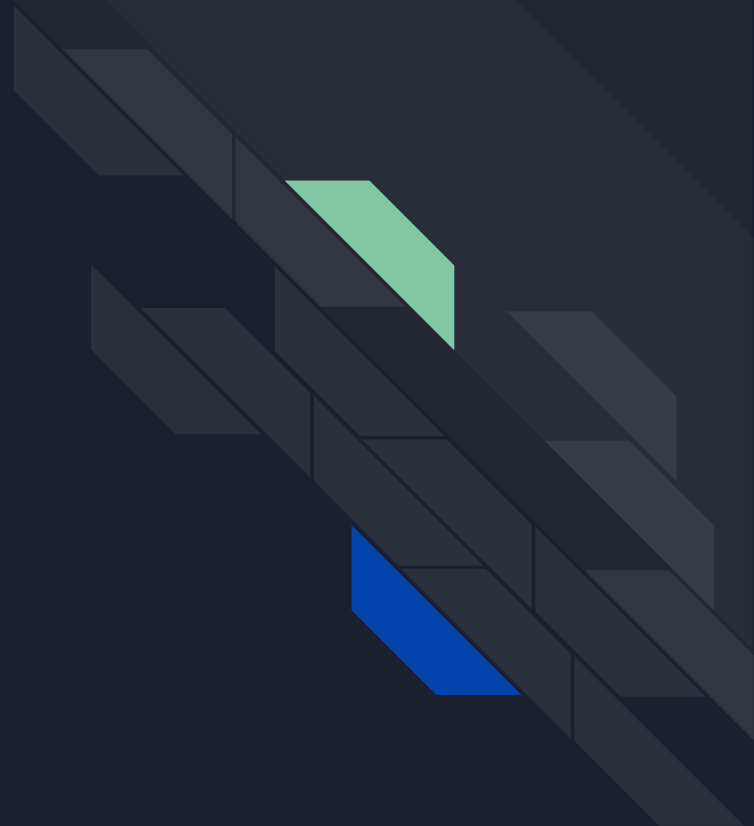




# 한국은행 논문 구현

1조: 김재경 원봄이 이현준 최예라

전처리



# 전처리

## 한국은행 금통위 의사록

- pdftotext로 얻은 raw 데이터 사용
- 문제점 1: 페이지 넘버
  - 해결책: regex
- 문제점 2: 띄어쓰기
  - 해결책:
    - KoSpacing
    - ChatSpace

이에 따라 **이 미** 소비를 제약하는 수준에 이르고 있는 소득대비 가계 부채비율은 앞으로...

이러한 점을 고려할 때 부채 **레 버리지**의 증가가 통화완화의 정도와 일정수준 연계...

더욱이 가계부채비율이 임계수준을 지나 증가세를 지속하는 경우 금융안정 **리스크**의 현...

금번 기준금리 조정으로 흑여 중기 기대인플레이션의 안정성이 저해되지 않도록...

향후 통화정책방향은 인플레이션 **기 조**가 안정적으로 정착되어 물가 목표 **수 령**에...

# 한국어 띄어쓰기: (Py)KoSpacing vs. ChatSpace

## (Py)KoSpacing

- 느린 편
- 공식 문서에서 더 잘 작동하는 듯

### PyKoSpacing

Python package for automatic Korean word spacing.

R version can be found [here](#).

**License** GPL v3

#### Introduction

Word spacing is one of the important parts of the preprocessing of Korean text. Proper word spacing affects the accuracy of subsequent text analysis. `PyKoSpacing` has fairly good performance, especially good for online text originated from SNS or SMS.

For example.

"아버지가방에들어가신다." can be spaced both of below.

1. "아버지가 방에 들어가신다." means "My father enters the room."
2. "아버지 가방에 들어가신다." means "My father goes into the bag."

## ChatSpace

- 빠른 편
- 채팅에 특화됨

### chatspace


build passing codecov 56%

핑퐁에서 만든 채팅체랑 잘 맞는 띄어쓰기 모델! 🏓😎

#### Getting Started

```
from chatspace import ChatSpace

spacer = ChatSpace()
spacer.space("안녕 만나서반가워 내이름은뽀로로라고해")
# '안녕 만나서 반가워 내 이름은 뽀로로라고 해'
```



