

캡스톤디자인I 중간보고서

제 목	국문		자연어 처리 기반 영화 리뷰 필터링 및 시각화	
	영문		Filter and Visualize movie reviews based on NLP	
진 행 상 황	중요마일스톤	<ul style="list-style-type: none">● 기초 아키텍처 작성 (시스템 구성도)● 필요한 핵심 기능 파악● API 서버 구축● 감성 분석 데이터 축적을 위한 DB 설계 및 구축● 웹 페이지 제작 - 메인 페이지, 검색 결과 페이지, 상세 페이지● 모델 제작 - 필터링, 감상 포인트 추출 모델		
	진행상황	<ul style="list-style-type: none">● 기초 아키텍처 작성 완료● Vue.js를 이용한 메인 페이지 및 영화 검색 페이지 설계 완료● Node.js를 이용한 API 작성과 서버 구축 완료● 속성 기반 감성 분석용 데이터셋 구축 중● 속성 기반 감성 분석 모델 설계 진행 중		
산출물	요구사항 정의서(별첨 1), 중간보고서(별첨 2)			
팀 구성원	학년	학 번	이 름	연락처(전화번호/이메일)
	4	20171602	한재혁	010-5584-9805 / hjh0633@gmail.com
	4	20171615	서민석	010-2856-4834 / seominseok4834@gmail.com
	4	20171598	정인상	010-9620-1518 / jisk101614@gmail.com
컴퓨터공학과와 프로젝트 관리규정에 따라 다음과 같이 요구사항 정의서와 중간보고서를 제출합니다				
2022 년 4 월 29 일				
책임자 : 한재혁 (인) 지도교수 : 황경호 (인)				

[별첨1]

프로젝트명 : 자연어 처리 기반 영화 리뷰 필터링 및 시각화

소프트웨어 요구사항 정의서

Version 1.0

개발 팀원 명(팀리더): 한재혁
서민석
정인상

대표 연락처: 010-5584-9805
e-mail: hjh0633@gmail.com

목차

1. 개요
2. 시스템 장비 구성요구사항
3. 기능 요구사항
4. 성능 요구사항
5. 인터페이스 요구사항
6. 데이터 요구사항
7. 테스트 요구사항
8. 보안 요구사항
9. 품질 요구사항
10. 제약 사항
11. 프로젝트 관리 요구사항

1. 시스템 개요

1) 목표

소비자가 영화를 선택하는 과정에 있어서 도움이 되지 않는 리뷰들(내용이 없는 리뷰, 영화의 내용과 관계없는 리뷰, 지나치게 편향된 리뷰 등)을 필터링하고 이를 관람 포인트별로 분석하여 어느 포인트에서 얼마만큼의 긍정도를 가지고 있는 시각화함으로써 소비자들의 영화 선택에 있어 도움이 되는 웹 어플리케이션을 제작한다.

2) 시스템 구성도



2. 시스템 장비 구성요구사항

요구사항 고유번호		ECR-001		
요구사항 명칭		모델 개발 서버	응락수준	필수
요구사항 분류		시스템 장비 구성 요구사항		
요구사항 상세설명	정의	- 모델 훈련 및 테스트		
	세부 내용	<ul style="list-style-type: none"> - 장비 품목 : Ubuntu Server - 장비 수량 : 1개 - 장비 기능 : 모델 학습 및 테스트 기능 수행 - 장비 성능 및 특징 <ul style="list-style-type: none"> - CPU : Intel(R) Xeon(R) Gold 6130 CPU @ 2.10GHz - GPU : NVIDIA GeForce RTX 2080 Ti (3개) - RAM : 187GB - 자원 제약사항 : 교내망으로만 접속할 수 있기 때문에 서버 사용을 위해선 교내망을 통한 접속이 이루어져야 한다. 		

요구사항 고유번호		ECR-002		
요구사항 명칭		인공지능	응락수준	필수
요구사항 분류		시스템 장비 구성 요구사항		
요구사항 상세설명	정의	- 리뷰 필터링 모델		
	세부 내용	<ul style="list-style-type: none"> - 장비 품목 : 머신러닝 모델 - 장비 수량 : 1개 - 장비 기능 : 영화 리뷰 필터링 - 장비 성능 및 특징 : 크롤링 해온 리뷰들에 대해 실시간으로 해당 리뷰를 보여줄 것인지 보여주지 않을 것인지 여부를 결정한다. 근거 없는 비판, 칭찬성 리뷰, 비관람객이 작성한 리뷰, 편향성이 짙은 리뷰 등이 필터링 된다. 		

요구사항 고유번호		ECR-003		
요구사항 명칭		인공지능	응락수준	필수
요구사항 분류		시스템 장비 구성 요구사항		
요구사항 상세설명	정의	- 감상포인트 추출 모델		
	세부 내용	<ul style="list-style-type: none"> - 장비 품목 : 속성 기반 감성 분석 딥러닝 모델 - 장비 수량 : 1개 - 장비 기능 : 감상 포인트 추출 - 장비 성능 및 특징 : 크롤링 해온 리뷰에 대해 각각의 리뷰별로 감상 포인트를 추출, 감성 분석을 진행한다. 감상 포인트별 감성 분석 결과를 종합해 영화별 감상 포인트를 반환한다. 		

요구사항 고유번호		ECR-004		
요구사항 명칭		감상 포인트 추출 모델용 학습 데이터셋	응락수준	필수
요구사항 분류		시스템 장비 구성 요구사항		
요구사항 상세설명	정의	- 인공지능 학습 데이터셋		
	세부 내용	<ul style="list-style-type: none"> - 장비 품목 : 모두의 말뭉치 속성 기반 감성 분석 데이터셋 - 장비 수량 : 1개 - 장비 기능 : 속성 기반 감성 분석 딥러닝 모델 학습 및 검증 - 장비 성능 및 특징 <ul style="list-style-type: none"> - 리뷰 텍스트 내에서 관람객의 의견이 들어간 substring을 begin, end 로 인덱싱 - 인덱싱한 부분의 내용을 고려하여 개체#속성 쌍으로 정의, 해당 부분에 대한 감성을 3개(긍정, 부정, 중립)로 라벨링 		

3. 기능 요구사항

요구사항 고유번호		SFR-001		
요구사항 명칭		박스 오피스 랭킹 시각화	응락수준	필수
요구사항 분류		기능 요구사항		
요구사항 상세 설명	정의	- 박스 오피스 랭킹 수집 및 시각화		
	세부 내용	<ul style="list-style-type: none"> - 박스 오피스 랭킹 상위 5개 목록 수집 - 수집 목록 시각화 - 수집 목록 클릭 시 해당 목록에 대한 영화 정보 시각화 		

요구사항 고유번호		SFR-002		
요구사항 명칭		영화 검색	응락수준	필수
요구사항 분류		기능 요구사항		
요구사항 상세 설명	정의	- 사용자가 원하는 영화의 목록을 검색		
	세부 내용	<ul style="list-style-type: none"> - 사용자가 원하는 영화의 목록을 검색 - 검색하는 문자열의 길이는 2 이상이어야 함 - 검색하는 문자열을 포함하고 있는 영화를 KMDB에서 검색해 결과 시각화 - KMDB에서 받은 영화의 포스터, 제목, 감독, 출시년도, 출연배우 표시 		

요구사항 고유번호		SFR-003		
요구사항 명칭		영화 리뷰 시각화	응락수준	필수
요구사항 분류		기능 요구사항		
요구사항 상세 설명	정의	- 사용자가 선택한 영화의 리뷰를 시각화		
	세부 내용	<ul style="list-style-type: none"> - 사용자가 선택한 영화의 리뷰를 시각화 - 사용자가 선택한 영화의 제목을 인자로 자체 제작한 API함수 호출 - API함수가 반환한 결과를 리뷰 페이지에 출력 		

요구사항 고유번호		SFR-004		
요구사항 명칭		영화 리뷰 분석 시각화	응락수준	필수
요구사항 분류		기능 요구사항		
요구사항 상세 설명	정의	- 사용자가 선택한 영화의 리뷰를 분석하여 결과를 시각화		
	세부 내용	<ul style="list-style-type: none"> - 사용자가 선택한 영화의 리뷰를 분석하여 결과를 시각화 - 사용자가 선택한 영화의 제목을 인자로 자체 제작한 API함수 호출 - API함수가 반환한 결과를 감상 포인트 페이지에 출력 		

