TEXT MINING

텍스트 마이닝을 활용한 금융통화위원회 의사록 분석

DECIPHERING MONETARY POLICY BOARD MINUTES THROUGH TEXT MINING APPROACH: THE CASE OF KOREA PPT

SAMSUNG MULTI CAMPUS 2020 인공지능 자연어처리(NLP) 기반 기업 데이터 분석 과정

| 김인용, 강승범, 박진영, 전수빈 |

구현 논문

[제2019-1호] Deciphering Monetary Policy Board Minutes through Text Mining Approach: The Case of Korea

주제 : 통화 | 저자 : 박기영, 이영준, 김수현

연구조정실(02-759-5362) 2019.01.06 9232

1. BOK경제연구 제2019-1호 합본.pdf

2. BOK경제연구 제2019-1호 요약본.pdf

제목 : 텍스트 마이닝을 활용한 금융통화위원회 의사록 분석
저자 : 박기영(연세대학교 경제학부), 이영준(연세대학교 경영대학), 김수현(한국은행 경제연구원 국제경제연구실)

- https://www.bok.or.kr/portal/bbs/P0002454/view.do?nttId=10049321&menuNo=200431&pageIndex=2

목차

CONTENTS

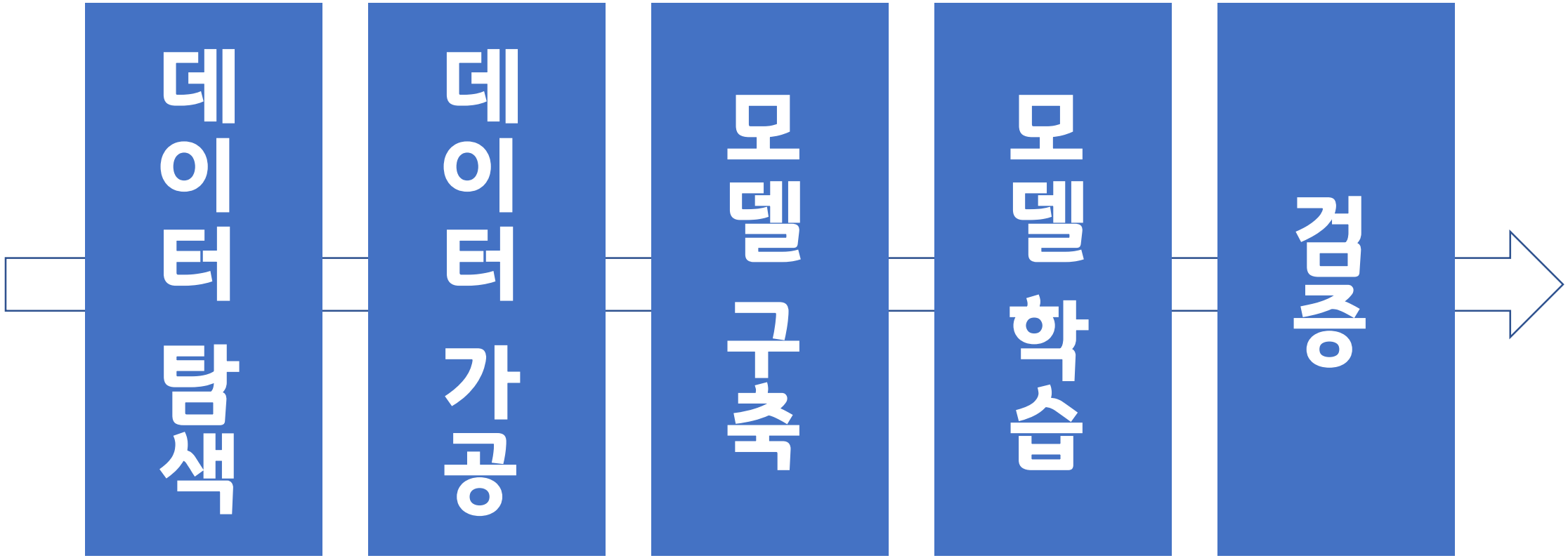
텍스트 마이닝을 활용한 금융통화위원회 의사록 분석

TEXT MINING

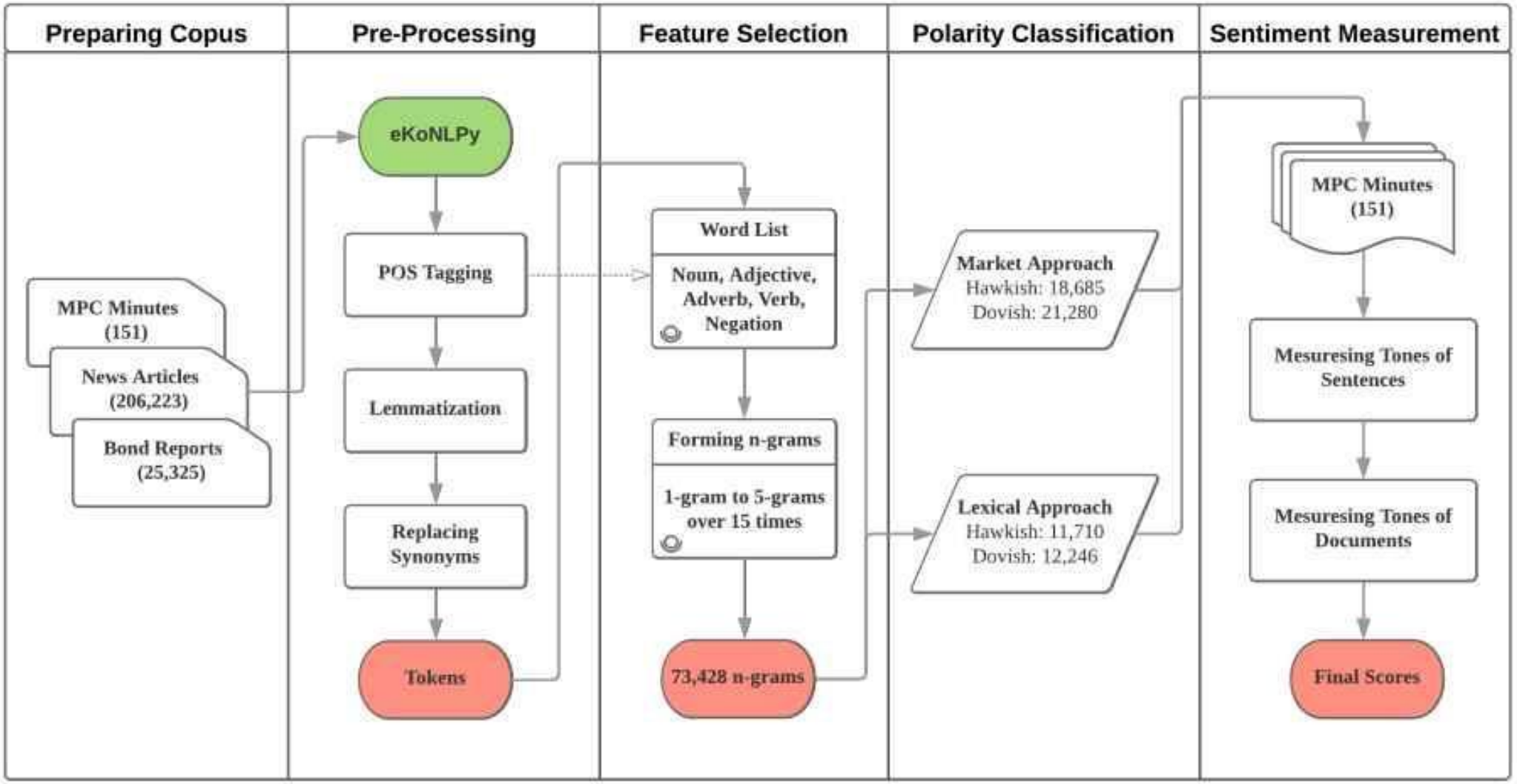
Processing

01

Processing



Processing


Preparing Copus
Pre-Processing
Feature Selection
Polarity Classification
Sentiment Measurement
MPC Minutes (151)
News Articles (206,223)
Bond Reports (25,325)
eKoNLPy
POS Tagging
Lemmatization
Replacing Synonyms
Tokens
Word List
Noun, Adjective, Adverb, Verb, Negation
Forming n-grams
1-gram to 5-grams over 15 times
73,428 n-grams
Market Approach
Hawkish: 18,685
Dovish: 21,280
Lexical Approach
Hawkish: 11,710
Dovish: 12,246
MPC Minutes (151)
Mesuresing Tones of Sentences
Mesuresing Tones of Documents
Final Scores

