CREATED BY

신조어 번역 챗봇

Group DMZ

- TEAM PROJECT OF PRESENTATION

KDT PROJECT OF PRESENTATION

Contents

Chat Bot
NEW WORD

01. 서비스 개요

1. 서비스 소개
2. 제작 배경 및 필요성
3. 데이터 수집 및 분석 과정
4. 웹 스크래핑 과정

서비스 소개

- ## 서비스 대상

  모든 세대의 사람들을 대상
  언어적으로 다양한 세대 간의 소통을 원활하게 돕는 유용한 도구

- ## 다른 서비스와의 차별성

  기존 자동 번역 서비스와 차별되어 신조어와 줄임 말에 특화된 해석을 제공
  다양한 지역과 문화에서 사용되는 신조어를 포함하여 빠르고 정확한 해석을 통해 세밀한 언어 이해를 제공

- ## 핵심 기능

  어려운 신조어와 줄임 말을 쉽게 이해하고 해석해주는 기능
  언어 학습을 강화 및 새로운 언어적 트렌드 습득

제작배경 및 필요성

세대 간 불통의 원인, '신조어'

신조어 발달

SNS와 디지털 커뮤니케이션으로 신조어가 쉽게 만들어지고 퍼져나가는 경향

세대 및 그룹 특정

신조어는 특정 세대나 그룹에서 주로 사용되어 커뮤니케이션 장벽 형성 가능

업데이트 어려움

신조어는 자주 변하고 업데이트되어 따라가기 어려운 특성.

공통된 세대 문화 마련 필요성 대두

신조어 번역기

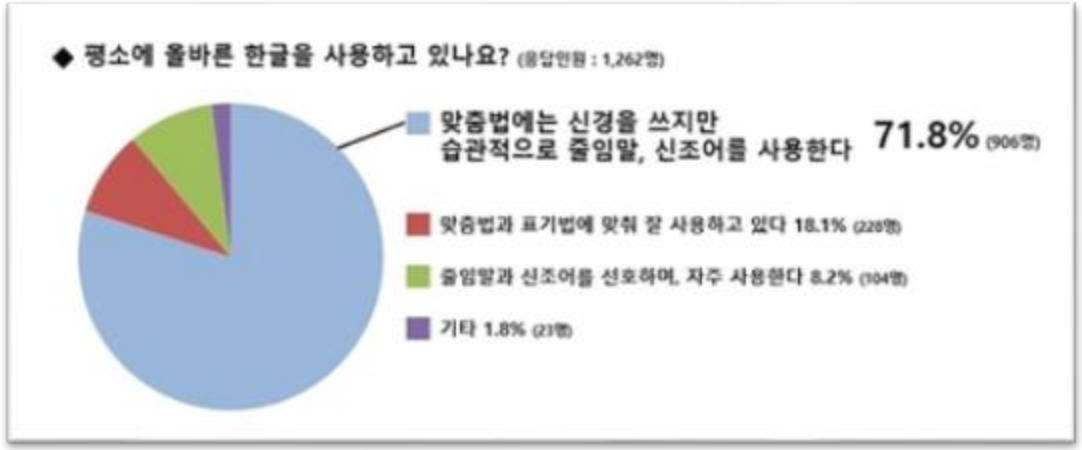
다양한 세대간 소통을 도와주고, 신조어에 대한 이해를 쉽게 도와주는 서비스

기술 익숙하지 않은 사용자

고령자 및 소셜 미디어에 미숙한 사람들이 신조어 이해에 도움

기업 및 광고 업계 활용

고객과의 효과적인 커뮤니케이션을 위해 이 서비스 활용 가능하며, 원활한 소통 향상 기대됨



데이터 수집 및 분석 과정

데이터 수집 (Web Site)

나무위키

blog

WIKIPEDIA
The Free Encyclopedia

데이터 수집 과정 (Web Scraping)