요구사항 고유번호		SFR-005		
요구사항 명칭		감상 포인트 추출 인공지능	응락수준	필수
요구사항 분류		기능 요구사항		
요구사항 상세설명	정의	속성 기반 감성 분석 딥러닝 모델 학습		
	세부 내용	<ul style="list-style-type: none"> - 리뷰 내에서 감상 포인트에 해당하는 부분에 대해 라벨링한 데이터셋으로 모델을 학습한다. - 연계 기능 : 웹 표현 - 산출물 : 학습된 딥러닝 모델 		

요구사항 고유번호		SFR-006		
요구사항 명칭		감상 포인트 추출 인공지능	응락수준	필수
요구사항 분류		기능 요구사항		
요구사항 상세설명	정의	속성 기반 감성 분석 딥러닝 모델 학습		
	세부 내용	<ul style="list-style-type: none"> - 리뷰 내에서 감상 포인트에 해당하는 부분에 대해 라벨링한 데이터셋으로 모델을 학습한다. - 연계 기능 : 웹 표현 - 산출물 : 학습된 딥러닝 모델 		

4. 성능 요구사항

요구사항 고유번호		PER-001		
요구사항 명칭		리뷰 필터링	응락수준	필수
요구사항 분류		성능 요구사항		
요구사항 상세설명	정의	영화 리뷰 필터링 시간		
	세부 내용	<ul style="list-style-type: none"> - 직접 설계한 크롤링 API를 통해 크롤링시 1000개당 약 11초가 소요될 것으로 예상된다. - 필터링 시간 및 웹 페이지 로드까지 고려했을 때, 필터링 모델은 1000개당 최소 5초 이내로 처리해야 한다. 		

5. 인터페이스 요구사항

요구사항 고유번호		SIR001		
요구사항 명칭		메인 페이지 인터페이스	응락수준	필수
요구사항 분류		인터페이스 요구사항		
요구사항 상세 설명	정의	- 메인 페이지의 인터페이스 요구사항		
	세부 내용	<ul style="list-style-type: none"> - 시스템명이 화면 중앙에 위치 - 시스템명 아래 영화를 검색할 수 있는 input박스 위치 - 스크롤시 박스 오피스 랭킹 출력 		

요구사항 고유번호		SIR002		
요구사항 명칭		박스 오피스 랭킹 인터페이스	응락수준	필수
요구사항 분류		인터페이스 요구사항		
요구사항 상세 설명	정의	- 메인 페이지의 박스오피스 랭킹 인터페이스 요구사항		
	세부 내용	<ul style="list-style-type: none"> - 메인 페이지에서 스크롤시 출력 - 시스템명과 검색 input박스가 화면 상단으로 이동 - 화면의 세로 중앙에 가로 방향으로 박스 오피스 랭킹 위치 		

요구사항 고유번호		SIR003		
요구사항 명칭		박스 오피스 상세 인터페이스	응락수준	필수
요구사항 분류		인터페이스 요구사항		
요구사항 상세 설명	정의	- 박스 오피스 랭킹의 영화를 클릭하면 나오는 인터네이스 요구사항		
	세부 내용	<ul style="list-style-type: none"> - 모달 형식으로 선택한 영화의 상세 정보 인터페이스 출력 - 영화 포스터, 제목, 감독, 출연 배우, 줄거리 출력 - 모달 창 하단에 '해당 영화의 리뷰 바로가기' 버튼 위치 		

요구사항 고유번호		SIR004		
요구사항 명칭		검색 결과 인터페이스	응락수준	필수
요구사항 분류		인터페이스 요구사항		
요구사항 상세 설명	정의	- 영화 검색 결과 페이지의 인터페이스		
	세부 내용	<ul style="list-style-type: none"> - 시스템명과 검색 input 박스가 화면 상단에 위치 - 총 검색 결과 수가 input 박스 아래에 위치 - 영화 검색 결과가 총 검색 결과 수 아래 list 형식으로 출력 - 각 결과는 영화 포스터, 제목, 감독, 출시연도, 출연 배우를 포함하여 출력 		

요구사항 고유번호		SIR005		
요구사항 명칭		영화 상세 인터페이스	응락수준	필수
요구사항 분류		인터페이스 요구사항		
요구사항 상세 설명	정의	- 검색 목록에서 영화 선택시 해당 영화 상세 페이지의 인터페이스		
	세부 내용	<ul style="list-style-type: none"> - 시스템명이 화면 상단에 위치 - 총 검색 결과 수가 input 박스 아래에 위치 - 영화 포스터, 제목, 감독, 출연 배우, 줄거리 출력 - 화면 하단에 해당 영화의 리뷰와 감상 포인트를 볼 수 있는 tab 위치 		

요구사항 고유번호		SIR006		
요구사항 명칭		영화 리뷰 데이터 인터페이스	응락수준	필수
요구사항 분류		인터페이스 요구사항		
요구사항 상세 설명	정의	- 선택한 영화의 필터링된 리뷰 데이터 출력 인터페이스		
	세부 내용	- 해당 리뷰를 작성한 사람의 닉네임, 내용, 리뷰를 가져온 플랫폼 출력		

요구사항 고유번호		SIR007		
요구사항 명칭		영화 감상 포인트 인터페이스	응락수준	필수
요구사항 분류		인터페이스 요구사항		
요구사항 상세 설명	정의	- 선택한 영화의 감상 포인트 출력 인터페이스		
	세부 내용	- 선택한 영화를 분석해 해당 영상의 감상 포인트 비중을 “연기력”, “영상미”, “연출”, “OST”, “스토리”를 원 그래프를 통해 시각화 - 각 감상 포인트에 대해 긍정적인 반응과 부정적인 반응의 정도를 가로 막대 그래프를 통해 시각화		

6. 데이터 요구사항

요구사항 고유번호		DAR-001		
요구사항 명칭		데이터 구축		
요구사항 분류		데이터 요구사항	응락수준	필수
요구사항 상세 설명		- 영화별 감상 포인트 추출 모델 실행 결과를 데이터베이스 테이블로 저장한다. - 테이블은 감상 포인트와 감상 포인트에 대한 긍·부정 개수로 구성된다.		

7. 테스트 요구사항

요구사항 고유번호		TER-001		
요구사항 명칭		테스트 방안	응락수준	필수
요구사항 분류		테스트		
요구사항 상세 설명	정의	- 단위테스트 요구사항		
	세부 내용	<ul style="list-style-type: none"> - 시스템은 최소 기능인 영화 리뷰 필터링을 베이스 라인으로 간주한다. - 사용자가 검색한 영화에 대한 리뷰를 크롤링해 필터링된 리뷰와 감상 포인트별 가이드라인을 제시하는 것을 목표로 한다. - 테스트를 위해서 임의의 영화를 선정하여 명시한 요구사항을 검증 활동을 통해 평가한다. - 기능 구현 정확성은 실제 사용자가 직접 테스트 수행 기간에 테스트를 수행함으로써 평가한다. 		

요구사항 고유번호		TER-002		
요구사항 명칭		단위 테스트	응락수준	필수
요구사항 분류		테스트 요구사항		
요구사항 상세 설명	정의	- 단위테스트 요구사항		
	세부 내용	<ul style="list-style-type: none"> - 기능 요구사항에 기술한 요구사항에 대하여 단위 테스트 수행 - 각 단위 테스트에 대하여 시나리오, 처리 절차, 수행 데이터, 예상 결과 등으로 사전에 정의하고 수행 		

8. 보안 요구사항

요구사항 고유번호		SER-001		
요구사항 명칭		API서버 보안	응락수준	필수
요구사항 분류		보안 요구사항		
요구사항 상세 설명	정의	- API서버 보안		
	세부 내용	- 해당 API서버를 사용함에 있어서 허가된 origin이 아닌 곳에서 정보를 요청할 경우 결과를 반환하지 않는다.		

요구사항 고유번호		SER-002		
요구사항 명칭		API키 보안	응락수준	필수
요구사항 분류		보안 요구사항		
요구사항 상세 설명	정의	- KMDB API키 보안		
	세부 내용	- KMDB의 API키를 코드 내부에 포함시켜 배포하지 않는다. - 해당 API키를 서버에 위치시켜 API형식으로 내려받도록 한다.		