Processing

데이터 수집

금통위 의사록	162 개
네이버금융 채권분석보고서	3,581 개

금리	기준 금리 콜 금리
뉴스 기사	253,027 개
연합인포맥스 116,069 이데일리 81,382 연합뉴스 55,576	

Processing

전처리

금통위 의사록	섹션 분리	외환, 국제금융	문장 분리	Token 화 N-gram 화
162 개		금융 시장		

Processing

전처리

뉴스 기사	253,027 개	Token 화 N-gram 화
연합인포맥스 116,069 이데일리 81,382 연합뉴스 55,576		

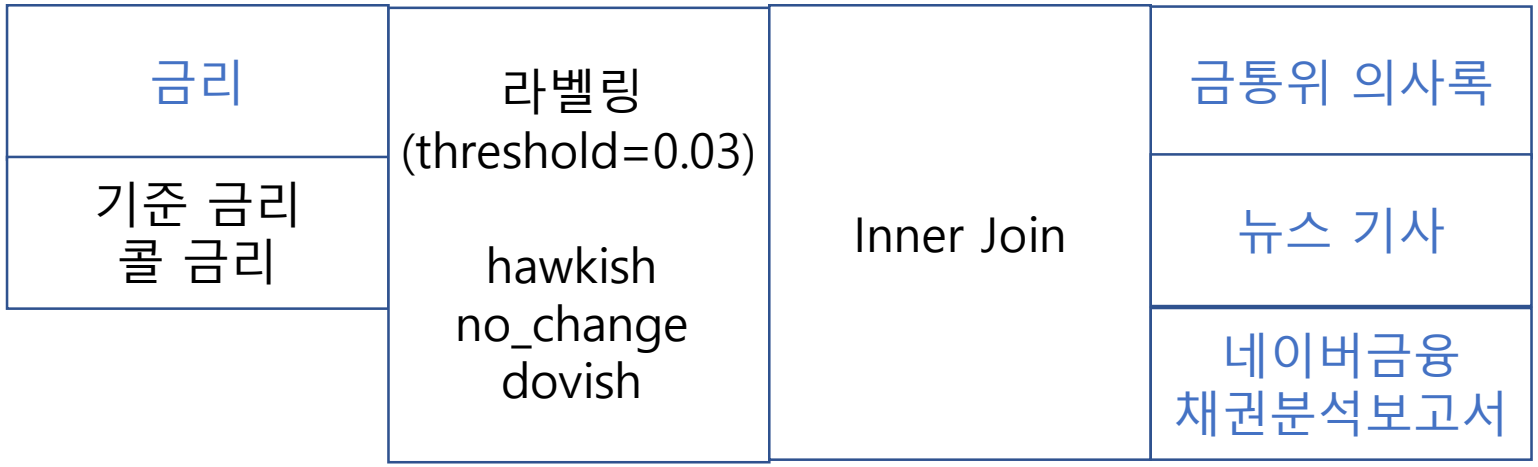
Processing

전처리

네이버금융 채권분석보고서	폭탄 처리	Token 화 N-gram 화
3,581 개	3,571 개	

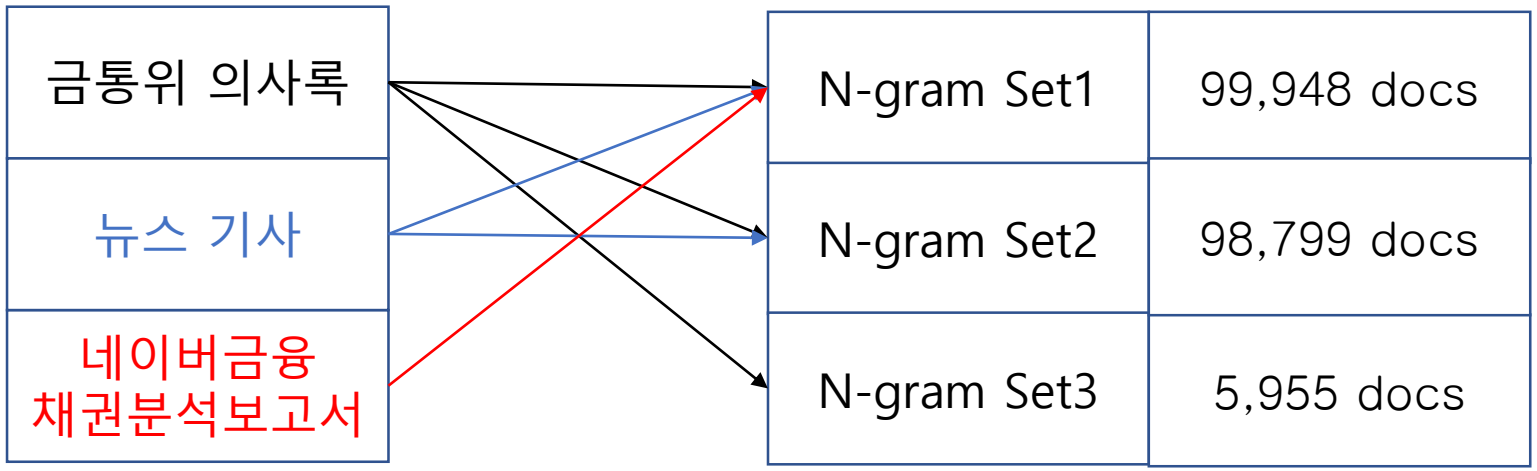
Processing

전처리



Processing

사전
구축



Processing

극성 분류

N-gram Set1	극성 분류	120,851 words	47,350 hawkish	46,673 dovish	Dictionary1
N-gram Set2		118,284 words	46,874 hawkish	45,389 dovish	Dictionary2
N-gram Set3		5,955 words	2,498 hawkish	2,544 dovish	Dictionary3

Processing

Tone 분석

Dictionary1	금통위 의사록 문장	금통위 의사록 문서	Tone_doc1
Dictionary2			Tone_doc2
Dictionary3			Tone_doc3

텍스트 마이닝을 활용한 금융통화위원회 의사록 분석

TEXT MINING

Process Details

02

데이터 가공

- 금통위 의사록, 네이버금융 채권 분석서 : PDF 형식

  PDF 파일 특징 상 텍스트를 온전히 추출하기 어려움
  여러 Library (poppler, pdftotxt, pdfminer, tika) 전부 실패

  HWP => DOCX => 텍스트로 변환 후 활용 (docx2txt)

N-gram 호환문제

- 전처리 미흡으로,
  각각의 문서의 형태가 통일되지 않음.

  그래서 같은 N-gram임에도 다른 형태로 저장.

  이로 계산한 Tone이 0값이 나옴.

N-gram 호환문제

- N-gram을 추가로 정제하여 형식을 맞춰주었음

```
[ ]   1 def preprocess_ngram(string):
      2     return ",".join([x.split("'")[1] for x in string.split(",")])
```

```
[ ]   1 minutes["ngram"]=minutes["ngram"].str.replace("@","")
      2
      3 news["ngram"]=news["ngram"].str.strip("[]")
      4 news["ngram"]=news["ngram"].apply(preprocess_ngram)
      5
      6 report["ngram"]=report["ngram"].str.strip("[]")
      7 report=report.drop(649)
      8 report["ngram"]=report["ngram"].apply(preprocess_ngram)
```

eKoNLPy_mpck

```
def text2ngram(text) :
    mpck = MPCK()
    tokens = mpck.tokenize(text)
    ngrams = mpck.ngramize(tokens)

    return tokens + ngrams
```

eKoNLPy_mpck

```
def __init__(self, classifier=None):
    if classifier is None:
        self.load_default_classifier()
    else:
        self.classifier = classifier
    self._tokenizer = Mecab()
    self._vocab = self.get_vocab(self.FILES['vocab'])
    self._positive_label = 'pos'
    self._negative_label = 'neg'
    self._min_ngram = 2
    self._ngram = 5
    self._delimiter = ';'
    self._start_tags = {'NNG', 'VA', 'VAX', 'MAG'}
    self._noun_tags = {'NNG'}
    self._aux_tags = aux_tags
    self._auxwords = {'못하/VX', '아니/VCN', '않/VX', '지만/VCP'}
```

