

# Word Senses: Similarity and Disambiguation

Kim, Ki Hyun and June Oh

FastCampus

2018.09.15

# Index

- Word Senses
- WordNet
- Preprocessing (Tokenization)
- TF-IDF
- Feature Vector
- Similarity Measures
- Word Sense Disambiguation
- Selectional Preference
- Conclusion

# '차' from 국립언어원

연관 의미 세부 의미 풀이

차#1 차1-1 좋은 향기나 맛이 있는 식물의 잎이나 뿌리, 열매 등을 달이거나 우려서 만든 마실 것.

차#2 차2-1 바퀴가 달려 있어 사람이나 짐을 실어 나르는 기관.

차2-2 사람이나 물건을 차에 실어 그 분량을 세는 단위.

차2-3 장기의 말 중에서 '車' 자를 새긴 말.

차#3 차3-1 둘 이상을 비교했을 때 서로 다르게 나타나는 수준이나 정도.

차3-2 어떤 수나 식에서 다른 수나 식을 뺀 나머지.

차#4 차4-1 어떤 일의 차례나 횟수를 나타내는 말.

차4-2 어떠한 일을 하던 기회나 순간.

차4-3 일정한 주기나 기간이 지난 해당 시기를 나타내는 말.

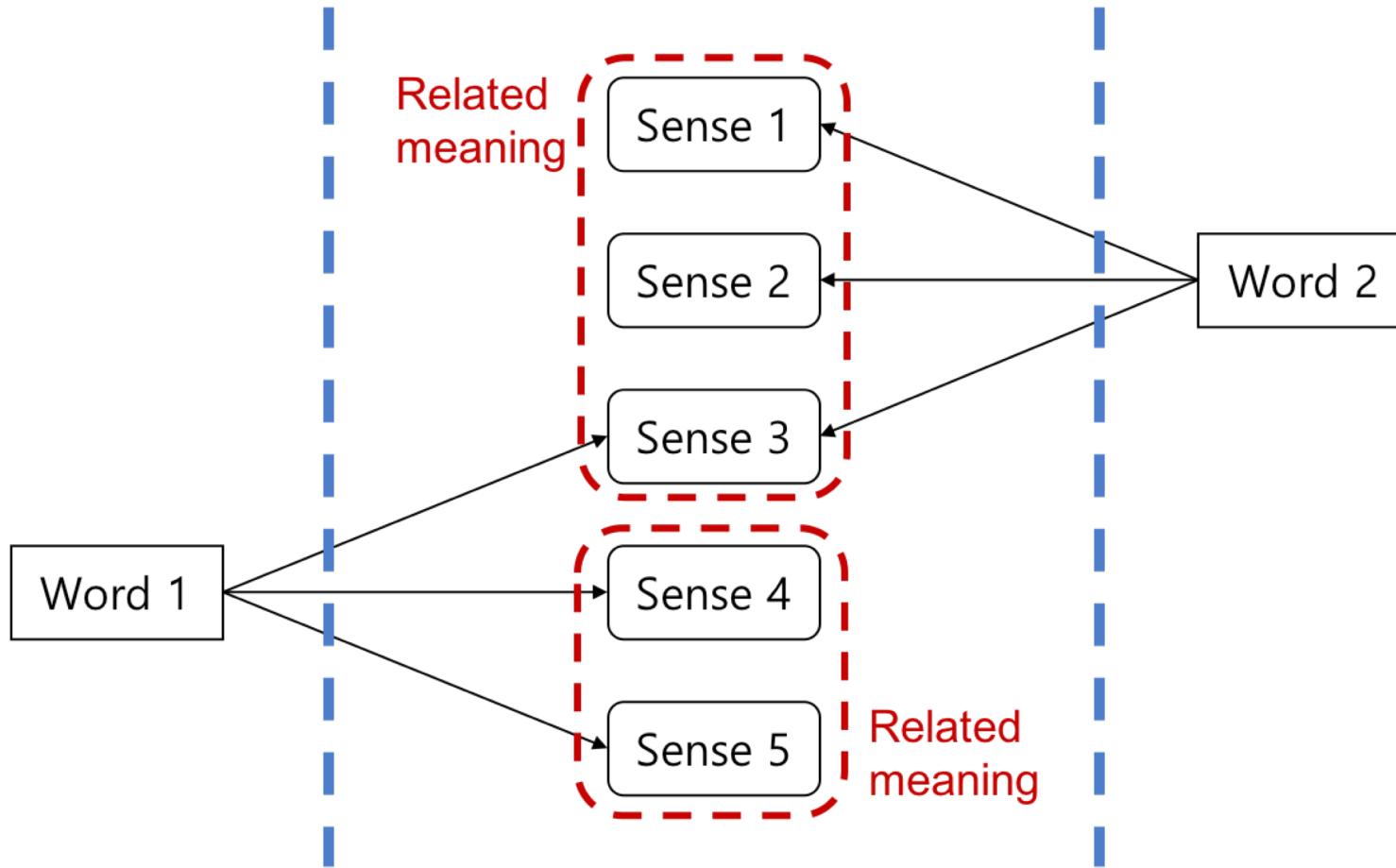
# '차' from 국립언어원

연관 의미	세부 의미	풀이
차#1	차1-1	좋은 향기나 맛이 있는 식물의 잎이나 뿌리, 열매 등을 달이거나 우려서 만든 마실 것.
차#2	차2-1	바퀴가 달려 있어 사람이나 짐을 실어 나르는 기관.
	차2-2	사람이나 물건을 차에 실어 그 분량을 세는 단위.
	차2-3	장기의 말 중에서 '車' 자를 새긴 말.
차#3	차3-1	둘 이상을 비교했을 때 서로 다르게 나타나는 수준이나 정도.
	차3-2	어떤 수나 식에서 다른 수나 식을 뺀 나머지.
차#4	차4-1	어떤 일의 차례나 횟수를 나타내는 말.
	차4-2	어떠한 일을 하던 기회나 순간.
	차4-3	일정한 주기나 기간이 지난 해당 시기를 나타내는 말.

