Minicycle

20210723

Goal

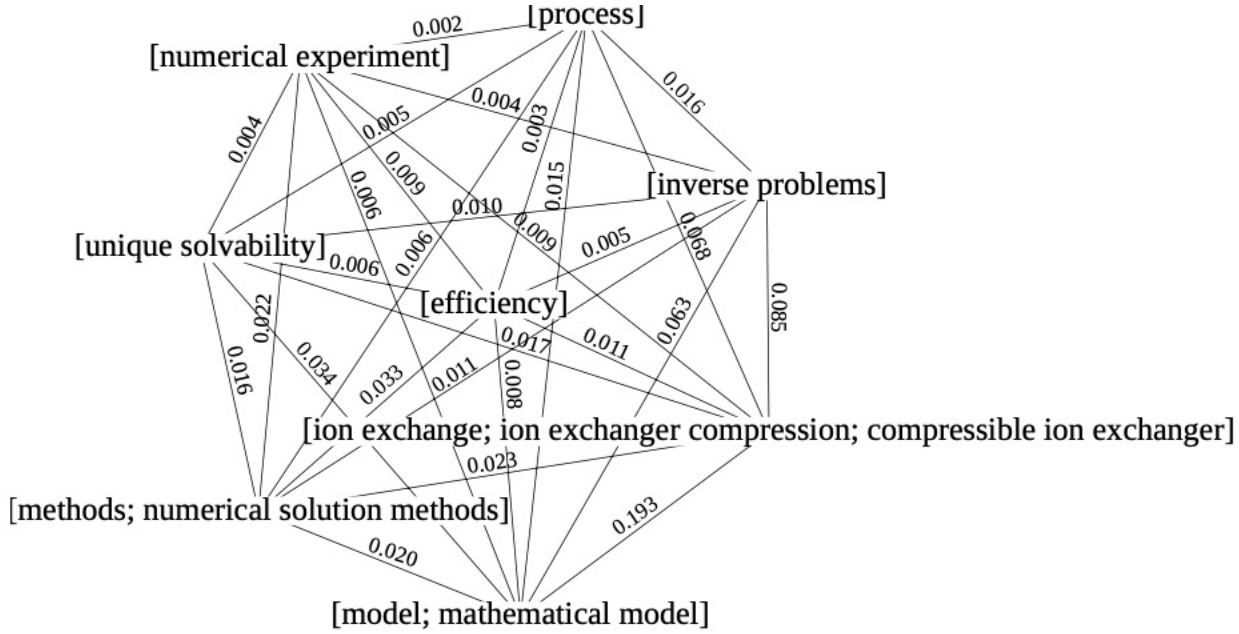
사용자

키워드 입력

뉴스 크롤링

텍스트 분석

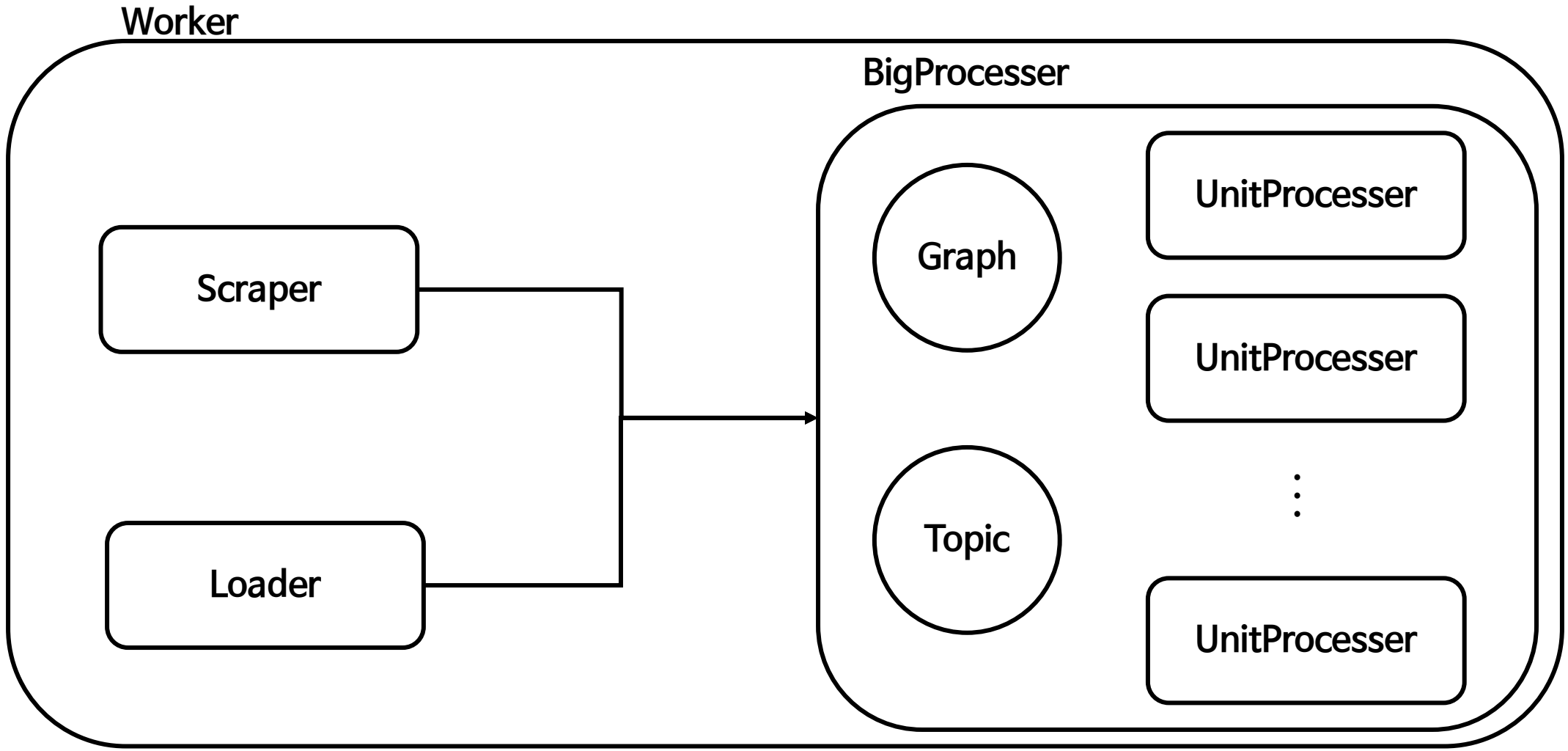
결과 시각화



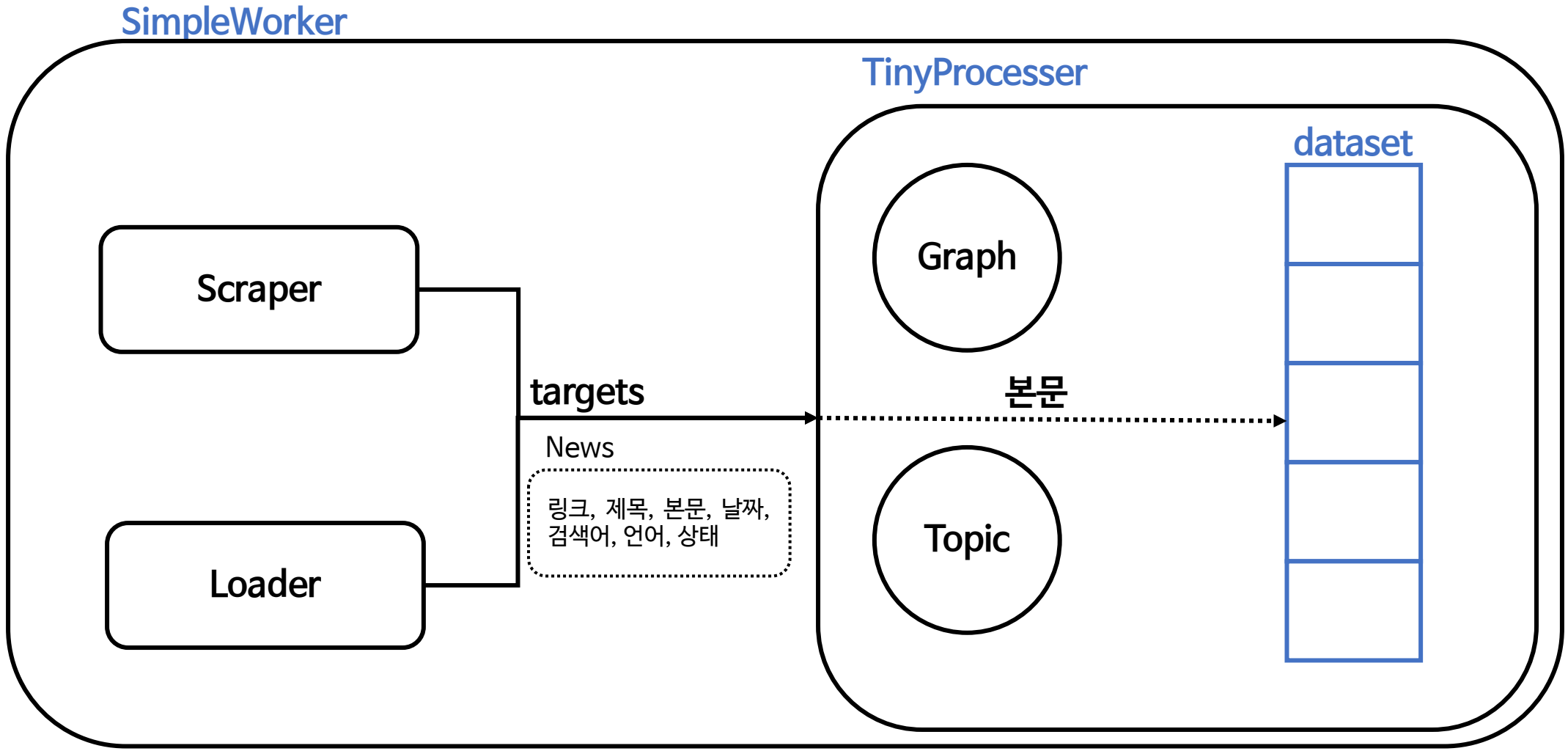
Keyphrases assigned by human annotators:
ion exchange; mathematical model; inverse problems; numerical solution methods; unique solvability; compressible ion exchanger; ion exchanger compression

Keyphrases assigned by TopicRank:
ion exchange; ***mathematical model***; ***inverse problems***; ***numerical solution methods***; process; ***unique solvability***; efficiency; numerical experiment