타겟 웹사이트 선별

원하는 데이터를 얻기 위해 분석할 대상 웹사이트를 선별
블로그, 소셜 미디어, 포럼 등의 웹사이트를 선택

Web Scraping 도구 선택

Beautiful Soup, Requests, Selenium 등의 Web Scraping 도구를 선택

웹 페이지 접속 및 데이터 추출

웹 페이지의 HTML 구조를 파악하고 필요한 정보를 추출
원하는 데이터가 어떤 태그와 속성에 있는지 확인

데이터 저장

추출한 데이터를 원하는 형식으로 가공하여 저장
CSV 파일에 저장 후 Pandas 라이브러리를 이용하여 Data Frame 생성

Web Scraping 과정

```python
import requests
from bs4 import BeautifulSoup

URL = 'https://clotheview.tistory.com/200'
response = requests.get(URL)

content_text = ""

if response.status_code == 200:
    soup = BeautifulSoup(response.text, 'html.parser')

    # Tistory의 블로그 글 본문의 태그와 클래스를 이용하여 텍스트 추출
    content_div = soup.find('div', class_='tt_article_useless_p_margin')

    # 각 단락을 개별적으로 저장
    for paragraph in content_div.find_all('p'):
        content_text += paragraph.get_text() + '₩n₩n'  # 단락 사이에 줄바꿈 추가

    # 텍스트 파일로 저장
    with open('신조어 블로그_content.txt', 'w', encoding='utf-8') as file:
        file.write(content_text)
    print("Content saved to 'blog_content.txt'")
else:
    print("Failed to retrieve the webpage.")
```

	mean	similar
뉴비	어떤 직업에 대한 무경험자를 지칭	초보자
Newb	어떤 직업에 대한 무경험자를 지칭	초보자
딸배	배달사원을 비하하는 말	배달원
믿거	믿고 거르는의 줄임말	의심의 여지 없이 걸러야 할, 믿고 거르는, 믿고 걸러야
인조새	인생 조진 새끼의 줄임말	힘든 일을 겪고 있는 사람, 힘든 일을 겪고 있는 사람이
낄끼빠빠	낄 때 끼고 빠질 때 빠져라	낄 때 끼고 빠질 때 빠진다, 낄 때 끼고 빠질 때 빠짐
답정너	답은 정해져 있고 너는 대답만 해라	답을 정해놓은 상태에서 대답만 해라, 답을 정해놓은 상태
비담	비주얼 담당	비주얼 담당
취존	취향 존중	취향존중
핑프	핑거 프린세스 시키기만 하는 사람	시키기만 하는 사람이, 시키기만 하는 사람
케바케	케이스 바이 케이스,Case by case	상황에 따라 다름
솔까말	솔직히 까놓고 말해서	솔직히 말해서
자낳괴	자본주의가 낳은 괴물	자본주의가 낳은 괴물, 자본주의가 낳은 괴물이
넌씨눈	넌 시X 눈치도 없냐	넌 눈치도 없다, 넌 눈치도 없음
자만추	자연스러운 만남 추구	자연스러운 만남을 추구
정줄놓	정신줄을 놓다	정신을 놓다, 정신을 놓은 사람
정뚝떨	정이 뚝 떨어진다	정이 뚝 떨어진다, 정이 뚝 떨어지는 사람
애빼시	애교 빼면 시체	애교를 빼면 시체
좋댓구알	좋아요 댓 글 구독 알림설정	좋아요와 댓글 구독 알림 설정
개이득	정말로라는 뜻으로도 쓰이는 접두사 개-와 이득이 함께 쓰인 파생어이다. 줄여서 ㄱㅇㄷ이다	정말로 유익하다, 정말로 유익, 정말로 유익함
주린이	주식 처음 해보는 사람	주식 초보자
복돌이	불법 복제품을 사용하는 사람을 일컫는 말	복제품 사용자
덕통사고	덕후' + '교통사고' 교통사고처럼 갑작스럽게 누군가나 무언가의 덕후가 되었음을 뜻하는 표	덕후가 갑작스럽게 만들어지는 것, 갑자기 덕후가 됨을
득템	획득'의 '得' + 'Item 얻는다는 뜻의 한자인 得(득)과 영어 아이템(item)의 합성어이다.	아이템을 획득하는 것, 아이템을 획득
발컨	발'+'control: 발로 하는 것처럼 컨트롤을 못하는 것을 의미	컨트롤하지 못하는 것, 컨트롤 못
어쩔티비	어쩌라고+티비	어쩌라고
웃프네	웃다'의 사동사 '웃기다' + 형용사 '슬프다	웃기고 슬프네
웃프다	웃다'의 사동사 '웃기다' + 형용사 '슬프다	웃기고 슬프다
역대급	(역)사에 남을만한 (대)단히 높은 (급)이라는 뜻	역사적인 수준의 높음
자존감	자존심'의 '自尊' + 감정을 나타내는 접미어	자아 존중감
미러링	컴퓨터 기술 용어지만 적대 진영에 대해 똑같은 방법으로 응징한다는 뜻	반대로 응징하는 것