9. 품질 요구사항

요구사항 고유번호		QUR-001		
요구사항 명칭		필터링 검출률	응락수준	필수
요구사항 분류		품질		
요구사항 상세설명	정의	인공지능의 성능(기술 관점)		
	세부 내용	- 인공지능을 통해 영화 리뷰를 필터링할 때, F1-score가 0.95 이상의 값을 갖도록 한다.		

요구사항 고유번호		QUR-002		
요구사항 명칭		감상포인트 검출률	응락수준	필수
요구사항 분류		품질		
요구사항 상세설명	정의	인공지능의 성능(기술 관점)		
	세부 내용	- 감상 포인트별 감성 분석을 수행할 때, 감상 포인트별로 F1-score가 0.9 이상의 값을 갖도록 한다.		

10. 제약 사항

요구사항 고유번호		COR-001		
요구사항 명칭		시스템 개발	응락수준	필수
요구사항 분류		제약사항		
요구사항 상세 설명	세부 내용	<ul style="list-style-type: none"> - 프로그래밍 언어: Python - 사용 라이브러리: Pytorch, Konlpy, KoBERT - 개발도구: Jupyter Notebook 		

요구사항 고유번호		COR-002		
요구사항 명칭		데이터셋 취득	응락수준	필수
요구사항 분류		제약사항		
요구사항 상세 설명	세부 내용	<ul style="list-style-type: none"> - 쉽게 취득할 수 있는 일반적인 감성 분석 데이터셋과 다르게 한글로 이루어진 속성기반 감성 분석(ABSA) 데이터셋 취득 제한으로 인해 모델에 학습하기 충분한 양의 데이터 구축이 필요하다. 		

요구사항 고유번호		COR-003		
요구사항 명칭		한국어 전처리	응락수준	필수
요구사항 분류		제약사항		
요구사항 상세 설명	세부 내용	<ul style="list-style-type: none"> - 영화 리뷰의 경우 은어, 비속어, 이모티콘 등이 많이 사용돼 클렌징 과정이 필요하고, 띄어쓰기나 맞춤법이 지켜지지 않은 리뷰가 많아 맞춤법 정제 및 띄어쓰기 교정이 필요하다. 		

요구사항 고유번호		COR-004		
요구사항 명칭		업무 모듈화 및 자원 활용 방안	응락수준	필수
요구사항 분류		제약사항		
요구사항 상세 설명	세부 내용	<ul style="list-style-type: none"> - 유연성, 확장성을 확보할 수 있도록 모듈화 개발전략을 반영한다. - 현재 활용 가능한 소프트웨어를 최대한 활용하여 수행한다. 		

11. 프로젝트 관리 요구사항

요구사항 고유번호		PMR-001		
요구사항 명칭		프로젝트 수행 조직	응락수준	필수
요구사항 분류		프로젝트 관리		
요구사항 상세 설명	세부 내용	<ul style="list-style-type: none"> - 한재혁 : Vue.js를 이용한 웹 페이지 구축 및 node.js를 이용한 API 작성과 서버 구축 - 서민석 : Glove를 활용한 한국어 단어 임베딩 구축 및 데이터셋 전처리 코드 작성, Pytorch를 이용한 ABSA 기반 감상 포인트 추출 모델 개발 - 정인상 : Python을 이용한 ABSA 데이터셋 가공 코드 설계 및 영화 감상 포인트 추출 모델 개발 		

요구사항 고유번호		PMR-002		
요구사항 명칭		프로젝트 일정계획	응락수준	필수
요구사항 분류		프로젝트 관리		
요구사항 상세 설명	세부 내용	<ul style="list-style-type: none"> - 필터링 모델 개발 : 7월 이전 - API서버 구축 및 웹 제작 : 8월 이전 - 관람 포인트 추출 모델 개발 : 9월 이전 - 모델 테스트 및 웹 디자인 개선 : 10월 이전 		

[별첨2]

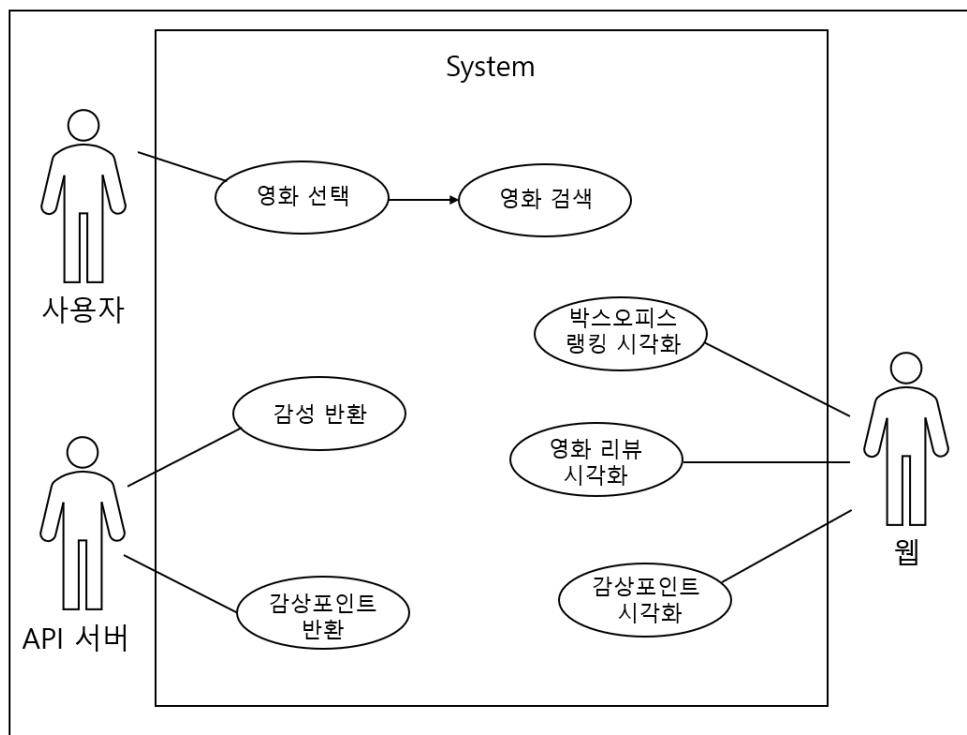
중간보고서

1. 요구사항 정의서에 명시된 기능에 대하여 현재까지 분석, 설계, 구현(소스코드 작성) 및 테스트한 내용을 기술하시오.