# 한국어 띄어쓰기: (Py)KoSpacing vs. ChatSpace

## (Py)KoSpacing

```
1 from pykospadding import spacing
2
3 spacing(test_text)
```

'향후 통화정책 방향은 인플레이션 기조가 안정적으로 정착되어 물가목표 수렴에 대한 전망이 더욱 견조해질 때까지 완화된 통화기조를 유지해 갈 필요성이 있으며, 추가 금리조정 여부와 속도는 근원인플레이션의 변화와 민간소비의 회복 상황, 글로벌 금융순환의 변화가 실질중립금리에 미치는 영향 등에 기초하여 신중히 결정되어야 할 것으로 생각함.'

```
1 from chatspace import ChatSpace
2
3 spacer = ChatSpace()
4 spacer.space(test_text)
```

## ChatSpace

'향후 통화 정책 방향은 인플레이션 기조가 안정적으로 정착되어 물가 목표 수렴에 대한 전망이 더욱 견조해질 때까지 완화된 통화기조를 유지해 갈 필요성이 있으며, 추가금리 조정 여부와 속도는 근원인플레이션의 변화와 민간 소비의 회복 상황, 글로벌 금융 순환의 변화가 실질 중립금리에 미치는 영향 등에 기초하여 신중히 결정되어야 할 것으로 생각함.'

# 전처리

## 채권분석 보고서

- Ngram 개수
  - 자체적으로 만든 ngram 개수와 mpck.ngramize로 만든 ngram 개수 차이

ngram 개수:	4854	mpck.ngramize 개수:	104
ngram 개수:	1301	mpck.ngramize 개수:	18
ngram 개수:	10697	mpck.ngramize 개수:	297
ngram 개수:	1220	mpck.ngramize 개수:	19
ngram 개수:	1939	mpck.ngramize 개수:	57

```
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)
```

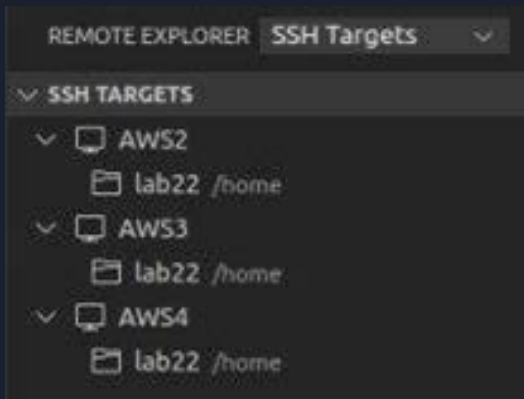
```
단기화/NNG:완화/NNG 30  
단발성/NNG:금리/NNG:인상/NNG 20  
단발성/NNG:금리/NNG:인상/NNG 54  
단발성/NNG:금리/NNG:인하/NNG 41  
단시간/NNG:어렵/VA 18  
단시간/NNG:어렵/VA:통화/NNG:강세/NNG 17  
단시간/NNG:어렵/VA:통화/NNG:강세/NNG:약세/NNG 17  
단시일/NNG:금리/NNG:인하/NNG 16  
단어/NNG:금리/NNG:인상/NNG 29  
단어/NNG:사용/NNG:금리/NNG:인상/NNG 16  
단어/NNG:삭제/NNG:금리/NNG:인상/NNG 28  
단일시장/NNG:개입/NNG:최대/NNG 26  
단칸지수/NNG:상승/NNG 17  
달러부채/NNG:늘/VV 16
```

```
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
```

# 전처리

## 뉴스 기사

- 연합뉴스
- 이데일리
- 연합인포맥스
  - 이유를 알 수 없이 오래 걸림
  - Colab 사용 제한
  - AWS 등에 분산해 처리



# 전처리

## 콜금리, 기준금리

- 값이 없는 빈 날짜도 채워줌
- pandas: 날짜 처리에 용이
  - `pd.to_datetime()`
  - `pd.date_range()`
  - `df.reindex(method='ffill')`

### 빈 날짜에 해당하는 금리 값 채우기

```
[ ] 1 # 날짜를 index로 설정
    2 baserate_df = baserate_df.set_index('time')
    3
    4 # 범위 내의 모든 날짜를 생성
    5 idx = pd.date_range('2005-05-01', '2020-05-28')
    6
    7 # ffill: 새로운 유효값이 나올 때까지 이전 유효값으로 이후의 비유효값을 채움
    8 baserate_df = baserate_df.reindex(idx, method='ffill')
    9
   10 # index 숫자로 재설정
   11 baserate_df.reset_index(inplace=True)
   12
   13 # 날짜 column 'time'으로 재명명
   14 baserate_df.rename(columns={'index': 'time'}, inplace=True)
   15
   16 baserate_df
```



