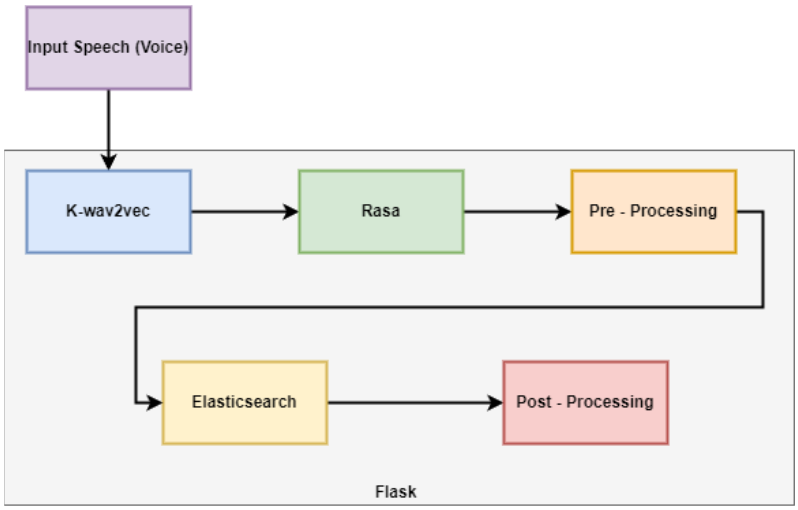
Clothes Recommendation System API Documents

Clothes Recommendation System Table of contents

★API 기능 요약



사용자에게 음성을 받아, 음성을 Flask서버에 전송 하면 STT후 해당하는 조건에 맞는 옷을 검색하여 리턴

★API 사용 설명서

해당 API는 포트번호 5050을 사용.

	/start_model
methods	POST, GET
end_point	/start_model
parameters	음성파일이 들어있는 경로 (이는 실제 음성이 들어올 경우 수정 필요)

★API 코드 설명

- **load_model.py**

> **실질적으로 돌아가는 Flask API code. 음성처리, STT처리, 검색 모든 것을 여기서 동작**

```python
import random, re
from flask import jsonify, request, Blueprint, g
from elasticsearch import Elasticsearch
from pprintpp import pprint
import torch

Load_model = Blueprint('load_model', __name__, url_prefix='/')

# STT pre-trained model load
import pyctcdecode
import librosa
import unicodedata

from pyctcdecode import build_ctcdecoder
from transformers import (
    AutoConfig,
    AutoFeatureExtractor,
    AutoModelForCTC,
    AutoTokenizer,
    Wav2Vec2ProcessorWithLM,
)
from transformers.pipelines import AutomaticSpeechRecognitionPipeline
from rasa.nlu.model import Interpreter

# 모델과 토크나이저, 예측을 위한 각 모듈들을 불러옵니다.
model = AutoModelForCTC.from_pretrained("42MARU/ko-spelling-wav2vec2-conformer-del-1s")
feature_extractor = AutoFeatureExtractor.from_pretrained("42MARU/ko-spelling-wav2vec2-conformer-del-1s")
tokenizer = AutoTokenizer.from_pretrained("42MARU/ko-spelling-wav2vec2-conformer-del-1s")
beamsearch_decoder = build_ctcdecoder(
    labels=list(tokenizer.encoder.keys()),
    kenlm_model_path=None,
)
processor = Wav2Vec2ProcessorWithLM(
    feature_extractor=feature_extractor, tokenizer=tokenizer, decoder=beamsearch_decoder
)


# STT pre-trained model
# 실제 예측을 위한 파이프라인에 정의된 모듈들을 삽입.
asr_pipeline = AutomaticSpeechRecognitionPipeline(
    model=model,
    tokenizer=processor.tokenizer,
    feature_extractor=processor.feature_extractor,
    decoder=processor.decoder,
    device=-1,
)

# rasa pre-trained mode
# 훈련시킨 rasa모델을 로드하여 intent, entity추출하기 위함
```

```python
interpreter = Interpreter.load('/home/user/rasa/rasa/models/nlu_20230213-144122')

# apply STT model
@Load_model.route('start_model', methods=['GET', 'POST'])
def start_model(audio_path=None):
    # audio파일이 아닌 text가 들어왔을 경우
    audio_path = request.args.get('audio_path', False)
#    text = request.args.get('text', False)

    if audio_path:
        # 음성파일을 불러오고 beamsearch 파라미터를 특정하여 예측을 수행합니다.
        raw_data, _ = librosa.load(audio_path, sr=16000)
        kwargs = {"decoder_kwargs": {"beam_width": 100}}
        pred = asr_pipeline(inputs=raw_data, **kwargs)["text"]
        # 모델이 자소 분리 유니코드 텍스트로 나오므로, 일반 String으로 변환해줄 필요가 있습니다.
        text = unicodedata.normalize("NFC", pred)
        result = interpreter.parse(text)

    elif text:
        result = interpreter.parse(text)

    # parameters initialization
    parameters = {}
    parameters['age'] = ''
    parameters['brand'] = ''
    parameters['color'] = ''
    parameters['gender'] = ''
    parameters['category'] = {'main_category': '', 'sub_category': ''}
    parameters['price'] = ''
    parameters['priceOperator'] = ''
    parameters['text'] = text

    # 데이터 정제
    if result['intent']['name'] == 'fashion_recommend':
        for e in result['entities']:
            if e['entity'] == 'age': # 나이
                parameters['age'] = e['value']
            elif e['entity'] == 'gender': # 성별
                parameters['gender'] = e['value']
            elif e['entity'] == 'top' or e['entity'] == 'pants' or e['entity'] == 'detail':
                parameters['category'] = {}
                if e['entity'] == 'detail': # 옷 종류
                    parameters['category']['main_category'] = 'detail'
                    parameters['category']['sub_category'] = e['value']
                else:
                    parameters['category']['main_category'] = e['value']
                    parameters['category']['sub_category'] = ''
            elif e['entity'] == 'brand': # 브랜드
                parameters['brand'] = e['value']
            elif e['entity'] == 'price': # 가격
                parameters['price'] = e['value']
            elif e['entity'] == 'priceOperator': # 가격 범위
                parameters['priceOperator'] = e['value']
            elif e['entity'] == 'color': # 색상
                parameters['color'] = e['value']

        parameters['isPopulary'] = True # 트렌드