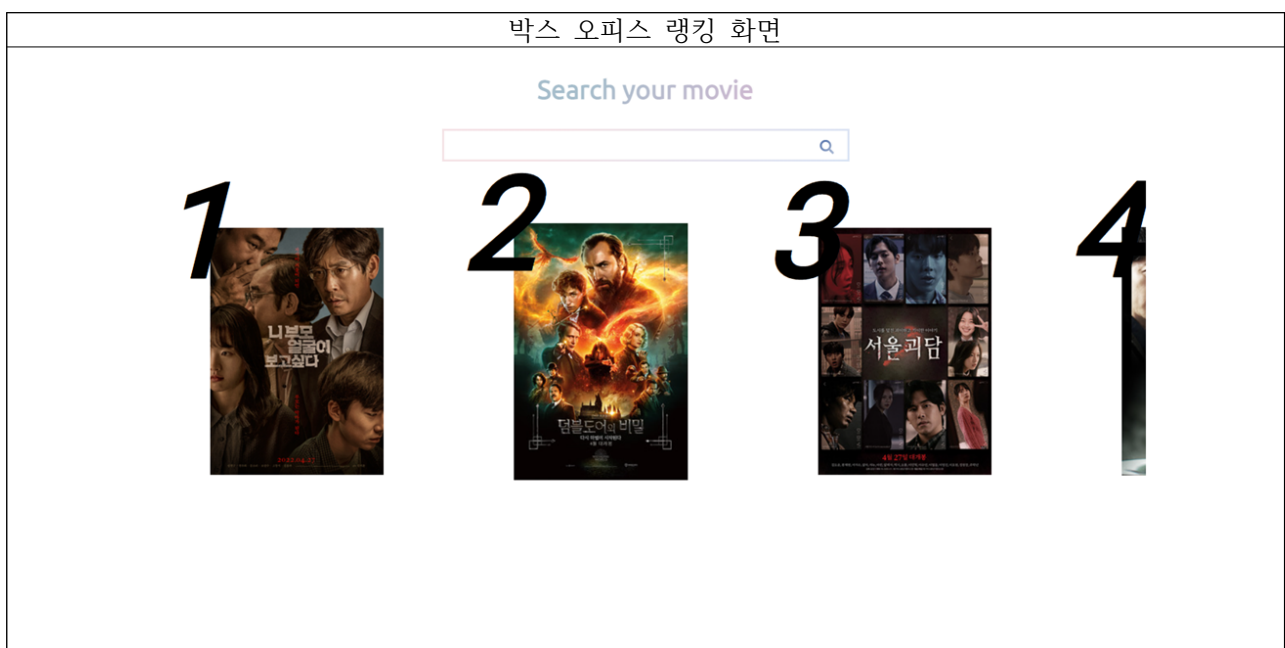
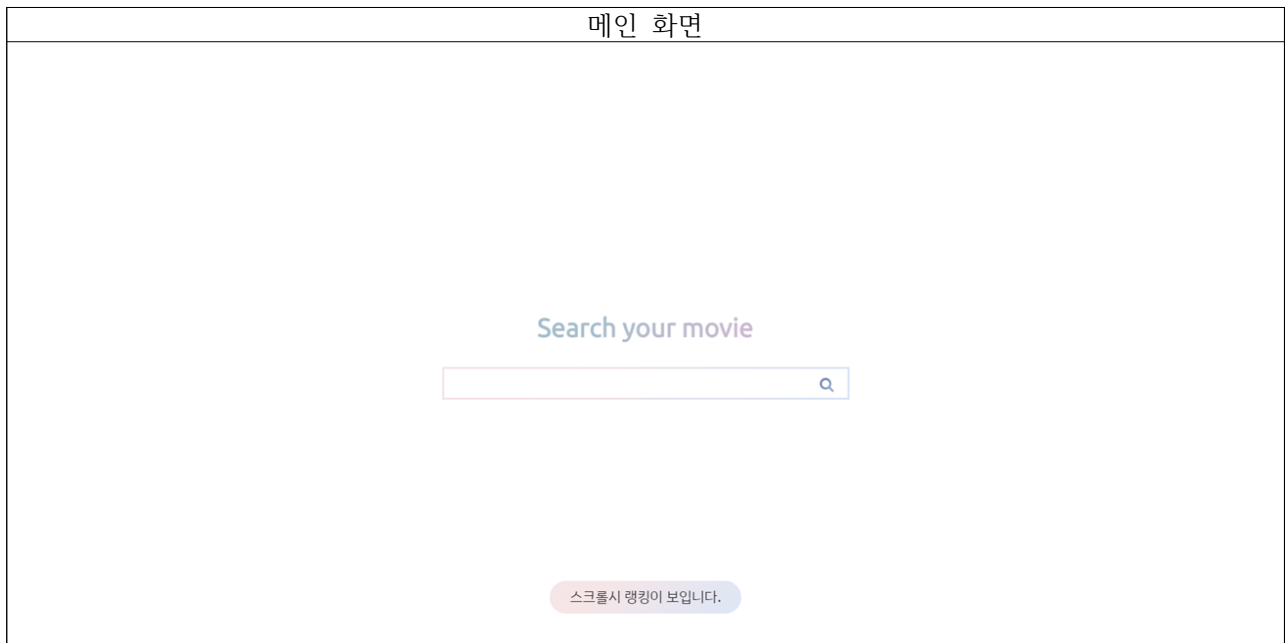
- 시스템(하드웨어, 시스템)구성도, 또는 소프트웨어 아키텍처

- 웹 : 사용자의 접근을 위한 웹을 제공한다. 해당 웹을 통해 사용자는 박스 오피스 랭킹의 상위 5개 영화 정보를 시각화하고, 사용자가 원하는 영화의 검색할 수 있는 UI를 제공하며 결과를 출력한다. 검색 결과에서 영화를 선택하면 해당 영화에 대한 상세한 정보를 열람할 수 있으며, 해당 영화의 리뷰들을 웹에서 크롤링한 후 필터링 및 분석하여 결과를 보여준다.
- API서버 : 사용자가 영화를 선택하게 되면 해당 영화의 영화명이 서버로 전달되면서 해당 영화의 리뷰들을 크롤링하는 API함수가 작성되어 있다. 추가적으로 크롤링된 리뷰들을 AI모델에 입력값으로 넣어 결과값을 반환하는 API함수 또한 작성되어 있다.
- 데이터 전처리 : 리뷰 데이터의 특성상 은어, 비속어, 이모티콘등이 포함돼 클렌징 및 띄어쓰기, 맞춤법 교정을 적용했다. 사전 훈련된 단어 임베딩(pre-trained word embedding)으로 위키백과, KCC(뉴스 기사 한국어 대용량 말뭉치), 영화 리뷰 말뭉치를 가공해 300차원의 glove 한국어 단어 임베딩(1,594,871개의 단어로 구성)을 만들어 사용했다. 감성사전은 KNU 한국어 감성사전을 사용했으며, tokenizer로는 mecab과 koBERT를 사용했다.
- 인공지능 : 영화 리뷰에서 사용자의 평가가 들어간 부분을 추출해 해당 평가가 긍정적인지 부정적인지 판단한다. 이를 위해 모두의 말뭉치에서 제공하는 속성 기반 감성 분석 데이터셋을 사용했으며, 모델은 capsnet-bert를 사용했고 PyTorch로 구현했다.

- 기능별 상세 요구사항(또는 유스케이스)



- 현재까지 구현한 소스 코드 및 캡처



박스 오피스 상세 정보 화면

Search your movie

신비한 동물들과 덤블도어의 비밀

개봉일 : 2022년 03월 31일

감독 : 데이빗 에이츠

장르 : 어드벤처, 가족, 판타지

출연 : 매즈 미켈슨, 주드 로, 케서린 워터스톤, 에디 레드메인, 이즈라 밀러, 댄 포글러, 리처드 코일, 제시카 윌리엄스

줄거리 : 가장 위험한 마법에 맞선, 세상을 구할 전쟁이 시작된다! 1930년대, 제2차 세계대전에 마법사들이 개입하게 되면서 강력한 어둠의 마법사 그린델왈드의 힘이 급속도로 커진다. 덤블도어는 뉴트 스캐먼더에게 위대한 마법사 가문 후손, 마법학교의 유능한 교사, 머글 등으로 이루어진 팀에게 임무를 맡긴다. 이에 뉴트와 친구들은 머글과의 전쟁을 선포한 그린델왈드와 추종자들, 그의 위험한 신비한 동물들에 맞서 세상을 구할 거대한 전쟁에 나선다. 한편 전쟁의 위기가 최고조로 달한 상황에서 덤블도어는 더 이상 방관자로 머물 수 없는 순간을 맞이하고, 서서히 숨겨진 비밀이 드러나는데...

리뷰 바로가기

검색 결과 화면

Search your movie

"스파이더맨"에 대한 검색결과 13건



스파이더맨: 노 웨이 홈 | 2021

감독명 : 존 왓츠

장르 : 액션, 어드벤처, SF



스파이더맨: 파 프롬 홈 | 2019

감독명 : 존 왓츠

장르 : 어드벤처, 코메디, SF, 액션, 판타지



스파이더맨: 뉴 유니버스 | 2018

감독명 : 밥 퍼시케티

장르 : 액션, 가족



스파이더맨: 홈 커밍 | 2017

감독명 : 존 왓츠

장르 : 액션, SF, 어드벤처

영화 선택시 화면

Search your movie

"스파이더맨"에 대한 검색결과 13건



스파이더맨: 노 웨이 홈 | 2021

감독명 : 존 왓츠

장르 : 액션,어드벤처,SF



스파이더맨: 파 프롬 홈 | 2019

감독명 : 존 왓츠

장르 : 어드벤처,코메디,SF,액션,판타지



스파이더맨: 뉴 유니버스 | 2018

감독명 : 밥 퍼시케티

장르 : 액션,가죽



스파이더맨: 홈 커밍 | 2017

감독명 : 존 왓츠

장르 : 액션,SF,어드벤처

영화 상세 정보 화면

Search your movie



개봉

일 : 2021년 12월 08일

감독 :

감독 : 존 왓츠

장르 :

장르 : 액션,어드벤처,SF

출연 :

