Classifier based Noun Extractor

Hyunjoong Kim

soy.lovit@gmail.com

github.com/lovit

Noun Extraction

- Word extractor & unsupervised tokenizers 에서는 분석할 데이터를 기반으로, 그 데이터에서 사용되는 **단어를 추출**하는 방법을 다뤘습니다.

- 이번에는 {'파스타', '좋아'} 와 같은 단어에서 '파스타'를 **명사로 인식**하는 방법을 다뤄봅니다.

A는 명사일까?

어제 A라는 가게에 가봤어

A에서 보자

A로 와줘

명사 우측에 등장하는 글자 분포를 이용하여

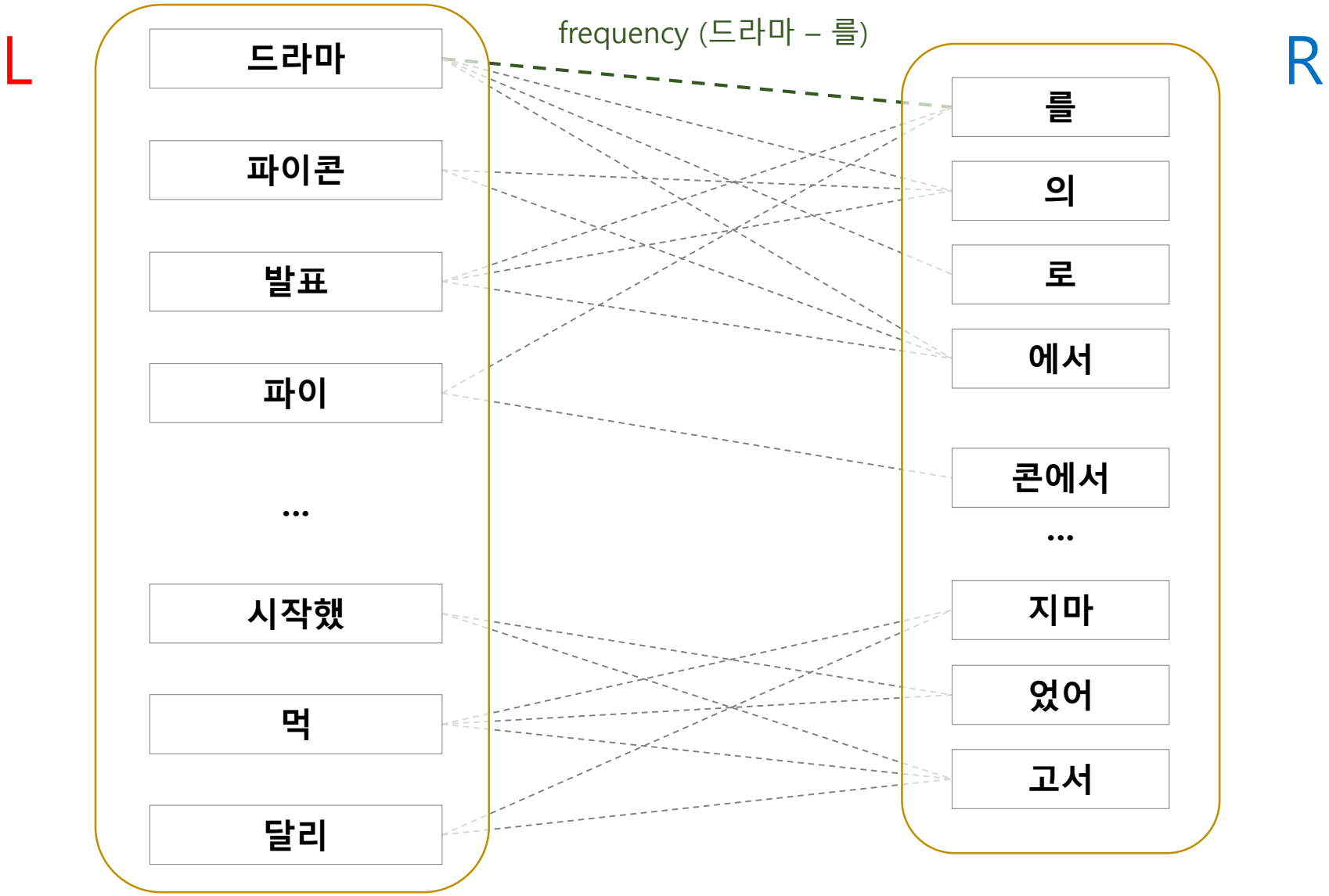
A가 명사임을 유추할 수 있다

L – R graph

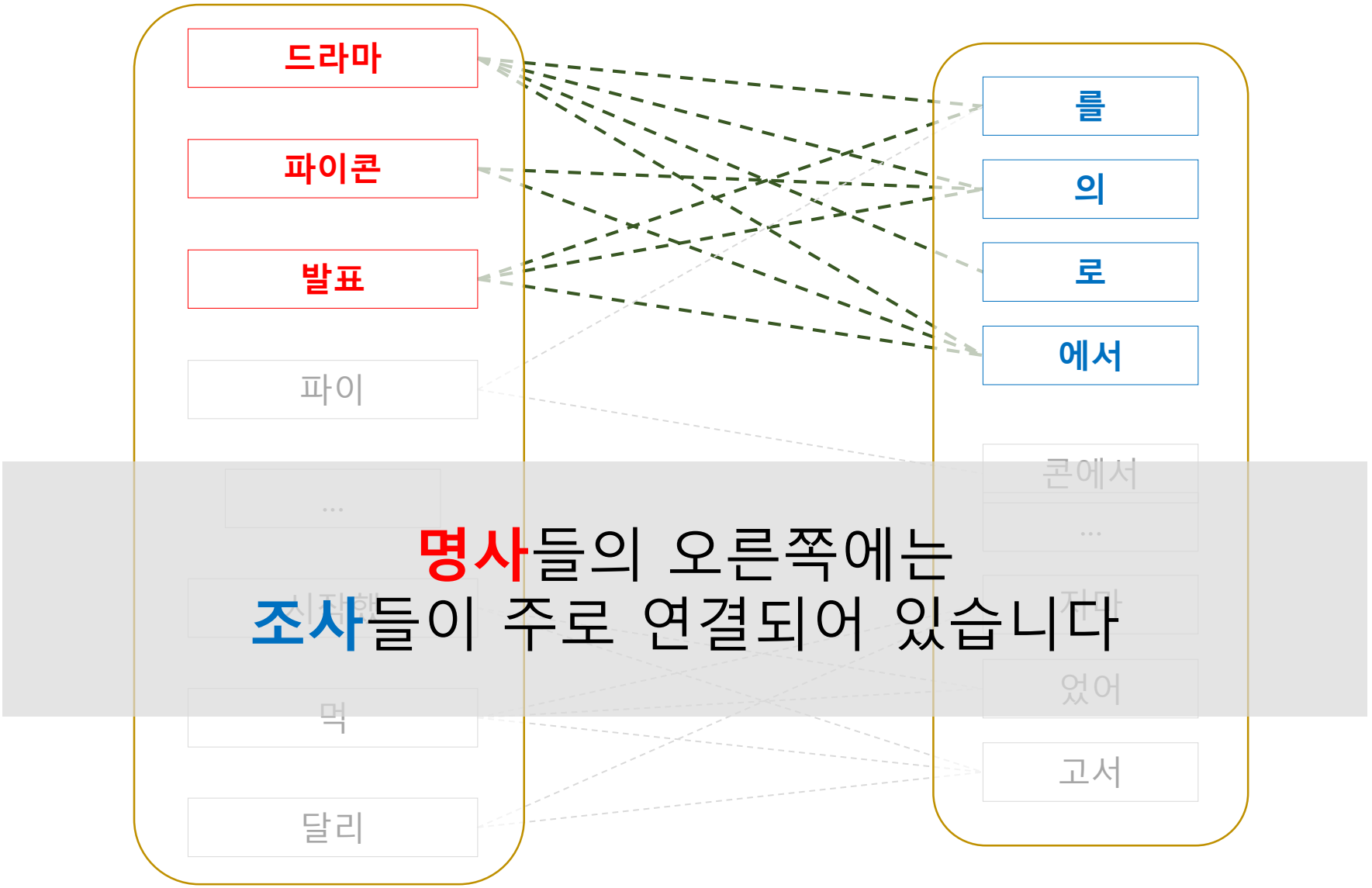
- 어절은 L + [R] 구조로 나눌 수 있습니다
  - 발표 + 를
  - 하 + 면서