	time	base_rate
0	2005-05-01	3.25
1	2005-05-02	3.25
2	2005-05-03	3.25



# POS tagging & n-gram 생성: 자체 시도

## 직접 논문에서 언급한 규칙을 적용하는 시도

- 잘 동작하나 eKoNLPy MPCK 사용 결과와 상당히 다름
- 부정 negation
  - 사용: VCN, 앓/VX
  - eKoNLPy: 못하/VX, 아니/VCN, 앓/VX, 지만/VCP
- 일부 요소는 구현하기 복잡
  - 겹치는 부분이 있는 n-gram은 제외하는 것 등

```
1 pos_list = ['NNG', 'VA', 'VAX', 'MAG', 'VA', 'VCN']
2
3 def get_pos_cleaned(col):
4     pos_clean = [(word, pos) for word, pos in col
5                   if (pos in pos_list) or (word, pos) == ('앓', 'VX')]
6
7     return pos_clean
```

```
[ ] 1 def generate_ngrams(pos_list, n):
2     # Use the zip function to help us generate n-grams
3     # Concatenate the tokens into ngrams and return
4     ngrams = list(zip(*[pos_list[i:] for i in range(n)]))
5
6     return ngrams
```

```
[ ] 1 from collections import defaultdict
2
3 ngrams_tmp_dict = defaultdict(list)
4
5 def get_sent_ngrams(col):
6     ngrams_tmp_dict[1].append(list(zip(col))) # n=1
7     for i in range(2, 6):
8         ngrams = generate_ngrams(col, n=i)
9         ngrams_tmp_dict[i].append(ngrams)
```

```
▶ 1 mpd_class_df['words'].apply(lambda x: get_sent_ngrams(x))
2
3 ngrams_tmp_dict[2]
```

```
☞ [('일부위원/NNG', '지난해/NNG'),
   ('지난해/NNG', '품작/NNG'),
   ('품작/NNG', '효과/NNG'),
```



## POS tagging & n-gram 생성: eKoNLPy 사용

### ekonlpy.sentiment.MPCK 사용

- 논문 내용이 잘 구현되어 있음
- tokens + ngrams 사용
  - mpck.tokenize()
  - mpck.ngramize()

```
1 from ekonlpy.sentiment import MPCK
2
3 def get_tokens_ngrams(text):
4     mpck = MPCK()
5
6     tokens = mpck.tokenize(text)
7     ngrams = mpck.ngramize(tokens)
8
9     ngrams_full = tokens + ngrams
10
11     return ngrams_full
```

# POS tagging & n-gram 생성: 자체 시도 vs. eKoNLPy 사용

자체 시도(모든 요소를 처리한 것은 아님)

ngrams
[(일부위원/NNG), (지 난해/NNG), (효 과/NNG), (금 년/NNG), ...
[(등위원/NNG), (건 설/NNG), (투 자/NNG), (선 행/NNG), ...
[(관련부서/NNG), (지 난해/NNG), (분 기/NNG), (건 설/NNG), ...
[(등위원/NNG), (수출 물량/NNG), (가 격/NNG), (요 인/NNG), ...
[(관련부서/NNG), (외 환위기/NNG), (직 후/NNG), (수출물 량/NN, ...

eKoNLPy 사용

mpck_ngrams
[풍작/NNG, 따르/VV, 효과/NNG, 담배가격/NNG, 인 상/NNG, 따르/V...
[건설/NNG, 투자/NNG, 선행/NNG, 지표/NNG, 면적/NNG, 건설/NN...
[대해/VV, 건설/NNG, 수주/NNG, 실적/NNG, 저조/NNG, 하/VV, ...
[수출/NNG, 가격/NNG, 요인/NNG, 해외/NNG, 수요/NNG, 요인/NN...
[대해/VV, 외환위기/NNG, 수출/NNG, 크/VA, 늘/VV, 앞/VX, 점/...