출연 : 톰 홀랜드, 켄다아 콜맨, 베네딕트 컴버배치, 존 파브로, 제이콥 베달런

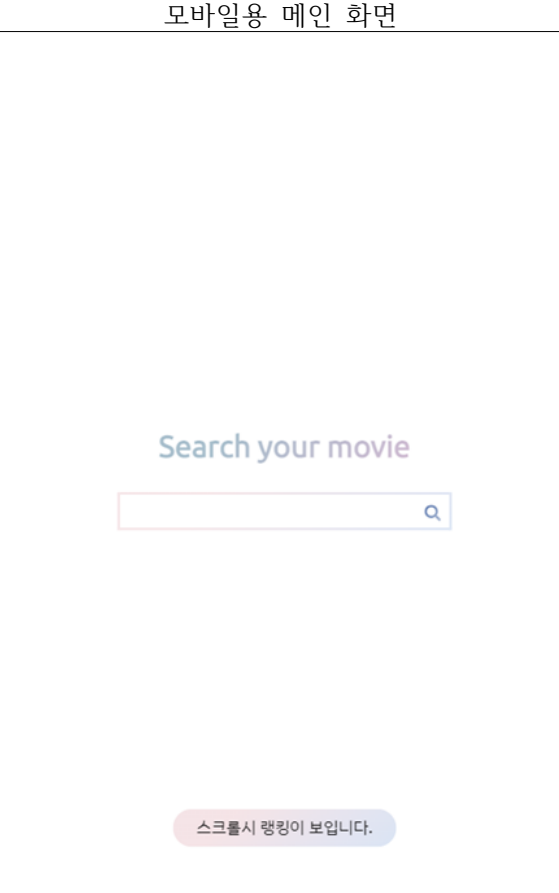
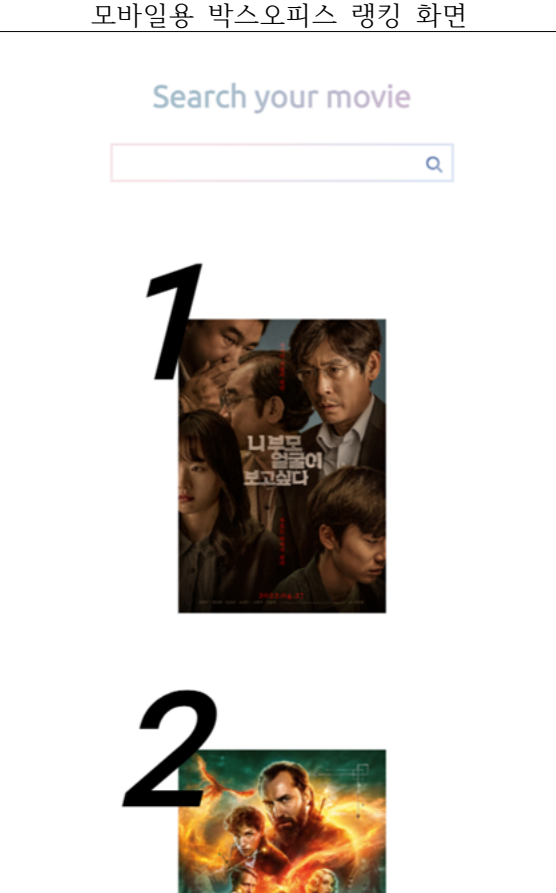
줄거

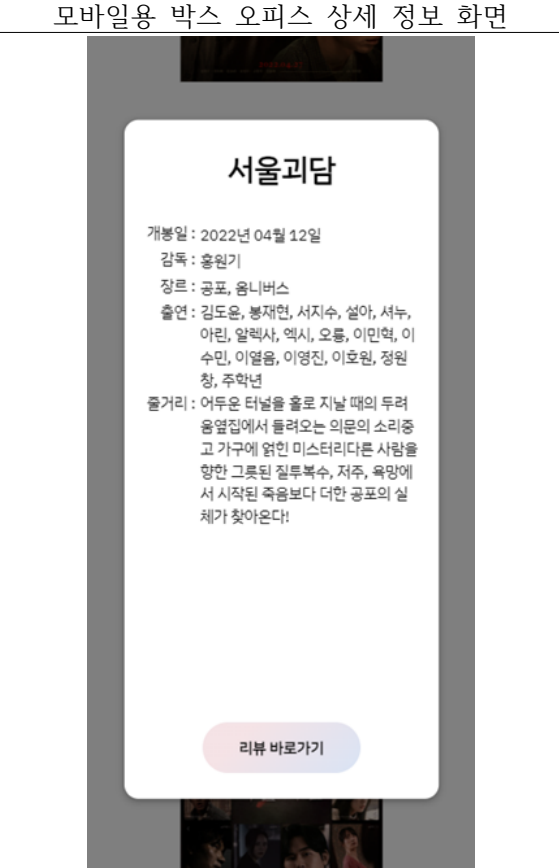

줄거 : 정체가 탄로나 스파이더맨 '피터 파커'(톰 홀랜드)가 문제를 해결하기 위해 '닥터 스트레인지'(베네딕트 컴버배치)의 도움을 받던 중 뜻하지 않게 멀티버스가 열리게 되고, 이를 통해 '닥터 옥토퍼스'(알프리드 물

리 : 리나) 등 각기 다른 차원의 숙적들이 나타나며 사상 최악의 위기를 맞게 되는 이야기를 그린 마블 액션 블록버스터.

ss

hh

모바일용 메인 화면	모바일용 박스오피스 랭킹 화면
 <p>Search your movie</p> <p>스크롤시 랭킹이 보입니다.</p>	 <p>Search your movie</p> <p>1</p> <p>2</p>

모바일용 박스 오피스 상세 정보 화면	모바일용 영화 검색 결과 화면
 <p>서울괴담</p> <p>개봉일 : 2022년 04월 12일 감독 : 홍원기 장르 : 공포, 유니버스 출연 : 김도운, 봉재현, 서지수, 설아, 서누, 아린, 알렉사, 엑시, 오홍, 이민혁, 이수민, 이열음, 이영진, 이호원, 정원창, 주학년 줄거리 : 어두운 터널을 홀로 지날 때의 두려움 열집에서 들려오는 의문의 소리 중 고 가구에 얽힌 미스터리다툼 사람을 향한 그릇된 질투복수, 저주, 욕망에서 시작된 죽음보다 더한 공포의 실체가 찾아온다!</p> <p>리뷰 바로가기</p>	 <p>Search your movie</p> <p>"스파이더맨"에 대한 검색결과 13건</p> <ul style="list-style-type: none"> 스파이더맨: 노 웨이 홈 2021 감독명 : 존 왓츠 장르 : 액션,어드벤처,SF 스파이더맨: 파 프롬 홈 2019 감독명 : 존 왓츠 장르 : 어드벤처,코메디,SF,액션,판타지 스파이더맨: 뉴 유니버스 2018 감독명 : 밥 페시케티 장르 : 액션,가죽 스파이더맨: 홈 커밍 2017 감독명 : 존 왓츠 장르 : 액션,SF,어드벤처 어메이징 스파이더맨 2 2014 감독명 : 마크 웹 장르 : 액션,어드벤처,판타지 어메이징 스파이더맨 2012 감독명 : 마크 웹 장르 : 액션,어드벤처,스릴러 스파이더맨 3 2007

