

# KoBERT

## 다중 분류 모델을 통한

## 한국어 텍스트 인식 정확도 향상

## 목차

문제점

기존 인식 정확도

개선 방안

핵심 함수

모델 테스트

Check Point

전체 코드

# 문제점



가격  
regex

OCR  
방식

물품  
한정

# 문제점

## 1. 정규표현식 수정

인식 범위가 한정적이므로  
이를 수정하여 캡처 범위 개선함

구어체 추가 (마년, 13만, 1.2)  
파운드, 달러, 유로 단위 삭제 등

```
100원, 100만원을, 100만원, 100마년,
100만원을,
13만원에 거래할 수 있어요?
100만원,
200만원을,
만원을,
12원,
```

가격  
regex

OCR  
방식

물품  
한정

```
/\d+(<원|만원|마년>)|\d+만|만원|\d.\d(<만원|만>)?/g
```

```
Text Tests NEW
100원, 100만원을, 100만원, 100마년,
100만원을,
13만원에 거래할 수 있어요?
100만원,
200만원을,
만원을,
12원,
1.3, 1.5만원에,
```

# 문제점

## 1. 정규표현식 수정

## 2. 채팅 대화문 형식

대화문이 분리되지 않은 형태의  
스크린샷을 제공하여 OCR 분석

채팅을 주고받는 형식임을 감안,  
Turn 단위로 분리하여 분석할 필요

가격  
regex

OCR  
방식

물품  
한정

Select Language:

☐ English

☒ Korean

Upload a Screenshot:



# 문제점

## 1. 정규표현식 수정

## 2. 채팅 대화문 형식

## 3. 거래 물품 추가

기존 4개로 주어지는 계약 타입을  
선택하지 않아도 물품 인식 가능

가격  
regex

OCR  
방식

물품  
한정

Select a Contract Type:

Type - 1

Used Bike Sales Contract

Type - 2

Used Mobile Phone Sales  
Contract

Type - 3

Gift Certificate Contract

Type - 4

Real Estate Sublease  
Agreement

# 기존 인식 정확도



안녕하세요  
네 안녕하세요  
물건 살려는 분 맞으시죠  
네네 자전거 캐논데일 사고싶어용  
서울역 근방에서 보시죠  
좋아요  
네  
얼마죠?  
네6만원입니다  
다음주 수요일에 거래 어떠세요?  
좋아요 그때 봐요

## 개선 전

날짜	품목	위치	가격
-	자전거	-	X

## 개선 후

날짜	품목	위치	가격
다음주	자전거	서울역	6만원

# 기존 인식 정확도



안녕하세요

자전거 사곤싶브슷디ㅏ

평택역앞버스정류장에서요

50만원에거래하고 싶습니다

10월 1일에봐어요.

개선 전

날짜	품목	위치	가격
-	자전거	-	X

\* 모델명: 사곤싶브슷디ㅏ

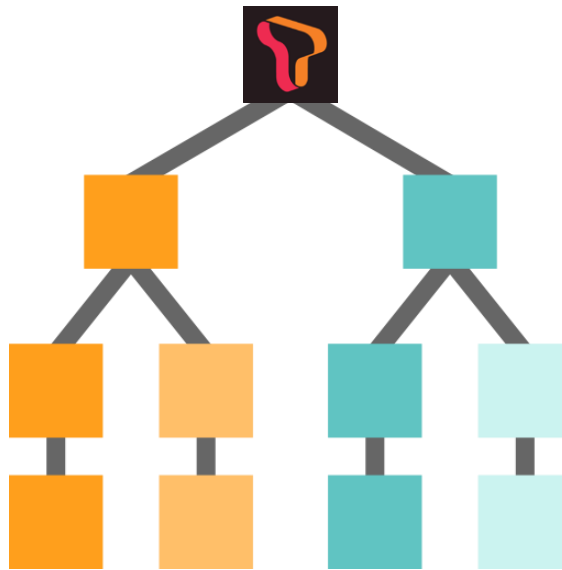
개선 후

날짜	품목	위치	가격
10월 1일	자전거	평택역	50만원

# 개선방안

## SKTBrain/**KoBERT**

Korean BERT pre-trained cased (KoBERT)



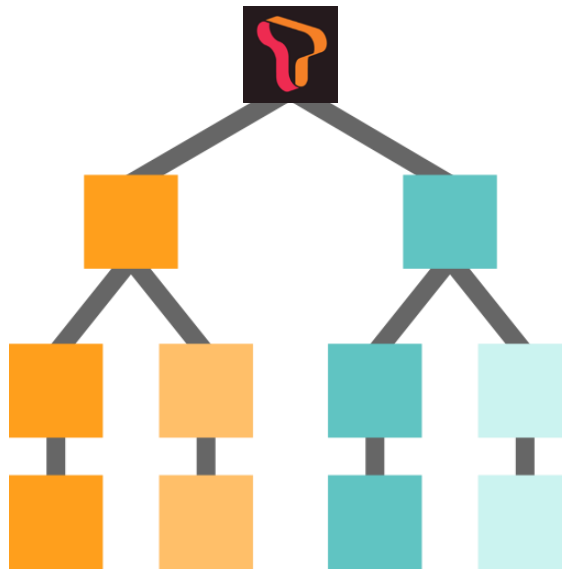


## SKTBrain/**KoBERT**

Korean BERT pre-trained cased (KoBERT)