Chat Bot
NEW WORD

02. 개발 목표 및 내용

1. 개발 목표
2. 개발 일정
3. 사용 기술

개발 목표

신조어를 이해하고
일반적 단어 또는 표현
대체 및 번역 기능

이해하기 쉽도록 신조어의
적절한 단어 예시 제공

신조어 업데이트를 주기적으로 반영
변화에 능동적 대응

신조어 번역을 통해
세대간 언어적 격차를 해소하는 역할

DMZ 개발 일정

DMZ SCHEDULE 2023년 7월

SUN	MON	TUE	WED	THR	FRI	SAT
9	10	11	12	13	14	15
	챗봇 서비스 기획			웹 스크래핑 및 DB 구축		
16	17	18	19	20	21	22
웹 스크래핑 및 DB 구축						
23	24	25	26	27	28	29
모델 구성 및 웹페이지 구성						
31	1	2	3	4	5	6
				수정 및 보완		
Front Back Connect 및 DB 수정						
7	8	9	10			
수정 및 보완			PT			
Front Back Connect 및 DB 수정						

사용 기술

언어모델(Language Model)

1. BERT 모델
2. GPT 모델

전이 학습(Transfer learning, TL)

1. 사전 훈련된 모델

자연어 처리(natural language processing)

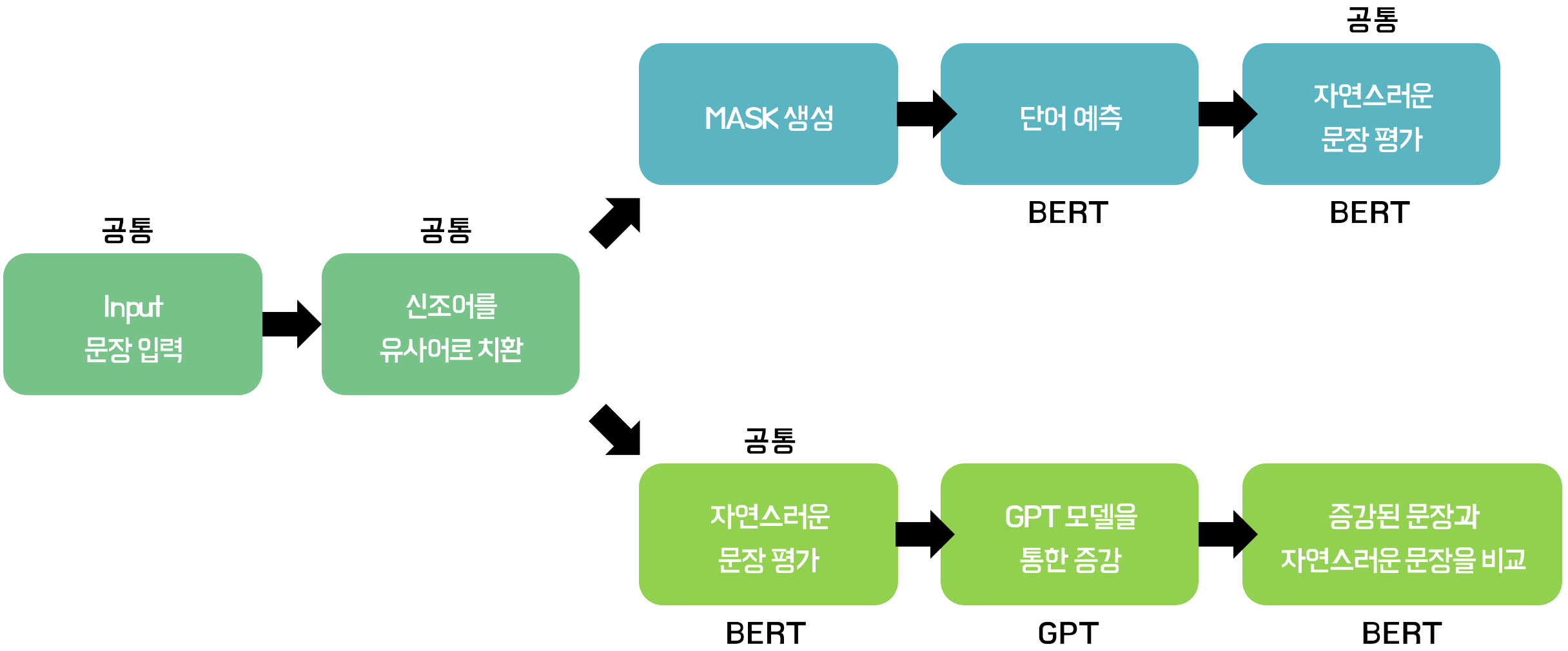
1. 개체명 인식(Named Entity Recognition, NER)
2. 품사 태깅 (Part-of-Speech Tagging, POS)

Chat Bot
NEW WORD

03. Logic Flow

1. Logic 및 코드 설명
2. BERT 모델의 한계와 GPT 모델 사용
3. 서비스 활용 기대효과

Logic


공통
Input
문장 입력
공통
신조어를
유사어로 치환
MASK 생성
단어 예측
BERT
공통
자연스러운
문장 평가
BERT
공통
자연스러운
문장 평가
BERT
GPT 모델을
통한 증강
GPT
증강된 문장과
자연스러운 문장을 비교
BERT

공통

Input

문장 입력

input Sentence

너 자낳괴지?

공통

신조어를
유사어로 치환

input Sentence

너 자낳괴지?

Replace Sentence

너 자본주의가
낳은 괴물지?

치환(공통)

```python
def check_word_in_sentence(sentence, word_list):
    found_words = []
    for word in word_list:
        if word in sentence:
            found_words.append(word)  # 캐치한 신조어를 리스트에 추가
    return found_words

def replace_words(sentence, word_replacements):
    sentences = [sentence]
    already_replaced = set()
    for word, replacements in word_replacements.items():
        if word in already_replaced:
            continue
        new_sentences = []
        for sent in sentences:
            if word in sent:
                replacement_words = []
                for replacement in replacements[1:]:
                    new_sentence = sent.replace(word, replacement)
                    replacement_words.append(replacement)
                    # print(replacement_words)
                    new_sentences.append(new_sentence)
                    already_replaced.add(replacement)
                    # 만약 새로운 대체어가 다시 신조어를 포함하고 있다면, 해당
                    신조어도 추가
                    for replaced_word, replaced_replacements in word_replacements.
                    items():
                        if replaced_word != word and replaced_word in replacement:
                            already_replaced.add(replaced_word)
            else:
                new_sentences.append(sent)
        sentences = new_sentences
    return sentences, replacement_words
```