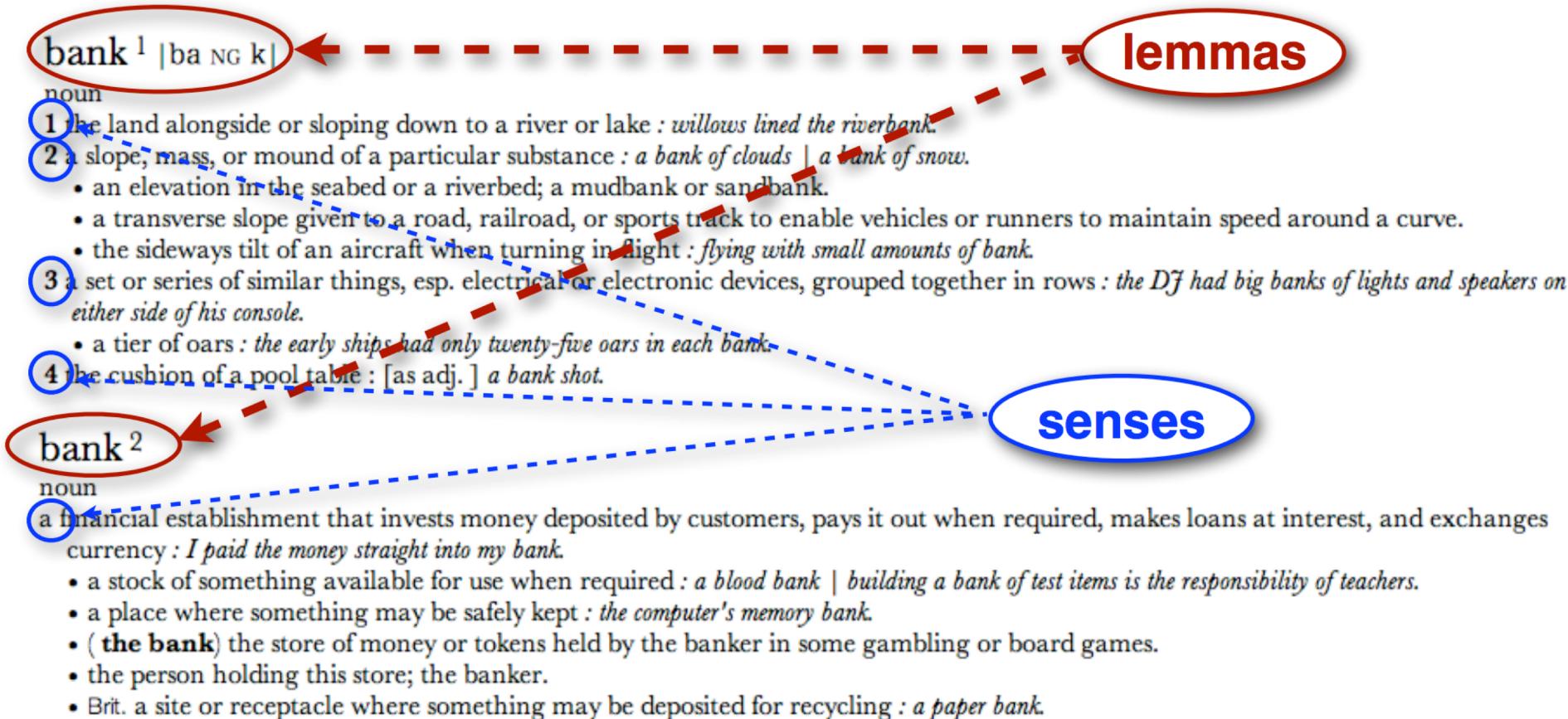
# '차' from 국립언어원

연관 의미	세부 의미	풀이
차#1	차1-1	좋은 향기나 맛이 있는 식물의 잎이나 뿌리, 열매 등을 달이거나 우려서 만든 마실 것.
차#2	차2-1	바퀴가 달려 있어 사람이나 짐을 실어 나르는 기관.
	차2-2	사람이나 물건을 차에 실어 그 분량을 세는 단위.
	차2-3	장기의 말 중에서 '車' 자를 새긴 말.
차#3	차3-1	둘 이상을 비교했을 때 서로 다르게 나타나는 수준이나 정도.
	차3-2	어떤 수나 식에서 다른 수나 식을 뺀 나머지.
차#4	차4-1	어떤 일의 차례나 횟수를 나타내는 말.
	차4-2	어떠한 일을 하던 기회나 순간.
	차4-3	일정한 주기나 기간이 지난 해당 시기를 나타내는 말.

# Word and Sense



# Word, Sense and Lemma



# Homonym and Polysemy

종류	단어 형태	의미1	의미2
동형이의어 (Homonym)	강남역	지하철역사	역 주변 상권
다의어 (Polysemy)	차	마시는 차(茶, tea)	달리는 차(車, car)

# Homonym and Polysemy

**Not a big problem**

종류	단어 형태	의미1	의미2
동형이의어 (Homonym)	강남역	지하철역사	역 주변 상권
다의어 (Polysemy)	차	마시는 차(茶, tea)	달리는 차(車, car)

# Homonym and Polysemy

종류	단어 형태	의미1	의미2
동형이의어 (Homonym)	강남역	지하철역사	역 주변 상권
다의어 (Polysemy)	차	마시는 차(茶, tea)	달리는 차(車, car)

**Big problem!**

# Homonym and Polysemy

종류	단어 형태	의미1	의미2
동형이의어 (Homonym)	강남역	지하철역사	역 주변 상권
다의어 (Polysemy)	차	마시는 차(茶, tea)	달리는 차(車, car)

**Big problem!**

**We need Word Sense Disambiguation!**

# Synonyms

형태	의미	동의어 집합(Synset)
home	home#1	place#7
home	home#2	dwelling#1, domicile#2, abode#2, habitation#2, ...
home	home#3	
home	home#4	home plate#1, home base#1, plate#1
home	home#5	base#14
...		
place	place#7	home#1
place	place#8	position#6, post#3, berth#1, office#7, spot#8, billet#3, place#8
...		

# Synonyms

형태	의미	동의어 집합(Synset)
home	home#1	place#7
home	home#2	dwelling#1, domicile#2, abode#2, habitation#2, ...
home	home#3	
home	home#4	home plate#1, home base#1, plate#1
home	home#5	base#14
...		
place	place#7	home#1
place	place#8	position#6, post#3, berth#1, office#7, spot#8, billet#3, place#8
...		

# Synonyms <> Antonyms

형태	의미	동의어 집합(Synset)
home	home#1	place#7
home	home#2	dwelling#1, domicile#2, abode#2, habitation#2, ...
home	home#3	
home	home#4	home plate#1, home base#1, plate#1
home	home#5	base#14
...		
place	place#7	home#1
place	place#8	position#6, post#3, berth#1, office#7, spot#8, billet#3, place#8
...		

# Taxonomy

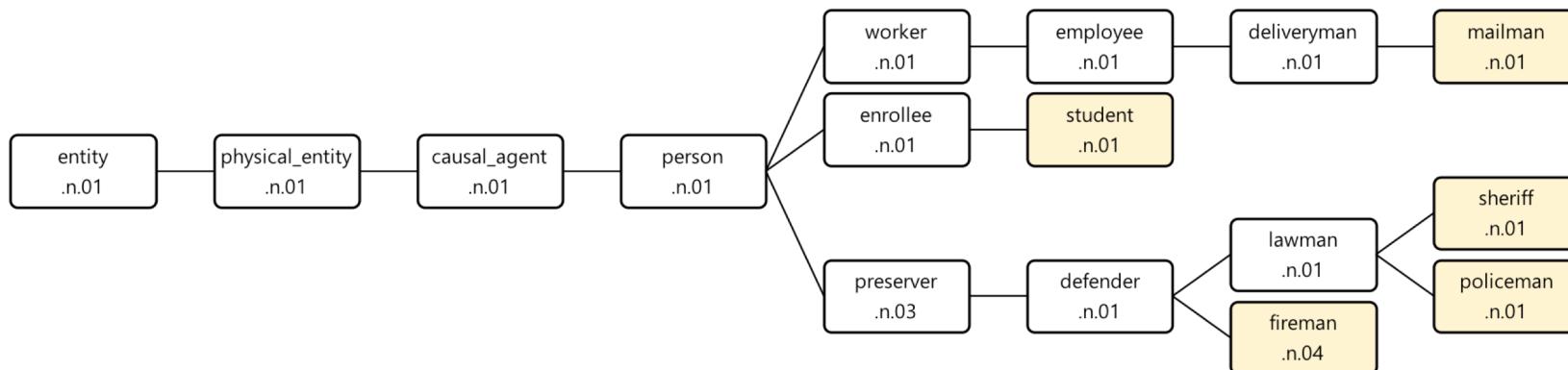
- 단어는 개념적 의미를 지님
- 따라서 개념을 포괄하는 상위 개념이 존재함
  - 하위 개념 또한 존재
- Hyponym / Hyponym
  - 동물 / 코끼리
  - 전화기 / 핸드폰
  - 컴퓨터 / 노트북

# Index

- Word Senses
- WordNet
- Preprocessing (Tokenization)
- TF-IDF
- Feature Vector
- Similarity Measures
- Word Sense Disambiguation
- Selectional Preference
- Conclusion

# WordNet

- Thesaurus(어휘분류사전, 시소러스)
- 심리학 교수인 George Armitage Miller 교수가 지도하에 프린스턴 대학에서 1985년부터 만든 프로그램
- 동의어 집합(Synset) 또는 상위어(Hypernym)나 하위어(Hyponym)에 대한 정보가 특히 잘 구축되어 있는 것이 장점
  - Directed Acyclic Graph(유방향 비순환 그래프)