1차  
분류

각 라벨 당 최고 확률 문장



2차  
분류

라벨 당 핵심 단어만

# 개선방안

1. 라벨 생성
2. 데이터 생성
3. 모델 학습
4. 문장 분리
5. Output



1차  
분류

# 개선방안

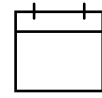
1. 라벨 생성

2. 데이터 생성

3. 모델 학습

4. 문장 분리

5. Output



날짜



품목



위치



가격

1차  
분류

# 개선방안

1. 라벨 생성

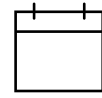
2. 데이터 생성

3. 모델 학습

4. 문장 분리

5. Output

1차  
분류



날짜



품목



위치



가격

800개

# 개선방안

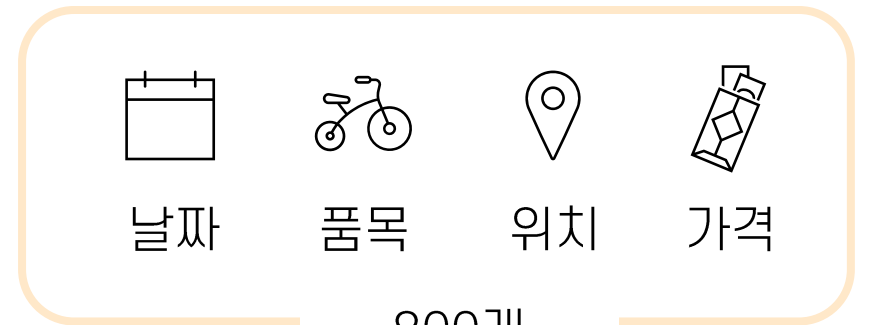
1. 라벨 생성

2. 데이터 생성

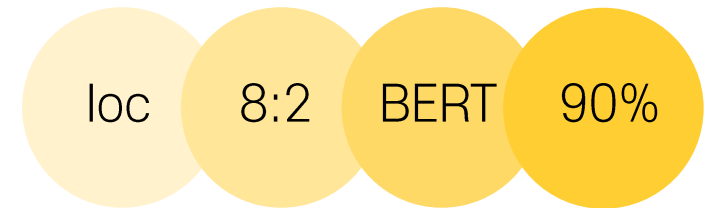
3. 모델 학습

4. 문장 분리

5. Output



800개



# 개선방안

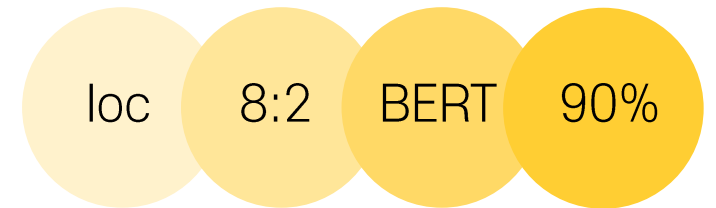
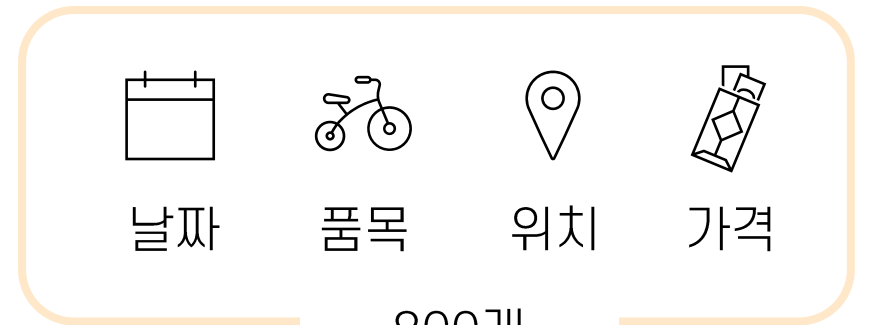
1. 라벨 생성

2. 데이터 생성

3. 모델 학습

4. 문장 분리

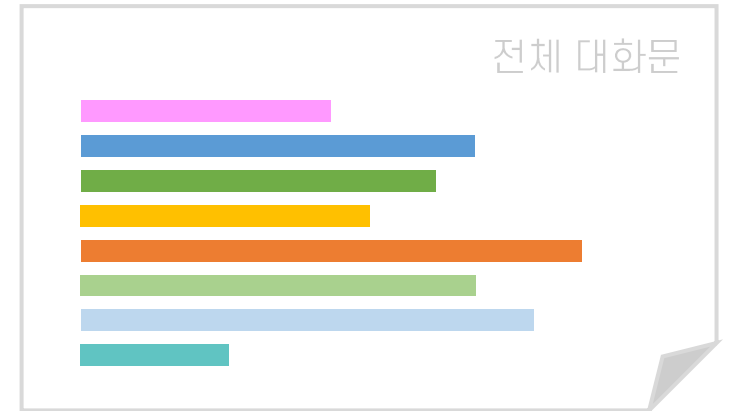
5. Output



# 개선방안

1. 라벨 생성
2. 데이터 생성
3. 모델 학습
4. 문장 분리
5. Output

1차  
분류



날짜    그럼 8월 5일 어떠세요?

품목    사서 얼마 타지도 않은 자전거예요ㅜㅜ

위치    저는 서울역이 가까우니까 거기서 뵙죠

가격    68만원에 해요

# 개선방안

1. 사용자 사전 추가
2. 코모란 명사 추출
3. 시간, 품목, 위치 추출
4. 가격 추출



2차  
분류



# 개선방안

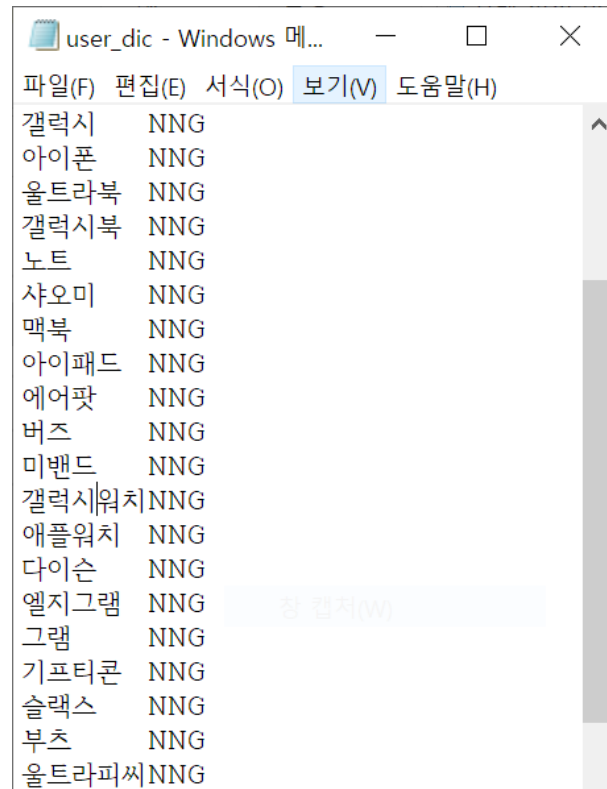
## 1. 사용자 사전 추가

## 2. 코모란 명사 추출

## 3. 시간, 품목, 위치 추출

## 4. 가격 추출

2차  
분류



갤럭시	NNG
아이폰	NNG
울트라북	NNG
갤럭시북	NNG
노트	NNG
샤오미	NNG
맥북	NNG
아이패드	NNG
에어팟	NNG
버즈	NNG
미밴드	NNG
갤럭시워치	NNG
애플워치	NNG
다이슨	NNG
엘지그램	NNG
그램	NNG
기프티콘	NNG
슬랙스	NNG
부츠	NNG
울트라파씨	NNG

# 개선방안

1. 사용자 사전 추가

2. 코모란 명사 추출

3. 시간, 품목, 위치 추출

4. 가격 추출

2차  
분류

날짜 그럼 8월 5일 어떠세요?