- 데이터에서 모든 어절들을 두 개의 subwords로 나눈 뒤 연결하면 L – R graph를 만들 수 있습니다

L – R graph



L – R graph



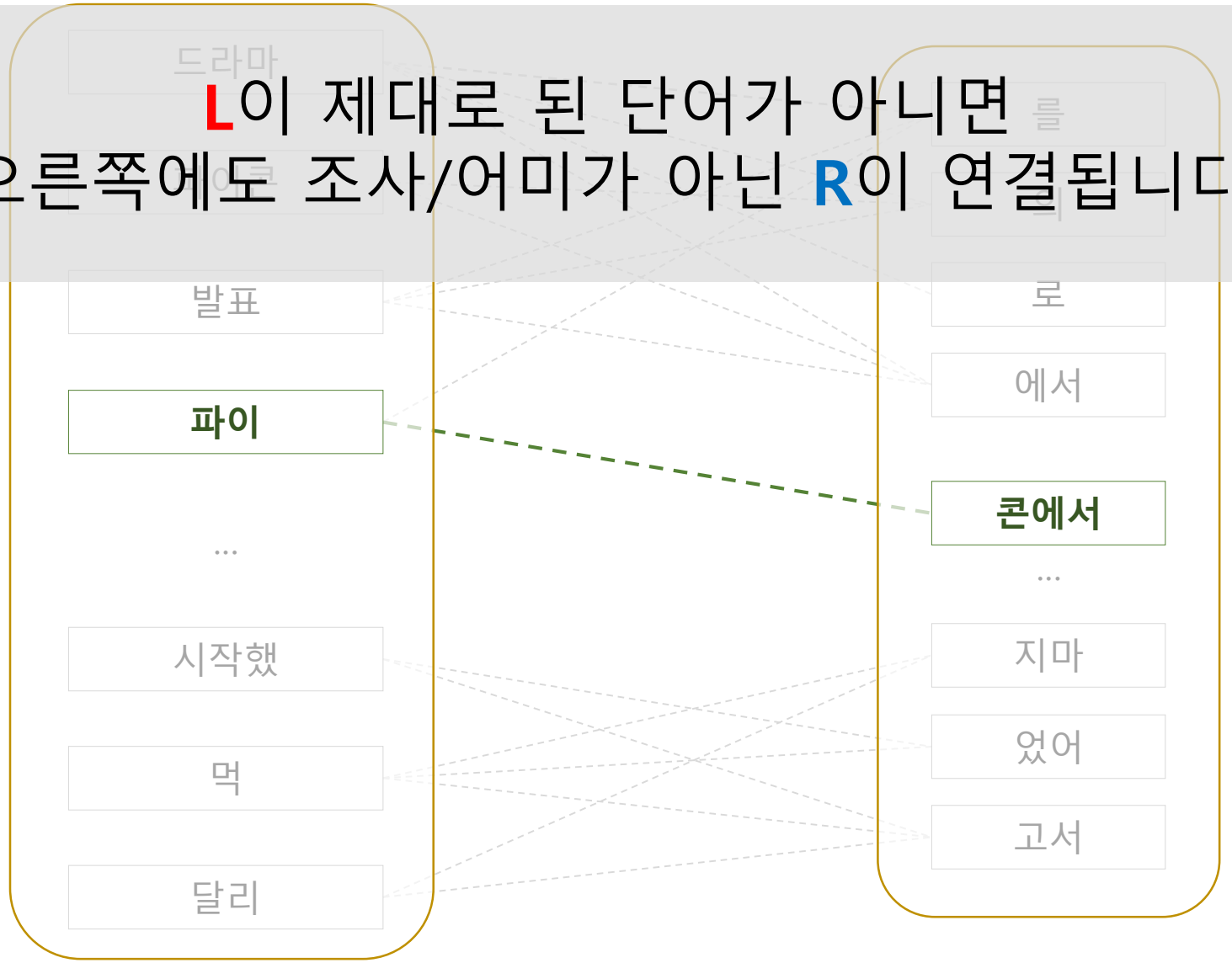
명사들의 오른쪽에는
조사들이 주로 연결되어 있습니다

L – R graph


동사/형용사들의 오른쪽에는
어미들이 주로 연결되어 있습니다
드라마
발표
파이
...
시작했
먹
달리
로
에서
콘에서
...
지마
었어
고서

L – R graph

L이 제대로 된 단어가 아니면
오른쪽에도 조사/어미가 아닌 **R**이 연결됩니다



Noun Extraction

- L – R graph: **[명사 + 조사]**, [동사 + 어미], [틀린 단어 + 틀린 단어]

get_r('드라마')

```
[('', 1268),
 ('를', 164),
 ('다', 152),
 ('의', 140),
 ('로', 138),
 ('에서', 98),
 ('와', 62),
 ('는', 55),
 ('에', 55),
 ('가', 48),
 ('이다', 24),
 ('인', 14),
 ... ]
```

get_r('시작했')

```
[('다', 567),
 ('고', 73),
 ('다고', 61),
 ('습니다', 42),
 ('는데', 26),
 ('으며', 16),
 ('지만', 15),
 ('던', 12),
 ('어요', 10),
 ('다는', 7),
 ('으나', 5),
 ('죠', 4),
 ... ]
```

get_r('드라')

```
[('마', 1268),
 ('마를', 164),
 ('마다', 152),
 ('마의', 140),
 ('마로', 138),
 ('마에서', 98),
 ('기', 65),
 ('마와', 62),
 ('마는', 55),
 ('마에', 55),
 ('마가', 48),
 ('이브', 28),
 ... ]
```

Noun Extraction

- L – R graph: [명사 + 조사], **[동사 + 어미]**, [틀린 단어 + 틀린 단어]

get_r(' 드라마')

```
[('', 1268),
 ('를', 164),
 ('다', 152),
 ('의', 140),
 ('로', 138),
 ('에서', 98),
 ('와', 62),
 ('는', 55),
 ('에', 55),
 ('가', 48),
 ('이다', 24),
 ('인', 14),
 ... ]
```

get_r('**시작했**')

```
[('다', 567),
 ('고', 73),
 ('다고', 61),
 ('습니다', 42),
 ('는데', 26),
 ('으며', 16),
 ('지만', 15),
 ('던', 12),
 ('어요', 10),
 ('다는', 7),
 ('으나', 5),
 ('죠', 4),
 ... ]
```

get_r('드라')

```
[('마', 1268),
 ('마를', 164),
 ('마다', 152),
 ('마의', 140),
 ('마로', 138),
 ('마에서', 98),
 ('기', 65),
 ('마와', 62),
 ('마는', 55),
 ('마에', 55),
 ('마가', 48),
 ('이브', 28),
 ... ]
```