# WordNet

- NLTK or Web
  - <http://wordnetweb.princeton.edu/perl/webwn>

## WordNet Search - 3.1

- [WordNet home page](#) - [Glossary](#) - [Help](#)

Word to search for:

Display Options:

Key: "S:" = Show Synset (semantic) relations, "W:" = Show Word (lexical) relations

Display options for sense: (gloss) "an example sentence"

Display options for word: word#sense number

### Noun

- S: (n) **bank#1** (sloping land (especially the slope beside a body of water)) "*they pulled the canoe up on the bank*"; "*he sat on the bank of the river and watched the currents*"
- S: (n) depository financial institution#1, **bank#2**, banking concern#1, banking company#1 (a financial institution that accepts deposits and channels the money into lending activities) "*he cashed a check at the bank*"; "*that bank holds the mortgage on my home*"
- S: (n) **bank#3** (a long ridge or pile) "*a huge bank of earth*"
- S: (n) **bank#4** (an arrangement of similar objects in a row or in tiers) "*he operated a bank of switches*"
- S: (n) **bank#5** (a supply or stock held in reserve for future use (especially in emergencies))
- S: (n) **bank#6** (the funds held by a gambling house or the dealer in some gambling games) "*he tried to break the bank at Monte Carlo*"
- S: (n) **bank#7**, cant#2, camber#2 (a slope in the turn of a road or track; the outside is higher than the inside in order to reduce the effects of centrifugal force)
- S: (n) savings bank#2, coin bank#1, money box#1, **bank#8** (a container (usually with a slot in the top) for keeping money at home) "*the coin bank was empty*"
- S: (n) **bank#9**, bank building#1 (a building in which the business of banking transacted) "*the bank is on the corner of Nassau and Witherspoon*"
- S: (n) **bank#10** (a flight maneuver; aircraft tips laterally about its longitudinal axis (especially in turning)) "*the plane went into a steep bank*"

### Verb

- S: (v) **bank#1** (tip laterally) "*the pilot had to bank the aircraft*"
- S: (v) **bank#2** (enclose with a bank) "*bank roads*"
- S: (v) **bank#3** (do business with a bank or keep an account at a bank) "*Where do you bank in this town?*"
- S: (v) **bank#4** (act as the banker in a game or in gambling)
- S: (v) **bank#5** (be in the banking business)
- S: (v) deposit#2, **bank#6** (put into a bank account) "*She deposits her paycheck every month*"
- S: (v) **bank#7** (cover with ashes so to control the rate of burning) "*bank a fire*"
- S: (v) count#8, bet#3, depend#2, swear#5, rely#1, **bank#8**, look#10, calculate#6, reckon#5 (have faith or confidence in) "*you can count on me to help you any time*"; "*Look to your friends for support*"; "*You can bet on that!*", "*Depend on your family in times of crisis*"

# WordNet

- Search Result: [bank](#)

## Noun

- [S: \(n\) bank#1](#) (sloping land (especially the slope beside a body of water))  
*"they pulled the canoe up on the bank"; "he sat on the bank of the river and watched the currents"*
- [S: \(n\) depository financial institution#1, bank#2, banking concern#1, banking company#1](#) (a financial institution that accepts deposits and channels the money into lending activities)  
*"he cashed a check at the bank"; "that bank holds the mortgage on my home"*
- [S: \(n\) bank#3](#) (a long ridge or pile)  
*"a huge bank of earth"*
- [S: \(n\) bank#4](#) (an arrangement of similar objects in a row or in tiers)  
*"he operated a bank of switches"*
- [S: \(n\) bank#5](#) (a supply or stock held in reserve for future use (especially in emergencies))
- [S: \(n\) bank#6](#) (the funds held by a gambling house or the dealer in some gambling games)  
*"he tried to break the bank at Monte Carlo"*
- [S: \(n\) bank#7, cant#2, camber#2](#) (a slope in the turn of a road or track; the outside is higher than the inside in order to reduce the effects of centrifugal force)
- [S: \(n\) savings bank#2, coin bank#1, money box#1, bank#8](#) (a container (usually with a slot in the top) for keeping money at home)  
*"the coin bank was empty"*
- [S: \(n\) bank#9, bank building#1](#) (a building in which the business of banking transacted)  
*"the bank is on the corner of Nassau and Witherspoon"*
- [S: \(n\) bank#10](#) (a flight maneuver; aircraft tips laterally about its longitudinal axis (especially in turning))  
*"the plane went into a steep bank"*

# Korean WordNet

이름	기관	웹사이트
KorLex	부산대학교	<a href="http://korlex.pusan.ac.kr/">http://korlex.pusan.ac.kr/</a>
Korean WordNet(KWN)	KAIST	<a href="http://wordnet.kaist.ac.kr/">http://wordnet.kaist.ac.kr/</a>

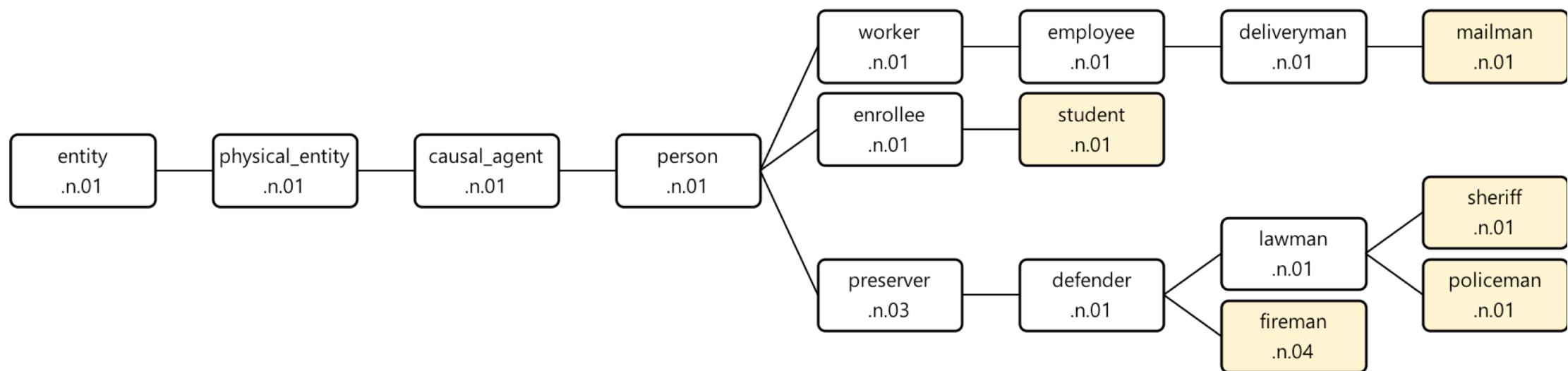
# WordNet Exercise

- Get hypernyms

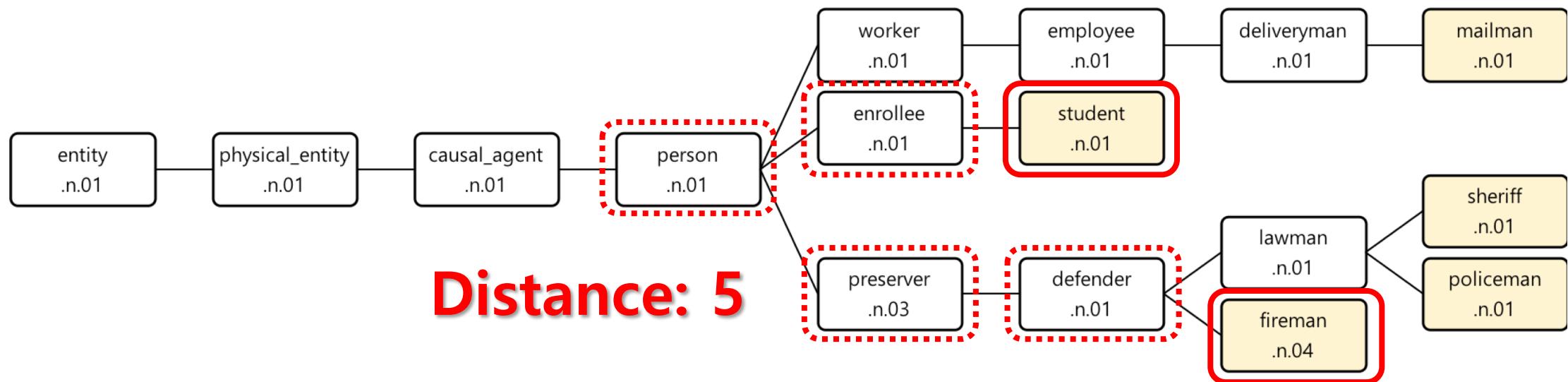
```
1 from nltk.corpus import wordnet as wn
2
3 def get_hypernyms(synset):
4     current_node = synset
5     while True:
6         print(current_node)
7         hypernym = current_node.hypernyms()
8         if len(hypernym) == 0:
9             break
10        current_node = hypernym[0]
```



