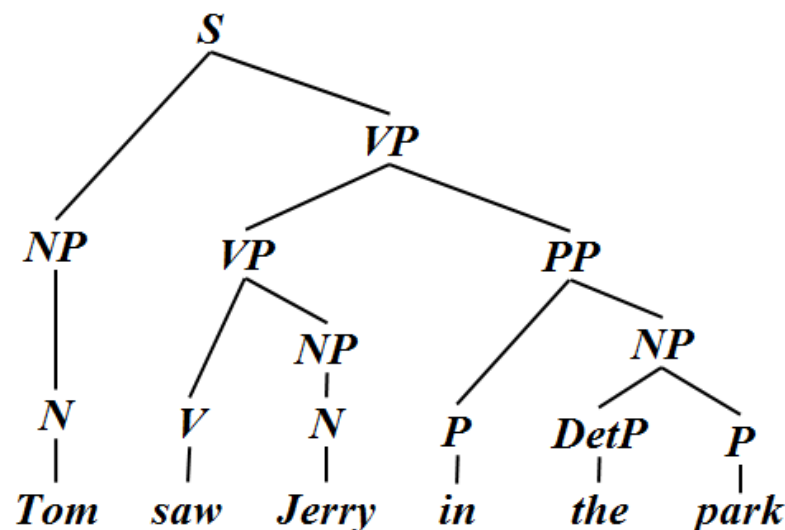
$PP \rightarrow P NP$

$N \rightarrow Tom \mid Jerry \mid park$

$V \rightarrow saw \mid caught$

$DetP \rightarrow a \mid the$

$P \rightarrow in$



어휘적 중의성(lexical ambiguity)

Time flies like an arrow.

$S \rightarrow NP VP$

$NP \rightarrow N \mid DetP N \mid Adj$

$VP \rightarrow V \mid VP NP \mid VP PP$

$PP \rightarrow P NP$

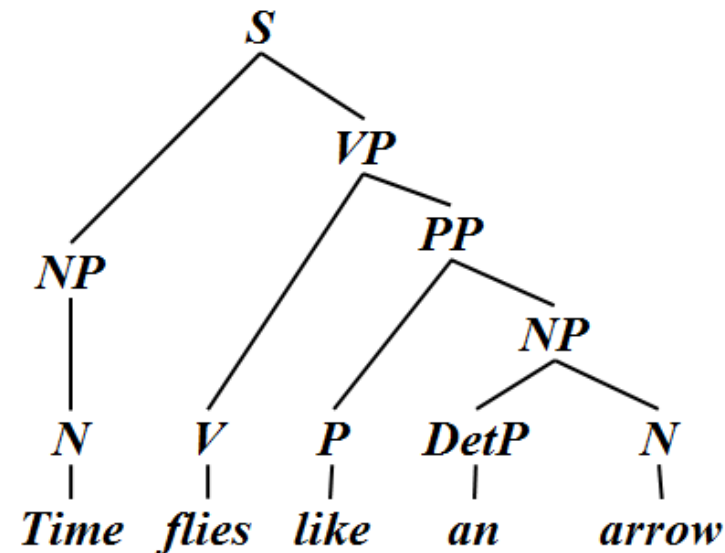
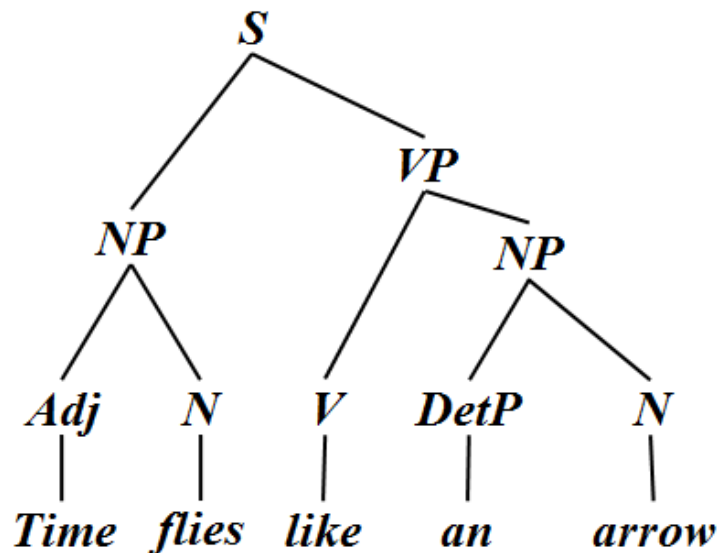
$N \rightarrow Time \mid flies \mid arrow$

$V \rightarrow flies \mid like$

$DetP \rightarrow a \mid an \mid the$

$Adj \rightarrow time$

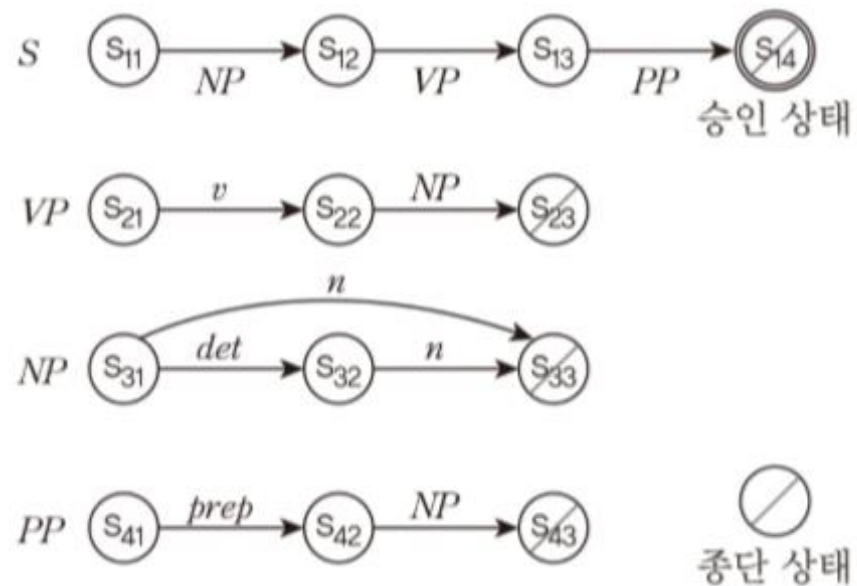
$P \rightarrow like$



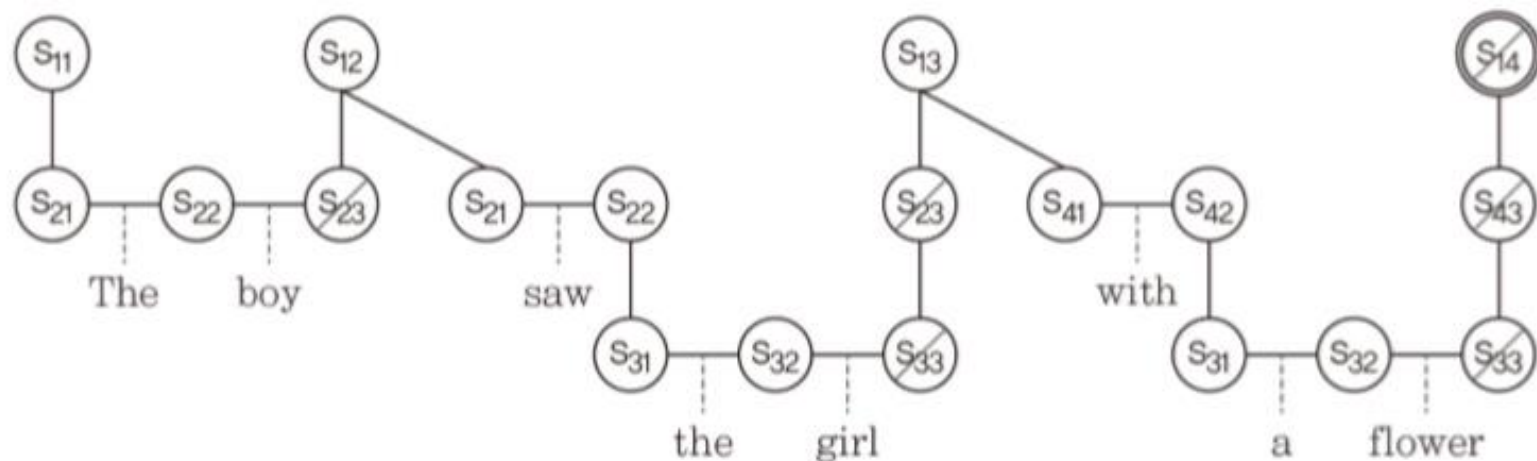
확장 전이망(augmented transition network) 파싱 기법

- 언어의 각 생성문법을 **유한상태기계**(finite state automata)로 표현
 - **노드**: 상태
 - 방향성 있는 **링크**(directed link): 전이(transition)
- **링크**
 - 단어, 품사, 다른 전이망 호출, 분기 등의 라벨 부착
 - 다른 전이망 호출 라벨이 있으면 해당 전이망을 이용하여 해당 전이부분 처리
- **파싱 결과**
 - 문장이 주어지면 시작 상태 노드에서 시작하여 전이를 통해 승인상태로 도달하게 하는 전이 과정

확장 전이망(augmented transition network) 파싱 기법



The boy saw the girl with a flower.