eKoNLPy_mpck

```
def tokenize(self, text):
    tokens = self._tokenizer.sent_words(text)
    tokens = [w for w in tokens
              if ((w.split('/')[1] if '/' in w else None) not in self._aux_tags
                  or w in self._auxwords)]
    return tokens
```

eKoNLPy_mpck

```
def ngramize(self, tokens, keep_overlapping_ngram=False):
    ngram_tokens = []

    for pos in range(len(tokens)):
        for gram in range(self._min_ngram, self._ngram + 1):
            token = self.get_ngram(tokens, pos, gram)
            if token:
                if token in self._vocab:
                    ngram_tokens.append(token)
    if not keep_overlapping_ngram:
        filtered_tokens = []
        if len(ngram_tokens) > 0:
            ngram_tokens = sorted(ngram_tokens, key=lambda item: len(item), reverse=True)
            for token in ngram_tokens:
                existing_token = False
                for check_token in filtered_tokens:
                    if token in check_token:
                        existing_token = True
                        break
                if not existing_token:
                    filtered_tokens.append(token)
        ngram_tokens = filtered_tokens

    return ngram_tokens
```

eKoNLPy_mpck

```
def get_ngram(self, tokens, pos, gram):
    if pos < 0:
        return None
    if pos + gram > len(tokens):
        return None
    token = tokens[pos]
    check_noun = False

    tag = token.split('/')[1] if '/' in token else None
    if tag in self._start_tags:
        if tag in self._noun_tags:
            check_noun = True
        for i in range(1, gram):
            if tokens[pos + i] != tokens[pos + i - 1]:
                tag = tokens[pos + i].split('/')[1] if '/' in tokens[pos + i] else None
                if tag in self._noun_tags:
                    check_noun = True
                token += self._delimiter + tokens[pos + i]
        if check_noun:
            return token
        else:
            return None
    else:
        return None
```

polarity_score

```
[ ]  1 concate_data #hawkish,dovish로 분류된 문서(문장)의 N-gram
```

```
[ ]  1 from collections import defaultdict
     2 polarity_score_ls=defaultdict(list)
     3
     4 for i in range(30):
     5     X_train,_,y_train,_= train_test_split(concate_data["ngram"],concate_data["label"],test_size=0.1,stratify=concate_data["label"],shuffle=True)
     6
     7     vectorizer=CountVectorizer(tokenizer=split,lowercase=False)
     8     ngram_vec=vectorizer.fit_transform(X_train)
     9     ngram_dict=vectorizer.vocabulary_
    10
    11     model=MultinomialNB()
    12     model.fit(ngram_vec,y_train)
    13     likelyhood_prop=np.exp(model.feature_log_prob_)
    14
    15     for ngram in ngram_dict.keys():
    16         polarity_score_ls[ngram].append((likelyhood_prop[1]/likelyhood_prop[0])[ngram_dict[ngram]])
    17
```

Process Details

tone

```
[ ]  1 def sen_tone(ls):
     2     return len([x for x in ls if x in hawkish_ngram_ls])-len([x for x in ls if x in dovish_ngram_ls])
```

```
[ ]  1 sentence_data["sen_tone"]=sentence_data["ngram"].str.split(",").apply(sen_tone)
```

```
[ ]  1 hawkish_num=sentence_data.loc[sentence_data["sen_tone"]>0].pivot_table(index="date",values=["sentences"],aggfunc="count")
     2 dovish_num=sentence_data.loc[sentence_data["sen_tone"]<0].pivot_table(index="date",values=["sentences"],aggfunc="count")
```

```
[ ]  1 haw_dov_sen_tone=hawkish_num.merge(dovish_num,how="outer",left_index=True,right_index=True).fillna(0)
     2 haw_dov_sen_tone.columns=["hawk_num","dov_num"]
     3 haw_dov_sen_tone
```

date	hawk_num	dov_num
20050512	5.0	7.0
20050609	5.0	4.0
20050707	4.0	3.0
20050811	12.0	3.0
20050908	10.0	5.0
...	...	...
20190228	10.0	25.0
20190418	13.0	12.0
20190531	11.0	14.0
20190718	16.0	34.0
20190830	9.0	35.0

161 rows × 2 columns

tone

```
[ ]  1 tone_doc=(haw_dov_sen_tone["hawk_num"]-haw_dov_sen_tone["dov_num"])/(haw_dov_sen_tone["hawk_num"]+haw_dov_sen_tone["dov_num"])
     2 tone_doc
```

```
date
20050512   -0.166667
20050609    0.111111
20050707    0.142857
20050811    0.600000
20050908    0.333333
               ...
20190228   -0.428571
20190418    0.040000
20190531   -0.120000
20190718   -0.360000
20190830   -0.590909
Length: 161, dtype: float64
```