```

```
    #pprint(parameters)

    # Elasticsearch server initialization
    es = Elasticsearch(
            'http://34.64.63.141:9200', maxsize=25, timeout=30, max_retries=10,
            retry_on_timeout=True)


    # age
    # if age is str ex) 이십대, 만원
    strToint = {
            '일': 1, '이': 2, '삼': 3, '사': 4, '오': 5, '육': 6, '칠': 7, '팔': 8, '구': 9,
            '십': 10, '백': 100, '천': 1000, '만': 10000}

    # 에러를 방지하기 위한 전처리
    if re.sub('[^0-9]', '', parameters['age']) == '':
        try:
            if parameters['age'][0] != '십' and parameters['age'][0] != '백' and parameters['age'][0] != '천' ar
                parameters['age'] = str(strToint[parameters['age'][0]] * strToint[parameters['age'][1]])
            else:
                parameters['age'] = str(strToint[parameters['age'][0]])
        except:
            parameters['age'] = '20'
    else:
        parameters['age'] = re.sub('[^0-9]', '', parameters['age'])


    age_range_list = [parameters['age'][0] + '0', parameters['age'][0] + '9']

    age_range = {
            'gte': int(age_range_list[0].strip()),
            'lte': int(age_range_list[1].strip())
            }


    age = {
            "bool": {
                "should": [
                    {
                        "range": {
                            "likesAgeRange.from": age_range
                        }
                    },
                    {
                        "range": {
                            "likesAgeRange.until": age_range
                        }
                    }
                ],
                "minimum_should_match": 1
            }
        }


    # brand
    if parameters['brand']:
        brand = {
                'match': {
```

```
                    'brandName': {
                        'query': parameters['brand']
                        }
                    }
                }
    else:
        brand = ''

    # category classification
    if parameters['category']['main_category'] != 'detail' and parameters['category']['main_category'] != '':
        if parameters['category']['main_category'] == '상의' or parameters['category']['main_category'] == '하의
            category = {
                    'match': {
                        'category.main.categoryName': {
                            'query': parameters['category']['main_category']
                            }
                        }
                    }
    elif parameters['category']['main_category'] == 'detail':
        category = {
                'match': {
                    'category.middle.categoryName': {
                        'query': parameters['category']['sub_category']
                        }
                    }
                }
    else:
        category = ''


    # color
    if parameters['color']:
        color = {
                'match': {
                    'color': {
                        'query': parameters['color']
                        }
                    }
                }

    # gender
    if parameters['gender']:
        gender = {
                'match': {
                    'gender': {
                        'query': parameters['gender']
                        }
                    }
                }
    else:
        gender = ''

    # price
    # text -> int
    parameters['price'] = re.sub('[^0-9ㄱ-ㅎ가-힣]', '', parameters['price'])
    tmp_price = 1
    for idx, s in enumerate(parameters['price']):
        if s.isdigit():
```

```
            tmp_price = int(parameters['price'][idx])
            continue
        elif s != '원':
            tmp_price = tmp_price * strToint[parameters['price'][idx]]
    parameters['price'] = tmp_price

    # price query
    if parameters['priceOperator'] == '사이':  # 사이
        sort_price = sorted(parameters['price'], key=lambda x: int(x))
        price_range = {
            'gte': int(sort_price[0].strip()),
            'lte': int(sort_price[1].strip())
        }
    elif parameters['priceOperator'] == '정도':  # 정도
        price_range = {
            'gte': int(parameters['priceOperator']) - 10000,  # gte: 이상 / gte: 초과
            'lte': int(parameters['priceOperator']) + 10000  # lte: 이하 / lt: 미만
        }
    elif parameters['priceOperator'] == '이상':  # 이상
        price_range = {
            'gte': int(parameters['priceOperator']),
        }
    elif parameters['priceOperator'] == '이하':  # 이하
        price_range = {
            'lte': int(parameters['priceOperator'])  # lte: 이하 / lt: 미만
        }
    else:
        price_range = {
            'gte': 0
        }

    # elasticsearch search body
    body = {
        'query': {
            'bool': {
                'must': [
                    {
                        'range': {
                            'price': price_range
                        }
                    },
                    color,
                    brand,
                    gender,
                    category,
                    age
                ],
            }
        },
        'size': 10,
        #'sort': isPopulary_sort
    }


    # elasticsearch search
    search_result = es.search(index='cublick-exhibition-musinsa-cloth-3', body=body)

    # return result
```

```
    result = []

    for i in search_result['hits']['hits']:
        res = {'id': '', 'brandName': '', 'color': '', 'gender': '', 'imgUrl': [],
                'likesAgeRange': {'from': '', 'until': ''},
                'likesCount': '', 'likesGender': '', 'name': '', 'price': '',
                'productName': '', 'siteUrl': ''}

        if i['_id'] not in res['id']:  # 중복제거
            res['id'] = i['_id']

            res['brandName'] = i['_source']['brandName']
            res['color'] = i['_source']['color']
            res['gender'] = i['_source']['gender']
            res['imgUrl'].append(i['_source']['imgUrl'])
            res['likesAgeRange']['from'] = i['_source']['likesAgeRange']['from']
            res['likesAgeRange']['until'] = i['_source']['likesAgeRange']['until']
            res['likesCount'] = i['_source']['likesCount']
            res['likesGender'] = i['_source']['likesGender']
            res['name'] = i['_source']['name']
            res['price'] = i['_source']['price']
            res['productName'] = i['_source']['productName']
            res['siteUrl'] = i['_source']['siteUrl']

        result.append(res)

    return jsonify(result)
```