품목 사서 얼마 타지도 않은 자전거예요ㅜㅜ

위치 저는 서울역이 가까우니까 거기서 뵙죠

가격 68만원에 해요

날짜 그럼 8월 5일 어떠세요?

품목 사서 얼마 타지도 않은 자전거예요ㅜㅜ

위치 저는 서울역이 가까우니까 거기서 뵙죠

가격 68만원에 해요

# 개선방안

1. 사용자 사전 추가

2. 코모란 명사 추출

3. 시간, 품목, 위치 추출

4. 가격 추출

2차  
분류

날짜 그럼 8월 5일 어떠세요?

품목 사서 얼마 타지도 않은 자전거예요ㅜㅜ

위치 저는 서울역이 가까우니까 거기서 뵙죠

가격 68만원에 해요

날짜 그럼 8월 5일 어떠세요?

품목 사서 얼마 타지도 않은 자전거예요ㅜㅜ

위치 저는 서울역이 가까우니까 거기서 뵙죠

# 개선방안

1. 사용자 사전 추가

2. 코모란 명사 추출

3. 시간, 품목, 위치 추출

4. 가격 추출

2차  
분류

날짜 그럼 8월 5일 어떠세요?

품목 사서 얼마 타지도 않은 자전거예요ㅜㅜ

위치 저는 서울역이 가까우니까 거기서 뵙죠

가격 68만원에 해요

날짜 그럼 8월 5일 어떠세요?

품목 사서 얼마 타지도 않은 자전거예요ㅜㅜ

위치 저는 서울역이 가까우니까 거기서 뵙죠

가격 68만원에 해요

# 핵심 함수 in 1차 분류

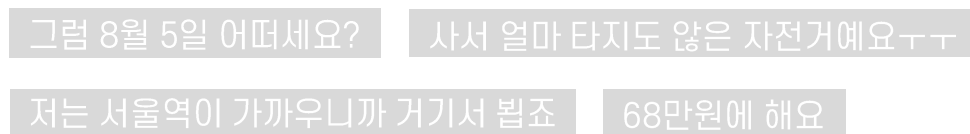
## 1. 문장 분리를 위한 KSS



KSS (Korean Sentence Splitter)

한 채팅에 2개 이상의 문장이 있을 경우 분리함  
문장 분리를 통해 **문장 당 라벨 배정 확률**을 높임

## 2. 1차 분류 수행 sorting



**라벨 당 최고 확률**을 가지는 문장을 하나씩 선정함  
'안녕하세요', '좋아요' 등 계약서 생성에 불필요한 문장 제거

# 핵심 함수 in 1차 분류

## 1. 문장 분리를 위한 KSS



좋아요

그럼 8월 5일 어떠세요?

좋아요

그럼 8월 5일 어떠세요?

## 2. 1차 분류 수행 sorting

그럼 8월 5일 어떠세요?

사서 얼마 타지도 않은 자전거예요ㅜㅜ

저는 서울역이 가까우니까 거기서 뵙죠

68만원에 해요

KSS (Korean Sentence Splitter)

한 채팅에 2개 이상의 문장이 있을 경우 분리함  
문장 분리를 통해 **문장 당 라벨 배정 확률**을 높임

라벨 당 최고 확률을 가지는 문장을 하나씩 선정함  
'안녕하세요', '좋아요' 등 계약서 생성에 불필요한 문장 제거

# 핵심 함수 in 1차 분류

## 1. 문장 분리를 위한 KSS



좋아요

그럼 8월 5일 어떠세요?

좋아요

그럼 8월 5일 어떠세요?

## 2. 1차 분류 수행 sorting

그럼 8월 5일 어떠세요?

사서 얼마 타지도 않은 자전거예요ㅜㅜ

저는 서울역이 가까우니까 거기서 뵙죠

68만원에 해요

KSS (Korean Sentence Splitter)

한 채팅에 2개 이상의 문장이 있을 경우 분리함  
문장 분리를 통해 **문장 당 라벨 배정 확률**을 높임

라벨 당 최고 확률을 가지는 **문장**을 하나씩 선정함  
'안녕하세요', '좋아요' 등 계약서 생성에 불필요한 문장 제거

# 핵심 함수 in 2차 분류

## 1. Komoran

그럼 8월 5일 어떠세요? 사서 얼마 타지도 않은 자전거예요ㅜㅜ  
저는 서울역이 가까우니까 거기서 뵙죠 68만원에 해요

계약서에 들어갈 **핵심어 추출**을 위해 명사 추출 사용  
코모란이 인식하지 못하는 NNG는 **사용자 사전**으로 보완  
**한 문장에 여러 개의 라벨이 포함되는 경우**를 백업 가능

## 2. 2차 분류 수행 sorting

그럼 8월 5일 어떠세요? 사서 얼마 타지도 않은 자전거예요ㅜㅜ  
저는 서울역이 가까우니까 거기서 뵙죠 68만원에 해요

**라벨 당 최고 확률**을 가지는 **핵심 단어**를 하나씩 선정함  
가격은 정규표현식 (코모란이 숫자 추출이 불가능하기 때문)  
최종 결과 보안을 위해 final\_check 함수로 보완



# 핵심 함수 in 2차 분류

## 1. Komoran

그럼 8월 5일 어떠세요? 사서 얼마 타지도 않은 자전거예요ㅜㅜ  
저는 서울역이 가까우니까 거기서 뵙죠 68만원에 해요

계약서에 들어갈 **핵심어 추출**을 위해 명사 추출 사용  
코모란이 인식하지 못하는 NNG는 **사용자 사전**으로 보완  
**한 문장에 여러 개의 라벨이 포함되는 경우**를 백업 가능

## 2. 2차 분류 수행 sorting

그럼 8월 5일 어떠세요? 사서 얼마 타지도 않은 자전거예요ㅜㅜ  
저는 서울역이 가까우니까 거기서 뵙죠 68만원에 해요

라벨 당 최고 확률을 가지는 **핵심 단어**를 하나씩 선정함  
가격은 정규표현식 (코모란이 숫자 추출이 불가능하기 때문)  
최종 결과 보안을 위해 final\_check 함수로 보완

