TAGFLIX

2nd Implementation Demo Presentation

안성관 이성민 이연우 홍인표

지도교수 : 하 영 국

TAGFLIX

INDEX

Software Design Specification

Table of Contents

개발 동기 및 배경

· 시청할 영화를 결정하는 데에는 장르를 보고, 줄거리를 읽고, 예고편을 시청하고, 평점과 관람평을 보는 등 소비자로 하여금 시간과 노력 등 자원의 투자가 요구되어왔다.

· 학습을 시켜 영화에 적절한 태그를 부여할 수 있다면, 사용자는 보다 적은 노력으로 더 취향에 맞는 영화를 선택할 수 있다.

· 기존 무비렌즈 같은 서비스는 투표를 기반으로 태그를 부여했는데, 이는 충분한 투표수가 확보되지 않는 영화의 태그분류가 잘 되지 않는 결점이 존재한다.

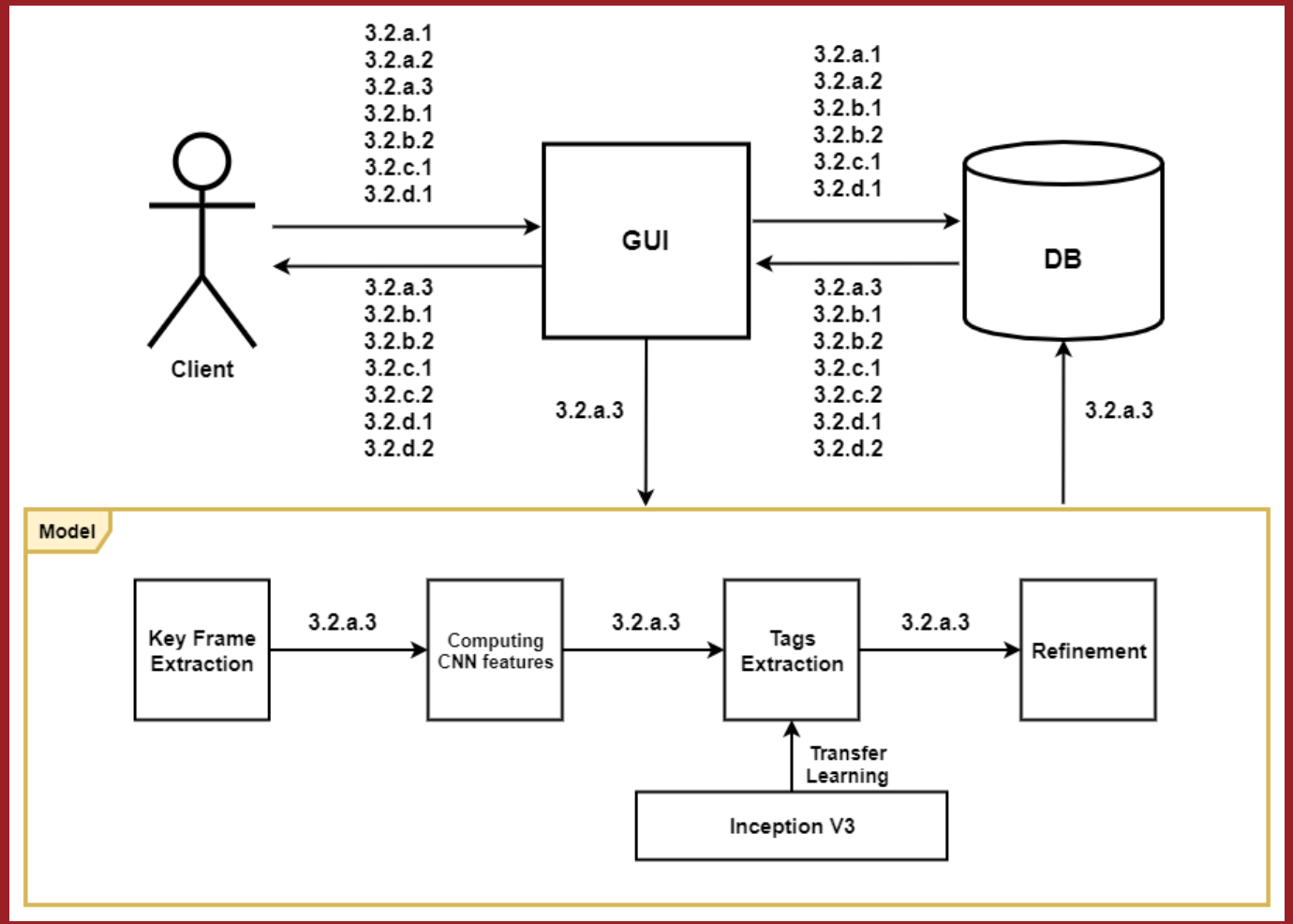
· 자동화를 통하면 사용자의 설명에 의존하지 않고 태그 서비스를 제공할 수 있다.