Noun Extraction

- L – R graph: [명사 + 조사], [동사 + 어미], **[틀린 단어 + 틀린 단어]**

```
get_r( ' 드라마')
```

```
[('', 1268),
 ('를', 164),
 ('다', 152),
 ('의', 140),
 ('로', 138),
 ('에서', 98),
 ('와', 62),
 ('는', 55),
 ('에', 55),
 ('가', 48),
 ('이다', 24),
 ('인', 14),
 ... ]
```

```
get_r('시작했')
```

```
[('다', 567),
 ('고', 73),
 ('다고', 61),
 ('습니다', 42),
 ('는데', 26),
 ('으며', 16),
 ('지만', 15),
 ('던', 12),
 ('어요', 10),
 ('다는', 7),
 ('으나', 5),
 ('죠', 4),
 ... ]
```

```
get_r('드라')
```

```
[('마', 1268),
 ('마를', 164),
 ('마다', 152),
 ('마의', 140),
 ('마로', 138),
 ('마에서', 98),
 ('기', 65),
 ('마와', 62),
 ('마는', 55),
 ('마에', 55),
 ('마가', 48),
 ('이브', 28),
 ... ]
```

Noun Extraction

- R 의 분포를 이용하여 L의 명사 점수를 계산할 수 있습니다

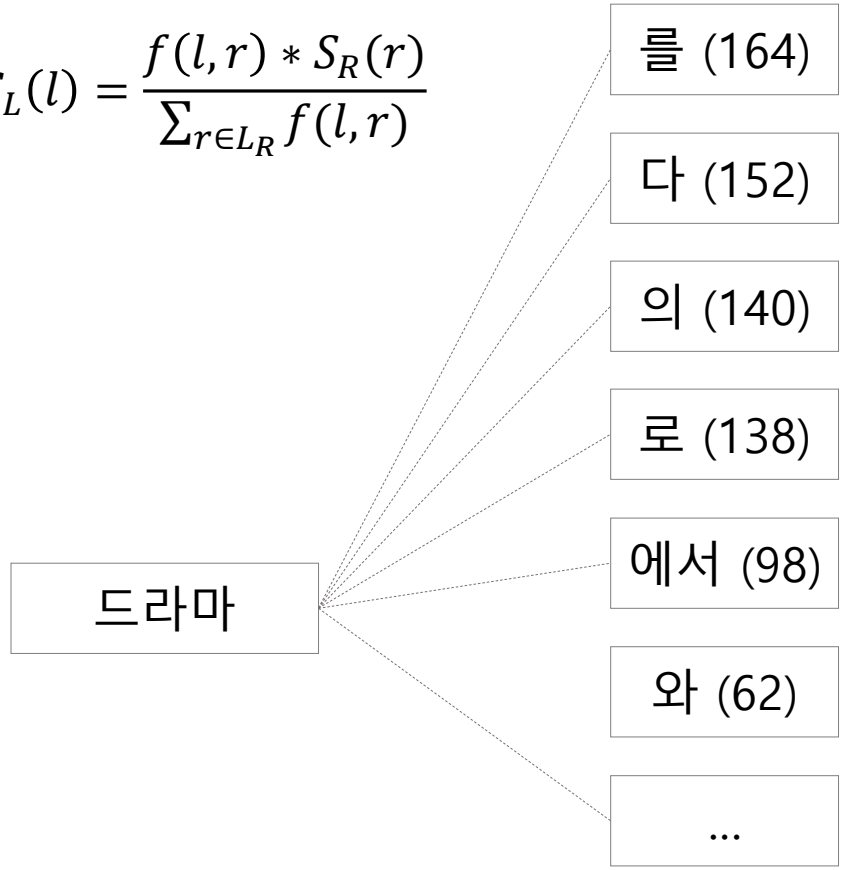
```
r_scores = {'은': 0.5, '었다' : -0.9, … }

def noun_score(L):
    (norm, score, _total) = (0, 0, 0)

    for R, frequency in get_r(L):
        _total += frequency
        if not R in r_scores:
            continue
        norm += frequency
        score += frequency * r_scores[R]

    score = score / norm if norm else 0
    prop = norm / _total if _total else 0
    return score, prop
```

$$S_L(l) = \frac{f(l,r) * S_R(r)}{\sum_{r \in L_R} f(l,r)}$$



Noun Extraction

- R 의 분포를 이용하여 L의 명사 점수를 계산할 수 있습니다

(명사 점수, 알려진 R 비율) = noun_score('Word')

```
noun_score('드라마')
```

```
(0.574, 0.921)
```

```
noun_score('시작했')
```

```
(-0.976, 0.999)
```

```
noun_score('드라')
```

```
(-0.661, 0.579)
```

틀린 단어 뒤에는 조사/어미 들이 아닌 글자들이 등장

Noun Extraction

- 세종 말뭉치의 품사가 태깅된 정보로부터 L – R 구조의 테이블을 만듭니다
  - R frequency vector를 이용하여 L이 명사인지 판단하는 Logistic Regression을 학습하여, 이의 coefficients를 r score table로 이용합니다.

```
…
예술가의   예술가/NNG+의/JKG   113
예술가는   예술가/NNG+는/JX    45
예술가가   예술가/NNG+가/JKS   43
예술가들의 예술가/NNG+들/XSN+의/JKG   30
…
```