# 핵심 함수 in 2차 분류

## 1. Komoran

그럼 8월 5일 어떠세요? 사서 얼마 타지도 않은 자전거예요ㅜㅜ  
저는 서울역이 가까우니까 거기서 뵙죠 68만원에 해요

계약서에 들어갈 **핵심어 추출**을 위해 명사 추출 사용  
코모란이 인식하지 못하는 NNG는 **사용자 사전**으로 보완  
**한 문장에 여러 개의 라벨이 포함되는 경우**를 백업 가능

## 2. 2차 분류 수행 sorting

그럼 8월 5일 어떠세요? 사서 얼마 타지도 않은 자전거예요ㅜㅜ  
저는 서울역이 가까우니까 거기서 뵙죠 68만원에 해요

**라벨 당 최고 확률**을 가지는 **핵심 단어**를 하나씩 선정함  
가격은 정규표현식 (코모란이 숫자 추출이 불가능하기 때문)  
최종 결과 보안을 위해 final\_check 함수로 보완

# 모델 테스트 1



안녕하세요

네 안녕하세요

물건 살려는 분 맞으시죠

네네 자전거 캐논데일 사고싶어용  
서울역 근방에서 보시죠

좋아요

네

얼마죠?

네6만원입니다

다음주 수요일에 거래 어떠세요?  
좋아요 그때 봐요

## 1차분류

날짜 정보 : 다음주 수요일에 거래 어떠세요?  
품목 정보 : 네네 자전거 캐논데일 사고싶어용  
위치 정보 : 서울역 근방에서 보시죠  
가격 정보 : 네6만원입니다

## 2차분류

날짜 정보 : 다음주  
품목 정보 : 자전거  
위치 정보 : 서울역  
가격 정보 : 6만원

# 모델 테스트 1



안녕하세요

네 안녕하세요

물건 살려는 분 맞으시죠

네네 자전거 캐논데일 사고싶어용

서울역 근방에서 보시죠

좋아요

네

얼마죠?

네6만원입니다

다음주 수요일에 거래 어떠세요?

좋아요 그때 봐요

품목

위치

가격

날짜

## 1차분류

날짜 정보 : 다음주 수요일에 거래 어떠세요?  
품목 정보 : 네네 자전거 캐논데일 사고싶어용  
위치 정보 : 서울역 근방에서 보시죠  
가격 정보 : 네6만원입니다

## 2차분류

날짜 정보 : 다음주  
품목 정보 : 자전거  
위치 정보 : 서울역  
가격 정보 : 6만원

# 모델 테스트 1



안녕하세요

네 안녕하세요

물건 살려는 분 맞으시죠

네네 자전거 캐논데일 사고싶어용

서울역 근방에서 보시죠

좋아요

네

얼마죠?

네 6만원입니다

다음주 수요일에 거래 어떠세요?

좋아요 그때 봐요

품목

위치

가격

날짜

## 1차분류

날짜 정보 : 다음주 수요일에 거래 어떠세요?  
품목 정보 : 네네 자전거 캐논데일 사고싶어용  
위치 정보 : 서울역 근방에서 보시죠  
가격 정보 : 네 6만원입니다

## 2차분류

날짜 정보 : 다음주  
품목 정보 : 자전거  
위치 정보 : 서울역  
가격 정보 : 6만원

# 모델 테스트 1



안녕하세요

네 안녕하세요

물건 살려는 분 맞으시죠

네네 자전거 캐논데일 사고싶어용

서울역 근방에서 보시죠

좋아요

네

얼마죠?

네 6만원입니다

다음주 수요일에 거래 어떠세요?

좋아요 그때 봐요

품목

위치

가격

날짜

## 1차분류

날짜 정보 : 다음주 수요일에 거래 어떠세요?  
품목 정보 : 네네 자전거 캐논데일 사고싶어용  
위치 정보 : 서울역 근방에서 보시죠  
가격 정보 : 네 6만원입니다

## 2차분류

날짜 정보 : 다음주 ✓  
품목 정보 : 자전거 ✓  
위치 정보 : 서울역 ✓  
가격 정보 : 6만원 ✓

# 모델 테스트 2



S10 폰케이스 8천원에 올리셨던데  
아직 파나요??

넵넵 어떤 디자인으로 원하세요??  
뒤에 고양이 그려진 케이스요!

아 그거는 만원이에요  
오늘 직거래해서 8000원에  
안되나요??

네 그럼 오늘 9시에  
구청앞에서 봅시다!!

## 1차분류

날짜 정보 :  
품목 정보 : S10 폰케이스 8천원에 올리셨던데 아직 파나요??  
위치 정보 : 네 그럼 오늘 9시에 구청앞에서 봅시다!!  
가격 정보 : 오늘 직거래해서 8000원에 안되나요??

## 2차분류

날짜 정보 : 오늘  
품목 정보 : 폰케이스  
위치 정보 : 구청  
가격 정보 : 8000원

# 모델 테스트 2



S10 폰케이스 8천원에 올리셨던데  
아직 파나요??

넵넵 어떤 디자인으로 원하세요??  
뒤에 고양이 그려진 케이스요!

아 그거는 만원이에요

오늘 직거래해서 8000원에  
안되나요??

네 그럼 오늘 9시에  
구청앞에서 봅시다!!

품목

가격

날짜  
위치

## 1차분류

날짜 정보 :  
품목 정보 : S10 폰케이스 8천원에 올리셨던데 아직 파나요??  
위치 정보 : 네 그럼 오늘 9시에 구청앞에서 봅시다!!  
가격 정보 : 오늘 직거래해서 8000원에 안되나요??

## 2차분류

날짜 정보 : 오늘  
품목 정보 : 폰케이스  
위치 정보 : 구청  
가격 정보 : 8000원



# 모델 테스트 2



S10 폰케이스 8천원에 올리셨던데  
아직 파나요??

넵넵 어떤 디자인으로 원하세요??  
뒤에 고양이 그려진 케이스요!

아 그거는 만원이에요

오늘 직거래해서 8000원에  
안되나요??

네 그럼 오늘 9시에  
구청앞에서 봅시다!!