확장 전이망(augmented transition network) 파싱 기법

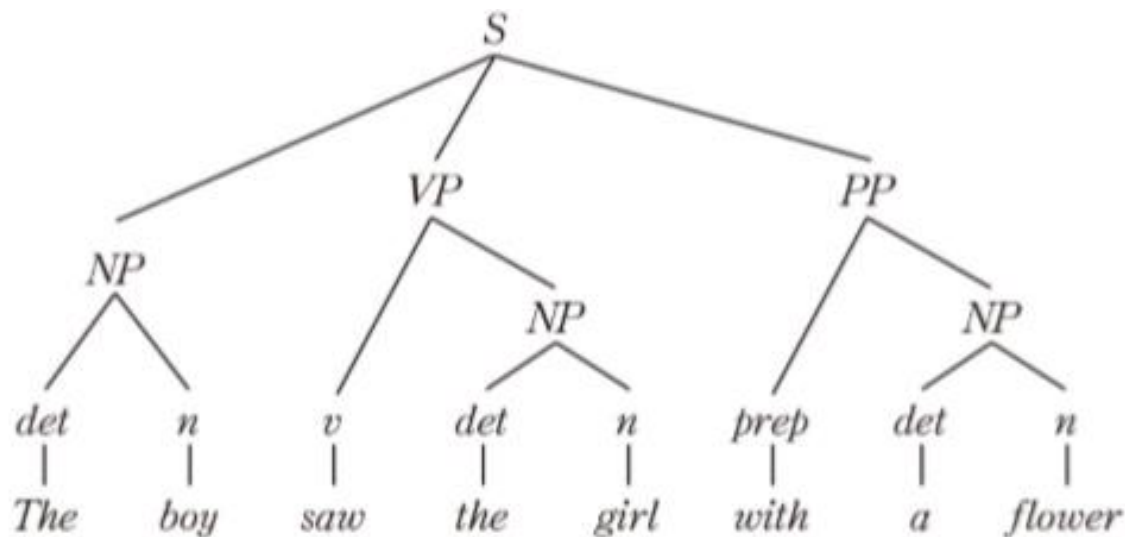
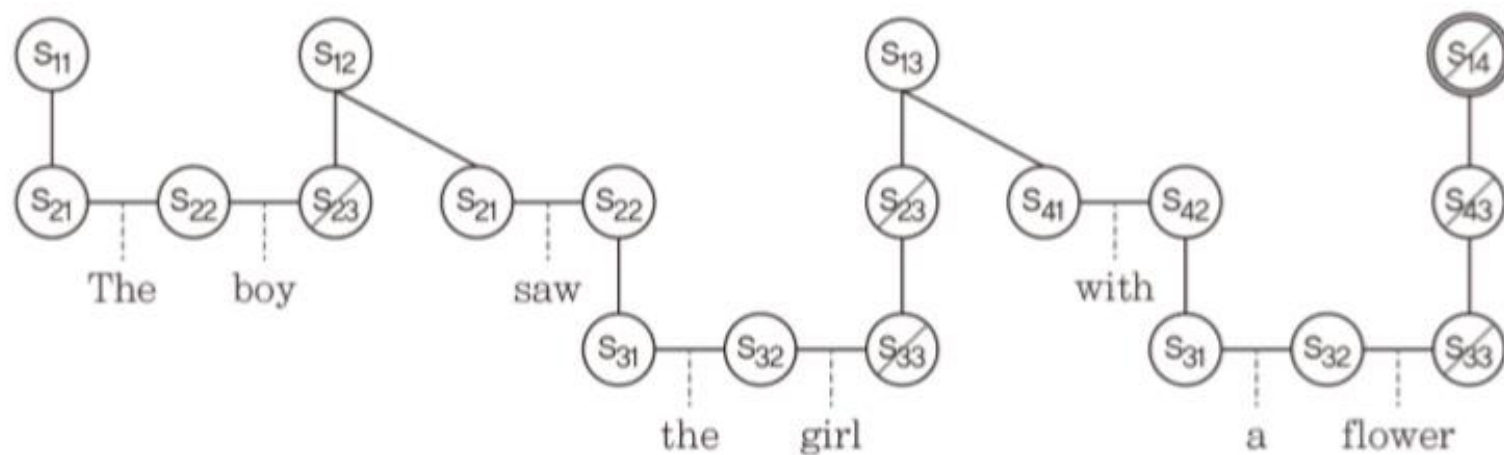


차트 파싱(chart parsing)

- 파싱의 중간과정에 부분적으로 완성되는 모든 구조들을 **차트(chart)**라는 **자료구조**에 저장
- 파싱 경로에 따라 **저장된 정보를 재활용**할 수 있게 만드는 **동적 계획법(dynamic programming)**에 기반한 방법
- **상향식(bottom-up)** 파싱에서 주로 이용
- 예. **CKY** (Cock-Kasami-Younger) **파싱** 알고리즘

CKY(Cock-Kasami-Younger) 파싱 알고리즘

- 차트 파싱의 일종
- Dynamic Programming 형태로 파싱 수행
- 상향식(bottom-up) 파싱

$$S \rightarrow NP VP$$
$$NP \rightarrow DetP N$$
$$VP \rightarrow V NP \mid VP PP$$
$$PP \rightarrow P NP$$
$$N \rightarrow boy \mid girl \mid flower \mid saw$$
$$V \rightarrow saw$$
$$P \rightarrow with$$

The boy saw the girl with a flower.

CKY(Cock-Kasami-Younger) 파싱 알고리즘

The boy saw the girl with a flower.

<i>DetP</i>							
the	<i>N</i>						
	boy	<i>V, N</i>					
		saw	<i>DetP</i>				
			the	<i>N</i>			
				girl	<i>P</i>		
					with	<i>DetP</i>	
						a	<i>N</i>
							flower

CKY(Cock-Kasami-Younger) 파싱 알고리즘

The boy saw the girl with a flower.

<i>DetP</i>	<i>NP</i>						
the	<i>N</i>						
	boy	<i>V, N</i>					
		saw	<i>DetP</i>	<i>NP</i>			
			the	<i>N</i>			
				girl	<i>P</i>		
					with	<i>DetP</i>	<i>NP</i>
						a	<i>N</i>
							flower

CKY(Cock-Kasami-Younger) 파싱 알고리즘

The boy saw the girl with a flower.

<i>DetP</i> ④	<i>NP</i> ②			<i>S</i>			<i>S</i> ①
the	<i>N</i> ⑤						
	boy	<i>V</i> ⑧, <i>N</i>		<i>VP</i> ⑥			<i>VP</i> ③
		saw	<i>DetP</i> ①	<i>NP</i> ⑨			
			the	<i>N</i> ②			
				girl	<i>P</i> ③		<i>PP</i> ⑦
					with	<i>DetP</i> ⑤	<i>NP</i> ④
						a	<i>N</i> ⑥
							flower

CKY(Cock-Kasami-Younger) 파싱 알고리즘

<i>DetP</i> ④	<i>NP</i> ②			<i>S</i>			<i>S</i> ①
the	<i>N</i> ⑤						
	boy	<i>V</i> ⑧, <i>N</i>		<i>VP</i> ⑥			<i>VP</i> ③
		saw	<i>DetP</i> ❶	<i>NP</i> ⑨			
			the	<i>N</i> ❷			
				girl	<i>P</i> ❸		<i>PP</i> ⑦
					with	<i>DetP</i> ⑤	<i>NP</i> ④
						a	<i>N</i> ⑥
							flower

S

NP

Detp
The

N
boy

VP

VP

V
saw

NP

Detp
the

N
girl

PP

P
with

NP

Detp
a

N
flower

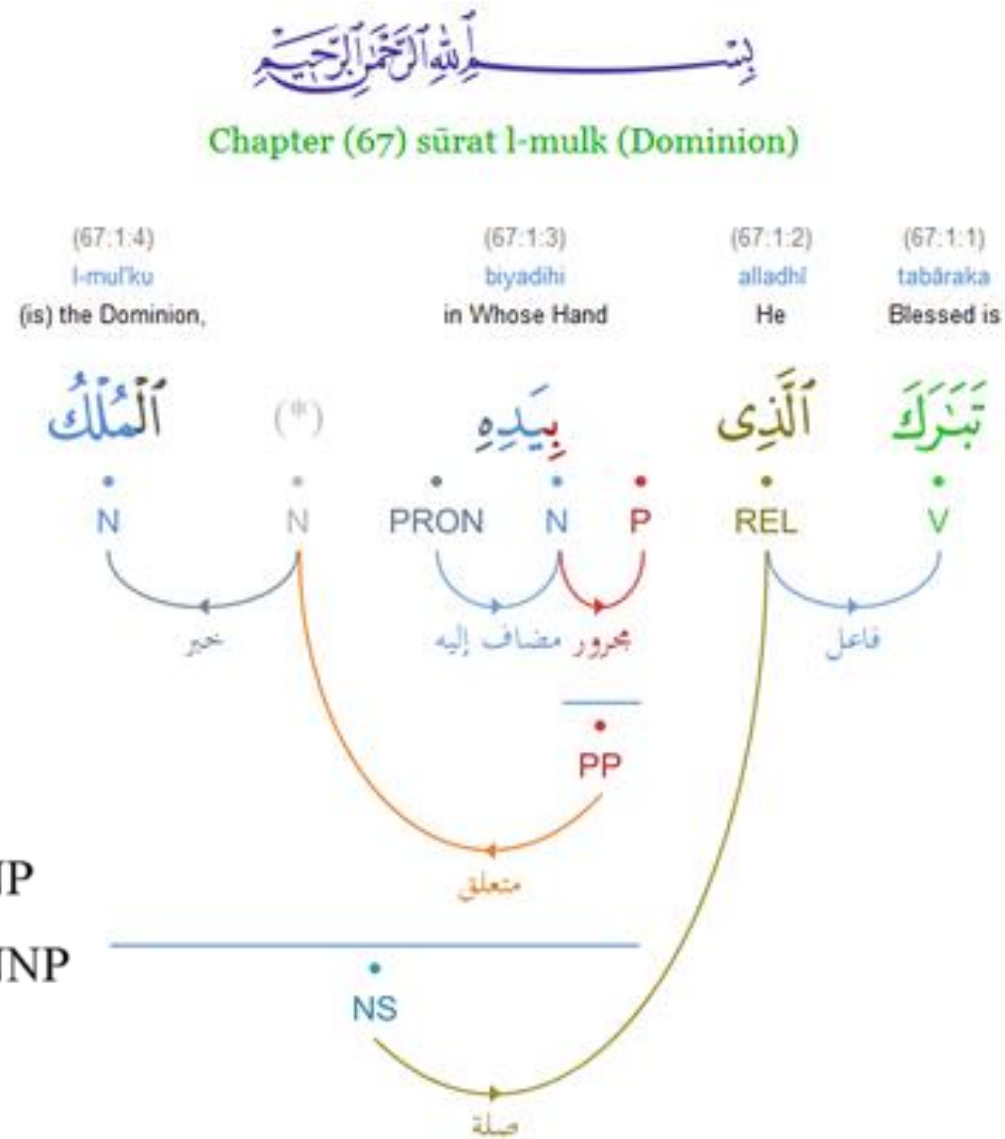
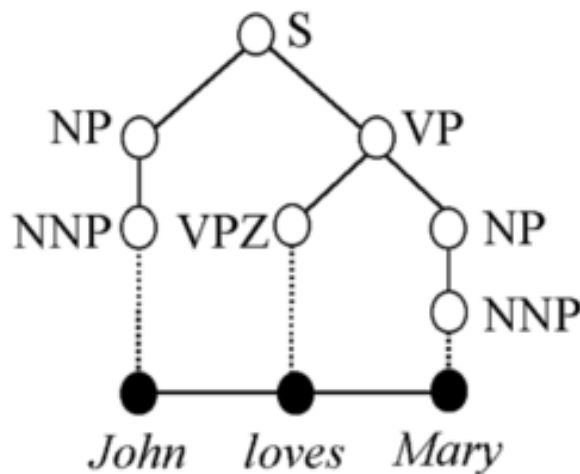
기계학습 기반 구문분석