단어 품사	단어 / R	- 는	- 의	- 고	- 었다	-었던
명사	예술가	45	113	2	0	0
동사	먹	33	0	27	0	27
…	…	…	…	…	…	…

Noun Extraction

- – R 앞의 명사 빈도를 이용하여 R score 를 계산할 수도 있습니다.
  - R의 명사 가능 점수 ($\boldsymbol{S_R}$)를 [-1, 1]로 스케일링 한 효과가 있습니다

$$2 * (p(N|R) - 0.5)$$

S_R(-다/R) = 2 * (0.056 – 0.5) = -0.888

S_R(-은/R) = 2 * (0.73– 0.5) = 0.46

Normalize R proportion by L distribution

R	Noun	Verb/Adj
-다/R	0.056	0.944
-은/R	0.73	0.265
…	…	…

Noun Extraction

- – R 앞의 명사의 빈도를 이용하여 명사가 등장할 점수를 계산

품사	단어 / R	- 는	- 의	- 고	- 었다	-구나
명사	예술가	45	113	2	0	0
명사	나이	54	87	27	0	27
동사	들	0	0	0	255	0
형용사	춥	0	0	0	0	87
…	…	…	…	…	…	…

품사	빈도수 (비율)
명사	704,197 (0.838)
^명사	136,598 (0.162)

품사	빈도수	보정빈도수	점수
명사	980	980	0.810
^명사	20	103.5 = 20 * (0.838/0.162)	= (980 – 103.5) / (980 + 103.5)

Noun Extraction: post-processing

- 오류를 막기 위한 후처리 기능이 필요합니다
  - R frequency vector를 데이터로 이용하는 classifier는 다음의 경우 잘못된 판단을 할 수 있습니다.

Noun Extraction: post-processing

- N = Nsub + J
  - 떡볶 + 이
  - '-이'는 대표적 조사이며, '떡볶이' 자체로 어절로 이용기도 하여 '떡볶' 이 명사로 잘못 판단될 수 있습니다

- N + Jsub
  - 대학생으 + 로
  - '-로'는 대표적 조사이기 때문에 '-으로'가 잘못 나뉘어질 경우, '대학생의' 역시 명사로 잘못 판단될 수 있습니다

Noun Extraction: post-processing

- ‘떡볶 + 이’의 경우,
  - {‘떡볶이’가 명사이고} and {끝 부분이 1음절 조사일 경우} ‘떡볶’을 명사에서 제외

- ‘대학생으 + 로’의 경우,
  - {‘대학생’이 명사이고} and { ‘-으 + 로 = -으로’가 조사일 경우} ‘대학생으’를 명사에서 제외

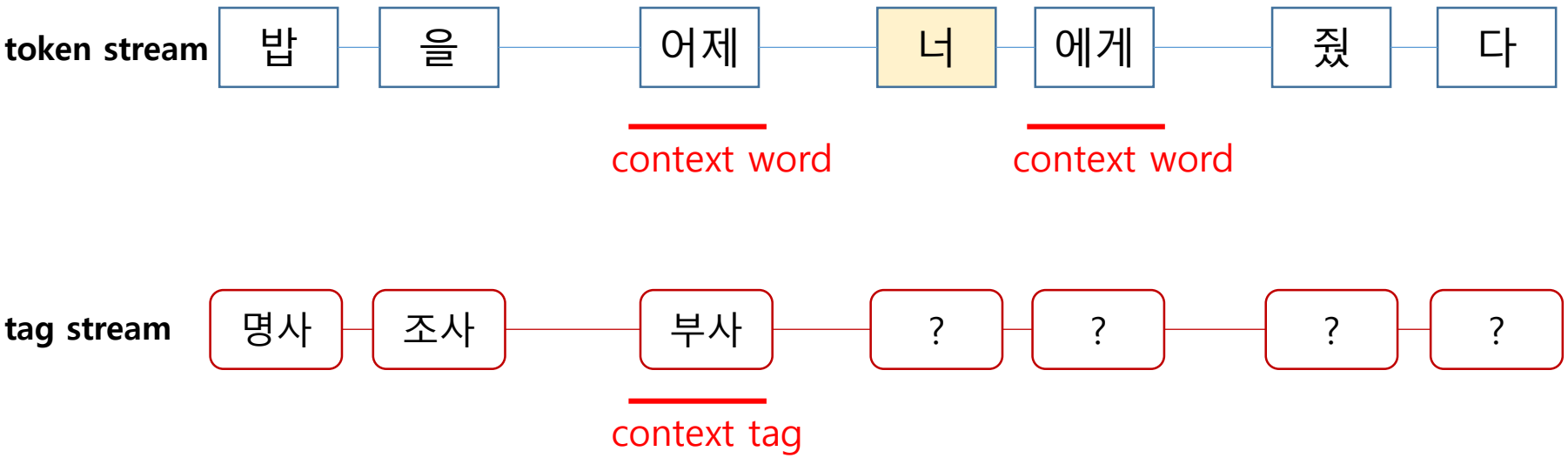
Noun Extraction

- Topic tracking, topic modeling 에는 concepts 을 나타내는 명사만을 이용하는 것도 좋습니다.