개발 목표

·접근하기 쉬운 GUI 환경을 통해 짧은 영화 예고편을 입력받아 해당 영화 전체에 대한 태그를 자동적으로 생성하는 프로그램을 제작하고, 사용자가 선택한 태그를 통해 영화를 추천하는 기능과 선택한 영화와 유사한 영화를 추천하는 기능을 추가적으로 구현하여, 사용자가 보다 편리하고 정확하게 영화를 선택할 수 있도록 한다.

시스템 설계

아키텍처 다이어그램



시스템 설계

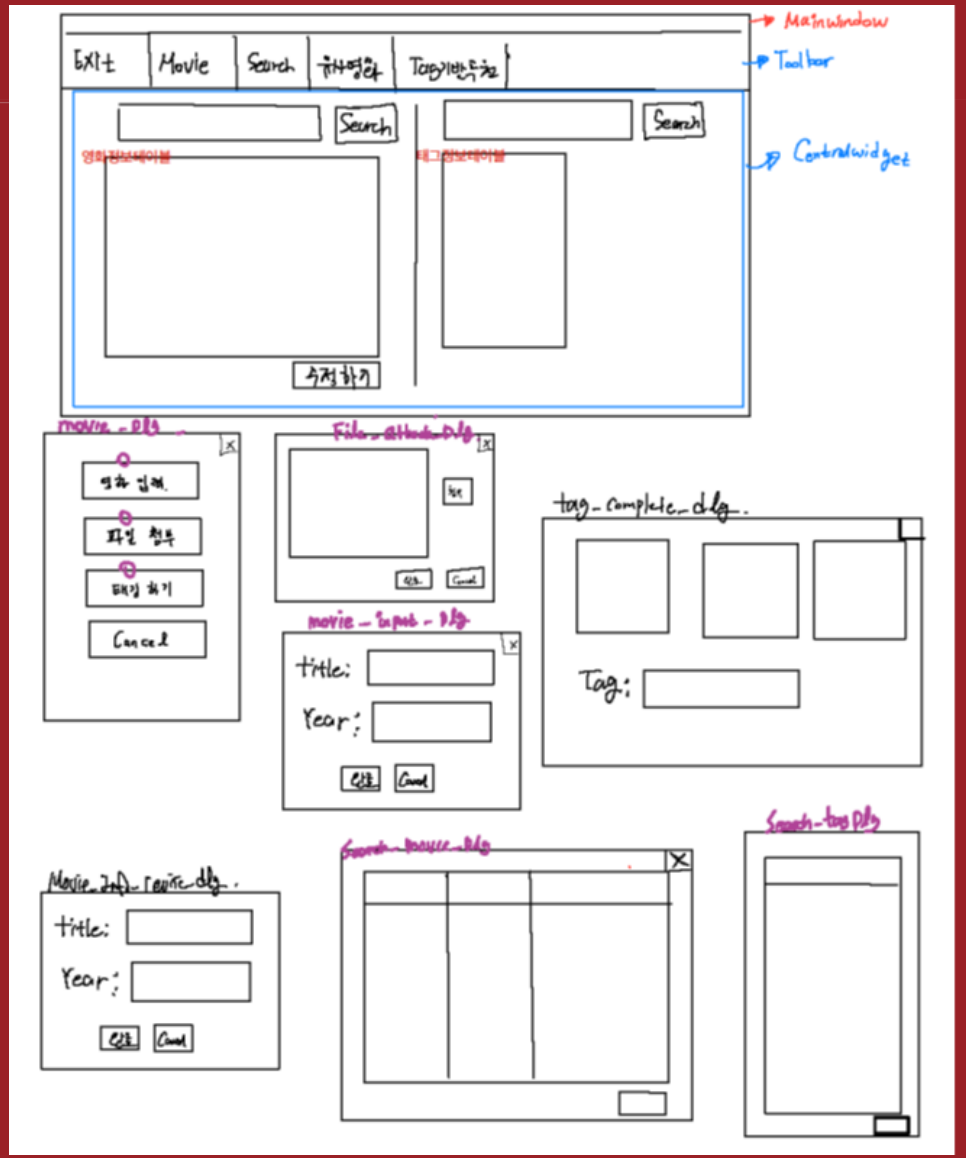
· 사용자는 GUI 환경을 통해 영화 정보와 영상정보를 입력한다.