- 구문 분석이 된 학습 데이터를 사용하여 구문 분석을 하는 모델을 학습을 통해 구축
- 사람이 구구조 문법을 정의할 필요가 없음
- 구문 분석된 정보를 포함한 말뭉치를 만들어서 제공
- 높은 신뢰도의 말뭉치 구축 필요
- **적용 모델**
 - SVM
 - 조건부 랜덤 필드(CRF) 모델
 - 딥러닝 신경망

말뭉치

- 문장 하나하나를 구문 분석하여 말뭉치(corpus)로 구성해 놓은 것
- 구문 구조를 트리(tree)형태로 표현
- 한국어 : '21세기 세종계획' 말뭉치
- 영어 : Penn Treebank 말뭉치

```
(S (NP (NNP John))
  (VP (VPZ loves)
    (NP (NNP Mary))))
(. .))
```



The Penn Treebank : 품사 태깅 테스트(POS-tagged text)

Part-of-speech tagging pipeline

Output of tagger

Battle-tested/NNP Japanese/NNP industrial/JJ managers/NNS
here/RB always/RB buck/VB up/IN nervous/JJ newcomers/NNS
with/IN the/DT tale/NN of/IN the/DT first/JJ of/IN
their/PP\$ countrymen/NNS to/TO visit/VB Mexico/NNP ,/,
a/DT boatload/NN of/IN samurai/NNS warriors/NNS blown/VBN
ashore/RB 375/CD years/NNS ago/RB ./.

Hand-corrected by annotator

Battle-tested/NNP*/JJ Japanese/NNP*/JJ industrial/JJ
managers/NNS here/RB always/RB buck/VB*/VBP up/IN*/RP
nervous/JJ newcomers/NNS with/IN the/DT tale/NN of/IN
the/DT first/JJ of/IN their/PP\$ countrymen/NNS to/TO
visit/VB Mexico/NNP ,/, a/DT boatload/NN of/IN
samurai/NNS*/FW warriors/NNS blown/VBN ashore/RB 375/CD
years/NNS ago/RB ./.

Final version

Battle-tested/JJ Japanese/JJ industrial/JJ managers/NNS
here/RB always/RB buck/VBP up/RP nervous/JJ newcomers/NNS
with/IN the/DT tale/NN of/IN the/DT first/JJ of/IN
their/PP\$ countrymen/NNS to/TO visit/VB Mexico/NNP ,/,
a/DT boatload/NN of/IN samurai/FW warriors/NNS blown/VBN
ashore/RB 375/CD years/NNS ago/RB ./.

한국어 treebank

- 괄호(bracket)를 사용하여 Penn Treebank와 유사하게 구성
- 구문분석 및 형태소 분석 포함

::A;25:1: 김 특무장, 아까 김 특무장이 지난 2주 동안 공병들이 대대 지역에서 열심히 작업을 해 왔다고 그랬다.

```
(S (NP-VOC 김/NPR
    특무장/NNC)
  ./SCM
  (ADVP 아까/ADV)
  (S (NP-SBJ 김/NPR
      특무장/NNC+이/PCA)
    (VP (S-COMP (NP-ADV 지난/DAN
                  2/NNU
                  주/NNX
                  동안/NNX)
          (S (NP-SBJ 공병/NNC+들/XSF+이/PCA)
            (VP (VP (NP-ADV 대대/NNC
                    지역/NNC+에서/PAD)
                  (ADVP 열심히/ADV)
                  (VP (NP-OBJ-LV 작업/NNC+을/PCA)
                    (LV 하/VV+어/EAU)))
                오/VX+았/EPF+다고/ECS)))
            그러/VV+었/EPF+다/EFN))
    ./SFN)
```



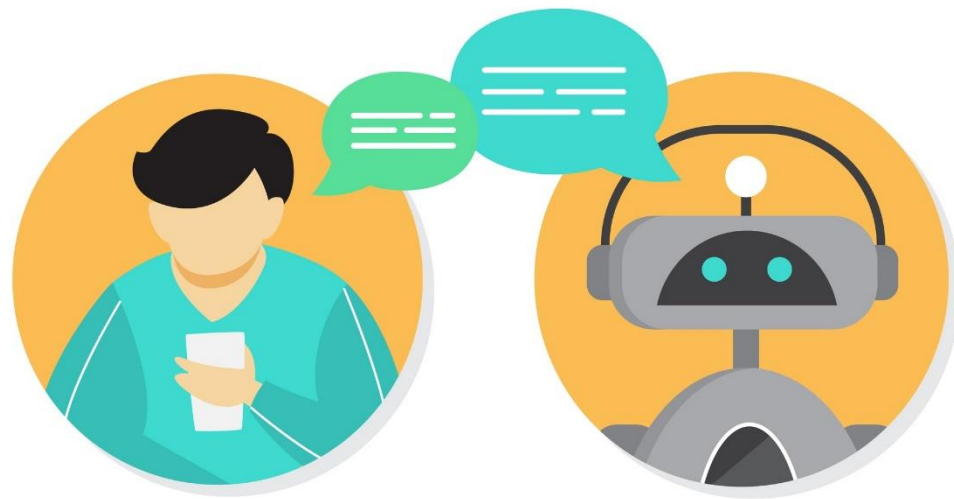
6 의미 분석

의미 분석(semantic analysis)

- **형태소 분석**과 **구문 분석** 결과를 해석하여 **문장**이 가진 **의미**를 파악하는 작업
- 형태소 각각의 의미에 대한 지식 표현 필요
- 담화가 이루어지는 상황에 대한 **세계 모델**(world model)과 **상식**(common sense)에 대한 지식 필요
- 일반적인 상황에 대한 담화를 이해하는 것은 어려움
- **담화 환경의 제약**을 통해 만족스러운 정도의 시스템 구축 가능

질의응답(question-answering) 서비스

- 의미 분석은 자연어로 제시한 질문에 대한 답변
- **검색 기반 질의응답**
 - 검색엔진과 같이 웹이나 데이터베이스에서 질문에 대한 답변을 찾아 제공
- **지식 기반 질의응답**
 - 질의의 의미를 술어 논리식(predicate logic expression)으로 변환하여, 온톨로지 등의 정형화된 지식 베이스에 대해 논리적인 추론을 하여 답변
- **혼합형 질의응답**
 - 웹 등의 텍스트 데이터 뿐만 아니라 지식 베이스도 함께 이용



단어 의미 중의성 해소(word sense disambiguation, 갈래뜻 판별)

- 특정 문장에 등장하는 단어가 어떤 의미인지 판별하는 작업

배를 타고 강을 건넜다.

배를 많이 먹었더니 배가 부르다.