텍스트 마이닝을 활용한 금융통화위원회 의사록 분석

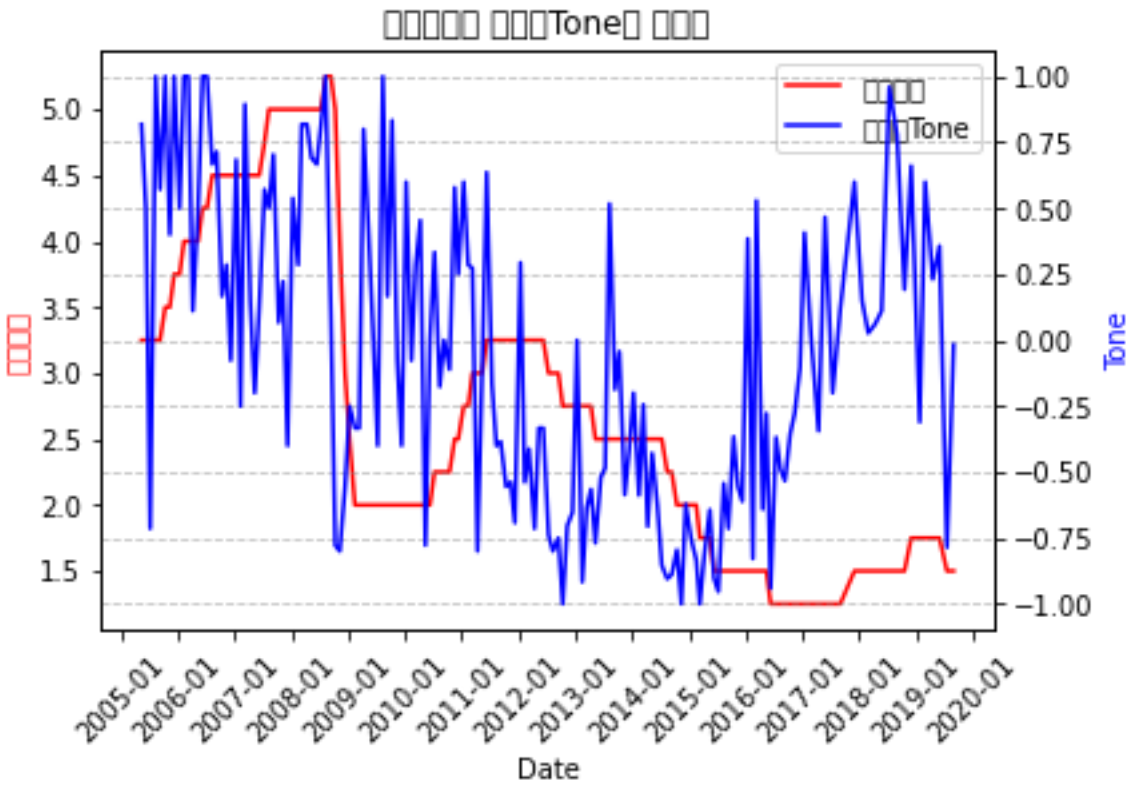
TEXT MINING

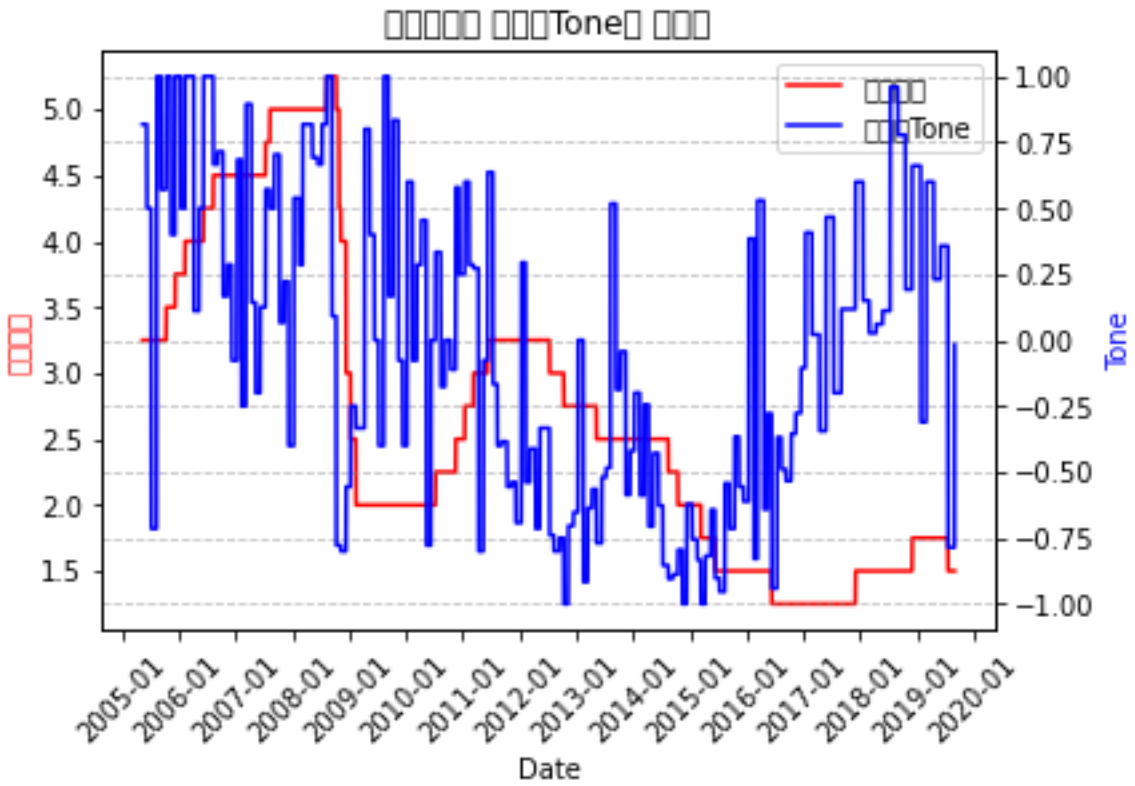