# 데이터 레이블링

문서 발행일

한 달 콜금리



날짜에 따른 매파/비둘기파 분류 dict 저장

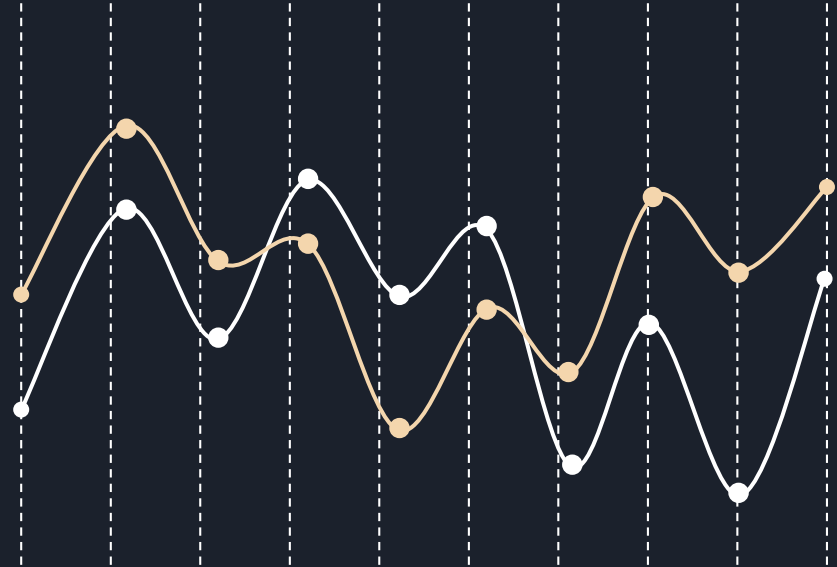
```
[ ] 1 from datetime import date
2 from dateutil.relativedelta import relativedelta
3
4 def check_hawk_dove(start_time):
5     end_time = start_time + relativedelta(months=+1)
6
7     start_callrate = callrate_df[callrate_df['time'] == start_time]['call_rate'].values[0]
8     end_callrate = callrate_df[callrate_df['time'] == end_time]['call_rate'].values[0]
9
10    change = ((end_callrate / start_callrate) - 1) * 100
11
12    # 각주 25) 의미 없는 변화는 제외하기 위해 +/-3bp를 threshold로 함, 1bp = 0.01%
13    if change >= 0.03:
14        return 1 # hawkish
15    elif change <= -0.03:
16        return -1 # dovish
17    else:
18        return 0 # no significant change
```

```
[ ] 1 callrate_hawk_dove_dict = {}
2
3 for start_time in callrate_df['time']:
4     end_time = start_time + relativedelta(months=+1)
5
6     if end_time <= max(callrate_df['time']):
7         callrate_hawk_dove_dict[start_time] = check_hawk_dove(start_time)
8
9 callrate_hawk_dove_dict
```

```
{Timestamp('2005-05-01 00:00:00'): -1,
Timestamp('2005-05-02 00:00:00'): -1,
Timestamp('2005-05-03 00:00:00'): -1,
```



# Tone 계산



# Polarity & tone 계산: eKoNLPy + NLTK

## mpck.bagging\_classifier()

- 옵션: 반복 수, train set 비율 등
- 결과(verbose=True) 예시:  
No. of iterations: 30. feature function: word, train ratio: 0.9, best words ratio: 0.8  
Best classifier: 5  
{'Accuracy': 0.5599789272848945, 'Pos precision': 0.5294918085794352, 'Pos recall': 0.5937962711147372, 'Neg precision': 0.5941396696959633, 'Neg recall': 0.5298465266558966}  
- Average metrics of classifiers -  
{'Accuracy': 0.5582353262236649, 'Pos precision': 0.5277308208333793, 'Pos recall': 0.5942203285689824, 'Neg precision': 0.5927134390892596, 'Neg recall': 0.5261714683180758}
- 실행 후 mpck.save\_classifier()로 저장
- 재사용시 mpck=MPCK(classifier=)로 로드

```
def bagging_classifier(self, dataset, iterations=20, feature_fn_name='word', train_ratio=0.8, best_words_ratio=0.8,
                      verbose=False, token_column='text', target_column='category',
                      pos_target_val=1, neg_target_val=-1):
    ...
    Bootstrap aggregating classifiers
    ...

    if verbose:
        print('\nNo. of iterations: {}. feature function: {}, train ratio: {}, best words ratio: {}'.format(
            iterations, feature_fn_name, train_ratio, best_words_ratio))

    clfs = []
    mlst = []

    for i in range(iterations):
        classifier, metrics = self.train_classifier(dataset, feature_fn_name=feature_fn_name, verbose=False,
                                                  train_ratio=train_ratio, best_ratio=best_words_ratio,
                                                  token_column=token_column, target_column=target_column,
                                                  pos_target_val=pos_target_val, neg_target_val=neg_target_val)

        clfs.append(classifier)
        mlst.append(metrics)

    mean_metrics = {}
    best_index = 0
    best_accuracy = 0
    for i, metrics in enumerate(mlst):
        if metrics['Accuracy'] > best_accuracy:
            best_accuracy = metrics['Accuracy']
            best_index = i
        if i == 0:
            for key in metrics.keys():
                mean_metrics[key] = metrics[key]
        else:
            for key in mean_metrics.keys():
                mean_metrics[key] += metrics[key]
    for key in mean_metrics.keys():
        mean_metrics[key] = mean_metrics[key] / len(mlst)

    if verbose:
        print('Best classifier: {}'.format(best_index))
        print(mlst[best_index])
        print('- Average metrics of classifiers -')
        print(mean_metrics)

    return best_index, clfs, mlst, mean_metrics
```