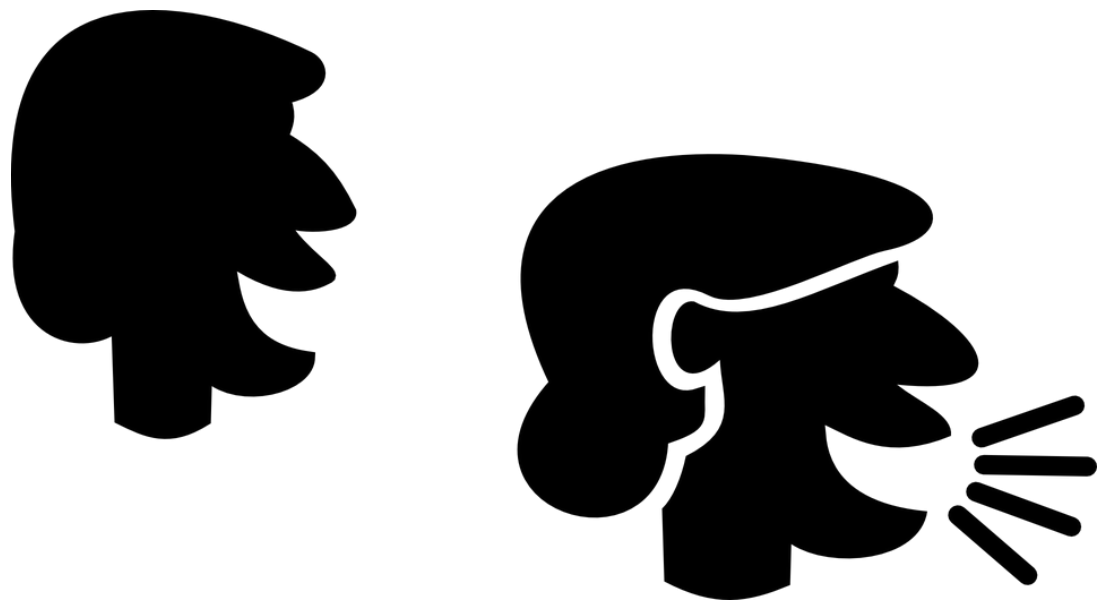
물가가 배로 올랐다.

우리 학과 축구팀은 총장 배에서 우승했다.



화행 분석

- 대화 중의 발화(utterance, 發話)가 어떤 종류인지 파악하는 것
- 설명, 의견, 동의, 거부, 감사, 예-아니오 질문, 주관식 질문, 혼잣말, 재확인 등으로 분류



문맥 함의

- 문장에 표면적으로 나타난 사실이외에 함의 된 사실을 파악하는 것

“한글은 세종대왕이 창제했다”



“한글은 조선시대에 창제되었다”

의미 분석

- 통사적으로 옳으나 의미적으로 틀린 문장이 있을 수 있음
 - 돌이 걸어간다 (cf. 사람이 걸어간다)
 - 바람이 달린다 (cf. 말이 달린다)
- 모호성 (ambiguity)
 - 말이 많다 (horse, speech)
 - 통사(統辭): 생각이나 감정을 말로 표현할 때 **완결된 내용**을 나타내는 최소의 단위



7 딥러닝 자연어 처리

단어의 수치화

- **one-hot 벡터:** 단어 별로 하나의 좌표 축을 대응시킨 공간에서, 해당되는 단어 위치에만 1을 설정하고 나머지에는 0을 설정하여, 공간 상에 단어 표현

0	0	0	1	0	...	0	0	0	0
딸기	오이	참외	수박	배	...	복숭아	사과	귤	바나나

‘수박’ = (0, 0, 0, 1, 0, ..., 0, 0, 0, 0)

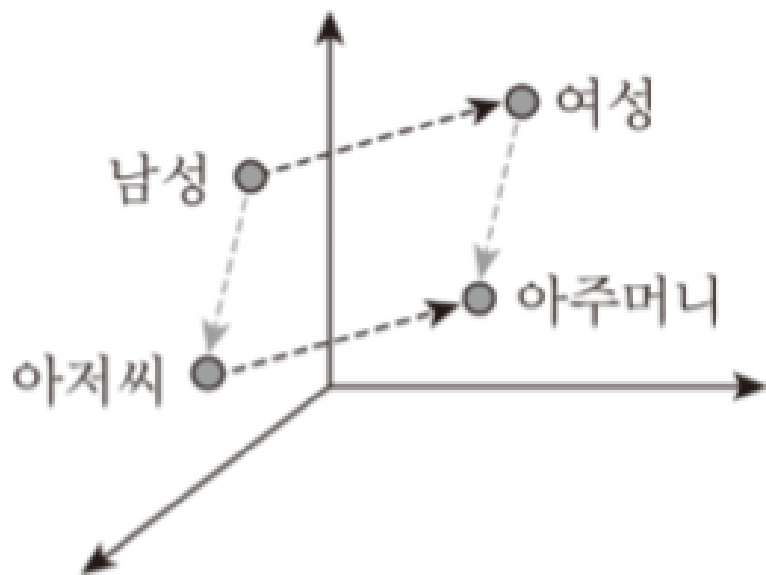
- **단점:** 단어간의 유사도를 계산하기 곤란

Word2Vec 또는 단어 임베딩(word embedding)

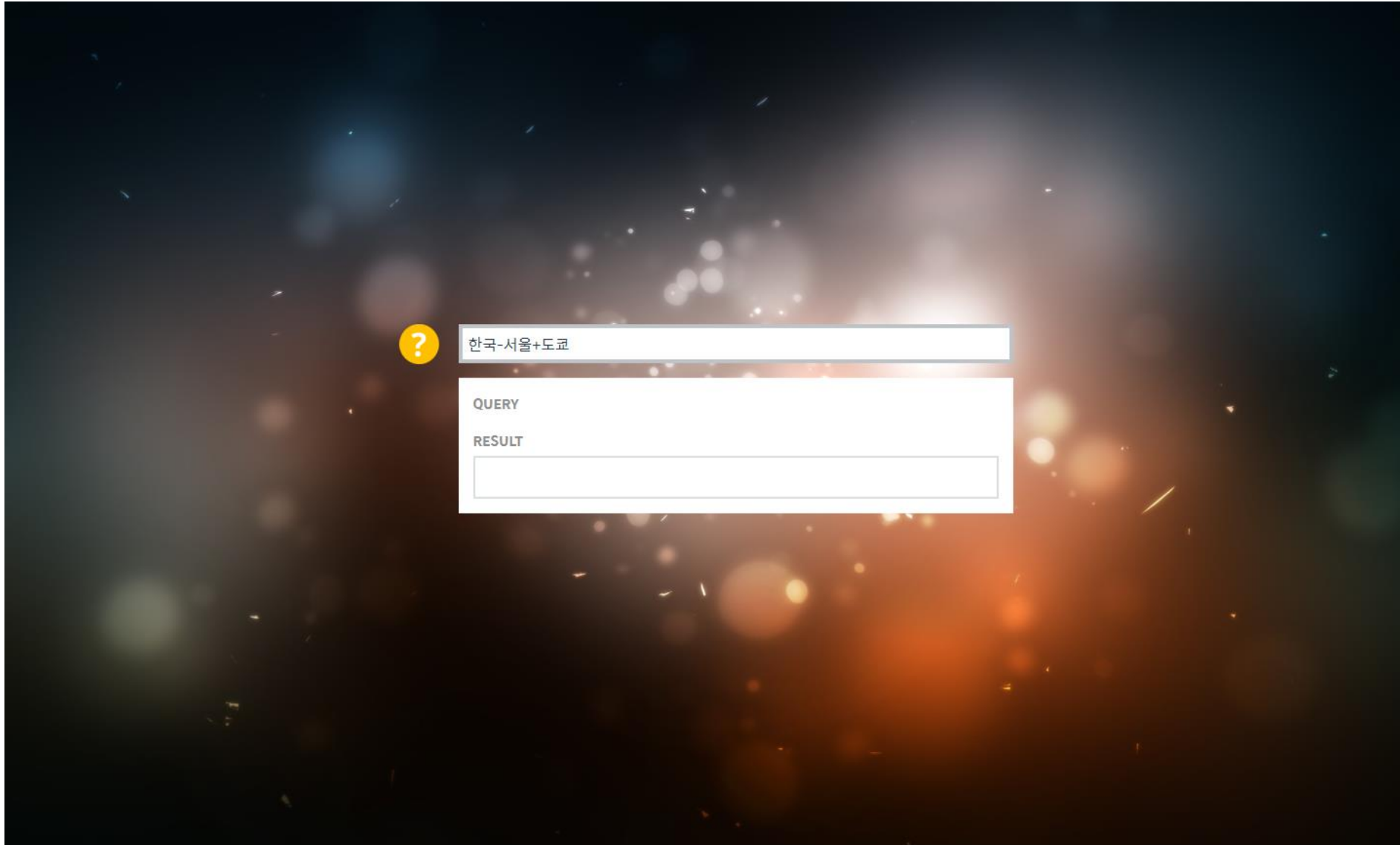
- 단어의 의미를 충분히 잘 나타내도록 단어를 공간상의 실수 벡터로 표현