· GUI는 Model로 입력 받은 동영상 데이터를 전달한다.

· GUI로 부터 동영상 데이터를 받은 Model은 Key Frame Extraction 모듈을 통해 동영상에서 키 프레임을 추출하고, Computing CNN features 모듈을 통해 부적합한 프레임을 선별하고, 사전에 InceptionV3로 부터 전이학습 된 Tag Extraction 모듈을 통해 키 프레임 당 태그를 3개씩 추출하여, 확률 분포에 기반해 추출된 태그를 정제, 최종 태그들을 결정한다. 결정된 태그는 영화 정보와 함께 DB로 저장이 된다.

· 이후 DB가 GUI와 상호작용을 위해 제공하는 함수들을 통해, 영화 정보의 수정, 영화와 태그 검색, 태그 기반 추천, 유사한 영화 추천 등의 기능을 사용할 수 있다.

UI 설계


Exit
Movie
Search
Tag관리
Search
Search
수정하기
Mainwindow
Toolbar
Centralwidget
Cancel
title:
Year:
Tag:

UI 설계

· MainWindow에 전체 목록을 보여주는 영화정보 테이블과 태그 테이블을 두고, Toolbar의 버튼들을 이용해서 프로그램의 기능은 버튼을 클릭할 때 발생하는 Slot에 작성하는 방식으로 연결한다.

· 버튼이나 Toolbar의 Icon을 클릭 하는 것에 대해 Slot을 이용해서 대화창을 하나씩 exec시키고, 해당 대화창에서 사용자의 작업이 끝나면 cancel 하는 방식으로 close한다.

· Table을 다이나믹하게 업데이트 하기 위해 MV아키텍쳐로 작성하였다.

· 사용자가 대화창에서 입력한 input들은 태깅 작업 까지 모두 끝내 태그 값을 얻은 후에 DB에 insert 되도록 DB와 connect하고 SQL문을 전달하도록 하였고, 하나의 영화정보(title,year,tag들)가 모두 얻어 지면 DB에 해당 영화의 정보가 Insert되도록 하였으며 , MainWindow의 table 또한 새로 추가된 데이터를 다이나믹하게 보여 줄 수 있도록 , 새로운 영화정보를 추가하는 작업을 모아놓은 movie_dlg 대화창을 최종적으로 close하는 순간에 Layoutchanged signal을 emit하여 Main의 영화정보 테이블의 뷰가 업데이트되도록 하였다.

· 사용자는 검색창에 문자열을 입력하고 테이블의 항목을 선택하는 것이 가능하며, 검색창에 입력한 문자열이 포함된 검색결과 테이블을 따로 결과 대화창을 통해 볼 수 있다. 검색결과 대화창은 MainWindow에 있는 Table과 다루고 있는 전체 Data가 다르기 때문에 서로 다른 Model로부터 View를 보여주고 있지만, 선택된 항에 관한 정보를 공유하여 검색결과 대화창에서 선택한 항목이 MainWindow에서도 선택되도록 하였다.

· 사용자는 검색결과 대화창의 테이블 또는 MainWindow에 있는 테이블에서 항목을 선택 할 수 있으며 이렇게 선택된 항목을 이용하여 영화 추천을 받을 수 있다.

인터페이스 설계

-소프트웨어 인터페이스
1.영상 이미지 분석 라이브러리
- Name: openCV
-Version number: 4.1.1
-Source: opencv.org

2. pre-trained 딥 러닝 모델
- Name : Inception
-Version number : 3
-Source: https://arxiv.org/abs/1512.00567