- 더 좋은 명사인식 / 품사판별을 위해서는 각 문장을 독립적으로 보지 않고, 문서 전체의 경향을 볼 필요가 있습니다 (global information)

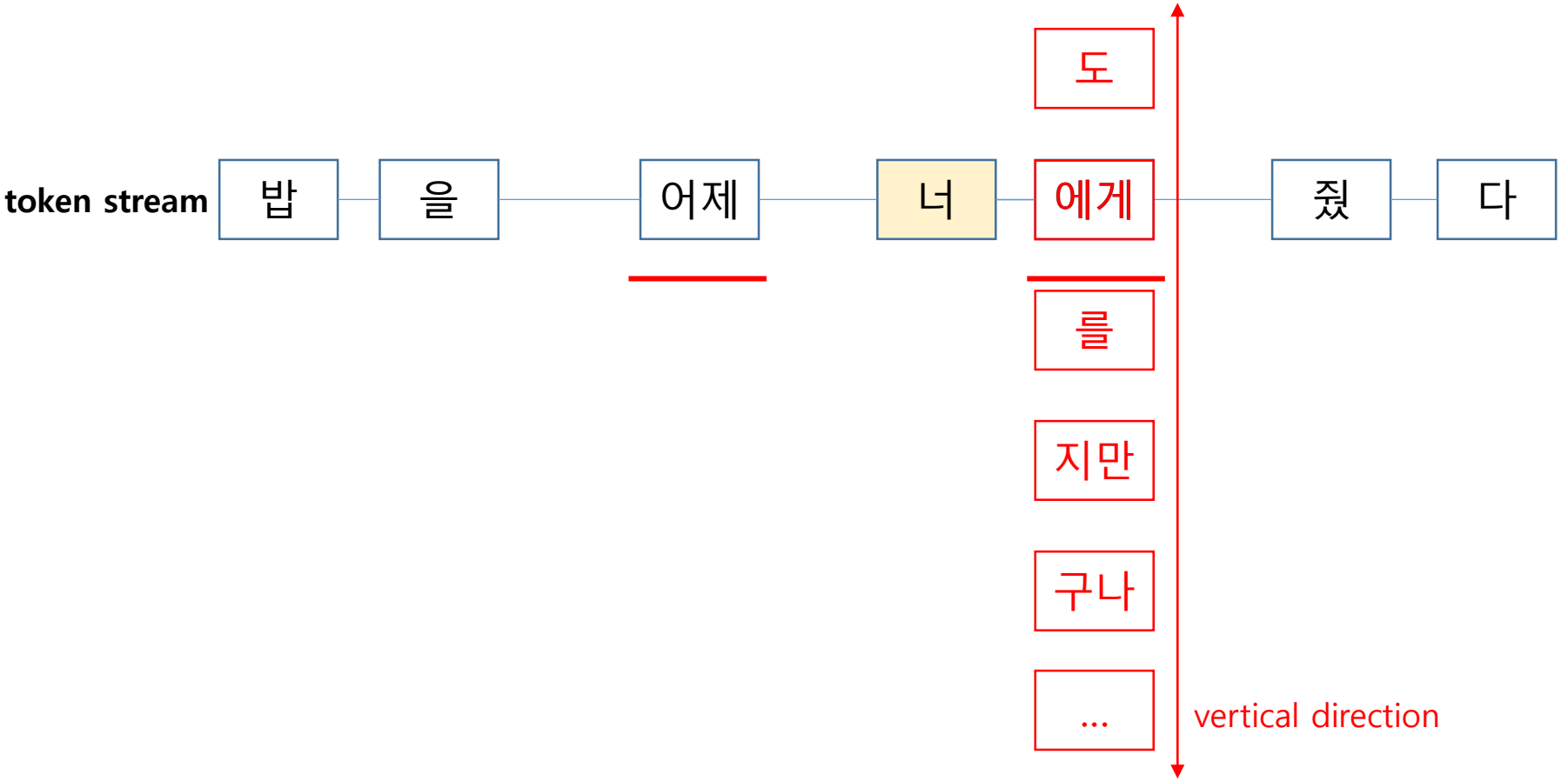
Noun Extraction

- Sequential labeling 기반 품사 판별기는 각 문장을 독립적으로 다룹니다.
  - 앞/뒤 단어의 문맥 정보를 이용하여 품사 판별을 수행합니다.