품목

가격

날짜  
위치

## 1차분류

날짜 정보 :  
품목 정보 : S10 폰케이스 8천원에 올리셨던데 아직 파나요??  
위치 정보 : 네 그럼 오늘 9시에 구청앞에서 봅시다!!  
가격 정보 : 오늘 직거래해서 8000원에 안되나요??

## 2차분류

날짜 정보 : 오늘  
품목 정보 : 폰케이스  
위치 정보 : 구청  
가격 정보 : 8000원

# 모델 테스트 2



S10 폰케이스 8천원에 올리셨던데  
아직 파나요??

넵넵 어떤 디자인으로 원하세요??  
뒤에 고양이 그려진 케이스요!

아 그거는 만원이에요

오늘 직거래해서 8000원에  
안되나요??

네 그럼 오늘 9시에  
구청앞에서 봅시다!!

품목

가격

날짜  
위치

## 1차분류

날짜 정보 :  
품목 정보 : S10 폰케이스 8천원에 올리셨던데 아직 파나요??  
위치 정보 : 네 그럼 오늘 9시에 구청앞에서 봅시다!!  
가격 정보 : 오늘 직거래해서 8000원에 안되나요??

## 2차분류

날짜 정보 :	오늘	●
품목 정보 :	폰케이스	✓
위치 정보 :	구청	✓
가격 정보 :	8000원	✓

# 모델 테스트 3



안녕하세요

자전거 사고싶습니다

평택역앞버스정류장에서요

50만원에거래하고 싶습니다

10월 1일에봐어요.

## 1차분류

날짜 정보 : 10월 1일에봐어요.  
품목 정보 : 자전거 사고싶습니다  
위치 정보 : 평택역앞버스정류장에서요  
가격 정보 : 50만원에거래하고 싶습니다

## 2차분류

날짜 정보 : 10월 1일  
품목 정보 : 자전거  
위치 정보 : 평택역  
가격 정보 : 50만원

# 모델 테스트 3



안녕하세요

자전거 사고싶습니다

품목

평택역앞버스정류장에서요

위치

50만원에거래하고 싶습니다

가격

10월 1일에보여요.

날짜

## 1차분류

날짜 정보 : 10월 1일에보여요.  
품목 정보 : 자전거 사고싶습니다  
위치 정보 : 평택역앞버스정류장에서요  
가격 정보 : 50만원에거래하고 싶습니다

## 2차분류

날짜 정보 : 10월 1일  
품목 정보 : 자전거  
위치 정보 : 평택역  
가격 정보 : 50만원

# 모델 테스트 3



안녕하세요

자전거 사고싶습니다

품목

평택역앞버스정류장에서요

위치

50만원에거래하고 싶습니다

가격

10월 1일에보여요.

날짜

## 1차분류

날짜 정보 : 10월 1일에보여요.  
품목 정보 : 자전거 사고싶습니다  
위치 정보 : 평택역앞버스정류장에서요  
가격 정보 : 50만원에거래하고 싶습니다

## 2차분류

날짜 정보 : 10월 1일  
품목 정보 : 자전거  
위치 정보 : 평택역  
가격 정보 : 50만원

# 모델 테스트 3



안녕하세요

자전거 사고싶습니다

품목

평택역앞버스정류장에서요

위치

50만원에거래하고 싶습니다

가격

10월 1일에보여요.

날짜

## 1차분류

날짜 정보 : 10월 1일에보여요.  
품목 정보 : 자전거 사고싶습니다  
위치 정보 : 평택역앞버스정류장에서요  
가격 정보 : 50만원에거래하고 싶습니다

## 2차분류

날짜 정보 : 10월 1일 ✓  
품목 정보 : 자전거 ✓  
위치 정보 : 평택역 ❶  
가격 정보 : 50만원 ✓

# 모델 테스트 4



ㅎㅎㅎ 소니 카메라 아직 파심??  
네네 아직 안팔림  
새 제품임??  
ㄴㄴ 올해 초에 사서 한 이틀 썼나..  
거의 새거임요  
음 그럼 600000원으로  
내일 바로 거래 어떨?  
60은 너무 싸고 70만으로 내일 가능  
오키오키 70만하시고  
내일 3시 롯데에서 봅시다  
네 그럼시다

## 1차분류

날짜 정보 : ㄴㄴ 올해 초에 사서 한 이틀 썼나..  
품목 정보 : ㅎㅎㅎ 소니 카메라 아직 파심??  
위치 정보 : 오키오키 70만하시고 내일 3시에 롯데에서 봅시다  
가격 정보 : 네 그럼시다

## 2차분류

날짜 정보 : 올해  
품목 정보 : 소니  
위치 정보 : 오키  
가격 정보 : 70만

# 모델 테스트 4



ㅎㅇㅎㅇ 소니 카메라 아직 파삼??

네네 아직 안팔림

새 제품임??

ㄴㄴ 올해 초에 사서 한 이틀 썼나..

거의 새거임요

음 그럼 600000원으로

내일 바로 거래 어떨?

60은 너무 싸고 70만으로 내일 가능

오키오키 70만하시고

내일 3시 롯데에서 봅시다

네 그럼시다

품목

위치  
가격  
날짜

## 1차분류

날짜 정보 : ㄴㄴ 올해 초에 사서 한 이틀 썼나..

품목 정보 : ㅎㅇㅎㅇ 소니 카메라 아직 파삼??

위치 정보 : 오키오키 70만하시고 내일 3시에 롯데에서 봅시다

가격 정보 : 네 그럼시다

## 2차분류

날짜 정보 : 올해

품목 정보 : 소니

위치 정보 : 오키

가격 정보 : 70만



# 모델 테스트 4



ㅎㅇㅎㅇ 소니 카메라 아직 파삼??

네네 아직 안팔림

새 제품임??

ㄴㄴ 올해 초에 사서 한 이틀 썼나..

거의 새거임요

음 그럼 600000원으로

내일 바로 거래 어떨?

60은 너무 싸고 70만으로 내일 가능

오키오키 70만하시고

내일 3시 롯데에서 봅시다

네 그럼시다

품목

위치  
가격  
날짜

## 1차분류

날짜 정보 : ㄴㄴ 올해 초에 사서 한 이틀 썼나..

품목 정보 : ㅎㅇㅎㅇ 소니 카메라 아직 파삼??