clfs: classifiers 저장됨

mlst: metrics (accuracy 등) 저장됨

iteration 횟수만큼 bagging 수행,  
개별 classifier는 train\_classifier()로

verbose=True 이면  
가장 좋은 classifier 및 metric 출력

# Polarity & tone 계산: eKoNLPy + NLTK

mpck.train\_classifier()

```
def train_classifier(self, dataset, feature_fn_name='word', train_ratio=0.8, verbose=False,
                    token_column='text', target_column='category', best_ratio=0.8,
                    pos_target_val=1, neg_target_val=-1):

    def word_feats(words):
        return dict([(word, True) for word in words])
```

```
negcutoff = int(len(negfeats) * train_ratio)
poscutoff = int(len(posfeats) * train_ratio)

trainfeats = negfeats[:negcutoff] + posfeats[:poscutoff]
testfeats = negfeats[negcutoff:] + posfeats[poscutoff:]

classifier = NaiveBayesClassifier.train(trainfeats) nltk.NaiveBayesClassifier
refsets = defaultdict(set)
testsets = defaultdict(set)

for i, (feats, label) in enumerate(testfeats):
    refsets[label].add(i)
    observed = classifier.classify(feats)
    testsets[observed].add(i)

metrics = {'Accuracy': nltk.classify.util.accuracy(classifier, testfeats),
          'Pos precision': precision(refsets[self._positive_label], testsets[self._positive_label]),
          'Pos recall': recall(refsets[self._positive_label], testsets[self._positive_label]),
          'Neg precision': precision(refsets[self._negative_label], testsets[self._negative_label]),
          'Neg recall': recall(refsets[self._negative_label], testsets[self._negative_label])}

if verbose:
    print(metrics)

return classifier, metrics
```

참고: nltk.NaiveBayesClassifier

```
1 classifier = nltk.NaiveBayesClassifier.train(training_set)
2
3 print("Classifier accuracy percent:",(nltk.classify.accuracy(classifier, testing_set))*100)
4
5 classifier.show_most_informative_features(15)
```

Models.py hosted with ❤ by GitHub

[view raw](#)

Classifier accuracy percent: 84.36

Most Informative Features

sleeve = True	neg : pos	=	15.8 : 1.0
lousy = True	neg : pos	=	13.0 : 1.0
unpretentious = True	pos : neg	=	12.9 : 1.0

<https://towardsdatascience.com/basic-binary-sentiment-analysis-using-nltk-c94ba17ae386>

# Polarity & tone 계산: eKoNLPy + NLTK

classifier.\_feature\_probdist

```
1 cpdist = classifier._feature_probdist
  cpdist: 클래스(매파/비둘기파)에 따른
  feature(n-grams) 확률 분포 dict
1 len(cpdist.keys())
269528
1 list(cpdist.keys())[0]
('neg', '중국은행/NNG')
```

mpck.classify()

```
def classify(self, tokens, intensity_cutoff=1.3):
    eps = 1e-6
    features = {token: True for token in tokens}
    result = self.classifier.prob_classify(features)
    pos_score = result.prob(self._positive_label)
    neg_score = result.prob(self._negative_label)
    polarity = pos_score - neg_score
    intensity = pos_score / (neg_score + eps) if polarity > 0 else neg_score / (pos_score + eps)
    polarity = polarity if intensity > intensity_cutoff else 0
    return {'Polarity': polarity, 'Intensity': intensity,
            'Pos score': pos_score, 'Neg score': neg_score}
```