- **app.py**

Flask API를 실행시키기 위한 code

```
from routes import app

#/apidocs 에서 확인 참고페이지 https://github.com/flasgger/flasgger
if __name__ == "__main__":
    app.run(host='0.0.0.0', port=5050, threaded=True, debug=True, use_reloader=False) # 포트5050번으로 실행
```

★ Elasticsearch Documents

Elasticsearch의 해당 Documents를 설명하기 위함.

해당 kibana 주소: http://34.64.63.141:5601/app/dev_tools#/console

Index name	cublick-exhibition-musinsa-cloth-3

```
# 해당 하는 Fields에 대해서만 검색을 진행하기에 해당 부분 초기화
    parameters = {}
    parameters['age'] = ''
    parameters['brand'] = ''
    parameters['color'] = ''
    parameters['gender'] = ''
    parameters['category'] = {'main_category': '', 'sub_category': ''}
```

```
    parameters['price'] = ''
    parameters['priceOperator'] = ''
    parameters['text'] = text
```

Example)


```
{
  "_index" : "cublick-exhibition-musinsa-cloth-3",
  "_type" : "_doc",
  "_id" : "aqVeQYQBINikir_MH0iE",
  "_score" : 1.0,
  "_source" : {
    "name" : "side zip sweatshirts light emera...",
    "siteUrl" : "https://www.musinsa.com/app/goods/1774672",
    "color" : "lightGreen",
    "category" : {
      "main" : {
        "categoryName" : "tops"
      },
      "middle" : {
        "categoryName" : "mantomanAndsweatshirt"
      }
    },
    "tags" : "['스웻셔츠', ' 오버사이즈']",
    "likesCount" : 6,
    "likesAgeRange" : {
      "from" : 0,
      "until" : 0
    },
    "likesGender" : "정보 없음",
    "gender" : "man",
    "brandName" : "LORD JOHN GREY",
    "productName" : "L21ATS101A",
    "imgUrl" : "['https://image.msscdn.net/images/goods_img/20210203/1774672/1774672_1_500.jpg?t
      =20210226160535']",
    "price" : 89000
  }
},
```