위치 정보 : 오키오키 70만하시고 내일 3시에 롯데에서 봅시다

가격 정보 : 네 그럼시다

## 2차분류

날짜 정보 : 올해

품목 정보 : 소니

위치 정보 : 오키

가격 정보 : 70만

# 모델 테스트 4



ㅎㅇㅎㅇ 소니 카메라 아직 파삼??

네네 아직 안팔림

새 제품임??

ㄴㄴ 올해 초에 사서 한 이틀 썼나..

거의 새거임요

음 그럼 600000원으로

내일 바로 거래 어떨?

60은 너무 싸고 70만으로 내일 가능

오키오키 70만하시고

내일 3시 롯데에서 봅시다

네 그럼시다

품목

위치  
가격  
날짜

## 1차분류

날짜 정보 : ㄴㄴ 올해 초에 사서 한 이틀 썼나..

품목 정보 : ㅎㅇㅎㅇ 소니 카메라 아직 파삼??

위치 정보 : 오키오키 70만하시고 내일 3시에 롯데에서 봅시다

가격 정보 : 네 그럼시다

## 2차분류

날짜 정보 : 올해 ✕

품목 정보 : 소니 〇

위치 정보 : 오키 ✕

가격 정보 : 70만 ✓

# Check Point

1. 외래어 및 줄임말
2. 상대적 시간
3. 어절 단위
4. 문맥 인식

# Check Point

## 1. 외래어 및 줄임말

e.g., 아메리카노, 갤럭시, 롯데

트레인 데이터셋 추가, 사용자 사전 업그레이드

## 2. 상대적 시간

## 3. 어절 단위

## 4. 문맥 인식

# Check Point

## 1. 외래어 및 줄임말

e.g., 아메리카노, 갤럭시, 롯데  
트레인 데이터셋 추가, 사용자 사전 업그레이드

## 2. 상대적 시간

e.g., 이틀 뒤, 모레, 다음주  
절대적 시간으로 변경할 수 있는 방법

## 3. 어절 단위

## 4. 문맥 인식

# Check Point

## 1. 외래어 및 줄임말

e.g., 아메리카노, 갤럭시, 롯데  
트레인 데이터셋 추가, 사용자 사전 업그레이드

## 2. 상대적 시간

e.g., 이틀 뒤, 모레, 다음주  
절대적 시간으로 변경할 수 있는 방법

## 3. 어절 단위

e.g., 갤럭시 s10, 강남역 8번 출구  
단위 조합 중 확률이 높은 단어 채택

## 4. 문맥 인식

# Check Point

1. 외래어 및 줄임말

2. 상대적 시간

3. 어절 단위

4. 문맥 인식

```
def item_word_maker(word_list):
    if len(word_list) == 1:
        return word_list[0][0]

    word_list = sorted(word_list, key=lambda x:x[1], reverse = True)

    candidates = [word_list[0][0], word_list[0][0] + " " + word_list[1][0], word_list[1][0] + " " + word_list[0][0]]
    candidate_and_value = []
    for candidate in candidates:
        test_eval, out = predict2(candidate)
        max_val, max_index = torch.max(out, 1)

        candidate_and_value.append([candidate, max_val])
    candidate_and_value = sorted(candidate_and_value, key=lambda x:x[1], reverse = True)
    print(candidate_and_value)

    return candidate_and_value[0][0]
```

e.g., 갤럭시 s10, 강남역 8번 출구  
단위 조합 중 확률이 높은 단어 채택

# Check Point

```
def item_word_maker(word_list):
    if len(word_list) == 1:
        return word_list[0][0]

    word_list = sorted(word_list, key=lambda x:x[1], reverse = True)

    candidates = [word_list[0][0], word_list[0][0] + " " + word_list[1][0], word_list[1][0] + " " + word_list[0][0]]
    candidate_and_value = []
    for candidate in candidates:
        test_eval, out = predict2(candidate)
        max_val = torch.max(out, 1)

        candidate_and_value.append([candidate, max_val])
    candidate_and_value = sorted(candidate_and_value, key=lambda x:x[1], reverse = True)
    return candidate_and_value[0][0]
```

```
word_list = [['중고', -0.8207], ['버즈', 3.2337], ['갤럭시', 3.3715]]
```

```
print("\n\n선택된 최종 단어 : " + item_word_maker(word_list))
```

>>품목 라벨 단어 리스트<<

['갤럭시', 3.3715]

['버즈', 3.2337]

['중고', -0.8207]

>>후보 조합 단어 리스트<<

['갤럭시 버즈', tensor([3.5306], device='cuda:0', grad\_fn=<MaxBackward0>)]

['갤럭시', tensor([3.3715], device='cuda:0', grad\_fn=<MaxBackward0>)]

['버즈 갤럭시', tensor([3.3211], device='cuda:0', grad\_fn=<MaxBackward0>)]