-0.1	0.03	0.5	0.8	-0.32	0.21
------	------	-----	-----	-------	------

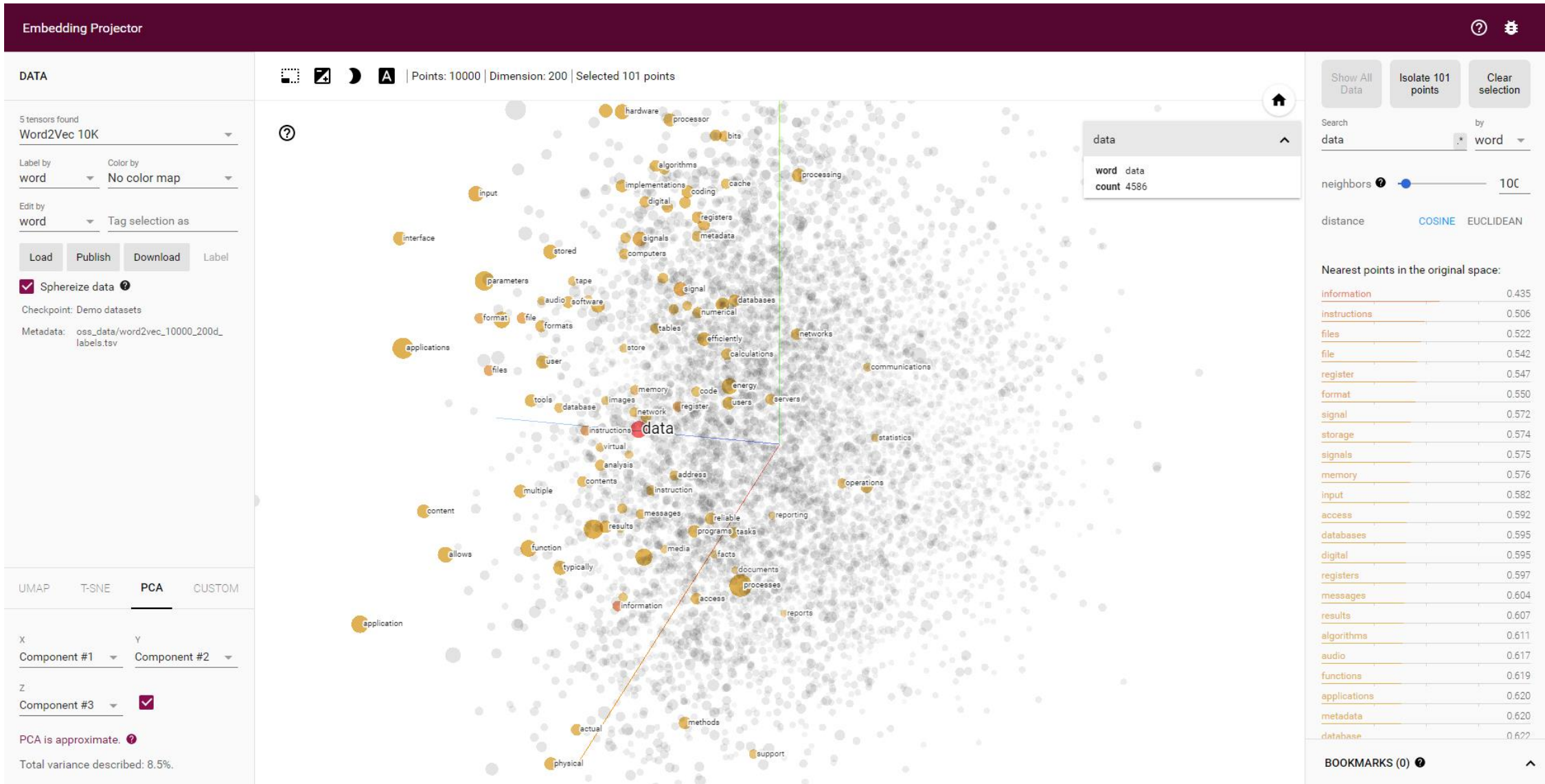
- 유사한 의미의 단어가 벡터공간 상에서 근처에 위치



Word2Vec (kor)

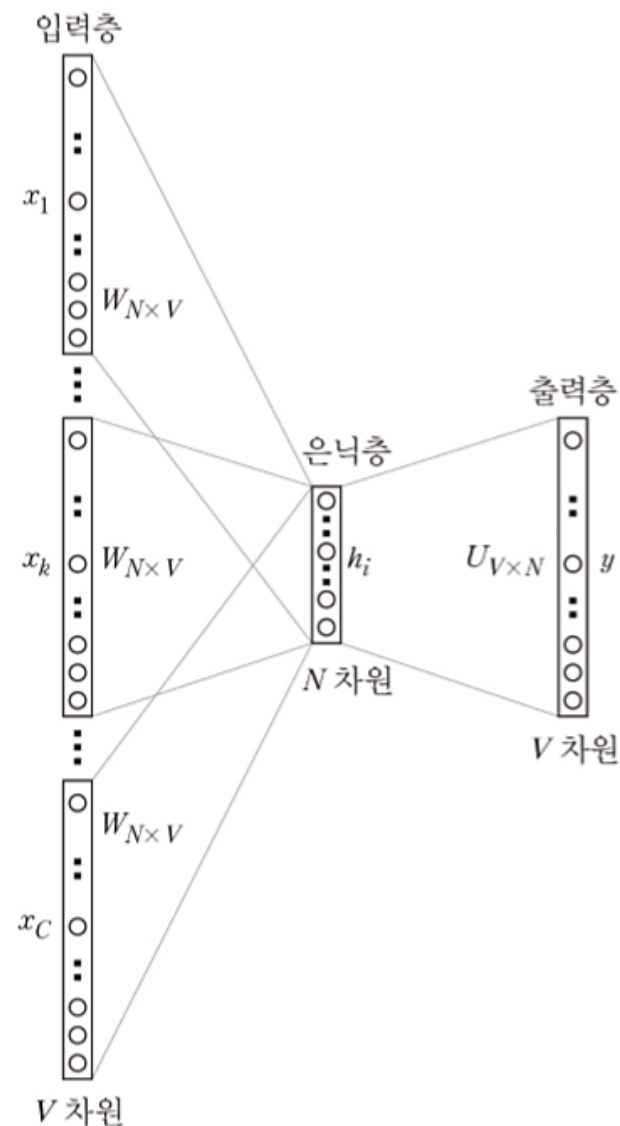


Word2Vec (TensorFlow)



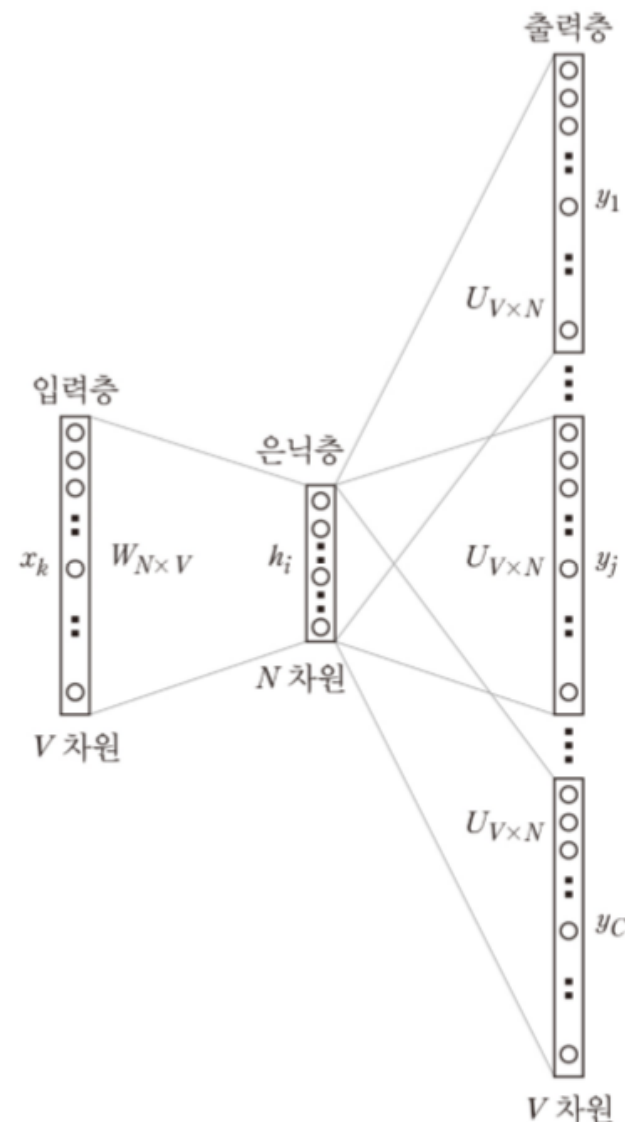
CBOW(Continuous Bag-of-Words) 모델

- V 차원의 one-hot 벡터로 표현된 단어를 N 차원의 실수 벡터로 바꾸는 역할
- **분포 가설**에 이론적 근거
 - '주변 단어들에 의해 단어 의미를 알 수 있다'
 - '나는 식사를 하기 전에 반드시 ____을 씻는다'
- 입력에 주변 단어들이 주어질 때, 출력에서는 해당 단어가 나타날 확률이 높아지도록 학습
- **단어를 표현하는 실수 벡터**
 - 학습된 모델에서 해당 단어의 노드에 연결된 가중치