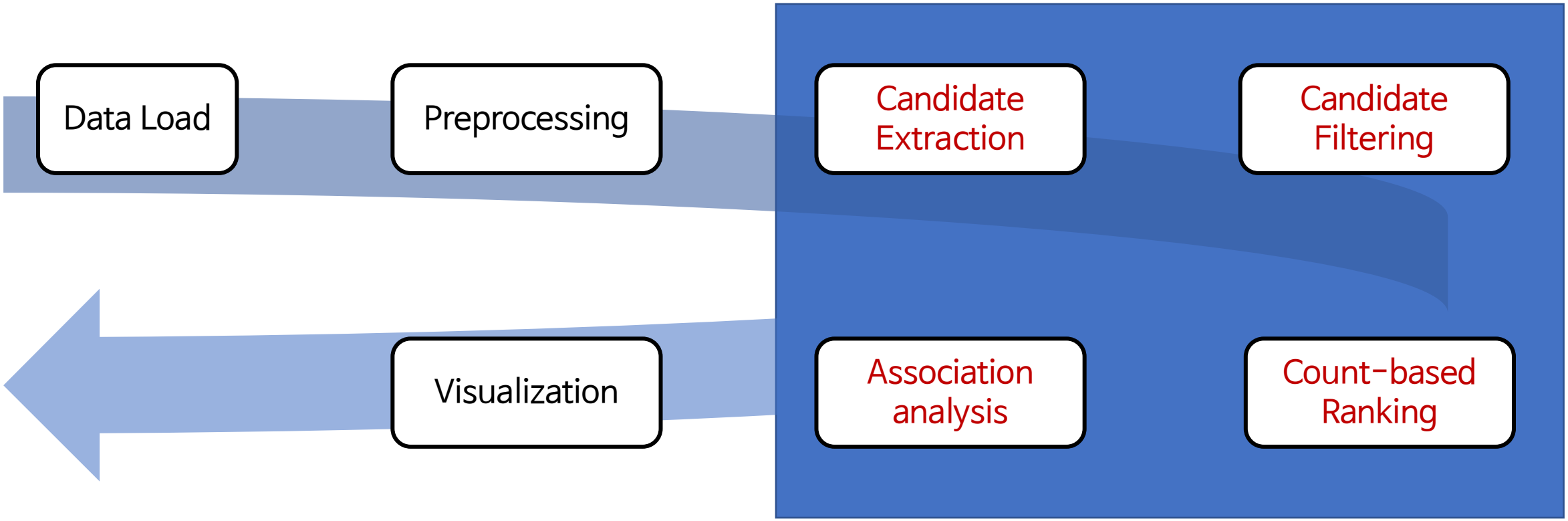
Architecture



What changed?



Workflow


Data Load
Preprocessing
Candidate Extraction
Candidate Filtering
Visualization
Association analysis
Count-based Ranking

Candidate Selection

```
def select_candidates(self):
        noun_extractor = LRNounExtractor_v2(verbose=True)
        nouns = noun_extractor.train_extract(self.dataset)
        self.candidates = nouns
```

- L-R Pattern

L + [R]

조사: 명사 가능성 ⬆️

어미: 명사 가능성 ⬇️

의미적 기능	문법적 기능
명사 / 동사 / 형용사 / 부사	조사 / 어미

재미있게	=	재미	+	있게	O
재미있는	=	재미	+	있는	O
서양 철학은	=	서양 철학	+	은	O
김정은	=	김정	+	은	?

soynlp

한국어 분석을 위한 pure python code 입니다. 학습데이터를 이용하지 않으면서 데이터에 존재하는 단어를 찾거나, 문장을 단어열로 분해, 혹은 품사 판별을 할 수 있는 비지도학습 접근법을 지향합니다.

About

한국어 자연어처리를 위한 파이썬 라이브러리입니다. 단어 추출/ 토크나이저 / 품사판별/ 전처리의 기능을 제공합니다.

https://github.com/lovit/soynlp

```
...
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
...	...	...	...	...	...	...

Count-based Ranking