선택된 최종 단어 : 갤럭시 버즈

e.g., 갤럭시 s10, 강남역 8번 출구

단위 조합 중 확률이 높은 단어 채택



# Check Point

## 1. 외래어 및 줄임말

e.g., 아메리카노, 갤럭시, 롯데  
트레인 데이터셋 추가, 사용자 사전 업그레이드

## 2. 상대적 시간

e.g., 이틀 뒤, 모레, 다음주  
절대적 시간으로 변경할 수 있는 방법

## 3. 어절 단위

e.g., 갤럭시 s10, 강남역 8번 출구  
단위 조합 중 확률이 높은 단어 채택

## 4. 문맥 인식

# Check Point

## 1. 외래어 및 줄임말

e.g., 아메리카노, 갤럭시, 롯데  
트레인 데이터셋 추가, 사용자 사전 업그레이드

## 2. 상대적 시간

e.g., 이틀 뒤, 모레, 다음주  
절대적 시간으로 변경할 수 있는 방법

## 3. 어절 단위

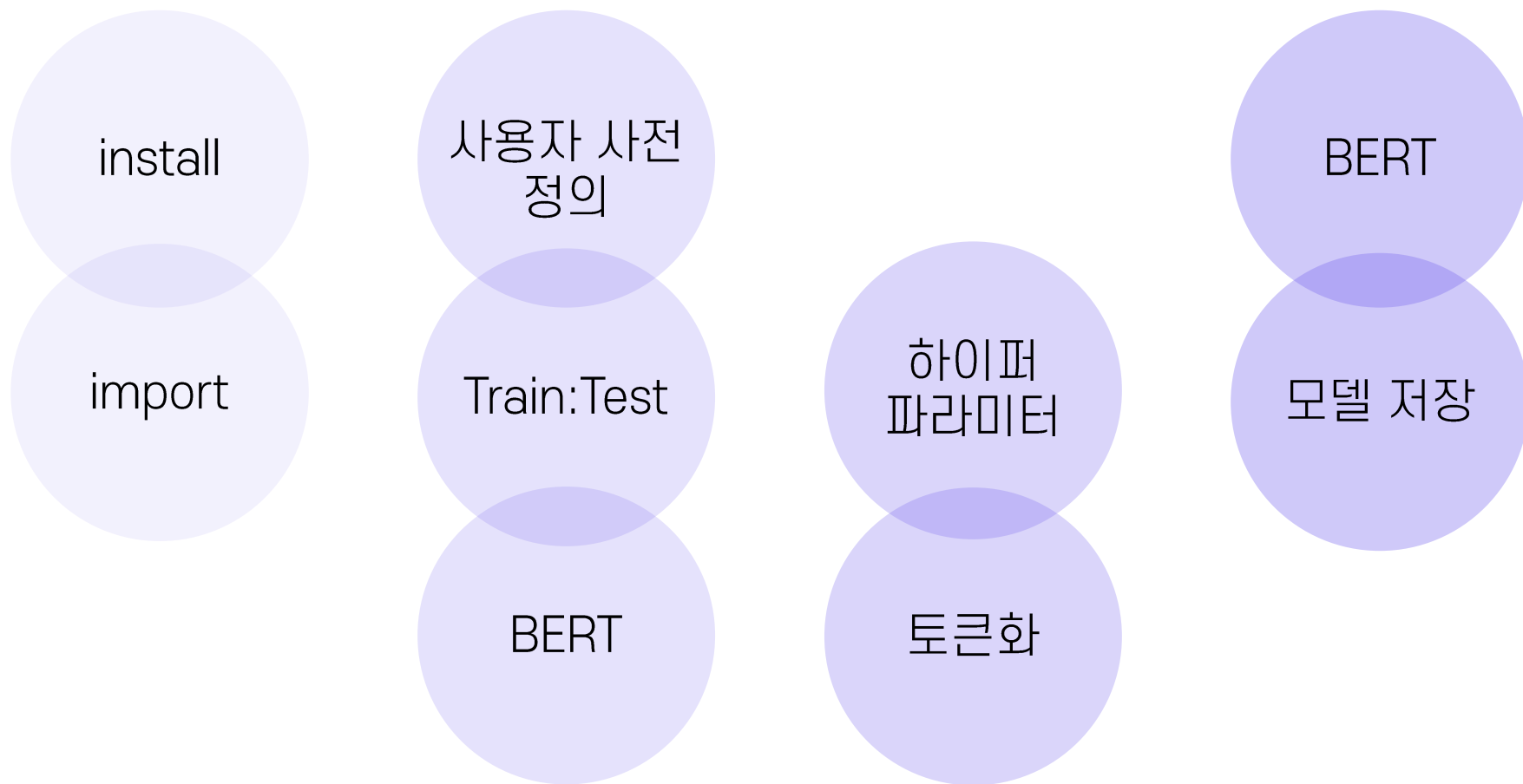
e.g., 갤럭시 s10, 강남역 8번 출구  
단위 조합 중 확률이 높은 단어 채택

## 4. 문맥 인식

e.g., 그럼 이전에 말했던 가격으로 하죠  
What is '이전'?

# 전체 코드 1/3

필요한 모듈 импорт부터 KoBERT 학습 모델 저장까지



```
[ ] 1 # 드라이브 마운트
    2 from google.colab import drive
    3 drive.mount('/content/drive')
```

```
▶ 1 # requirements 다운로드
   2 !pip install mxnet
   3 !pip install gluonnlp pandas tqdm
   4 !pip install sentencepiece
   5 !pip install transformers==3.0.2
   6 !pip install torch
   7 !pip install git+https://git@github.com/SKTBrain/KoBERT.git@master
   8 !pip install konlpy
   9 !pip install PyKomoran
  10 !pip install kss
```

install

```
▶ 1 import pandas as pd
   2 from sklearn.model_selection import train_test_split
   3 import torch
   4 from torch import nn
   5 import torch.nn.functional as F
   6 import torch.optim as optim
   7 from torch.utils.data import Dataset, DataLoader
   8 import gluonnlp as nlp
   9 import numpy as np
  10 from tqdm import tqdm, tqdm_notebook
  11 from kobert.utils import get_tokenizer
  12 from kobert.pytorch_kobert import get_pytorch_kobert_model
  13 from transformers import AdamW
  14 from transformers.optimization import get_cosine_schedule_with_warmup
  15 import re
  16 from konlpy.tag import Komoran
  17 import kss
```

import



```
[ ] 1 # 하이퍼 파라미터
2 max_len = 64
3 batch_size = 64
4 warmup_ratio = 0.1
5 num_epochs = 10
6 max_grad_norm = 1
7 log_interval = 200
8 learning_rate = 5e-5
```

하이퍼  
파라미터

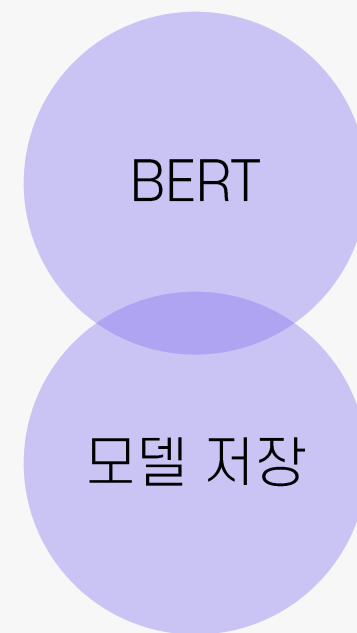
```
[ ] 1 # 토큰화
2 tokenizer = get_tokenizer()
3 tok = nlp.data.BERTSPTokenizer(tokenizer, vocab, lower=False)
4 data_train = BERTDataset(dataset_train, 0, 1, tok, max_len, True, False)
5 data_test = BERTDataset(dataset_test, 0, 1, tok, max_len, True, False)
```