신조어 검출 및 문장 내 단어 치환 함수

두 개의 함수는 주어진 문장 내에서 **신조어를 검출**하고 **해당 단어를 치환**하여 새로운 문장을 생성하는 기능

"check_word_in_sentence" 함수는 주어진 **문장과 신조어 목록을 비교**하여 문장 내에서 발견한 신조어를 리스트로 반환

"replace_words" 함수는 문장과 신조어에 대한 사전 정보를 활용하여 해당 **신조어를 유사어로 치환**하고, 여러 치환 단어가 만들어지는 경우 모든 가능한 문장을 생성

이를 통해 신조어 검출 및 치환 작업을 효율적으로 수행하여 **원하는 텍스트 변환 작업을 수행**

평가(공통)

```python
def evaluate_naturalness(replaced_sentences):
    tokenizer = BertTokenizerFast.from_pretrained('klue/bert-base')
    model = BertForSequenceClassification.from_pretrained('klue/bert-base')

    max_naturalness_score = -1
    most_natural_sentence = None

    for i in range(len(replaced_sentences)):
        for j in range(i+1, len(replaced_sentences)):
            sentence1 = replaced_sentences[i]
            print(sentence1)
            sentence2 = replaced_sentences[j]
            print(sentence2)
            inputs = tokenizer(sentence1, sentence2, add_special_tokens=True,
            return_tensors='pt', truncation=True, padding=True)
            with torch.no_grad():
                outputs = model(**inputs)
            logits = outputs.logits
            prob = torch.softmax(logits, dim=1)
            similarity_score = prob[:, 1].item()
            if similarity_score > max_naturalness_score:
                max_naturalness_score = similarity_score
                most_natural_sentence = sentence1 if similarity_score > 0.5 else
                sentence2
    return most_natural_sentence
```

BERT를 활용한 문장 자연성 평가

주어진 치환된 문장들 중에서 **문장의 자연성을 평가** 하고 가장 **자연스러운 문장을 선택**

"evaluate_naturalness" 함수는 "klue/bert-base" 사전 훈련된 BERT 모델을 사용하여 **문장 간 유사도를 측정**합니다.

주어진 치환된 문장들을 순회하며 문장 쌍의 유사도를 측정하고 가장 자연스러운 문장을 선택하여 반환

이를 통해 자연스러운 문장을 선택하는 과정을 자동화하고 변환 작업 결과의 자연성을 평가

MASK

치환된 문장에
MASK 생성

Replace Sentence

너 자본주의가
낳은 괴물지?

Replace Mask

너 자본주의가 낳은
괴물<mask>?

MASK

예측된 문장
생성

Replace Mask

너 자본주의가 낳은
괴물<mask>?

Mask Predict

Bert : 아냐 / 아니야
Xlm : 인지 / 이니

공통

KLUE/BERT
모델을 통한 평가

Predicted Sentence

1. 너 자본주의가 낳은 괴물아냐?
2. 너 자본주의가 낳은 괴물아니야?
3. 너 자본주의가 낳은 괴물이지?
4. 너 자본주의가 낳은 괴물이니?

Output

너 자본주의가 낳은
괴물아니야?

증강(MASK 모델)

```python
def replace_mask(masked_sentences, predictions):
    replaced_sentences = []

    for sentence, prediction in zip(masked_sentences, predictions):
        if '<mask>' in sentence:
            if prediction != 'None':
                # <mask>를 예측된 단어로 치환
                for key, value in prediction.items():
                    words_to_replace = value.split(' / ')
                    for word in words_to_replace:
                        replaced_sentence = sentence.replace('<mask>', word)
                        replaced_sentences.append(replaced_sentence)
            else:
                replaced_sentences.append(sentence)
        else:
            replaced_sentences.append(sentence)

    return replaced_sentences
```

```python
def mask(sentences, replacement_words):
    masked_sentences = []
    for sent in sentences:
        for word in replacement_words:
            if word in sent:
                word_idx = sent.index(word)
                try:
                    replace_mask = sent[word_idx + len(word)]
                except IndexError:
                    replace_mask = ""

                if replace_mask.strip() and replace_mask.isalnum():
                    mask = sent[:word_idx + len(word)] + '<mask>' + sent[word_idx
                    + len(word)+1:]
                    masked_sentences.append(mask)
                else:
                    masked_sentences.append(sent)

    return masked_sentences
```

```python
def predict_mask(masked_sentences):
    predictions = []

    for sentence in masked_sentences:
        if '<mask>' in sentence:
            prediction = predict(sentence)
            predictions.append(prediction)
        else:
            predictions.append('None')

    return predictions
```