```
def simpleTfIdf(self):
    c = list(self.candidates.keys())
    f_dataset = list()
    for i, _ in enumerate(self.dataset):
        f_dataset.append(helper.filter_text(self.dataset[i], c))
    total_words = sum([len(text.split()) for text in f_dataset])
    total_num_text = len(f_dataset)
    for each_c in c:
        for i, _ in enumerate(self.dataset):
            cnt = len(self.dataset[i].split(each_c)) - 1
            num_words = len(self.datset[i].split()) - 1
            self.tf_score[each_c] = self.tf_score.get(each_c, 0) + cnt / num_words
        self.idf_score[each_c] = sum([ 1 if each_c in dataset else 0 for dataset in f_dataset])
    self.tf_score.update([[candidate, tf/total_words] for candidate, tf in self.tf_score.items()])
    self.idf_score.update([[candidate, math.log(total_num_text / idf)] for candidate, idf in self.idf_score.items()])
    self.tf_idf_score = {key: self.tf_score[key] * self.idf_score[key] for key in self.tf_score.keys()}
```

$$\mathrm{tf}(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}},$$

term frequency

$$f_{t,d} \Big/ \sum_{t' \in d} f_{t',d}$$

$$\mathrm{idf}(t,D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

inverse document frequency

$$\log \frac{N}{n_t} = -\log \frac{n_t}{N}$$

$$\mathrm{tfidf}(t,d,D) = \mathrm{tf}(t,d) \cdot \mathrm{idf}(t,D)$$

Association Analysis