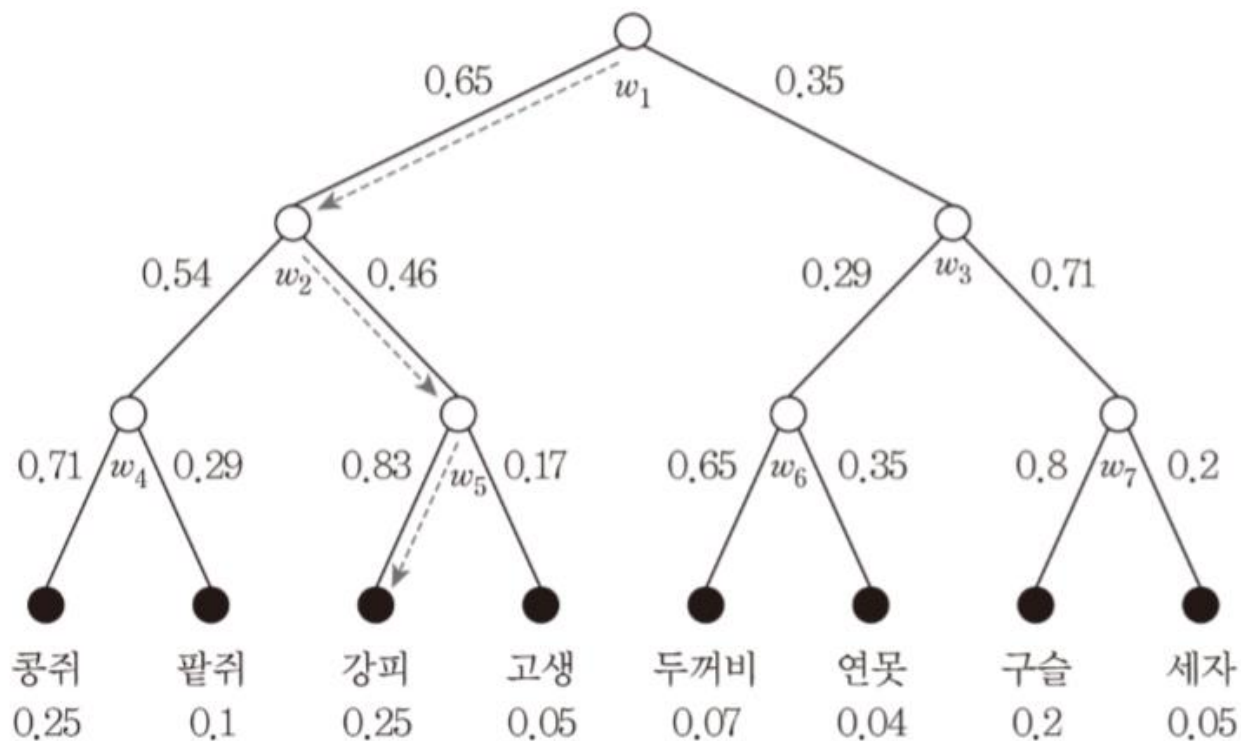
Skip-gram 모델

- CBOW와 대칭적인 구조
- 입력에 학습 대상이 되는 단어가 주어질 때, 출력에서는 해당 단어의 주위 단어들이 나타날 확률이 높아지도록 학습
- 단어를 표현하는 실수 벡터
 - 학습된 모델에서 해당 단어의 노드에 연결된 가중치



계층적 소프트맥스

- 소프트맥스값을 계산할 때, 분모에서 모든 노드에 대한 계산 결과값이 필요
→ 높은 계산 비용
- 소프트맥스를 이진 트리로 근사하여 계산시간을 절감하는 방법



$$p('강피') = v_{1l}v_{2r}v_{3l} = \sigma(w_1 \cdot h)(1 - \sigma(w_2 \cdot h))\sigma(w_3 \cdot h)$$

반례 표본 추출

- 소프트맥스를 사용할 때 매번 많은 출력 벡터를 계산하는 대신, **표본을 추출**해서 이들에 대해서만 계산하는 방법
- 표본 구성: 학습 데이터에 주어진 전체 출력 단어들 w_i 와 소수의 반례 단어들 w_{n_j} 로 구성
- 각 학습 데이터의 목표 출력 단어 w_i 와 반례 단어들 w_{n_j} 에 한 오차 함수 E 를 최소화하도록 학습

$$E = -\log \sigma(u_i \cdot h) - \sum_{w_{n_j}} \log \sigma(-u_{n_j} \cdot h)$$

- 오차 함수가 소프트맥스의 값을 직접 사용하지 않기 때문에, 일반 소프트맥스를 사용하는 경우에 비하여 매우 짧은 시간에 CBOW나 Skip-gram 모델 학습 가능

단어 벡터 표현의 활용

- **단어 간의 유사도**는 실수 벡터의 거리를 사용하여 직접 계산 가능
 - 유의어, 어근 비교, 시제 등에 한 정보 추출 또는 분석
 - 품사 태깅, 의미 분석, 관계 추출, 단위 의미 식별 등
- 수치 기반의 **다양한 기계학습 알고리즘**의 자연어 처리 **적용 가능**
 - 기계번역
 - 영상 주석달기 등

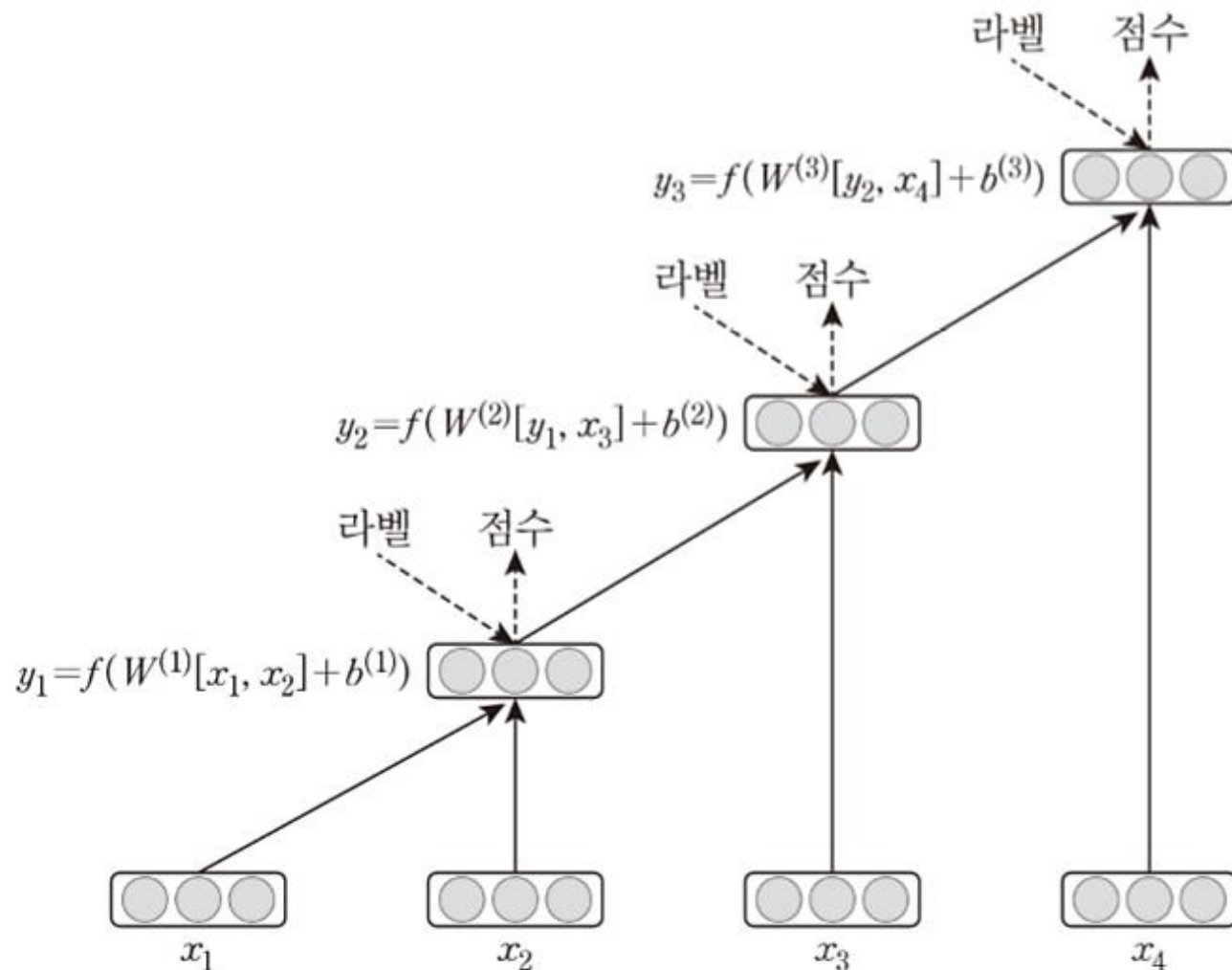
언어 모델(language model)

- 직전 n 개의 단어로 부터 다음에 나올 단어를 예측하는 모델

$$p(w_t | w_{t-1}, \dots, w_{t-n})$$

- 구와 문장의 벡터 표현

- 구(phase)와 문장(sentence)의 벡터표현



기계 번역(machine translation)

- 입력 언어의 문장 s 를 다른 언어의 문장 t 로 변환할 때, 제로 된 번역일 확률이 가장 큰 문장 t' 을 선택하는 작업

$$t' = \operatorname{argmax}_t p(t|s)$$

■ 전통적 기계 번역

- 입력 문장 → 단어나 구의 서열 또는 부분 트리의 집합으로 분할하여 번역 → 조각 번역된 것들 결합 → 확률이 가장 큰 것 선택

■ 통계적 기계 번역

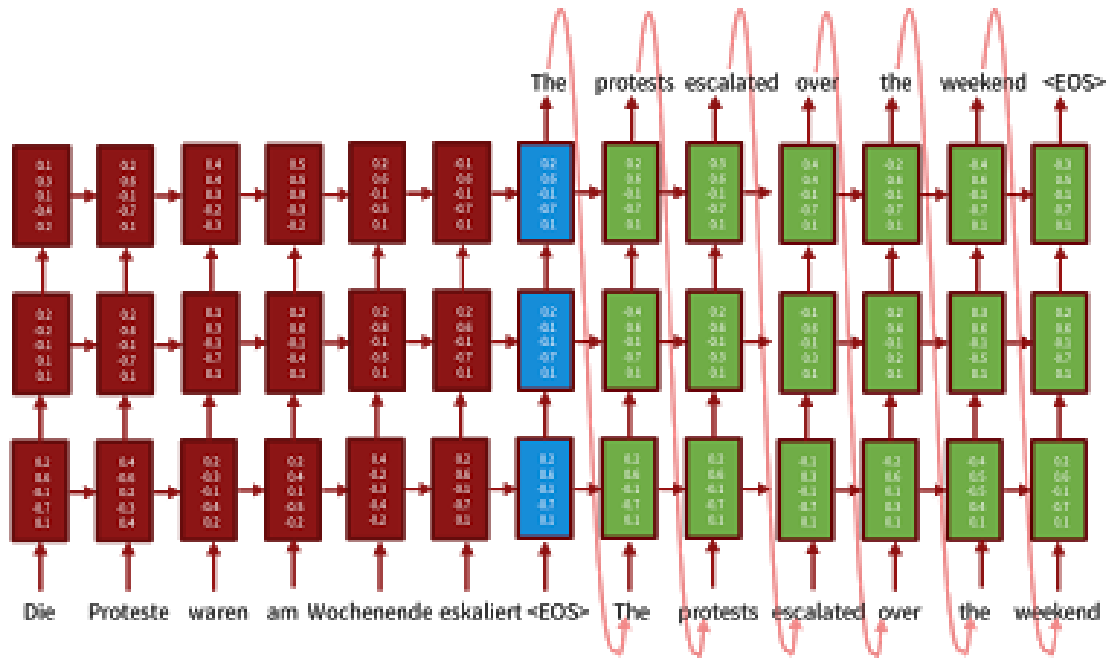
- 같은 내용을 표현하는 두 언어의 문장 쌍들을 많이 사용하여, 통계적으로 대응되는 단어, 구, 또는 문장을 찾아 번역