# WordNet Exercise



# Distance between Words



# Word Similarity based on WordNet

$$\text{similarity}(w, w') = -\log \text{distance}(w, w')$$

# Index

- Word Senses
- WordNet
- Preprocessing (Tokenization)
- TF-IDF
- Feature Vector
- Similarity Measures
- Word Sense Disambiguation
- Selectional Preference
- Conclusion

# Part of Speech Tagging, Tokenization (Segmentation)

- 띄어쓰기만 제공하는 프로그램이 아니더라도, tag정보 등을 제외하게 되면 띄어쓰기(segmentation)를 수행
- 띄어쓰기
  - Sparsity 감소 – better Language Modeling
  - Normalization (특히 한국어)
- 한국어 이외 언어
  - 영어: 띄어쓰기가 이미 잘 되어 있음. NLTK를 사용하여 comma 등 후처리
  - 중국어: 기본적인 띄어쓰기가 없음. Character 단위로 사용해도 무방
  - 일본어: 기본적인 띄어쓰기가 없음.

# Part of Speech Tagging, Tokenization (Segmentation)

언어	프로그램명	제작언어	특징
한국어	Mecab	C++	일본어 Mecab을 wrapping하였으며, 속도가 가장 빠름. 설치가 종종 까다로움
한국어	KoNLPy	Python Wrapping(복합)	PIP을 통해 설치 가능하며 사용이 쉬우나, 일부 tagger의 경우에는 속도가 느림
일본어	Mecab	C++	속도가 빠르다
중국어	Stanford Parser	Java	미국 스탠포드에서 개발
중국어	PKU Parser	Java	북경대에서 개발. Stanford Parser와 성능 차이가 거의 없음
중국어	Jieba	Python	가장 최근에 개발됨. Python으로 제작되어 시스템 구성에 용이

# Korean (Mecab)

- <http://konlpy-ko.readthedocs.io/ko/v0.4.3/install/#ubuntu>

```
$ sudo apt-get install curl  
$ bash <(curl -s https://raw.githubusercontent.com/konlpy/konlpy/master/scripts/mecab.sh)
```

# Korean (Mecab)

```
$ echo "안녕하세요, 반갑습니다!" | mecab
안녕      NNG,*,T,안녕,*,*,*,*
하        XSV,*,F,하,*,*,*,*
세요      EP+EF,*,F,세요,Inflect,EP,EF,시/EP/*+어요/EF/*
,        SC,*,*,*,*,*,*,*
반갑      VA,*,T,반갑,*,*,*,*
습니다    EF,*,F,습니다,*,*,*,*
!        SF,*,*,*,*,*,*,*
EOS
```

# Korean (Mecab)

- -O **wakati** 옵션을 사용

```
$ echo "안녕하세요, 반갑습니다!" | mecab -O wakati  
안녕 하 세요 , 반갑 습니다 !
```

# Korean (KoNLPy)

- <http://konlpy-ko.readthedocs.io/>
- 여러 한국어 tokenizer 또는 tagger들을 모아놓은 wrapper를 제공
  - [Hannanum Class](#)
  - [Kkma Class](#)
  - [Komoran Class](#)
  - [Mecab Class](#)
  - [Twitter Class](#)
- 시스템 연동 및 구성이 용이
  - 내부 라이브러리들은 각기 다른 언어(Java, C++ 등)로 이루어져 있어 호환성에 문제가 생기기도
  - 일부는 C++로 구현되어 있는 Mecab에 비해 속도면에서 불리함

# English (Moses with NLTK)

- <http://www.nltk.org/api/nltk.tokenize.html#module-nltk.tokenize.moses>

```
import sys, fileinput
from nltk.tokenize.moses import MosesTokenizer

t = MosesTokenizer()

if __name__ == "__main__":
    for line in fileinput.input():
        if line.strip() != "":
            tokens = t.tokenize(line.strip(), escape=False)

            sys.stdout.write(" ".join(tokens) + "\n")
```

# English (Moses with NLTK)

before:

North Korea's state mouthpiece, the Rodong Sinmun, is also keeping mum on Kim's summit with Trump while denouncing ever-tougher U.S. sanctions on the rogue state.

after:

North Korea 's state mouthpiece , the Rodong Sinmun , is also keeping mum on Kim 's summit with Trump while denouncing ever-tougher U.S. sanctions on the rogue state .

# English (Moses with NLTK)

before:

North Korea's state mouthpiece, the Rodong Sinmun, is also keeping mum on Kim's summit with Trump while denouncing ever-tougher U.S. sanctions on the rogue state.

after:

North Korea's state mouthpiece, the Rodong Sinmun, is also keeping mum on Kim's summit with Trump while denouncing ever-tougher U.S. sanctions on the rogue state.

# Tokenization Conclusion

- 일반적인 또는 전형적인 쉬운 문장(표준어를 사용하며 문장 구조가 명확한 문장)의 경우, 성능이 비슷
- 하지만 **신조어나 보지 못한 고유명사를 처리하는 능력**이 다름.
- 따라서 프로그램을 선택 할 때에, 그런 부분에 초점을 맞추어 성능을 평가하고 선택해야 함

# Index

- Word Senses
- WordNet
- Preprocessing (Tokenization)
- **TF-IDF**
- Feature Vector
- Similarity Measures
- Word Sense Disambiguation
- Selectional Preference
- Conclusion

# TF-IDF

- 텍스트 마이닝(Text Mining)에서 중요하게 사용
- 어떤 단어  $w$ 가 문서  $d$  내에서 얼마나 중요한지 나타내는 수치

# TF-IDF

- TF(Term Frequency)
  - 단어의 문서 내에 출현한 횟수
  - 숫자가 클수록 문서 내에서 중요한 단어
  - 하지만, 'the'와 같은 단어도 TF값이 매우 클 것
- IDF(Inverse Document Frequency)
  - 그 단어가 출현한 문서의 숫자의 역수(inverse)
  - 값이 클수록 'the'와 같이 일반적으로 많이 쓰이는 단어

$$\text{TF-IDF}(w, d) = \frac{\text{TF}(w, d)}{\text{DF}(w)}$$

# TF-IDF Exercise

```
1 def get_term_frequency(document, word_dict=None):
2     if word_dict is None:
3         word_dict = {}
4     words = document.split()
5
6     for w in words:
7         word_dict[w] = 1 + (0 if word_dict.get(w) is None else word_dict[w])
8
9 return word_dict
```

# TF-IDF Exercise

```
1 def get_document_frequency(documents):
2     dicts = []
3     vocab = set([])
4     df = {}
5
6     for d in documents:
7         tf = get_term_frequency(d)
8         dicts += [tf]
9         vocab = vocab | set(tf.keys())
10
11    for v in list(vocab):
12        df[v] = 0
13        for dict_d in dicts:
14            if dict_d.get(v) is not None:
15                df[v] += 1
16
17    return df
```