Noun Extraction

- 한 문장을 독립적으로 다루지 않고, 모든 문장에서의 "너/L" 의 오른쪽에 등장하는 R 의 분포를 이용하면 사전에 존재하지 않는 명사도 인식합니다.



soynlp

- 깔끔한 뉴스 데이터에서는 명사가 잘 추출됩니다

덴마크	웃돈	너무너무너무	가락동	매뉴얼	지도교수
전망치	강구	언니들	신산업	기뢰전	노스
할리우드	플라자	불법조업	월스트리트저널	2022년	불허
고씨	어플	1987년	불씨	적기	레스
스퀘어	충당금	건축물	뉴질랜드	사각	하나씩
근대	투자주체별	4위	태권	네트웍스	모바일게임
연동	런칭	만성	손질	제작법	현실화
오해영	심사위원들	단점	부장조리	차관급	게시물
인터폰	원화	단기간	편곡	무산	외국인들
세무조사	석유화학	워킹	원피스	서장	공범

soynlp

- 아직 노이지 데이터에서는 성능 개선이 필요합니다

변심	인시디어스	코디님	강만후	병원예약	전공필수
이런의미	바깥양반	프레시아	전리품상자	난심일기	근육몬
동피랑	상태이상	흐흐르	수입과자	모래사장	알엔에이
오션윙	옥매미	산넘어산	클럽하우스	물한번	손에다
따른사람	모진말	노들섬	한신아파트	괜차니	핵고생
다음장	아이도루	본사사람	마스크걸	브런치카페	클렌징워터
배아플라	일일미션	버스정거장	리장	안산도착	다리하나
빠른사과	용계	아홉시꺼	걸신	컴푸타	필터빨
성큼성큼	서가엔쿡	경중	스카이워크	인형만	하루시작
퍈의점	신청하기	미모쨜	재밌는시간	운동하규	간사들

soynlp

- 후처리과정이 추가된 명사 추출기를 이용중입니다

```
from soynlp.noun import LRNounExtractor


noun_extractor = LRNounExtractor(min_count=50)
nouns = noun_extractor.train_extract(corpus, minimum_noun_score=0.5)
```

```
nouns['설입']
```

```
NounScore(frequency=67, score=0.926, known_r_ratio=0.529)
```

```
nouns['드라마']
```

```
NounScore(frequency=4976, score=0.522, known_r_ratio=0.601)
```

Classifier based Noun Extraction (version 2)

Hyunjoong Kim

soy.lovit@gmail.com

github.com/lovit

- 명사는 어절의 왼쪽에 위치합니다.
  - 명사의 종류는 다양하지만, 조사의 종류는 제한적입니다.
  - 조사는 명사의 오른쪽에 위치합니다.
- 단어의 오른쪽에 등장한 subwords 의 분포로부터 명사의 유무를 판단하였습니다.

- 그러나 각각의 명사 후보에 대하여 독립적인 classification 을 수행할 경우, “매우 확실한 경우”에만 명사로 제대로 인식되었습니다.
- 위 방법의 문제점들을 유형화하고, 이를 고쳐나가며 두번째 버전의 명사 추출기를 개발하였습니다. 이 과정을 문서화 하였습니다.

Problem 1. 어미, 조사에 모두 해당하는 R parts

- 명사 추출용 features 를 만들기 위해 어절의 (L, R) 구조를 이용합니다.
  - L 에는 명사 (positive) 와 형용사, 동사의 어간 (negative) 을 이용합니다.
  - R 에는 명사의 오른쪽에 등장하는
    - 조사 (positive) : -은, -는, -이, -가, …
    - 전성어미가 포함된 어미 (positive) : -하는, -하려고, -있는데, …
    - 형용사, 동사의 어간에 결합되는 어미 (negative) : -은, -는, -던, -다고, …
  - 가 포함되어 있습니다.

Problem 1. 어미, 조사에 모두 해당하는 R parts

- 대부분의 단어는 어미와 조사 중 하나에 포함되지만,
  몇 개는 어미와 조사 모두에 포함됩니다.
  - 하나의 종류에 포함되는 단어의 coefficient 는 1/-1 입니다.
  - 어미/조사에 모두 포함되면 모호한 coefficient 를 지닙니다.

은	-0.606532
을	0.504724
의	0.999867
이	0.999185
를	0.999605
는	-0.619916
만	0.951425
다만	-0.953206

- 자주 등장하지 않는 명사의 오른쪽에 조사 '-은' 이 주로 등장한다면,
  그 단어는 명사로 인식될 수 없습니다.

Problem 1. 어미, 조사에 모두 해당하는 R parts

- 이를 해결하기 위하여 classifier 의 features 를 세 종류로 나눕니다.
  - positive : 조사에만 해당하거나 전성어미에 어미가 결합된 feature
  - negative : 어미에만 해당하는 feature
  - common : pos, neg 에 모두 해당하는 feature

Problem 2. “감사합니다+만” vs “감사합니+다만”

- Features 의 일부가 포함된 단어에 대하여 잘못된 판단을 할 수 있습니다.
  - ‘-만’은 대표적인 조사입니다.
  - ‘-다만’은 대표적인 어미입니다.
  - ‘감사합니다’의 다음에 ‘-만’이 자주 등장하여 false positive 가 발생합니다.

Problem 2. “감사합니다+만” vs “감사합니+다만”

- L 의 뒷부분의 글자와 R 을 합쳤을 때 이에 해당하는 negative features 가 존재하면 negative score 를 높입니다.
  - ‘-다만’이 negative features 이기 때문에 ‘감사합니다’의 명사 점수가 낮아집니다.