■ 신경망 기계 번역

- 번역 자체를 딥러닝으로 수행

Language Translation

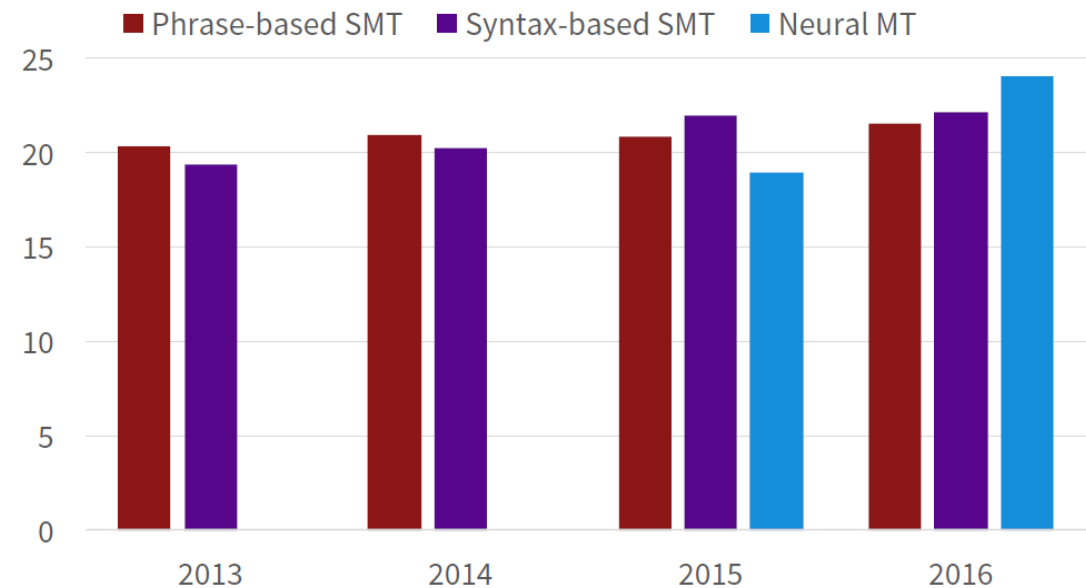
- Sequence-to-sequence models with LSTMs and attention



Source Luong, Cho, Manning ACL Tutorial 2016.

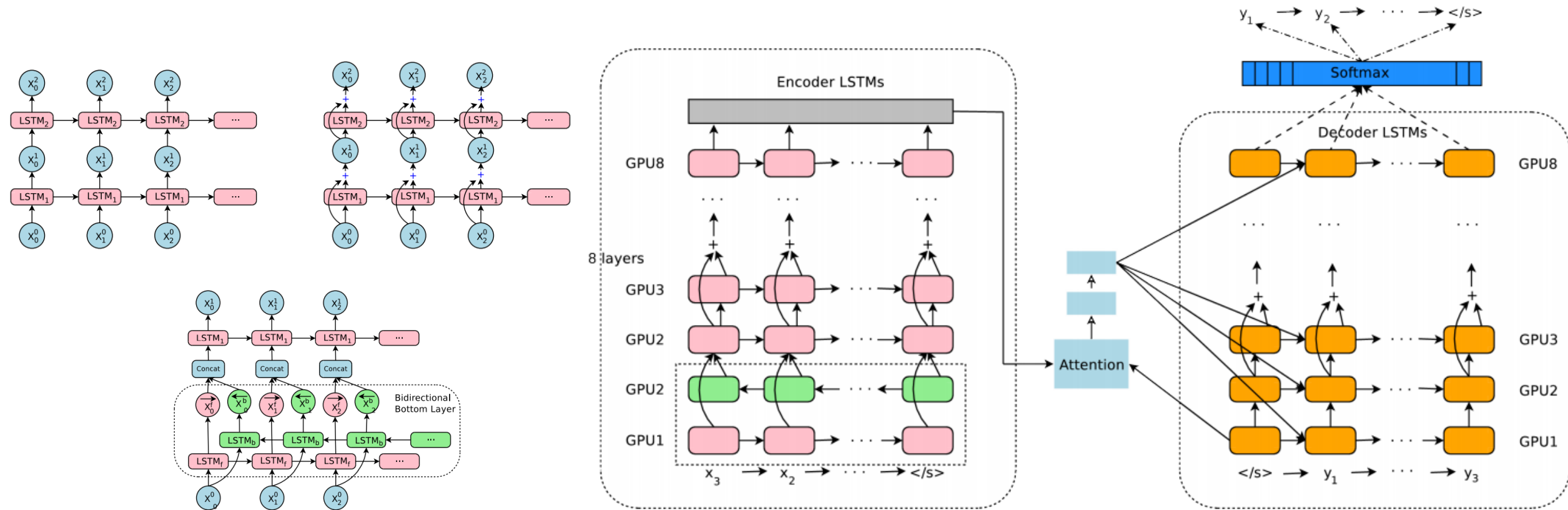
Progress in Machine Translation

[Edinburgh En-De WMT newstest2013 Cased BLEU; NMT 2015 from U. Montréal]



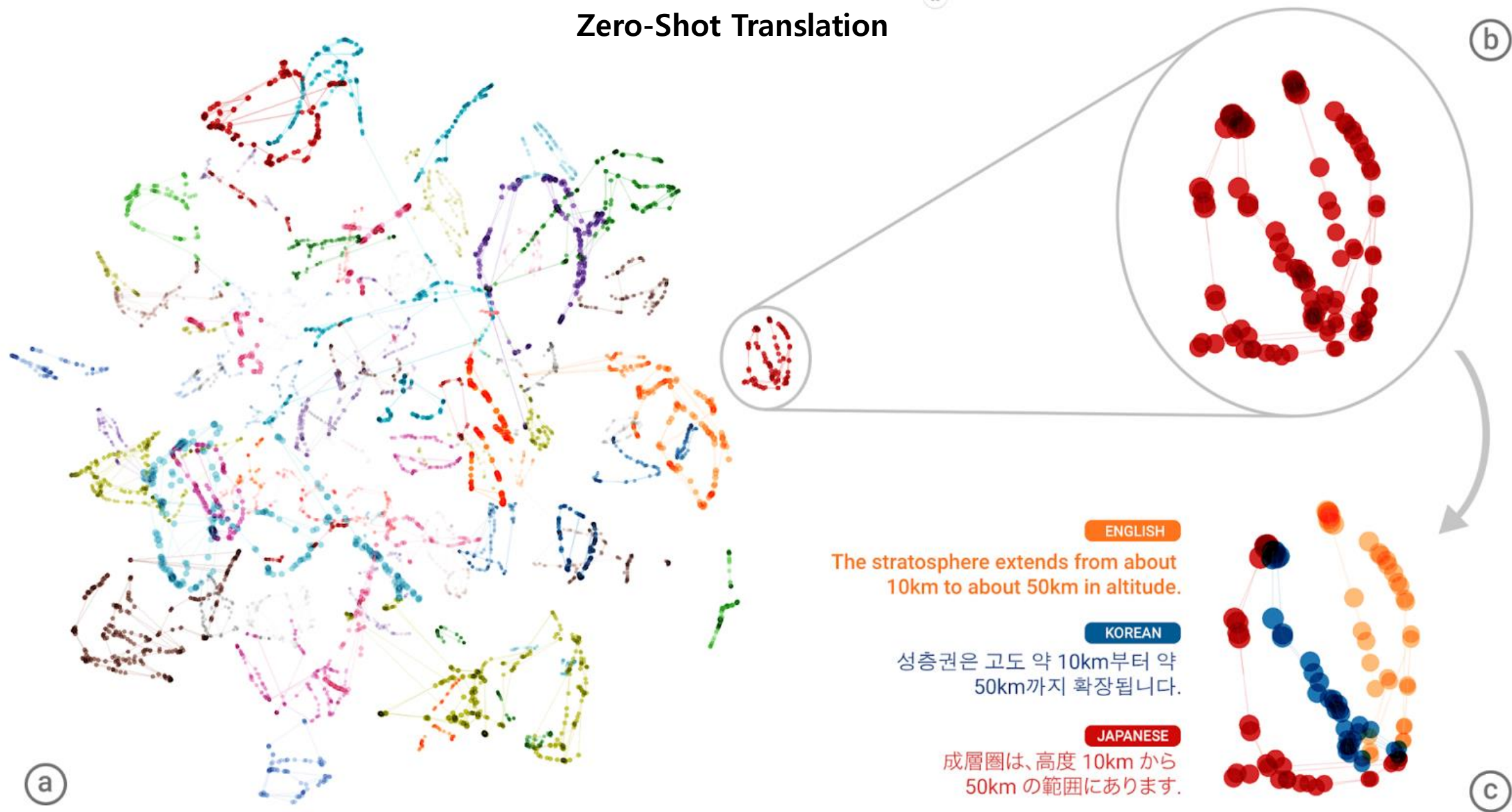
From [Sennrich 2016, http://www.meta-net.eu/events/meta-forum-2016/slides/09_sennrich.pdf]

Google Neural Machine Translation (GNMT)



Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation, <https://arxiv.org/abs/1609.08144>

Zero-Shot Translation



Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation, <https://arxiv.org/abs/1611.04558>

사람이 준 주제

충격적인 발견에서, 과학자들은 이전에 탐험되지 않았던 안데스 산맥의 외딴 계곡에 살고 있는 유니콘 무리들을 발견했다. 연구자들에게 더욱 놀라운 것은 유니콘들이 완벽한 영어를 구사한다는 사실이었다.