Consequence

03

Consequence

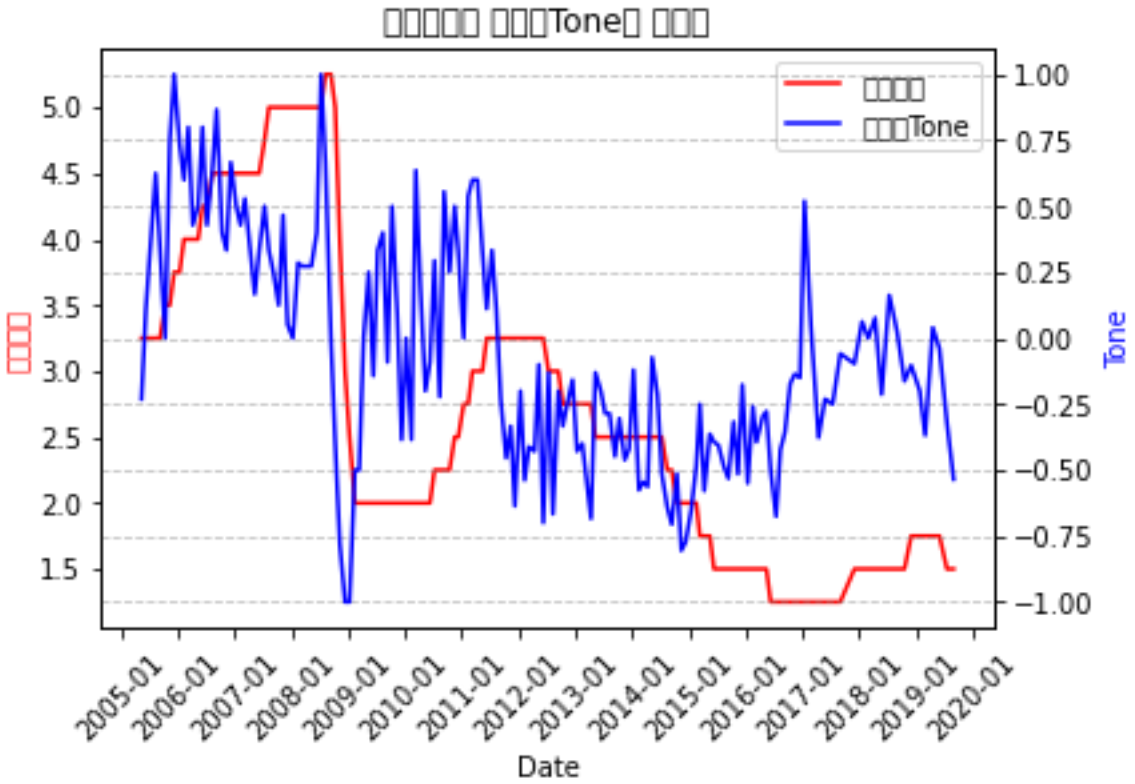
Corpus = [금통위 의사록]	
금통위 의사록 발행일 – 금리	매일 - 금리
0.3540	0.2988