word_embedding.py

glove 모델 훈련 및 저장

```
import pickle
from glove import Corpus, Glove

corpus = Corpus()
corpus.fit(total_morph, window=15)

# glove model 생성
glove = Glove(no_components=300, learning_rate=0.01)
glove.fit(corpus.matrix, epochs=15, no_threads=4, verbose=False)
glove.add_dictionary(corpus.dictionary)

# word dict 생성
word_dict = {}
for word in glove_model.dictionary.keys():
    word_dict[word] = glove_model.word_vectors[glove_model.dictionary[word]]
print('[Success] Length of word dict: ', len(word_dict))

# save word_dict
with open('glove_word_dict_300.pickle', 'wb') as f:
    pickle.dump(word_dict, f)
print('[Success] Save word dict')
```

```
{'스품': array([ 0.01687806,  0.04371413,  0.05712648,  0.01844265,  0.01108963,
 -0.04780463,  0.01532296, -0.01851136, -0.02451378, -0.02058796,
 -0.01775614,  0.01853415,  0.02120506,  0.02445549, -0.02428252,
 -0.02709526,  0.09301033,  0.02942713, -0.06487419, -0.02545111,
 -0.03039146, -0.04807996, -0.02898595,  0.02289183,  0.03328399,
  0.02173328, -0.02402653, -0.07460099, -0.01935587, -0.01917952,
 -0.02295648,  0.02418459,  0.02497789,  0.02491846, -0.01635201,
 -0.04941483, -0.01912335,  0.05026753, -0.0101685 ,  0.01920339,
  0.02235636, -0.01869214, -0.01807253,  0.02058045,  0.02119259,
 -0.01915527,  0.0252072 ,  0.07071469, -0.00935074,  0.01799365,
  0.02514742,  0.01891136,  0.06653911,  0.06390345,  0.06357712,
  0.02186423, -0.02287705, -0.0169937 ,  0.02268015, -0.01748731,
 -0.0193435 , -0.02292906, -0.02569123,  0.01690702,  0.05190504,
 -0.06012266, -0.02089435,  0.01992355,  0.08949602, -0.03099131,
  0.01829355,  0.02281876,  0.01869702,  0.01999496,  0.02599778,
  0.10532122, -0.04398261,  0.02483481,  0.02581101,  0.02349536,
  0.03188759, -0.02505857, -0.02158529,  0.06148403,  0.0166215 ,
  0.03269424,  0.01690354,  0.02937467,  0.01860201,  0.02005839,
 -0.01653808, -0.02270048, -0.02920488, -0.0675908 ,  0.03336313,
  0.03063314, -0.03126409,  0.02831555,  0.01952382, -0.02207489,
 -0.02186787, -0.02441415, -0.02831206, -0.02403111, -0.02445845,
  0.02174035, -0.03736613,  0.08589584,  0.01909522,  0.02101641,
 -0.04684634, -0.0163512 ,  0.02036522, -0.02414935, -0.03461472,
  0.01843837,  0.04186685,  0.02962485,  0.02253044,  0.04512058,
 -0.02064011, -0.03836357, -0.01714932,  0.02178195,  0.06236261,
 -0.02276859,  0.01707268,  0.01733467,  0.02068501,  0.11432796,
 -0.02837636, -0.07231355, -0.02161056,  0.02140141,  0.02368186,
 -0.02180653, -0.02401641, -0.02174603,  0.02292262,  0.02892909,
 -0.02051234, -0.0176458 , -0.03503321,  0.0317562 , -0.01971699,
  0.01906822,  0.02412252,  0.02243512,  0.01724962,  0.01908415,
 -0.01603695,  0.04270295,  0.0472204 , -0.03954613, -0.05179545,
 -0.07657435,  0.01705042, -0.0195309 ,  0.05618824, -0.01777991,
  0.01679466,  0.04924609,  0.01946444, -0.01860381,  0.01520276,
  0.01897445,  0.05107498, -0.02334759,  0.01828282, -0.08808766,
  0.02882235, -0.01930556, -0.02068386,  0.02744959,  0.04625075,
  0.02035622,  0.02041421,  0.03172785,  0.02119361,  0.02217874,
 -0.03352227,  0.02113889,  0.0193071 , -0.0559799 ,  0.03594135,
 -0.06702676,  0.01564932,  0.02138522, -0.02333784, -0.02205841,
  0.02153316,  0.01985227,  0.02223904,  0.02839317,  0.10108196,
```

한국어 단어 임베딩

preprocess.py

전처리 코드 - 함수 호출 부분

```
import os
import numpy as np
import pickle
import yaml
from data_process.utils import *
def data_process(config):
    mode = config['mode']
    assert mode in ('term', 'category')
    base_path = config['base_path']
    raw_train_path = os.path.join(base_path, 'raw/train.xml')
    raw_val_path = os.path.join(base_path, 'raw/val.xml')
    raw_test_path = os.path.join(base_path, 'raw/test.xml')
    lowercase = config['lowercase']
    if mode == 'term':
        # 리뷰를 다음 포맷으로 변환 (리뷰__split__split__감정__split__시작__split__끝)
        train_data = parse_sentence_term(raw_train_path, lowercase =lowercase)
        val_data = parse_sentence_term(raw_val_path, lowercase =lowercase)
        test_data = parse_sentence_term(raw_test_path, lowercase =lowercase)
    else:
        train_data = parse_sentence_category(raw_train_path, lowercase =lowercase)
        val_data = parse_sentence_category(raw_val_path, lowercase =lowercase)
        test_data = parse_sentence_category(raw_test_path, lowercase =lowercase)
    # conflict로 라벨링된 데이터를 제거
    remove_list = ['conflict']
    train_data = category_filter(train_data, remove_list)
    val_data = category_filter(val_data, remove_list)
    test_data = category_filter(test_data, remove_list)

    # word2index, index2word dictionary 생성
    word2index, index2word = build_vocab(train_data, max_size =config['max_vocab_size'],
min_freq =config['min_vocab_freq'])
    if not os.path.exists(os.path.join(base_path, 'processed')):
        os.makedirs(os.path.join(base_path, 'processed'))
    if mode == 'term':
        save_term_data(train_data, word2index, os.path.join(base_path, 'processed/train.npz'))
        save_term_data(val_data, word2index, os.path.join(base_path, 'processed/val.npz'))
        save_term_data(test_data, word2index, os.path.join(base_path, 'processed/test.npz'))
    else:
        save_category_data(train_data, word2index, os.path.join(base_path,
'processed/train.npz'))
        save_category_data(val_data, word2index, os.path.join(base_path, 'processed/val.npz'))
        save_category_data(test_data, word2index, os.path.join(base_path, 'processed/test.npz'))
    # glove 단어 임베딩 로드
    glove = load_glove(config['glove_path'], len(index2word), word2index)
    # 감정 사전 로드
    sentiment_matrix = load_sentiment_matrix(config['glove_path'], config['sentiment_path'])
    np.save(os.path.join(base_path, 'processed/glove.npy'), glove)
    np.save(os.path.join(base_path, 'processed/sentiment_matrix.npy'), sentiment_matrix)
    with open(os.path.join(base_path, 'processed/word2index.pickle'), 'wb') as handle:
        pickle.dump(word2index, handle)
    with open(os.path.join(base_path, 'processed/index2word.pickle'), 'wb') as handle:
        pickle.dump(index2word, handle)
    analyze = analyze_term if mode == 'term' else analyze_category
    log = {
```

```

        'vocab_size': len(index2word),
        'oov_size': len(word2index) - len(index2word),
        'train_data': analyze(train_data),
        'val_data': analyze(val_data),
        'test_data': analyze(test_data),
        'num_categories': 3
    }
    if not os.path.exists(os.path.join(base_path, 'log')):
        os.makedirs(os.path.join(base_path, 'log'))
    with open(os.path.join(base_path, 'log/log.yml'), 'w') as handle:
        yaml.safe_dump(log, handle, encoding='utf-8', allow_unicode =True, default_flow_style
=False)

```

utils.py

전처리 코드 - 함수 구현 부분

```

import os
import numpy as np
import random
from xml.etree.ElementTree import parse
from gluonnlp.data import SentencepieceTokenizer
from kobert_transformers import get_tokenizer
from data_process.vocab import Vocab
from src.module.utils.constants import UNK, PAD_INDEX, ASPECT_INDEX
from konlpy.tag import Mecab
import pickle
import re
import json

url = re.compile('<url>.*</url>')
mecab_tokenizer = Mecab().morphs

def check(x):
    return len(x) >=1 and not x.isspace()

# tokenizer 로드 (mecab 사용)
def tokenizer(text):
    tokens = [tok for tok in mecab_tokenizer(url.sub('@URL@', text))]
    return list(filter(check, tokens))

# koBERT 로드
bert_tokenizer = get_tokenizer()

# 리뷰__split__split__감정__split__시작__split__끝 포맷으로 변경
def parse_sentence_term(path, lowercase =False):
    tree = parse(path)
    sentences = tree.getroot()
    data = []
    split_char = '__split__'
    for sentence in sentences:
        text = sentence.find('text')
        if text is None:
            continue