# Index

- Word Senses
- WordNet
- Preprocessing (Tokenization)
- TF-IDF
- **Feature Vector**
- Similarity Measures
- Word Sense Disambiguation
- Selectional Preference
- Conclusion

# Term-Frequency Matrix

- 각 문서(문장)별 단어가 출현한 횟수로 채워진 매트릭스
  - Row: 단어
  - Column: 문서
- 문서가 많을수록 vector의 차원이 높아지고, 좀 더 정교해질 수 있음
- 출현빈도 대신 TF-IDF 수치로 대체 가능
- Vector 대부분의 값들은 0들로 채워져 있을 것  
→ Sparse Vector

# Exercise

# Based on Context Window (Co-occurrence)

- 함께 나타나는 단어들을 활용
- 가정
  - 의미가 비슷한 단어라면 **쓰임새가 비슷**할 것
  - 쓰임새가 비슷하기 때문에, 비슷한 문장 안에서 **비슷한 역할**로 사용될 것
  - **따라서 함께 나타나는 단어들이 유사**할 것
- Context Window를 사용하여 windowing을 실행
- 좀 더 정확하지만, window의 크기라는 hyper-parameter 추가
- 적절한 window 크기를 정하는 것이 중요

# Exercise

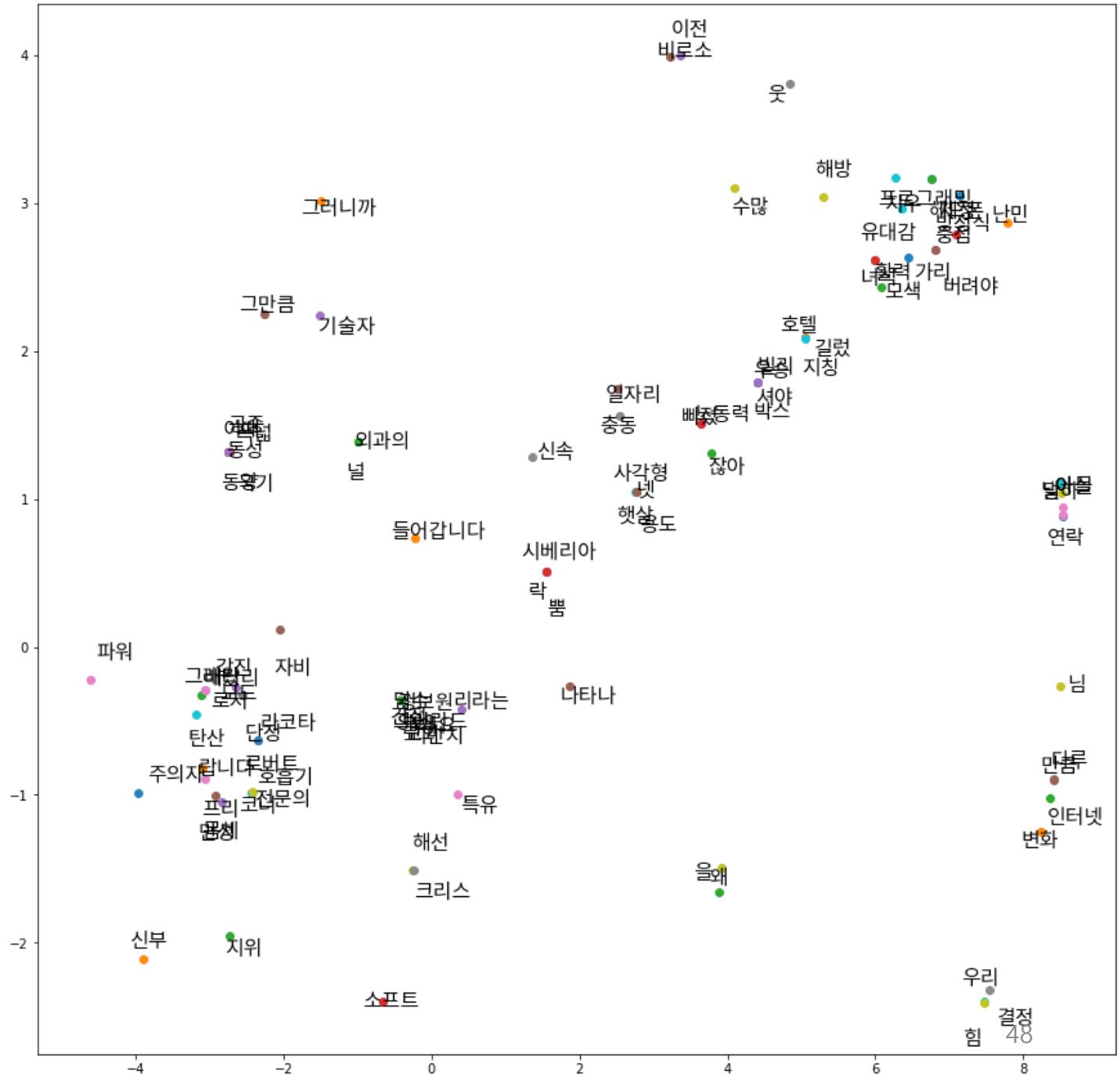
# Feature Vectors

	는	이	을	은	하	의	에	,	들	있	...	다다를	유리병	엉터리	켠	기념관	외침	스프를	이야	금성	악상어
는	0	4227	3554	320	13174	1324	6800	4049	315	8655	...	0	1	0	1	0	0	0	0	2	3
이	8466	0	706	3342	494	3450	1423	2797	12935	4470	...	1	0	0	0	0	1	1	0	1	0
을	5496	3031	0	3064	2249	5075	1875	1119	6720	1783	...	0	1	2	1	2	2	0	1	1	0
은	3298	2764	394	0	40	1994	442	2514	9052	59	...	0	2	2	0	0	0	1	0	3	0
하	13364	2407	9382	717	0	112	1945	700	232	336	...	0	0	0	0	0	0	0	0	0	0
의	3289	1965	295	2386	29	0	409	1858	3699	21	...	0	1	1	0	0	6	3	1	2	3
에	5965	2431	625	1574	391	2945	0	1935	1299	3145	...	7	7	0	5	3	1	1	0	0	2
,	2053	2858	969	1541	2899	675	1305	0	1297	1932	...	0	1	1	1	2	0	0	1	2	0
들	3958	13284	6696	10531	23	5851	1447	2311	0	16	...	0	0	3	0	0	1	0	4	0	0
있	8674	4684	2692	169	2997	20	3802	71	932	0	...	3	0	0	0	0	0	0	0	0	0
습니다	465	3532	1406	218	621	1	663	470	167	11875	...	0	0	0	0	0	0	0	0	0	3
가	2788	1517	351	2008	365	1791	1236	2067	211	2880	...	0	0	1	0	1	0	0	0	0	1
를	2274	2219	96	1768	1247	3336	780	1044	641	179	...	0	0	0	0	0	0	0	0	0	3
고	456	3332	5220	250	8667	43	828	6303	291	12122	...	0	0	0	0	0	1	0	0	0	0
것	10318	7502	5762	5080	3916	414	1296	334	1884	2445	...	3	0	0	0	0	0	0	0	0	0

# Sparsity?

베네수엘라	1	0	0	0	0	3	1	3	0	0	...	0	0	0	0	0	0	0	0	0	0	0
표하	0	0	2	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0	0
쿠데타	0	0	0	1	0	1	0	0	1	0	...	0	0	0	0	0	0	0	0	0	0	0
소용없	1	2	0	1	0	0	1	0	0	0	...	0	0	0	0	0	0	0	0	0	0	0
유실	0	3	1	2	0	2	0	1	2	0	...	0	0	0	0	0	0	0	0	0	0	0
거장	0	2	0	1	0	5	0	1	2	0	...	0	0	0	0	0	0	0	0	0	0	0
물어봅니다	1	2	0	0	0	0	0	0	1	0	...	0	0	0	0	0	0	0	0	0	0	0
자연스레	2	0	3	0	1	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0	0
대포	0	0	0	0	0	1	1	0	1	0	...	0	0	0	0	0	0	0	0	0	0	0
분당	4	0	0	0	0	1	0	0	1	0	0	...	0	0	0	0	0	0	0	0	0	0