신조어 마스킹과 BERT 기반 예측을 활용한 문장 변환

1. 문장 내 대체어 위치 마스킹

"mask" 함수는 주어진 문장 내에서 **단어 위치를 찾아 마스킹 처리**

대체어 다음의 알파벳이나 숫자가 아닌 문자에는 마스킹을 적용 X

2. BERT 마스킹 예측 및 단어 대체

"replace_mask" 함수는 예측된 결과를 활용하여 '<mask>'를
예측된 단어로 대체하며, 예측 결과가 None인 경우 원래 문장을 유지

"predict_mask" 함수는 마스킹된 문장들의 '<mask>' 토큰을
BERT 기반의 "predict" 함수를 사용하여 예측

이를 통해 BERT의 예측 결과를 활용하여 **문장 내 단어를 자동으로 대체**하고
변환 작업을 수행하여 **신조어 변환에 자연스러움과 정확성 제공**

예측

(BERT Multilingual, XLM-RoBERTa Base)

```python
def predict(text_sentence, top_k=10, top_clean=2):
    # 입력된 문장에 <mask> 토큰이 포함되어 있는지 확인
    if '<mask>' not in text_sentence:
        print('<mask> 를 입력해주세요.')  # <mask>가 없는 경우 경고 메시지 출력
        return
    # ======================== BERT MULTILINGUAL 예측 시작 ================================
    # BERT Multilingual 모델에 입력하기 위해 문장을 인코딩
    input_ids, mask_idx = encode(bert_multilingual_tokenizer, text_sentence,
                                 mask_token=bert_multilingual_tokenizer.mask_token,
                                 mask_token_id=bert_multilingual_tokenizer.mask_token_id)

    # 그라디언트 계산 비활성화하며 BERT Multilingual 모델로 예측 수행
    with torch.no_grad():
        predict = bert_multilingual_model(input_ids)[0]

    # 상위 예측값을 디코딩하여 문자열로 변환
    bert_multilingual = decode(bert_multilingual_tokenizer, predict[0, mask_idx, :].topk(top_k).indices.tolist(),
                               top_clean)
    # ======================== XLM ROBERTA BASE 예측 시작 ================================

    # XLM-RoBERTa Base 모델에 입력하기 위해 문장을 인코딩
    input_ids, mask_idx = encode(xlmroberta_tokenizer, text_sentence,
                                 mask_token=xlmroberta_tokenizer.mask_token,
                                 mask_token_id=xlmroberta_tokenizer.mask_token_id)
    # 그라디언트 계산 비활성화하며 XLM-RoBERTa Base 모델로 예측 수행
    with torch.no_grad():
        predict = xlmroberta_model(input_ids)[0]

    # 상위 예측값을 디코딩하여 문자열로 변환
    xlm = decode(xlmroberta_tokenizer, predict[0, mask_idx, :].topk(top_k).indices.tolist(), top_clean)

    # 두 모델의 예측 결과를 딕셔너리에 저장
    results = {'bert_multilingual': bert_multilingual, 'xlm': xlm}

    # 결과 반환
    return results
```