```
def pmi(self):
    corpus = list()
    c = list(self.candidates.keys())
    for doc in self.dataset:
        sents = kss.split_sentences(doc)
        for i, _ in enumerate(sents):
            corpus.append(helper.filter_text(sents[i], c))
    x, idx2vocab = sent_to_word_contexts_matrix(
        corpus,
        windows=5,
        min_tf=10,
        dynamic_weight=False,
        verbose=True
    )
    pmi_dok, px, py = pmi(
        x,
        min_pmi=0,
        alpha=0.0001
    )
    return idx2vocab, pmi_dok
```

	from	swerve	of	shore	to	bend	of	bay	,	brings
Window: 3	4	3	2	1	0	1	2	3	4	5
Scaling: flat	0	1	1	1	1	1	1	1	0	0
Scaling: $\frac{1}{n}$	0	$\frac{1}{3}$	$\frac{1}{2}$	$\frac{1}{1}$	1	$\frac{1}{1}$	$\frac{1}{2}$	$\frac{1}{3}$	0	0

- Example corpus:
  - I like deep learning.
  - I like NLP.
  - I enjoy flying.

counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

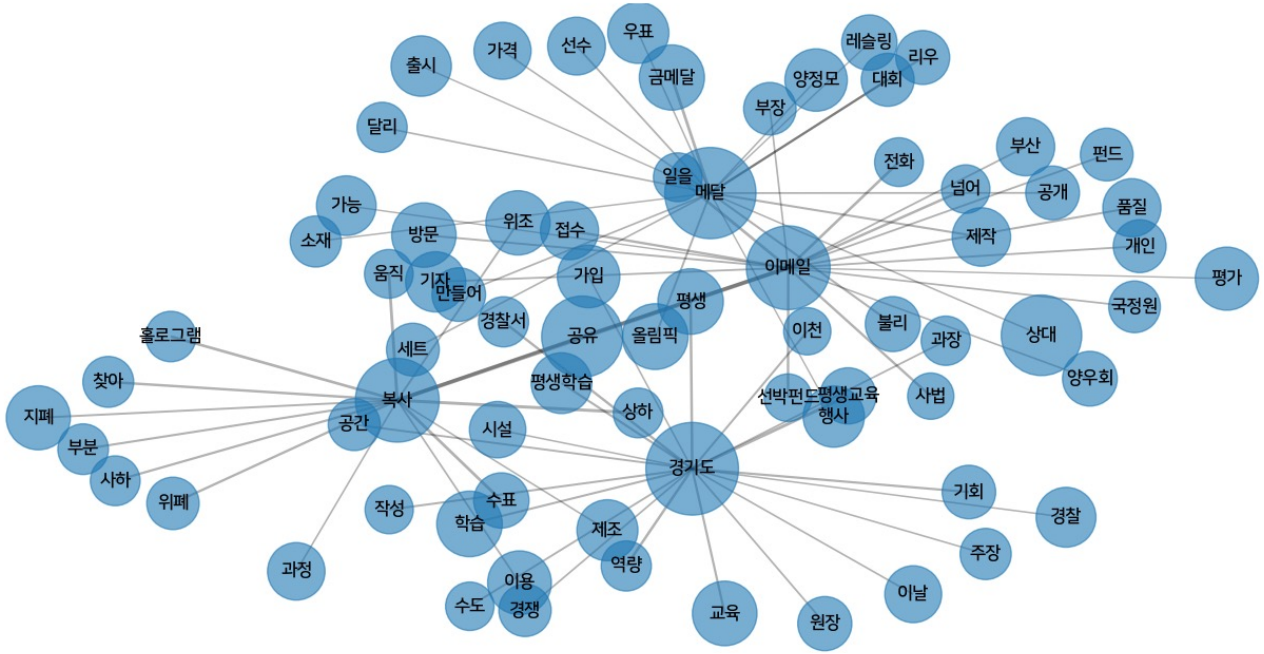
$$\text{pmi}(x;y) \equiv \log \frac{p(x,y)}{p(x)p(y)} = \log \frac{p(x|y)}{p(x)} = \log \frac{p(y|x)}{p(y)}.$$

Visualization

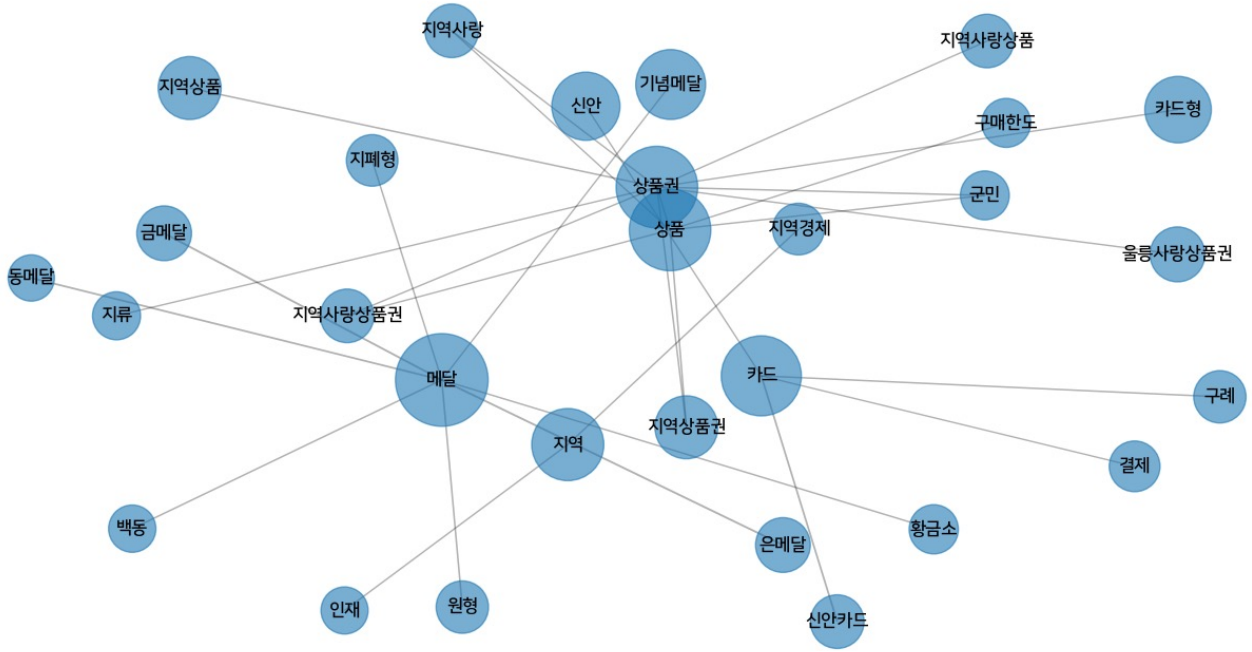
2016년

2021년

Visualization



2016년



2021년