Problem 2. “감사합니다+만” vs “감사합니+다만”

- L 의 뒷부분의 글자와 R 을 합쳤을 때 이에 해당하는 positive features 가 존재하면 이는 무시합니다.
  - ‘-로’, ‘-으로’ 는 대표적인 조사입니다.
  - ‘대학생으로’의 어절에서 ‘대학생으’ 다음에는 ‘-로’가 자주 등장합니다.
  - ‘대학생으’가 정말로 명사라면 ‘-로’ 외에도 다양한 pos 가 등장할 것입니다. 하지만, 조사의 일부가 포함된 경우라면 명사의 점수가 올라가지 않습니다.

- predict 에서 common features 는 판단을 유보합니다.
  - common features 가 없어도 명사 점수가 클 때, common 를 positive 로 취급합니다.
  - 반대라면 negative 로 취급합니다.

```
def _predict(word, features):
    pos, common, neg, unk, end = 0, 0, 0, 0, 0
    for r, freq in features:
        if r == '':
            end += freq
            continue
        if _exist_longer_pos(word, r): # ignore
            continue
        if _exist_longer_neg(word, r): # negative -다고
            neg += freq
            continue
        if r in _common_features:
            common += freq
        elif r in _pos_features:
            pos += freq
        elif r in _neg_features:
            neg += freq
        else:
            unk += freq
return pos, common, neg, unk, end
```

- predict 에서 common features 는 판단을 유보합니다.
  - common features 가 없어도 명사 점수가 클 때, common 를 positive 로 취급합니다.
  - 반대라면 negative 로 취급합니다.

```
def predict(word, minimum_noun_score=0.3):
    pos, common, neg, unk, end = _predict(word, features)
    base = pos + neg score = 0 if base == 0 else (pos - neg) / base
    support = pos + end + common if score >= minimum_noun_score else neg + end + common
```

Problem 3. R features 가 다양하지 않을 경우 판단을 유보

- L 이 명사라면 그 오른쪽에는 다양한 조사가 등장해야 합니다.
  그렇지 않다면 '감사합니다 + 만' 과 비슷한 경우일 수 있습니다.
- 최소한의 R features 의 개수를 만족하지 않으면 prediction score 를 명사 점수로 이용하지 않습니다.
  - 도매인에 맞는 적절한 후처리를 수행합니다.

Problem 4. 짧은 단어에 불리한 prediction

- 짧은 L 단어의 R features 에 긴 단어의 일부가 포함됩니다. 긴 단어의 부분 글자는 짧은 단어의 prediction 에 영향을 줍니다.
  - 명사 '아이디' 오른쪽에는 다양한 조사가 위치합니다.
  - 명사 '아이디어'의 오른쪽에도 다양한 조사가 위치합니다.
  - '-어'는 대표적 어미이기 때문이 '아이디'의 명사 점수가 낮아집니다.
  - 그러나 '아이디어'의 명사 점수는 '아이디'에 영향을 받지 않습니다.

Problem 4. 짧은 단어에 불리한 prediction

- 긴 단어부터 명사 점수를 계산한 뒤, 해당 어절로부터 만들어진 (L, R) 을 L-R graph 에서 제거합니다.
  - L('아이디'), L('아이디어')의 빈도수가 각각 400, 300 라면,
  - L('아이디어')가 명사로 판단되면, 이를 포함하는 어절을 (L, R) 을 제거합니다.
    - '아이디어를' → [(아, 이디어를), (아이, 디어를), (아이디, 어를), (아이디어, 를), (아이디어를, '')
    - '아이디어' → [(아, 이디어), (아이, 디어), (아이디, 어), (아이디어, '')
  - L('아이디')에 연결된 R 의 총 빈도수 합은 100 보다 작아집니다.

Problem 4. 짧은 단어에 불리한 prediction

- ‘아이디어’가 포함된 어절을 지우면, ‘아이디‘, ‘아이’가 제대로 인식됩니다.

단어	(명사 점수, 빈도수) 확인된 어절을 제거하기 전	(명사 점수, 빈도수) 확인된 어절을 제거한 후
아이디어들	(1.0, 13)	(1.0, 13)
아이디어에	(0, 4)	(0, 4)
아이디어	(1.0, 329)	(1.0, 329)
아이디	(-0.526, 231)	(0.903, 100)
아이	(0.766, 568)	(1.0, 568)

Problem 5. 단일 어절로 기술되는 복합명사들

- 뉴스에서는 복합명사들이 조사 없이 단일 어절을 이루는 경우가 많습니다.
- 단일어절로만 이용되는 단어에 대해서 이들이 명사 집합으로 이뤄진 경우, 복합명사로 인식합니다.
  - {'경찰', '병원', '국립'} 이 명사이면 '국립경찰병원'을 명사로 인식합니다.

soynlp

- 위 과정을 포함한 명사 추출기입니다.

```
from soynlp.noun import LRNounExtractor_v2


noun_extractor = LRNounExtractor_v2()
nouns = noun_extractor.train_extract(sentences)
```