번역 by Naver Papago

AI가 쓴 이야기

그 과학자는 개체군의 이름을 그들의 독특한 뿔인 오비드의 유니콘의 이름을 따서 지었다. 이 네 개의 뿔이 달린 은백색의 유니콘들은 이전에는 과학에 알려지지 않았다.

이제 거의 2세기가 지난 후, 무엇이 이 기묘한 현상을 촉발시켰는지에 대한 미스터리가 마침내 풀리게 된다.

라파즈 대학의 진화생물학자 조르헤 페레스 박사와 여러 동료들이 안데스 산맥을 탐험하고 있을 때 그들은 다른 동물이나 인간이 없는 작은 계곡을 발견했다. 페레스는 그 계곡에 암봉과 은설기 두 봉우리에 둘러싸인 자연분수로 보이는 것이 있다는 것을 알아챘다.

그러자 페레스와 다른 사람들은 위험을 무릅쓰고 계곡으로 더 들어갔다. 페레스는 "한 봉우리 꼭대기에 다다랐을 때 물이 푸른색으로 보였고 결정체도 몇 개 있었다"고 말했다.

페레스와 그의 친구들은 유니콘 무리를 보고 놀랐다. 이 생물들은 너무 많이 움직이지 않고도 공중에서 볼 수 있었다. - 그들은 너무 가까이 있어서 뿔을 만질 수 있었다.

...

과학자들은 이 기괴한 생물들을 조사하는 동안 이 생물들이 꽤 규칙적인 영어를 구사한다는 것을 발견했다. 페레스는 예를 들어 그들이 방언이나 변증법 같은 공통의 '언어'를 가지고 있다는 것을 알 수 있다고 말했다.

...

Word Prediction with Transformers

Using transformers to predict next word and predict <mask> word

Deep Learning in Action

Word Prediction with Transformers

Type a word and press space to get suggestions

Top k words: 5

h

Predictions

BERT	XLNET	XLM-Roberta	BART	ELECTRA	ROBERTA

Masked Word Prediction

Type a sentence, include <mask> to predict the word and hit Predict.

Top k words: 5

Predict

Predictions

BERT	XLNET	XLM-Roberta	BART	ELECTRA	ROBERTA



<https://aibril-pi-fb-demo-korean.sk.kr.mybluemix.net/>

Personality Insights 서비스를 사용해 보세요

귀하는 귀하가 관심있는 성격의 사람이 작성한 텍스트가 필요합니다. 그것은 매일의 경험, 생각 및 반응에 관한 단어가 포함되어 있어야 합니다.통계적으로 중요한 결과를 얻으려면,적어도 3,500단어 이상, 이상적으로 6000단어가 필요합니다.100단어 정도의 데모로 실행할 수 있지만, 보다 정확한 분석을 위해서는 더 많은 단어가 필요합니다.

[초기화](#) | [사용약관\(영어\)](#)

텍스트

귀하의 트위터 성향

선택

항소이유서 – 유시민(한국어)

2012 논쟁 – 버락 오바마(영어)

반성 – 간디 (영어)

귀하의 텍스트

인공지능이 산업 전반에 영향을 미치며 기존 기술을 대체하거나 지능화되고 있다. 이러한 변화는 인공지능을 이용한 악의적인 공격과 개인정보 침해, 범죄의 지능화에도 영향을 주고 있다. 대표적으로 인공지능 발전의 핵심인 딥러닝 기술을 이용해 가짜를 만들어 내는 딥페이크(Deepfake) 기술이 있다.

딥페이크 기술을 활용한 예로는 성인비디오 출연 배우 얼굴을 연예인 등 유명인의 얼굴로 교체하거나 가짜 기사를 만들어 사회적인 혼란 조성, 특정 기업의 대표와 유사한 합성 음성을 만들어 공급사에 돈을 보내게 한 사례 등이 있다. 최근에는 구글 Vision AI가 체온계를 들고 있는 사람의 피부가 밝을 때는 체온계로 판단한 반면, 피부가 어두울 때는 총으로 인식해 논란이 된 적이 있다.

언어선택 ☐ 영어 ☒ 한국어

분석하기

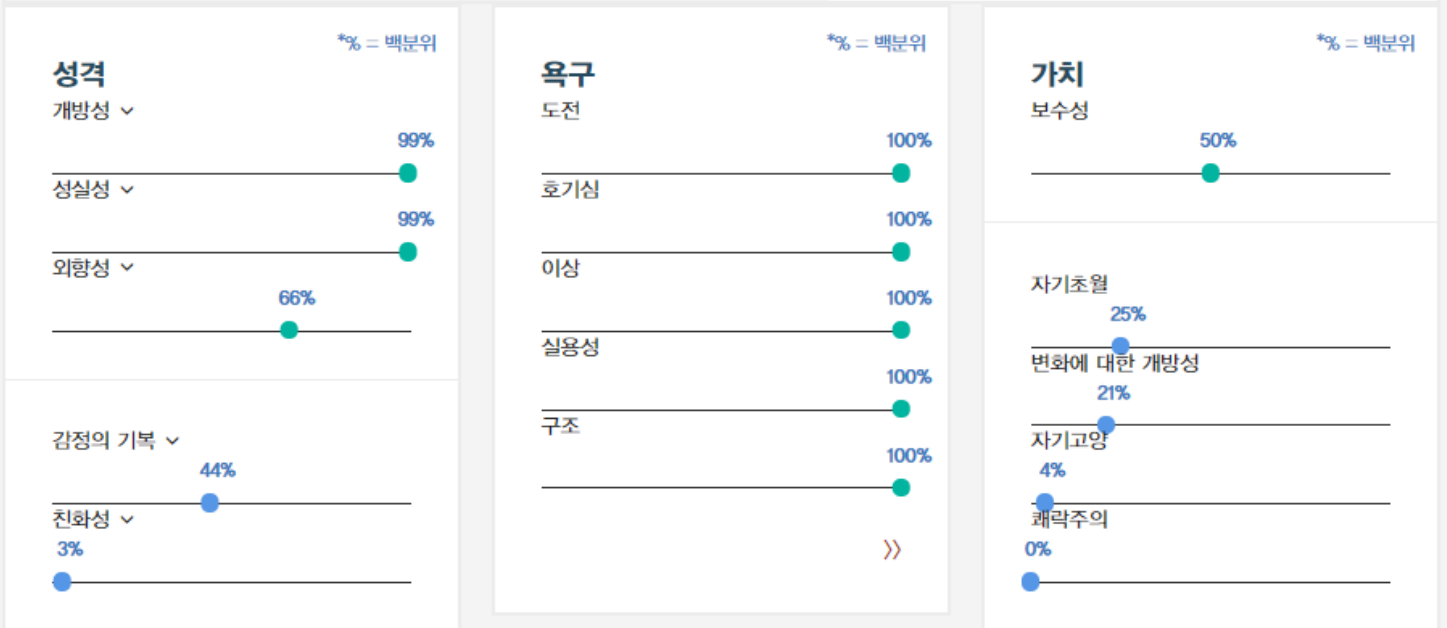
요약

귀하는 분석적입니다 그리고 상황판단이 빠릅니다.

귀하는 온순한 기질입니다: 귀하는 화가 나는데 많은 시간이 필요합니다. 귀하는 압박감을 받아도 침착합니다: 귀하는 예기치 않은 사건을 침착하고 효과적으로 처리합니다. 또한 귀하는 타협하지않습니다: 귀하는 출세하기 위해 다른 사람들을 이용하는 것이 잘못 되었다고 생각합니다.

귀하의 선택들은 발견에 대한 열망에 의해 좌우됩니다.

귀하는 인생의 즐거움을 즐기는 것, 성공 달성에 대해 비교적 신경쓰지않는 편입니다. 귀하는 단지 개인적인 즐거움보다 더 큰 목적을 가진 활동을 선호합니다. 그리고 귀하는 자신의 재능을 과시하는 방법을 거의 고려하지 않고 결정합니다.







Select Kuki's Avatar

Meet Kuki!

Mitsuku, or Kuki to her close friends, is a [record-breaking](#), five-time winner of the Loebner Prize Turing Test and the world's best conversational chatbot (according to folks like [Google AI Research](#)). As featured in the [New York Times](#), [Wall Street Journal](#), [BBC](#), [Guardian](#), [Wired](#), and [Radiolab](#).





[Get the app](#) [Help](#) [Log in](#)

The AI companion who cares

Always here to listen and talk.
Always on your side. Join the millions growing with their AI friends now!

Create your Replika

Log in



이수안 컴퓨터 연구소

suan computer laboratory