<mask> 토큰의 예측 값 제공

<mask> 토큰의 가능한 예측 값을 두 가지 모델을 통해 제공

BERT Multilingual, XLM-RoBERTa Base를 사용

"encode" 함수를 사용하여 문장을 BERT Multilingual, XLM-RoBERTa 모델에 맞게 **토큰화 한뒤 인코딩 진행**

MASK 토큰을 예측

"decode" 함수를 사용하여 **단어나 문자열 표현으로 반환**

인코딩(MASK 모델)

```python
def encode(tokenizer, text_sentence, add_special_tokens=True,
mask_token='[MASK]', mask_token_id=4):
    # mask_token = tokenizer.mask_token
    # mask_token_id = tokenizer.mask_token_id

    text_sentence = text_sentence.replace('<mask>', mask_token)
    # if <mask> is the last token, append a "." so that models
    dont predict punctuation.
    if mask_token == text_sentence.split()[-1]:
        text_sentence += ' .'

    input_ids = torch.tensor([tokenizer.encode(text_sentence,
    add_special_tokens=add_special_tokens)])
    mask_idx = torch.where(input_ids == mask_token_id)[1].tolist
    ()[0]
    return input_ids, mask_idx
```

텍스트 인코딩

다국어 BERT와 XLM-RoBERTα 모델을 활용하여 텍스트를 인코딩하는 "encode" 함수

텍스트 문장을 토큰화 하고, 인코딩된 토큰 ID와 마스크 토큰의 위치를 반환

'<mask>'는 주어진 마스크 토큰으로 대체되며,
만약 마스크 토큰이 문장의 끝에 위치한다면 마침표를 추가하여 모델이 문장 종료를 인식

함수를 통해 텍스트 문장을 모델에 입력 가능한 형태로 변환

디코딩(MASK 모델)

```python
def decode(tokenizer, pred_idx, top_clean):
    ignore_tokens = string.punctuation + '[PAD][UNK]<pad><unk> '
    tokens = []
    for w in pred_idx:
        token = ''.join(tokenizer.decode(w).split())
        if token not in ignore_tokens:
            tokens.append(token.replace('##', ''))
    return ' / '.join(tokens[:top_clean])
```

텍스트 디코딩

디코딩 과정에서 **불필요한 토큰을 제거**(special token)
정제된 텍스트를 반환하는 "decode" 함수

예측된 **토큰 인덱스를 활용**하여 **디코딩한 결과를 정제**

불필요한 구두점 및 특수 토큰을 제거하고, "##" 접두사를
공백으로 변환하여 단어를 복원하여 반환

최종적으로 **상위 N개의 정제된 토큰을 연결하여 반환**
하고, 의미 있는 문장 형태로 변환

모델의 예측 결과를 읽기 쉽고 의미 있는 문장으로 변환

공통

Input

문장 입력

input Sentence

바퀴벌레는 극혐이다

공통

신조어를
유사어로 치환

input Sentence

바퀴벌레는 극혐이다

Replace Sentence

1. 바퀴벌레는 극도로 혐오하다이다
2. 바퀴벌레는 극도록 혐오이다
3. 바퀴벌레는 극도로 혐오스럽다이다

공통

KLUE/BERT
모델을 통한 평가

Replace Sentence

1. 바퀴벌레는 극도로 혐오하다이다
2. 바퀴벌레는 극도록 혐오이다
3. 바퀴벌레는 극도로 혐오스럽다이다

Natural Sentence

바퀴벌레는 극도로 혐오하다이다

GPT

GPT 모델을 통한
증강

Natural Sentence

바퀴벌레는 극도로 혐오하다이다

Improve Sentence

바퀴벌레에 대한 혐오는 매우
강하다

GPT

Naturalness Sentence

vs

Improve Sentence

Natural Sentence

바퀴벌레는 극도로 혐오하다이다

Improve Sentence

바퀴벌레에 대한 혐오는 매우 강하다

Output

바퀴벌레에 대한 혐오는 매우 강하다

증강(GPT 모델)