# Polarity & tone 계산: eKoNLPy + NLTK

mpck.get\_informative\_features()

```
def get_informative_features(self, cutoff_ratio=1.2):
    cpdist = self.classifier._feature_probdist # probability distribution for feature values given labels
    fcnt = len(set([w for l, w in cpdist.items()]))
    feature_list = []
    epsilon = 1e-6
    feature = namedtuple('Feature', ['Word', 'Label', 'Polarity', 'Intensity'])

    for (fname, fval) in self.classifier.most_informative_features(n=fcnt):
        def labelprob(l):
            return cpdist[l, fname].prob(fval)

        labels = sorted([l for l in self.classifier._labels if fval in cpdist[l, fname].samples()],
                        key=labelprob)
        l0 = labels[0]
        l1 = labels[-1]
        l0_p = cpdist[l0, fname].prob(fval) + epsilon
        l1_p = cpdist[l1, fname].prob(fval) + epsilon
        ratio = l1_p / l0_p
        if ratio > cutoff_ratio:
            polar = ratio if l1 == self._positive_label else 1 / ratio
            label = 1 if l1 == self._positive_label else -1
            feature_list.append(feature(fname, label, polar, ratio))

    p = [f.Polarity for f in feature_list if f.Label > 0]
    n = [f.Polarity for f in feature_list if f.Label < 0]
    for i, f in enumerate(feature_list):
        if f.Label > 0:
            feature_list[i] = f._replace(Polarity=(f.Polarity - np.min(p)) / (np.max(p) - np.min(p)))
        elif f.Label < 0:
            feature_list[i] = f._replace(Polarity=(f.Polarity - np.max(n)) / (np.max(n) - np.min(n)))

    return feature_list
```

cpdist: 클래스(매파/비둘기파)에 따른  
feature(n-grams) 확률 분포


```
1 best_features = mpck.get_informative_features(cutoff_ratio=1.3)
```

```
1 len(best_features)
```

```
48703
```

```
1 best_features best_features: namedtuple
```

```
Feature(Word='그린카/NGG', Label=1, Polarity=0.23014070803804285, Intensity=4.747425119318334),
Feature(Word='통화정책/NGG; 경제/NGG; 견조/NGG; 성장/NGG', Label=1, Polarity=0.23014070803804285, Intensity=4.747425119318334),
Feature(Word='대토보상/NGG', Label=1, Polarity=0.23014070803804285, Intensity=4.747425119318334),
Feature(Word='소기업낙관지수/NGG; 상승/NGG', Label=-1, Polarity=-1.0729625153045743, Intensity=4.8278),
Feature(Word='머지않/VA; 금리/NGG; 인상/NGG', Label=-1, Polarity=-1.0729625153045743, Intensity=4.8278),
Feature(Word='지수/NGG; 실망/NGG', Label=-1, Polarity=-1.0731808975859014, Intensity=4.90530175392754),
Feature(Word='긴급/NGG; 금리/NGG; 인하/NGG', Label=-1, Polarity=-1.0731808975859014, Intensity=4.90530175392754),
Feature(Word='미국발/NGG; 금동/NGG; 충격/NGG', Label=-1, Polarity=-1.0720449347051046, Intensity=4.52740551137769),
Feature(Word='공공기관/NGG; 부채/NGG; 감축/NGG', Label=-1, Polarity=-1.0720449347051046, Intensity=4.52740551137769),
Feature(Word='하락/NGG; 압력/NGG; 줄/VV', Label=-1, Polarity=-1.0720449347051046, Intensity=4.52740551137769),
Feature(Word='악화/NGG; 악재/NGG', Label=-1, Polarity=-1.0720449347051046, Intensity=4.52740551137769),
Feature(Word='실물/NGG; 경제/NGG; 우려/NGG', Label=-1, Polarity=-1.0720449347051046, Intensity=4.52740551137769),
Feature(Word='스케이팅/NGG', Label=-1, Polarity=-1.0720449347051046, Intensity=4.52740551137769),
Feature(Word='cd/NGG; 금리/NGG; 인하/NGG', Label=-1, Polarity=-1.0724990696274652, Intensity=4.6712730)
```



# Polarity & tone 계산: eKoNLPy + NLTK

mpck.get\_informative\_features()

```
1 best_features_pos = [feature.Word for feature in best_features if feature.Label > 0]
2 print(len(best_features_pos))
3 print(best_features_pos[:10])
```

27768

['리타/NNG', '우월주의/NNG', '경제/NNG;전망/NNG;단기/NNG;위험/NNG;감소/NNG', '사이버먼데이/NNG', '포위사격/NNG', '금리/NNG;인상/NNG;절대/MAG;없/VA',

```
1 best_features_neg = [feature.Word for feature in best_features if feature.Label < 0]
2 print(len(best_features_neg))
3 print(best_features_neg[:10])
```

20935

['신뉴딜정책/NNG', '쌀직불금/NNG', '그린본드/NNG', '비밀/NNG', 'ff/NNG;금리/NNG;목표/NNG;높/VV', '영결식/NNG', '구제금융법안/NNG', '토빈/NNG', '대주