# Visualization



# Index

- Word Senses
- WordNet
- Preprocessing (Tokenization)
- TF-IDF
- Feature Vector
- **Similarity Measures**
- Word Sense Disambiguation
- Selectional Preference
- Conclusion

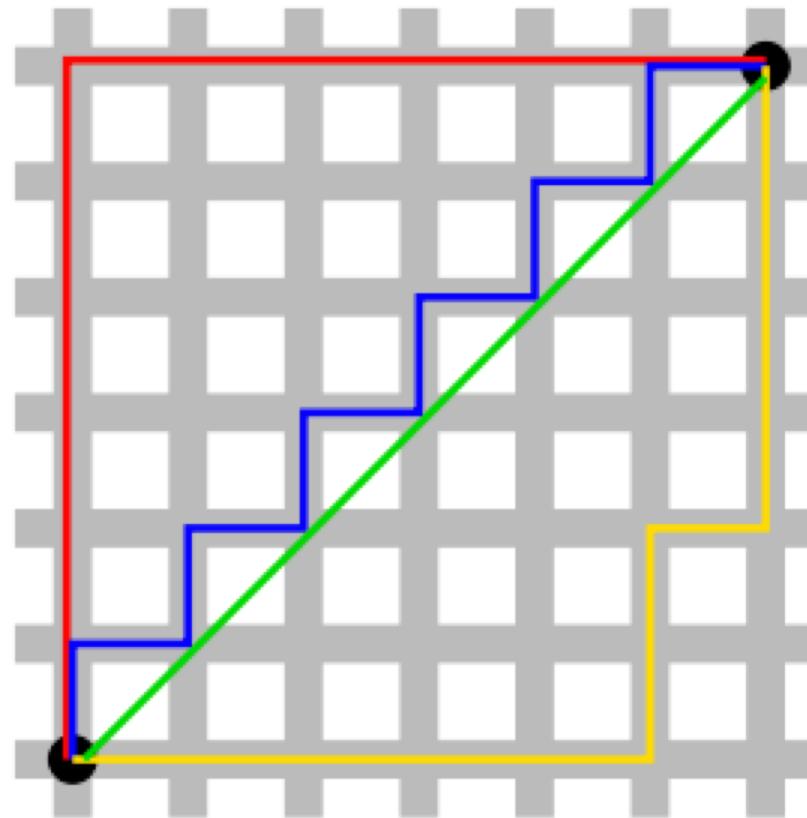
# Manhattan Distance (L1 distance)

$$d_{L1}(w, v) = \sum_{i=1}^d |w_i - v_i|, \text{ where } w, v \in \mathbb{R}^d.$$

# Euclidean Distance (L2 distance)

$$d_{L2}(w, v) = \sqrt{\sum_{i=1}^d (w_i - v_i)^2}, \text{ where } w, v \in \mathbb{R}^d.$$

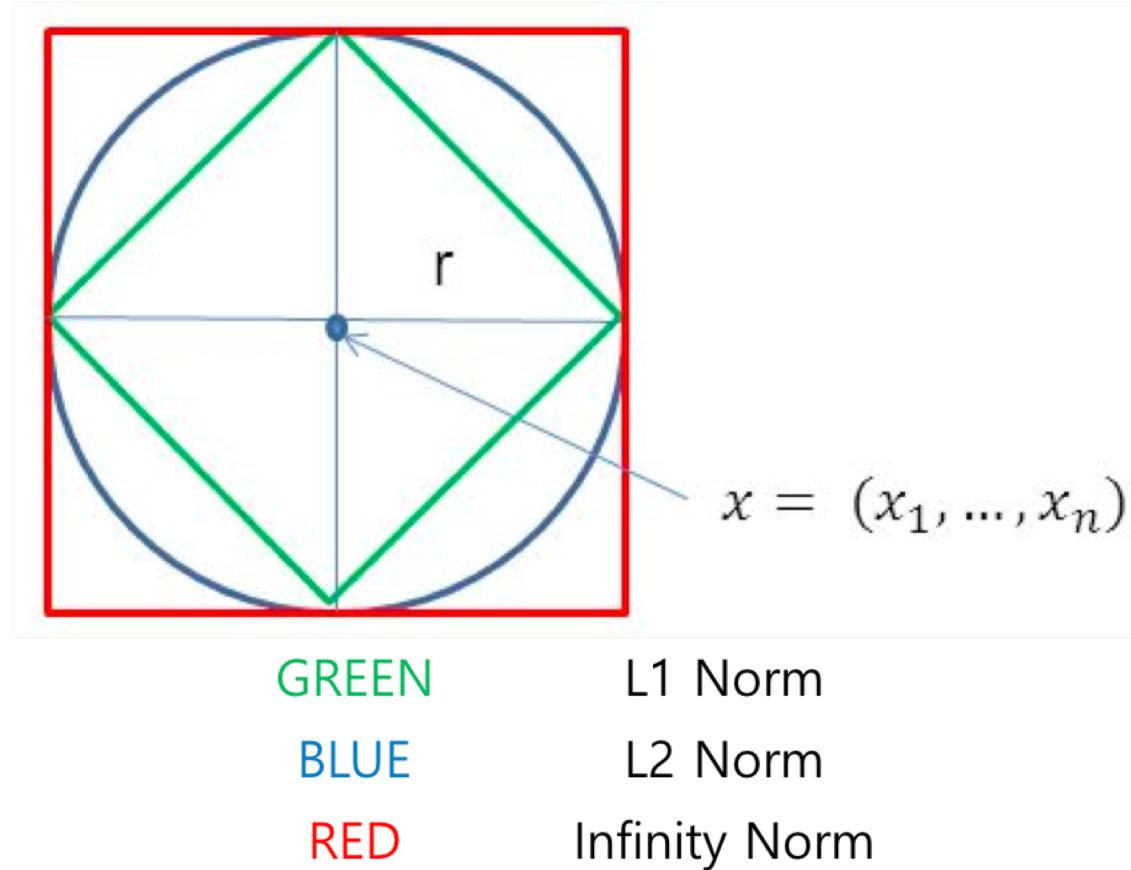
# L1 vs L2



# Infinity Norm

$$d_\infty(w, v) = \max(|w_1 - v_1|, |w_2 - v_2|, \dots, |w_d - v_d|), \text{ where } w, v \in \mathbb{R}^d$$

# L1, L2 and Infinity



# Cosine Similarity

$$\begin{aligned}\text{sim}_{\cos}(w, v) &= \frac{\overbrace{w \cdot v}^{\text{dot product}}}{\overbrace{|w||v|}^{\text{unit vector}}} = \frac{\widehat{w}}{|w|} \cdot \frac{v}{|v|} \\ &= \frac{\sum_{i=1}^d w_i v_i}{\sqrt{\sum_{i=1}^d w_i^2} \sqrt{\sum_{i=1}^d v_i^2}}\end{aligned}$$

where  $w, v \in \mathbb{R}^d$

# Jaccard Similarity

$$\begin{aligned}\text{sim}_{\text{jaccard}}(w, v) &= \frac{|w \cap v|}{|w \cup v|} \\ &= \frac{|w \cap v|}{|w| + |v| - |w \cap v|} \\ &\approx \frac{\sum_{i=1}^d \min(w_i, v_i)}{\sum_{i=1}^d \max(w_i, v_i)}\end{aligned}$$

where  $w, v \in \mathbb{R}^d$ .