```



```

text = text.text
if lowercase:
    text = text.lower()
aspectTerms = sentence.find('aspectTerms')
if aspectTerms is None:
    continue
for aspectTerm in aspectTerms:
    term = aspectTerm.get('term')
    if lowercase:
        term = term.lower()
    polarity = aspectTerm.get('polarity')
    start = aspectTerm.get('from')
    end = aspectTerm.get('to')
    piece = text + split_char + term + split_char + polarity + split_char + start +
split_char + end
    data.append(piece)
return data

# conflict로 라벨링된 데이터를 제거
def category_filter(data, remove_list):
    remove_set = set(remove_list)
    filtered_data = []
    for text in data:
        if not text.split('__split__')[2] in remove_set:
            filtered_data.append(text)
    return filtered_data

def build_vocab(data, max_size, min_freq):
    if max_size == 'None':
        max_size = None
    vocab = Vocab()
    for piece in data:
        text = piece.split('__split__')[0]
        text = tokenizer(text)
        vocab.add_list(text)
    return vocab.get_vocab(max_size = max_size, min_freq = min_freq)

# 모델 인풋 데이터 포맷으로 변환
def save_term_data(data, word2index, path):
    dirname = os.path.dirname(path)
    if not os.path.exists(dirname):
        os.makedirs(dirname)
    sentence = []
    aspect = []
    label = []
    context = []
    bert_token = []
    bert_segment = []
    td_left = []
    td_right = []
    f = lambda x: word2index[x] if x in word2index else word2index[UNK]
    g = lambda x: list(map(f, tokenizer(x)))
    d = {
        'positive': 0,
        'negative': 1,
        'neutral': 2,
        'conflict': 3
    }
}

```

```

for piece in data:
    text, term, polarity, start, end = piece.split('__split__')
    start, end = int(start), int(end)
    assert text[start: end] == term
    sentence.append(g(text))
    aspect.append(g(term))
    label.append(d[polarity])
    left_part = g(text[:start])
    right_part = g(text[end:])
    context.append(left_part + [ASPECT_INDEX] + right_part)
    bert_sentence = bert_tokenizer.tokenize(text)
    bert_aspect = bert_tokenizer.tokenize(term)
    bert_token.append(bert_tokenizer.convert_tokens_to_ids(['[CLS]'] + bert_sentence +
['[SEP]'] + bert_aspect + ['[SEP]']))
    bert_segment.append([0] * (len(bert_sentence) + 2) + [1] * (len(bert_aspect) + 1))
    td_left.append(g(text[:end]))
    td_right.append(g(text[start:])[::-1])
    assert len(bert_token[-1]) == len(bert_segment[-1])
max_length = lambda x: max([len(y) for y in x])
sentence_max_len = max_length(sentence)
aspect_max_len = max_length(aspect)
context_max_len = max_length(context)
bert_token_max_len = max_length(bert_token)
td_left_max_len = max_length(td_left)
td_right_max_len = max_length(td_right)
num = len(data)
for i in range(num):
    sentence[i].extend([0] * (sentence_max_len - len(sentence[i])))
    aspect[i].extend([0] * (aspect_max_len - len(aspect[i])))
    context[i].extend([0] * (context_max_len - len(context[i])))
    bert_token[i].extend([0] * (bert_token_max_len - len(bert_token[i])))
    bert_segment[i].extend([0] * (bert_token_max_len - len(bert_segment[i])))
    td_left[i].extend([0] * (td_left_max_len - len(td_left[i])))
    td_right[i].extend([0] * (td_right_max_len - len(td_right[i])))
sentence = np.asarray(sentence, dtype = np.int32)
aspect = np.asarray(aspect, dtype = np.int32)
label = np.asarray(label, dtype = np.int32)
context = np.asarray(context, dtype = np.int32)
bert_token = np.asarray(bert_token, dtype = np.int32)
bert_segment = np.asarray(bert_segment, dtype = np.int32)
td_left = np.asarray(td_left, dtype = np.int32)
td_right = np.asarray(td_right, dtype = np.int32)
np.savez(path, sentence=sentence, aspect =aspect, label =label, context =context,
bert_token =bert_token, bert_segment =bert_segment,
        td_left=td_left, td_right =td_right)

# 로그 파일 생성
def analyze_term(data):
    num = len(data)
    sentence_lens = []
    aspect_lens = []
    log = {'total': num}
    for piece in data:
        text, term, polarity, _, _ = piece.split('__split__')
        sentence_lens.append(len(tokenizer(text)))
        aspect_lens.append(len(tokenizer(term)))
        if not polarity in log:

```

```

        log[polarity] =0
        log[polarity] +=1
    log['sentence_max_len'] = max(sentence_lens)
    log['sentence_avg_len'] = sum(sentence_lens) /len(sentence_lens)
    log['aspect_max_len'] = max(aspect_lens)
    log['aspect_avg_len'] = sum(aspect_lens) /len(aspect_lens)
    return log

# glove 단어 임베딩을 로드해서 데이터셋에 있는 단어들을 맵핑
def load_glove(path, vocab_size, word2index):
    if not os.path.isfile(path):
        raise IOError('Not a file', path)
    glove = np.random.uniform(-0.01, 0.01, [vocab_size, 300])
    with open(path, 'rb') as f:
        content = pickle.load(f)
    for word in content:
        if word in word2index:
            glove[word2index[word]] = np.array(list(map(float, content[word])))
    glove[PAD_INDEX, :] =0
    return glove

# 감성 사전을 로드해서 glove 단어 임베딩으로 맵핑
def load_sentiment_matrix(glove_path, sentiment_path):
    sentiment_matrix = np.zeros((3, 300), dtype =np.float32)
    sd = json.load(open(sentiment_path, 'r', encoding ='utf-8'))
    sd['positive'] = set(sd['positive'])
    sd['negative'] = set(sd['negative'])
    sd['neutral'] = set(sd['neutral'])
    with open(glove_path, 'rb') as f:
        content = pickle.load(f)
    for word in content:
        vec = np.array(list(map(float, content[word])))
        if word in sd['positive']:
            sentiment_matrix[0] += vec
        elif word in sd['negative']:
            sentiment_matrix[1] += vec
        elif word in sd['neutral']:
            sentiment_matrix[2] += vec
    sentiment_matrix -= sentiment_matrix.mean()
    sentiment_matrix = sentiment_matrix / sentiment_matrix.std() * np.sqrt(2.0 / (300.0 +3.0))
    return sentiment_matrix

```

train.py

모델 훈련 코드

```

import torch
from torch import nn
from torch import optim
from train import make_aspect_term_model, make_aspect_category_model
from train.make_data import make_term_data, make_category_data
from train.make_optimizer import make_optimizer
from train.eval import eval
import os
import time
import pickle

```

```

from src.module.utils.loss import CapsuleLoss

def train(config):
    mode = config['mode']
    if mode == 'term':
        model = make_aspect_term_model.make_model(config)
        train_loader, val_loader = make_term_data(config)
    else:
        model = make_aspect_category_model.make_model(config)
        train_loader, val_loader = make_category_data(config)

    model = model.cuda() # 모델 파라미터를 GPU로 로드
    base_path = config['base_path'] # ./data/MAMS-ATSA
    model_path = os.path.join(base_path, 'checkpoints/%s.pth' % config['aspect_' + mode
+ '_model']['type']) # 모델 저장 경로

    if not os.path.exists(os.path.dirname(model_path)): # 디렉토리가 없는 경우 생성
        os.makedirs(os.path.dirname(model_path))
    with open(os.path.join(base_path, 'processed/index2word.pickle'), 'rb') as handle: # 전처리
과정에서 저장한 index2word.pickle 파일 로드
        index2word = pickle.load(handle)

    criterion = CapsuleLoss() # loss function 생성 (/src/module/utils/loss.py)
    optimizer = make_optimizer(config, model) # optimizer 생성

    # 훈련시 사용하기 위한 변수
    max_val_accuracy = 0
    min_val_loss = 100
    global_step = 0
    config = config['aspect_' + mode + '_model'][config['aspect_' + mode + '_model']['type']] #
recurrent_capsnet

    for epoch in range(config['num_epochs']): # 20 epoch
        total_loss = 0
        total_samples = 0
        correct_samples = 0
        start = time.time()

        for i, data in enumerate(train_loader): # batch_size만큼 자동으로 데이터를 뽑아줌
            global_step += 1
            model.train() # 모델을 훈련모드로 변경 (dropout 같은 훈련을 위한 설정을 활성화)
            input0, input1, label = data # context, aspect, label (ABSADataset __getitem__에
서 정의)
            input0, input1, label = input0.cuda(), input1.cuda(), label.cuda() # 데이터를 gpu로
올림

            optimizer.zero_grad() # gradient 계산을 위해 0으로 초기화
            logit = model(input0, input1) # 모델에 context와 aspect를 입력 데이터로 전달 ->
forward 함수가 실행됨
            loss = criterion(logit, label) # 예측값과 실제값을 비교해 loss 계산
            batch_size = input0.size(0)
            total_loss += batch_size * loss.item() # 전체 loss 계산
            total_samples += batch_size # 훈련을 실행한 데이터 개수 카운트
            pred = logit.argmax(dim = 1) # 예측값에 argmax를 취해 감정 예측 (긍정 / 부정)
            correct_samples += (label == pred).long().sum().item() # 맞은 개수를 카운트
            loss.backward() # backward 진행
            torch.nn.utils.clip_grad_norm_(model.parameters(), 5.0) # gradient clipping
(gradients vanishing, exploding을 방지하기 위해 사용)