# Polarity & tone 계산: eKoNLPy + NLTK

## 문장 tone

$$\text{tone}_s = \frac{\text{No. of hawkish features} - \text{No. of dovish features}}{\text{No. of hawkish features} + \text{No. of dovish features}} \quad (2)$$

```
1 def calc_tone(ngrams):
2     eps = 1e-6
3
4     num_pos = sum(feature in best_features_pos for feature in ngrams)
5     num_neg = sum(feature in best_features_neg for feature in ngrams)
6
7     tone_sent = (num_pos - num_neg) / (num_pos + num_neg + eps)
8
9     return tone_sent
```

```
>>> from tqdm import tqdm
>>> tqdm.pandas(desc="my bar!")
C:\Users\Vera Choi\..conda\envs\py37\lib\site-packages\tqdm\std.py:668: FutureWarning: The Panel class is removed from pandas. Accessing it from the top-level
namespace will also be removed in the next version
  from pandas import Panel
>>> mpd_df_crop['tone'] = mpd_df_crop['mpck_ngrams'].progress_apply(lambda x: calc_tone(x))
my bar!: 7%|██████████|
```

```
def get_tone_bin(value):
    if value > 0:
        return 1
    elif value < 0:
        return -1
    else:
        return 0
```

mpd_ngram_df_crop						
	time class			mpck_ngrams	polarity	tone tone_bin
0	2005-05-12	-1	[풍작/NNG, 따르/VV, 효과/NNG, 담배가격/NNG, 인상/NNG, 따르/V...		-0.780861	-1.000000 -1
1	2005-05-12	-1	[건설/NNG, 투자/NNG, 선행/NNG, 지표/NNG, 면적/NNG, 건설/NN...		0.000000	0.000000 0
2	2005-05-12	-1	[대해/VV, 건설/NNG, 수주/NNG, 실적/NNG, 저조/NNG, 하/VV, ...		0.000000	0.999999 1
3	2005-05-12	-1	[수출/NNG, 가격/NNG, 요인/NNG, 해외/NNG, 수요/NNG, 요인/NN...		0.390964	0.000000 0
4	2005-05-12	-1	[대해/VV, 외환위기/NNG, 수출/NNG, 크/VA, 늘/VV, 앞/VX, 점/...		0.000000	-0.999999 -1
...	...	...		...	...	...
49068	2017-11-30	1	[이미/MAG, 소비/NNG, 제약/NNG, 하/XSV, 수준/NNG, 이르/VV,...		0.800545	0.999999 1
49069	2017-11-30	1	[이러/NNG, 점/NNG, 고려/NNG, 하/VV, 때/NNG, 부채/NNG, 레...		0.894988	0.000000 0
49070	2017-11-30	1	[가계/NNG, 부채비율/NNG, 임계/NNG, 수준/NNG, 지나/VV, 증가/N...		0.829999	0.500000 1
49071	2017-11-30	1	[금리/NNG, 조정/NNG, 흑여/MAG, 중기/NNG, 기대/NNG, 인플레이션...		0.487696	0.000000 0
49072	2017-11-30	1	[통화정책/NNG, 방향/NNG, 인플레이션/NNG, 기조/NNG, 안정/NNG, ...		0.746070	0.999999 1

tqdm.pandas(),  
progress\_apply()

# Polarity & tone 계산: eKoNLPy + NLTK

## 문서 tone

$$tone_i = \frac{No. of hawkish tone_{s,i} - No. of dovish tone_{s,i}}{No. of hawkish tone_{s,i} + No. of dovish tone_{s,i}} \quad (3)$$

```
1 def calc_tone_doc(tones):
2     eps = 1e-6
3
4     tones = list(tones)
5     num_pos = tones.count(1)
6     num_neg = tones.count(-1)
7
8     tone_doc = (num_pos - num_neg) / (num_pos + num_neg + eps)
9
10    return tone_doc
```