# Index

- Word Senses
- WordNet
- Preprocessing (Tokenization)
- TF-IDF
- Feature Vector
- Similarity Measures
- Word Sense Disambiguation
- Selectional Preference
- Conclusion

# Word Sense Disambiguation

원문	차를 마시러 공원에 가던 차 안에서 나는 그녀에게 차였다.
G*	I was kicking her in the car that went to the park for tea.
M*	I was a car to her, in the car I had a car and went to the park.
N*	I got dumped by her on the way to the park for tea.
K*	I was in the car going to the park for tea and I was in her car.
S*	I got dumped by her in the car that was going to the park for a cup of tea.

# Simple Principle

- 단어의 의미는 주변 문맥에 따라 정해진다.

# Thesaurus Based Method: Lesk Algorithm

- 간단한 사전 기반 중의성 해소 방법
  - 문장 내에 같이 등장하는 단어(context)들은 **공통 토픽을 공유**한다는 가정
- 
- ① 중의성을 해소하고자 하는 단어의 사전(주로 WordNet)내의 미별 설명 및 문장과, 주어진 문장 사이의 유사도를 구함
  - ② 가장 유사도가 높은 (또는 겹치는 단어가 많은) 의미가 선택

# Exercise

# Index

- Word Senses
- WordNet
- Preprocessing (Tokenization)
- TF-IDF
- Feature Vector
- Similarity Measures
- Word Sense Disambiguation
- Selectional Preference
- Conclusion

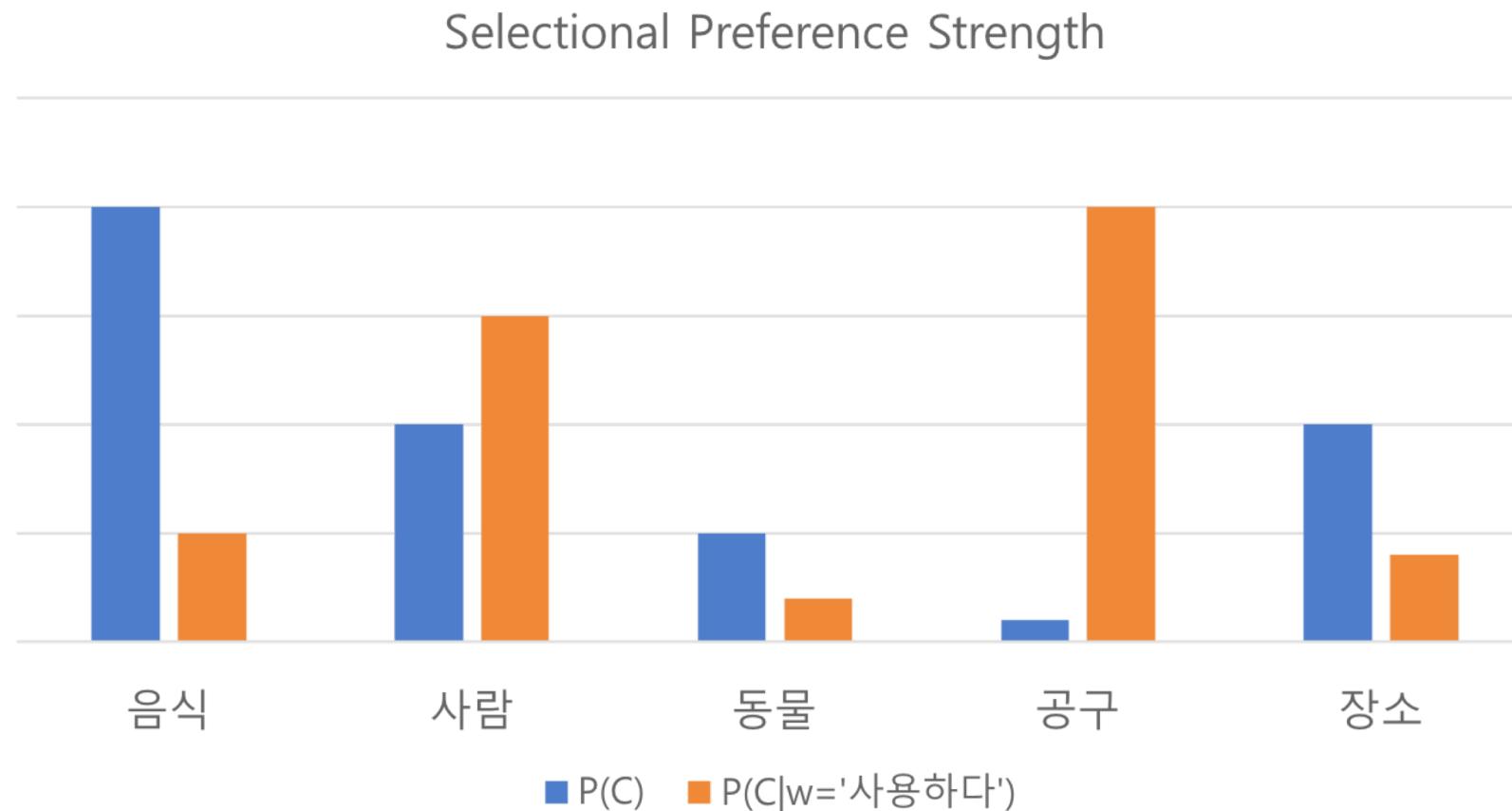
# Selectional Preference

- '마시다'라는 동사에 대한 목적어는 '<음료>' 클래스에 속하는 단어가 올 확률이 매우 높음
- 따라서 우리는 '차'라는 단어가 '<탈 것>' 클래스에 속하는지 '<음료>' 클래스에 속하는지 쉽게 알 수 있음

# Selectional Preference

- 단어 사이의 관계(예: verb-object)가 좀 더 특별한 경우에 대해 수치화 하여 나타냄
- **술어(predicate)** 동사(예: verb)가 주어졌을 때, 목적어(예: object)관계에 있는 **headword 단어**(보통은 명사가 될 겁니다.)들의 분포는, 평소 문서 내에 해당 명사(예: object로 써 noun)가 나올 분포와 다를 것
- 분포의 차이가 크면 클수록 해당 술어(predicate)는 더 강력한 selectional preference를 가짐

# Selectional Preference Strength



# Selectional Preference Strength

$$\begin{aligned} S_R(w) &= \text{KL}(P(C|w) || P(C)) \\ &= - \sum_{c \in \mathcal{C}} P(c|w) \log \frac{P(c)}{P(c|w)} \\ &= -\mathbb{E}_{C \sim P(C|w)} [\log \frac{P(C)}{P(C|W=w)}] \end{aligned}$$

# Selectional Association

$$A_R(w, c) = -\frac{P(c|w) \log \frac{P(c)}{P(c|w)}}{S_R(w)}$$

# Selectional Preference and WSD

- '마시다'라는 동사에 '차'라는 목적어가 함께 있을 때,
- 우리는 selectional preference를 통해서
- '차'는 '<음료>' 클래스에 속한다고 말할 수 있을 것
- 문제
  - '차'가 '<탈 것>' 또는 '<음료>' 클래스에 속하는 것을 알아내는 것
  - 코퍼스는 단어들로 표현되어 있지, 클래스로 표현되어 있지는 않음
  - 이를 위해서는 사전에 정의되어 있는 지식 또는 데이터셋이 필요

# Selectional Preference and WSD

- '마시다'라는 동사에 '차'라는 목적어가 함께 있을 때,
- 우리는 selectional preference를 통해서
- '차'는 '<음료>' 클래스에 속한다고 말할 수 있을 것
- 문제
  - '차'가 '<탈 것>' 또는 '<음료>' 클래스에 속하는 것을 알아내는 것
  - **코퍼스는 단어들로 표현되어** 있지, 클래스로 표현되어 있지는 않음
  - 이를 위해서는 **사전에 정의되어 있는 지식 또는 데이터셋이 필요**