```

검증

```
optimizer.step() # backpropagation에서 수집된 변화량으로 매개변수 조정
if i % 10 == 0 and i > 0: # 10번의 iteration마다 total_loss, total_accuracy 계산 및

train_loss = total_loss / total_samples
train_accuracy = correct_samples / total_samples
total_loss = 0
total_samples = 0
correct_samples = 0
val_accuracy, val_loss = eval(model, val_loader, criterion) # /train/eval.py 실행
print('[epoch %2d] [step %3d] train_loss: %.4f train_acc: %.4f val_loss: %.4f
val_acc: %.4f'

        % (epoch, i, train_loss, train_accuracy, val_loss, val_accuracy))
if val_accuracy > max_val_accuracy:
    max_val_accuracy = val_accuracy
    # torch.save(aspect_term_model.state_dict(), model_path)
if val_loss < min_val_loss: # validation loss가 최소가 될 때마다 모델 갱신
    min_val_loss = val_loss
    if epoch > 0:
        torch.save(model.state_dict(), model_path)
end = time.time()
print('time: %.4fs' % (end - start))
print('max_val_accuracy:', max_val_accuracy)
```

```
seominseok@mobicom-SYS-7049GP-TRT:~/MAMS-for-ABSA$ python3 train.py
/usr/lib/python3/dist-packages/requests/_init_.py:89: RequestsDependencyWarning: urllib3 (1.26.9) or chardet (3.0.4) doesn't match a supported version!
  warnings.warn("urllib3 ({}), or chardet ({}), doesn't match a supported version")
[epoch 0] [step 10] train_loss: 0.3384 train_acc: 0.5124 val_loss: 0.3021 val_acc: 0.7585
time: 0.3588s
[epoch 1] [step 10] train_loss: 0.2910 train_acc: 0.7130 val_loss: 0.2562 val_acc: 0.7585
time: 0.4003s
[epoch 2] [step 10] train_loss: 0.2660 train_acc: 0.7423 val_loss: 0.2314 val_acc: 0.7585
time: 0.3923s
[epoch 3] [step 10] train_loss: 0.2420 train_acc: 0.7423 val_loss: 0.2130 val_acc: 0.7585
time: 0.4127s
[epoch 4] [step 10] train_loss: 0.2255 train_acc: 0.7423 val_loss: 0.2105 val_acc: 0.7585
time: 0.4621s
[epoch 5] [step 10] train_loss: 0.2160 train_acc: 0.7423 val_loss: 0.2160 val_acc: 0.7585
time: 0.4164s
[epoch 6] [step 10] train_loss: 0.2118 train_acc: 0.7423 val_loss: 0.2227 val_acc: 0.7585
time: 0.4156s
[epoch 7] [step 10] train_loss: 0.2086 train_acc: 0.7423 val_loss: 0.2135 val_acc: 0.7585
time: 0.4280s
[epoch 8] [step 10] train_loss: 0.2038 train_acc: 0.7423 val_loss: 0.2050 val_acc: 0.7517
time: 0.4633s
[epoch 9] [step 10] train_loss: 0.1967 train_acc: 0.7452 val_loss: 0.2026 val_acc: 0.7585
time: 0.4690s
[epoch 10] [step 10] train_loss: 0.1924 train_acc: 0.7570 val_loss: 0.2106 val_acc: 0.7517
time: 0.4110s
[epoch 11] [step 10] train_loss: 0.1822 train_acc: 0.7555 val_loss: 0.1968 val_acc: 0.7653
time: 0.4819s
[epoch 12] [step 10] train_loss: 0.1742 train_acc: 0.7628 val_loss: 0.1984 val_acc: 0.7653
time: 0.4234s
[epoch 13] [step 10] train_loss: 0.1700 train_acc: 0.7687 val_loss: 0.2316 val_acc: 0.7585
time: 0.4131s
[epoch 14] [step 10] train_loss: 0.1660 train_acc: 0.7657 val_loss: 0.1901 val_acc: 0.7823
time: 0.4626s
[epoch 15] [step 10] train_loss: 0.1550 train_acc: 0.7804 val_loss: 0.2105 val_acc: 0.7211
time: 0.4078s
[epoch 16] [step 10] train_loss: 0.1389 train_acc: 0.8155 val_loss: 0.2207 val_acc: 0.7211
time: 0.4259s
[epoch 17] [step 10] train_loss: 0.1349 train_acc: 0.8067 val_loss: 0.2269 val_acc: 0.7041
time: 0.4036s
[epoch 18] [step 10] train_loss: 0.1188 train_acc: 0.8360 val_loss: 0.2198 val_acc: 0.7415
time: 0.4122s
[epoch 19] [step 10] train_loss: 0.1116 train_acc: 0.8492 val_loss: 0.2197 val_acc: 0.7551
time: 0.4175s
max_val_accuracy: 0.782312925170068
```

Validation accuracy: 78.23%

모두의 말뭉치 데이터셋을 가지고 모델을 실행한 결과
데이터셋 개수가 적어 train, validation만 진행하였다. (훈련 데이터셋: 686개, 검증 데이터셋: 294개)

2. 프로젝트 수행을 위해 적용된 추진전략, 수행 방법의 결과를 작성하고, 만일 적용과정에서 문제점이 도출되었다면 그 문제를 분석하고 해결방안을 기술하시오.

- 문제해결을 위해 적용한 방법(또는 기법) 결과, (문제점, 해결방안)

[문제점]

I. 진행 과정에서 오류 발생

[해결방안]

II. 필요한 라이브러리나 패키지 등의 버전 정보를 확인하고 해당 오류의 정확한 원인을 파악

- 팀원의 책임 및 역할 수행에 대한 결과, (문제점, 해결방안)

[문제점]

I. 자연어 처리 분야에 대한 지식 부족으로 인한 진행 차질

II. pytorch로 설계된 모델에 대한 이해 미숙

[해결방안]

I. 자연어 처리 전공 교수님과의 상담을 통해 확립된 방향성과 제시받은 아이디어를 토대로 진행

II. 해당 모델에 대한 논문 참조 및 코드 해석을 통한 동작 방식 이해

캡스톤 디자인 | 중간보고서 채점표

평가도구	평 가 항 목	평 가 점 수				
		1	2	3	4	5
중간 보고서 및 실행 결과	1. 요구사항 정의서(기능, 성능, 인터페이스 등)가 구체적으로 작성되었는가?					
	2. 요구분석, 설계 산출물(모델, 프로토타입 등)의 내용이 충실한가?					
	3. 설계 및 구현 문제를 위해 적용한 이론, 문제해결 방법이 제시되었으며 그 적용이 적합한가?					
	4. 구현된 소프트웨어(또는 이와 동등한 하드웨어 시스템)가 버그 없이 실행되었는가?					
	5. 구현된 소프트웨어(또는 이와 동등한 하드웨어 시스템)의 성능 요구사항은 충족되었는가?					
도구활용	6. 설계 및 구현을 위해 도구가 적절히 활용되었는가?					
	7. 도구의 활용수준(능숙도)은 프로젝트 수행에 적합한가?					
팀원의 업무 및 역할	8. 팀원의 업무분담에 따른 역할 및 협력이 충실히 이루어졌는가? (평가자에 의한 질의)					
	9. 프로젝트 중간 진척상황에 대해 팀원이 충분히 인지하고 있는가?(평가자에 의한 질의)					
합계						
*검토 의견(최종완료 때까지 보완해야할 점에 대해 작성 요망)						
심사위원(소속):		(이름)			(인)	