신조어 검출 및 문장 내 단어 치환 함수

```python
def improve_sentence(sentence):
    response = openai.ChatCompletion.create(
        model="gpt-3.5-turbo",
        messages=[
            {
                "role": "user",
                "content": f"{sentence} 를 자연스러운 문장으로 바꿔줘",
            },
        ]
    )

    bot_response = response['choices'][0]['message']['content']
    return bot_response
```

주어진 **문장을 개선하여 자연스러운 문장으로 변환**

"improve_sentence" 함수는 OpenAI의 GPT-3.5 Turbo 모델을 활용하여 입력 문장을 바탕으로 사용자 역할의 메시지를 생성하여 모델 응답을 받아 **자연스러운 문장으로 개선**

API 호출을 통해 **문장을 개선**하는 작업을 수행하며 변환된 결과를 반환

사용자의 입력 문장을 더 자연스럽고 표현력 있는 형태로 개선

Bert 모델의 한계와 GPT 모델 사용 이유

01

KLUE/BERT모델 사용이유

모두의 말뭉치, CC-100-Kor, 나무위키, 뉴스, 청원 등 문서에서 추출한 63GB의 데이터 학습되어 신조어 번역 챗봇 데이터와 적합

02

BERT모델의 한계점

BERT는 주어진 텍스트의 문맥을 이해하는 능력이 뛰어난 것으로 알려져 있지만, 여전히 사람처럼 심도 있는 의미 이해나 추론능력에는 한계가 있습니다.

03

GPT모델 사용

GPT 모델은 좀 더 좋은 생성 모델 기능을 제공합니다. GPT는 문장 생성 등의 태스크에 특히 강점을 보이며, 순차적인 정보를 처리하는 데 능하여 BERT모델의 한계점을 보완할 수 있어 GPT모델을 사용했습니다

서비스 활용 기대효과

기대효과1: 세대 간 원활한 소통

신조어와 줄임 말을 사용하는 젊은 세대와 그렇지 않은 세대 간의 소통을 원활하게 돕습니다. 어려운 신조어를 이해하고 해석함으로써 대화의 장벽을 줄여주며, 더욱 원활한 소통이 가능해집니다.

기대효과2: 사회 현상 파악 및 분석

특정 신조어의 유행을 추적하고 분석함으로써 특정 사회 현상의 동향을 파악할 수 있습니다. 이를 통해 문화적인 변화와 트렌드를 이해하는 데 도움을 줄 수 있습니다.

기대효과3: 세대 간 이해 증진

신조어와 줄임 말을 이해하는 데 어려움을 겪는 사람들이 이를 해석해주는 챗봇을 통해 서로의 언어와 문화를 더 잘 이해하게 되며, 세대 간의 갈등을 줄일 수 있습니다.

기대효과4: 소통의 확장

신조어 해석 챗봇을 통해 언어적 장벽을 극복하고 더 다양한 사람들과 소통이 가능해집니다. 이는 사회와 개인 간의 관계 개선을 도모할 수 있습니다.

Chat Bot
NEW WORD

04. 서비스 시연

1. 시연
2. Trouble Shooting
3. Q&A
4. 개발 환경

시연 준비중

Trouble shooting

1. 치환부분에서 왜 학습을 시키지 않았을까

2. 신조어가 치환 됐을 때 발생하는
어색함 문제는 무엇인가?

3. 전체적인 문장의 뜻은 이어가면서 자연스럽게 바꾸는
문제를 해결하기위해 어떤 로직을 구성했는가?

개발 환경

Tool

Jupyter
Notebook

Google
COLAB

Pycharm

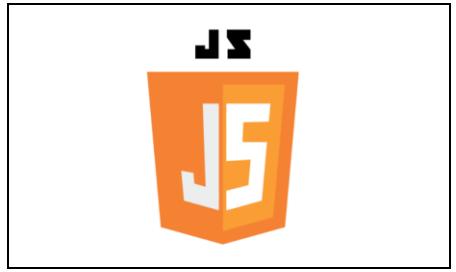
FastAPI

Language

Python

HTML

CSS



JavaScript


QnA

감사합니다