**WordNet**

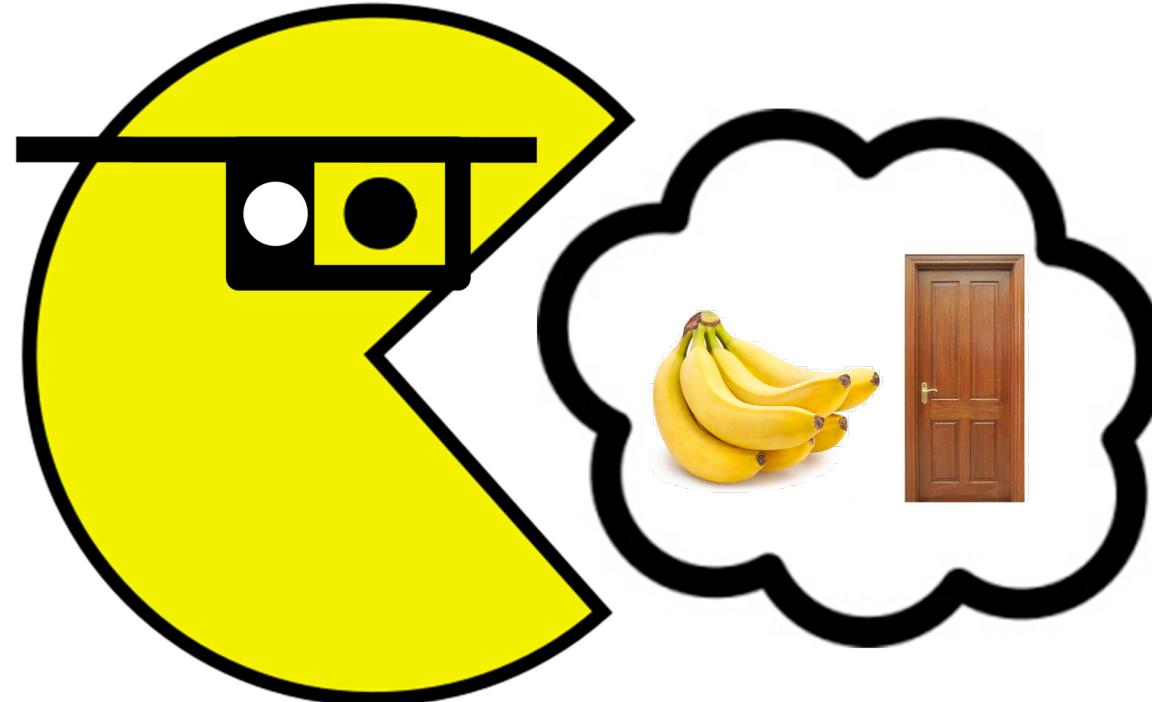
# Selectional Preference based on WordNet

$$\text{Count}_R(w, c) \approx \sum_{h \in c} \frac{\text{Count}_R(w, h)}{|\text{Classes}(h)|}$$

$\hat{c} = \operatorname{argmax}_{c \in \mathcal{C}} A_R(w, c)$ , where  $\mathcal{C} = \text{hypernym}(h)$ .

# Selectional Preference Evaluation using Pseudo Word

- Pseudo Word
  - 두 단어를 인공적으로 합성한 단어: banana-door
  - eat과 verb-object 관계인 경우, banana-door의 의미는?



# Selectional Preference Evaluation using Pseudo Word

- Pseudo Word
  - 두 단어를 인공적으로 합성한 단어: banana-door
  - eat과 verb-object 관계인 경우, banana-door의 의미는?



**Now, we induced WSD problem  
to select a proper word between words  
given predicate.**

# Similarity-based Selectional Preference

- A Simple, Similarity-based Model for Selectional Preferences

$(w, h, R)$ , where  $R$  is a relationship, such as verb-object.

$$A_R(w, h_0) = \sum_{h \in \text{Seen}_R(w)} \text{sim}(h_0, h) \cdot \phi_R(w, h)$$

$$\phi_R(w, h) = \text{IDF}(h)$$

# Exercise

# Result

```
1 >>> wsd('피우', ['담배', '맥주', '사과'])
2 [tensor(6.1853), tensor(3.9723), tensor(3.8503)]
```



# Index

- Word Senses
- WordNet
- Preprocessing (Tokenization)
- TF-IDF
- Feature Vector
- Similarity Measures
- Word Sense Disambiguation
- Selectional Preference
- Conclusion

# Conclusion

- 단어는 겉의 discrete한 형태와 달리 내부적으로는 non-discrete 한 '의미(sense)'를 가짐
- 의미가 유사한 단어들간의 유사도를 계산 할 수 있다면, 코퍼스 (corpus)로부터 분포나 특징(feature)들을 훈련 할 때 좀 더 많은 정보를 얻고 정확한 훈련을 할 수 있음
- 추출된 feature vector의 차원은 단어 사전의 크기와 맞먹음
- 그리고 대부분은 0으로 가득함
- 이러한 sparsity(희소성)문제는 자연어처리의 가장 큰 특징

# Conclusion

- 단어는 겉의 discrete한 형태와 달리 내부적으로는 non-discrete 한 '의미(sense)'를 가짐
- 의미가 유사한 단어들간의 유사도를 계산 할 수 있다면, 코퍼스 (corpus)로부터 분포나 특징(feature)들을 훈련 할 때 좀 더 많은 정보를 얻고 정확한 훈련을 할 수 있음
- 추출된 feature vector의 차원은 단어 사전의 크기와 맞먹음
- 그리고 대부분은 0으로 가득함
- 이러한 sparsity(희소성)문제는 자연어처리의 가장 큰 특징

**Welcome to Deep Learning!!**

# Appendix I

Kim, Ki Hyun and June Oh

FastCampus

2018.09.15

# Maximum A Posterior

$$\underbrace{P(Y|X)}_{posterior} = \frac{\overbrace{P(X|Y)P(Y)}^{likelihood \ prior}}{\underbrace{P(X)}_{evidence}}$$

수식	영어 명칭	한글 명칭
$P(Y X)$	Posterior	사후 확률
$P(X Y)$	Likelihood	가능도(우도)
$P(Y)$	Prior	사전 확률
$P(X)$	Evidence	증거

# Maximum A Posterior

$$\hat{y}_{MAP} = \operatorname*{argmax}_{y \in \mathcal{Y}} P(Y = y | X)$$

# Naïve Bayes

$$P(Y = c | X = w_1, w_2, \dots, w_n)$$

# Naïve Bayes

$$\begin{aligned} P(Y = c | X = w_1, w_2, \dots, w_n) &\propto P(X = w_1, w_2, \dots, w_n | Y = c)P(Y = c) \\ &\approx P(w_1 | c)P(w_2 | c) \cdots P(w_n | c)P(c) \\ &= \prod_{i=1}^n P(w_i | c)P(c) \end{aligned}$$

# Naïve Bayes

$$\begin{aligned}\hat{c}_{MAP} &= \operatorname{argmax}_{c \in \mathcal{C}} P(Y = c | X = w_1, w_2, \dots, w_n) \\ &= \operatorname{argmax}_{c \in \mathcal{C}} \prod_{i=1}^n P(w_i | c) P(c)\end{aligned}$$

# Naïve Bayes

$$\tilde{P}(Y = c) = \frac{\text{Count}(c)}{\sum_{i=1}^{|C|} \text{Count}(c_i)}$$

$$\tilde{P}(w|c) = \frac{\text{Count}(w, c)}{\sum_{j=1}^{|V|} \text{Count}(w_j, c)}$$

# Appendix II

Kim, Ki Hyun and June Oh

FastCampus

2018.09.15

# Samsung AI Forum Day 2 Track 1



**Yejin Choi** | University of Washington  
The Missing Representation in Neural Language Models

**Sam Bowman** | New York University  
Toward Task-Independent Sentence Understanding



**Yejin Choi**

Paul G. Allen School of  
Computer Science & Engineering,  
University of Washington.  
Allen Institute for  
Artificial Intelligence



**Sam Bowman**

New York University

# Samsung AI Forum Day 2 Track 1

- The Missing Representation in Neural Language Models
- Toward Task Independent Sentence Understanding