토큰화

```

[ ] 1 class BERTClassifier(nn.Module):
2     def __init__(self,
3         bert,
4         hidden_size = 768,
5         num_classes=4,
6         dr_rate=None,
7         params=None):
8         super(BERTClassifier, self).__init__()
9         self.bert = bert
10        self.dr_rate = dr_rate
11
12        self.classifier = nn.Linear(hidden_size , num_classes)
13        if dr_rate:
14            self.dropout = nn.Dropout(p=dr_rate)
15
16    def gen_attention_mask(self, token_ids, valid_length):
17        attention_mask = torch.zeros_like(token_ids)
18        for i, v in enumerate(valid_length):
19            attention_mask[i][:v] = 1
20        return attention_mask.float()
21
22    def forward(self, token_ids, valid_length, segment_ids):
23        attention_mask = self.gen_attention_mask(token_ids, valid_length)
24
25        _, pooler = self.bert(input_ids = token_ids, token_type_ids = segment_ids.long(), attention_mask = attention_mask.float().to(token_ids.device))
26        if self.dr_rate:
27            out = self.dropout(pooler)
28        return self.classifier(out)

```



```

[ ] 1 PATH = "/content/drive/MyDrive/Colab Notebooks/kobertModel_v0728.pt"
2     model = torch.load(PATH, map_location=torch.device('cpu'))
3     model.eval()

```

# 전체 코드 2/3

1차 분류 모델 생성



KSS

문장  
분류



```
[ ] 1 #문장분리를 위한 kss
2 def KSS(sent):
3     kss_sent = []
4     for i in sent:
5         j = kss.split_sentences(i)
6         kss_sent.extend(j)
7     print('원본 텍스트:', sent)
8     print('분리 텍스트:', kss_sent)
9     return kss_sent
```

KSS

```
[ ] 1 def predict2(sent):
2     data = [sent, '0']
3     dataset_another = [data]
4     another_test = BERTDataset(dataset_another, 0, 1, tok, max_len, True, False)
5     test_dataloader = torch.utils.data.DataLoader(another_test, batch_size=batch_size, num_workers=5)
6     model.eval()
7     for batch_id, (token_ids, valid_length, segment_ids, label) in enumerate(test_dataloader):
8         token_ids = token_ids.long().to(device)
9         segment_ids = segment_ids.long().to(device)
10        valid_length= valid_length
11        label = label.long().to(device)
12        out = model(token_ids, valid_length, segment_ids)
13        test_eval=[]
14        for i in out:
15            logits = i
16            logits = logits.detach().cpu().numpy()
17            if np.argmax(logits) == 0:
18                test_eval.append("날짜")
19            elif np.argmax(logits) == 1:
20                test_eval.append("위치")
21            elif np.argmax(logits) == 2:
22                test_eval.append("품목")
23            elif np.argmax(logits) == 3:
24                test_eval.append("가격")
25        return test_eval[0], out
```

문장  
분류

# 전체 코드 3/3

2차 분류 모델 생성

komoran

명사  
리스트

final  
check

단어  
분류

```
[ ] 1 class ClassifierModule():
2     def __init__(self):
3         self.text = ""
4
5         self.date_sent = ""
6         self.item_sent = ""
7         self.place_sent = ""
8         self.price_sent = ""
9
10        self.words = []
11
12        self.date_word = ""
13        self.item_word = ""
14        self.place_word = ""
15        self.price_word = ""
16
17    def sentence_to_words(self): # 선정된 문장에 대해 명사만 추출하여 리스트 생성
18        self.words = komoran.nouns(self.date_sent) + komoran.nouns(self.item_sent) + komoran.nouns(self.place_sent)
19        #print(self.words)
20
21    def sorting(self, text): # 전체 문장 리스트에 대해 정확도가 가장 높은 문장 분류
22        date_max = -1
23        item_max = -1
24        place_max = -1
25        price_max = -1
26
```



komoran

```

27 for sent in text:
28     test_eval, out = predict2(sent)
29     max_val, max_index = torch.max(out, 1)
30     #print(sent)
31     #print(max_val)
32     #print("\n")
33
34     if test_eval == '날짜':
35         #result = ClassifierModule.date_regex.search(sent)
36         #if result:
37             #self.date_word = result.group(0)
38         if date_max < max_val:
39             self.date_sent = sent
40             date_max = max_val
41     elif test_eval=='품목':
42         if item_max < max_val:
43             item_max = max_val
44             self.item_sent = sent
45     elif test_eval == '위치':
46         if place_max < max_val:
47             place_max = max_val
48             self.place_sent = sent
49     elif test_eval == '가격':
50         if price_max < max_val:
51             money_regex = re.compile("#d+(원 |만원 |마년 )|#d+만 |만원 |#d.#d(만원 |만)?")
52             result = money_regex.search(sent)
53             if result:
54                 self.price_sent = result.group(0)
55 print(' ')
56 print(
57     '* * * 1차 분류 결과 * * *',
58     '\n날짜 정보 : ', self.date_sent,
59     '\n품목 정보 : ', self.item_sent,
60     '\n위치 정보 : ', self.place_sent,
61     '\n가격 정보 : ', self.price_sent)

```

명사  
리스트

```
63
64 def final_check(self):
65     if self.price_word == "":
66         money_regex = re.compile("#d+(원 |만원 |마년 )|#d+만 |만원 |#d.#d(만원 |만)?")
67         for sent in self.text:
68             result = money_regex.search(sent)
69             if result:
70                 self.price_word = result.group(0)
71
72 def words_sorting(self, text):      # 명사 리스트 중에서 가장 정확도가 높은 명사 분류
73     self.text = text
74     self.sorting(text)
75     self.sentence_to_words()
76     date_max = -1
77     item_max = -1
78     place_max = -1
79     price_max = -1
80     for word in self.words:
81         test_eval, out = predict2(word)
82         max_val, max_index = torch.max(out, 1)
83         #print(word)
84         #print(max_val)
85         #print("#n")
```

final  
check

```
87     if test_eval == '날짜':
88         if self.date_word == "" and date_max < max_val:
89             self.date_word = word
90             date_max = max_val
91     elif test_eval == '품목':
92         if item_max < max_val:
93             self.item_word = word
94             item_max = max_val
95     elif test_eval == '위치':
96         if place_max < max_val:
97             self.place_word = word
98             place_max = max_val
99
100     self.price_word = self.price_sent
101
102     self.final_check()
103
104     print(' ')
105     print(
106         '* * * 2차 분류 결과 * * *',
107         '\n날짜 정보 : ', self.date_word,
108         '\n품목 정보 : ', self.item_word,
109         '\n위치 정보 : ', self.place_word,
110         '\n가격 정보 : ', self.price_word)
```



단어  
분류