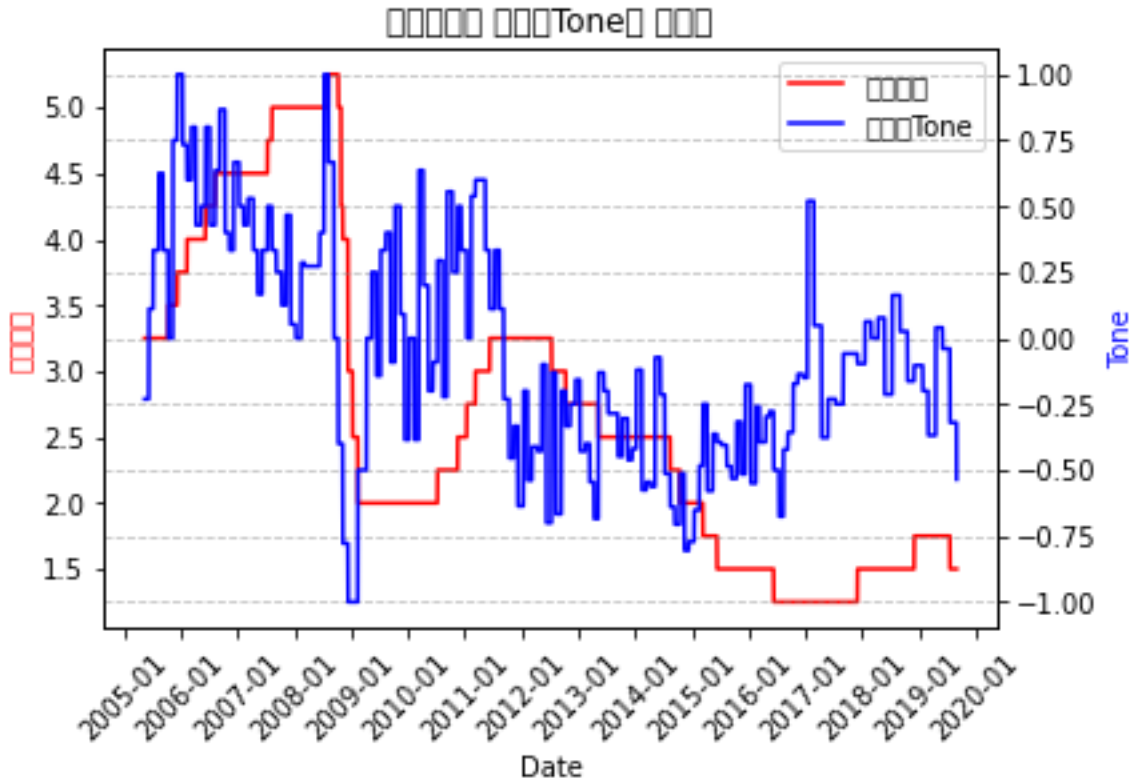




Consequence

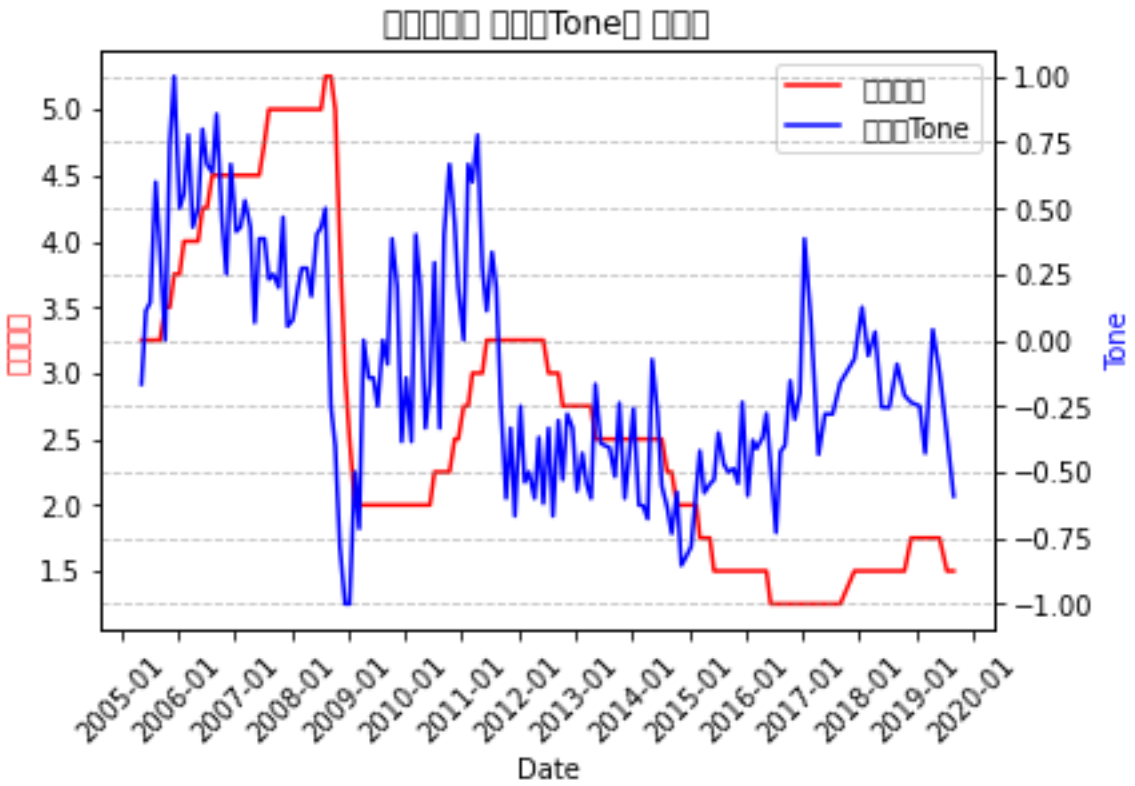
Corpus = [금통위 의사록, 뉴스 기사]	
금통위 의사록 발행일 – 금리	매일 - 금리
0.4967	0.4721