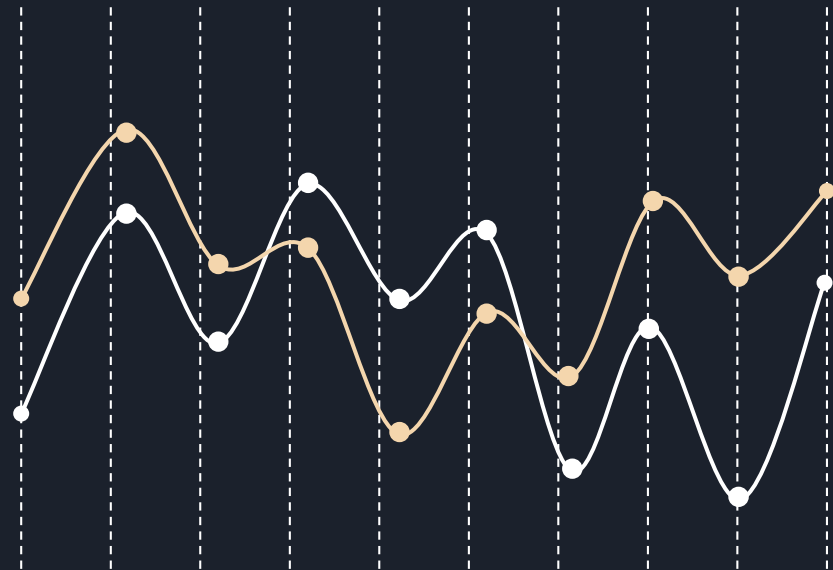
```
1 mpd_tone_df = mpd_ngram_df_crop.groupby('time')['tone_bin'].agg(lambda x: calc_tone_doc(x)).reset_index()
2 mpd_tone_df
```

문장별 tone 값을 시간(=의사록 발행일) 기준으로 합치고,  
매파(+1)/비둘기파(-1)인 문장의 수를 계산해 문서별 tone 계산

	time	tone_bin
0	2005-05-12	-0.019608
1	2005-06-09	-0.177570
2	2005-07-07	0.195652
3	2005-08-11	0.039216
4	2005-09-08	0.242718



## 정책금리와 비교



# 정책금리와 비교

## 상관계수

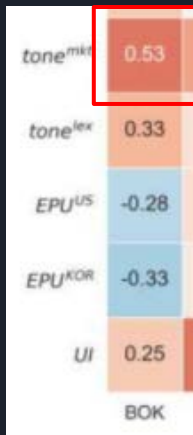
```
1 def get_mpd(value):
2     if value >= 0.25:
3         return 1
4     elif value < 0 and value >= -0.25:
5         return -1
6     elif value < -0.25:
7         return -2
8     else:
9         return 0
```

```
1 mpd_tone_df['mpd'] = mpd_tone_df['base_rate_change'].apply(lambda x: get_mpd(x))
2
3 mpd_tone_df
```

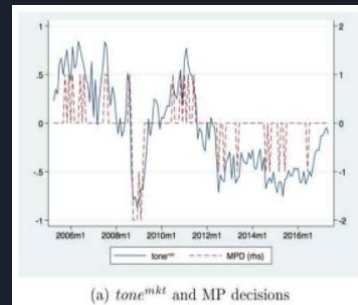
	time	tone_bin	base_rate	base_rate_change	mpd
0	2005-05-12	-0.019608	3.25	0.00	0
1	2005-06-09	-0.177570	3.25	0.00	0
2	2005-07-07	0.195652	3.25		
3	2005-08-11	0.039216	3.25		
4	2005-09-08	0.242718	3.25		

```
1 mpd_tone_df[['tone_bin', 'mpd']].corr()
```

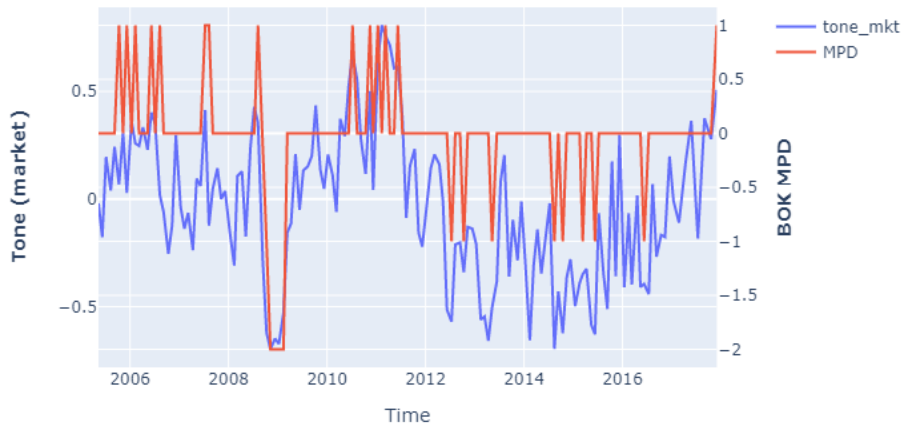
	tone_bin	mpd
tone_bin	1.00000	0.56767
mpd	0.56767	1.00000



## Tone (market) vs. 정책금리



## Tone (market) and BOK MPD



감사합니다