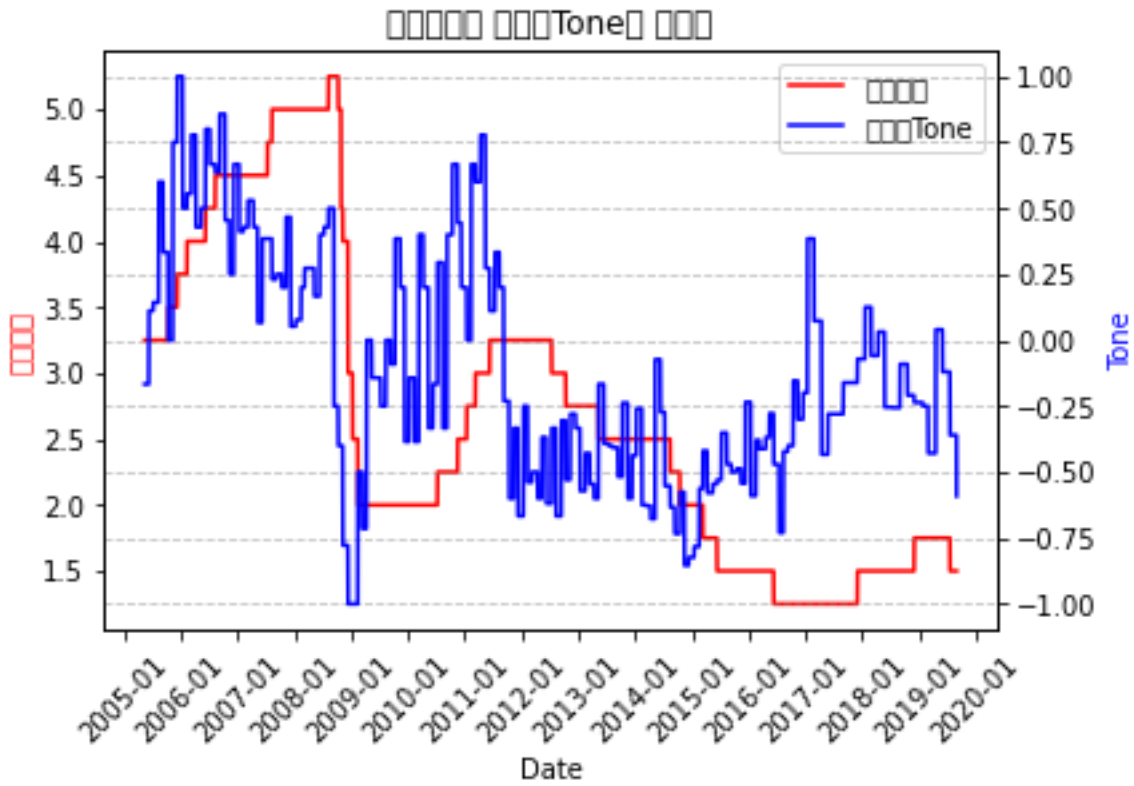




Consequence

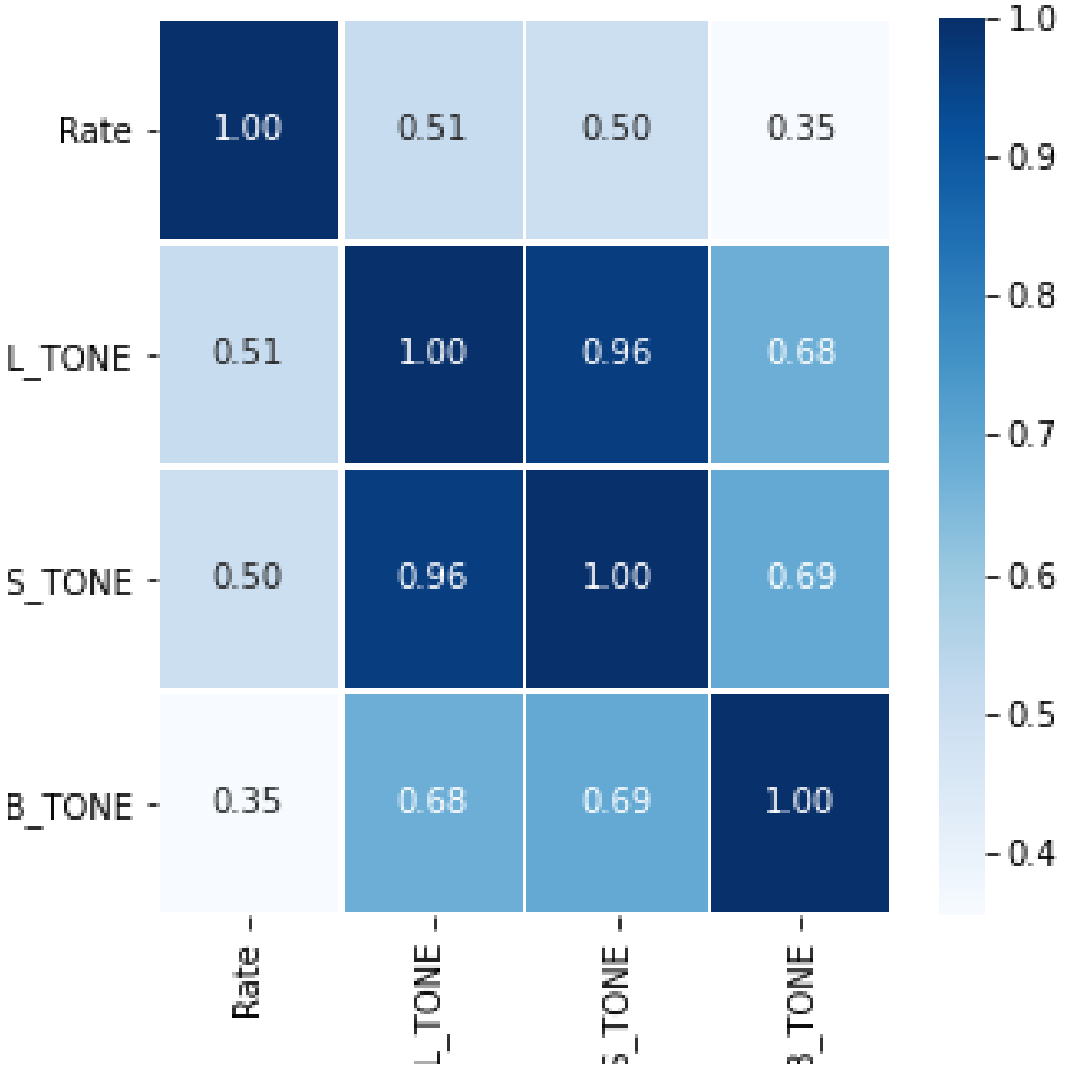
Corpus = [금통위 의사록, 뉴스기사, 채권 분석 보고서]	
금통위 의사록 발행일 – 금리	매일 - 금리
0.5126	0.4930





Consequence

Corpus 별 상관 계수 CorrPlot



Consequence

데이터 전처리를 충분히 하지 않았음에도
논문과 비슷한 결과를 얻을 수 있었다.

채권 분석 보고서 데이터가
부정적인 영향을 줄 것이라 예상했지만,
다행히 결과를 향상시켰다.

상관 계수 향상이 상당히 어렵다.

추후 연구 주제 제언

Corpus만 다르게 만들지 말고, 그에 따라서 threshold를 다르게 설정하면 어떨까

eKoNLPy(MPCK) 내의 parameter들을 수정하여 분석하면 어떨까

(컴퓨팅 파워가 된다면) 뉴스 기사를 기반으로 금리와의 상관관계를 Tone 분석하면 어떨까

THANKS

텍스트 마이닝을 활용한 금융통화위원회 의사록 분석

DECIPHERING MONETARY POLICY BOARD MINUTES THROUGH TEXT MINING APPROACH: THE CASE OF KOREA PPT

SAMSUNG MULTI CAMPUS 2020 인공지능 자연어처리(NLP) 기반 기업 데이터 분석 과정

|김인용, 강승범, 박진영, 전수빈 |