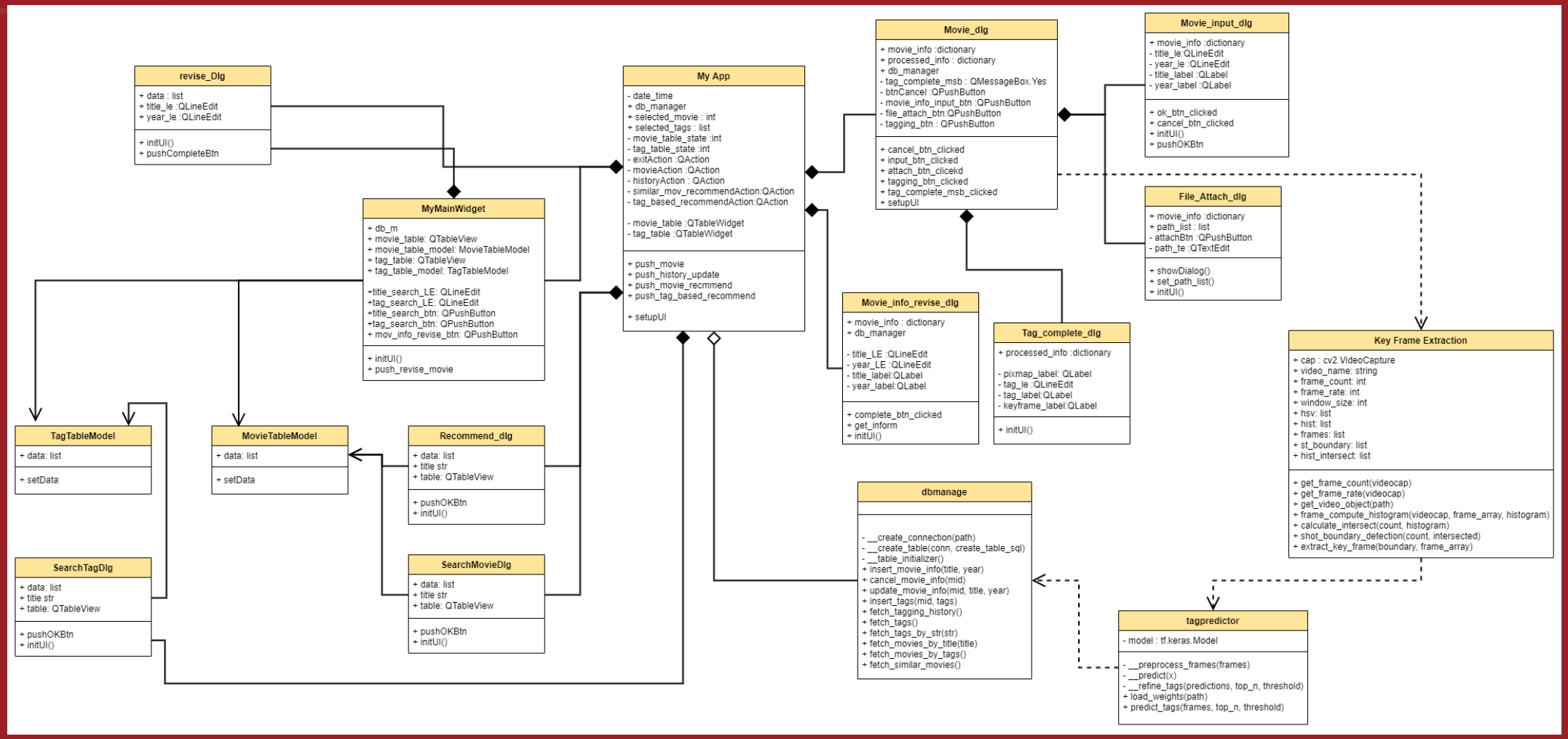
3. DBMS
- Name: SQLite
-Version number : 3.19.2
-Source: sqlite.org

4. OS
-Name: Windows
-Version number : 10
-Source: MicroSoft

-하드웨어 인터페이스
특별한 H/W requirement가 없다.

-통신 인터페이스
본 시스템에서는 네트워크가 관여하지 않으므로 해당사항이 없다.

기타 모듈 설명


revise_Dlg
+ data : list
+ title_le :QLineEdit
+ year_le :QLineEdit
+ initUI()
+ pushCompleteBtn
My App
- date_time
+ db_manager
+ selected_movie : int
+ selected_tags : list
- movie_table_state :int
- tag_table_state :int
- exitAction :QAction
- movieAction :QAction
- historyAction : QAction
- similar_mov_recommendAction:QAction
- tag_based_recommendAction:QAction
- movie_table :QTableWidget
- tag_table :QTableWidget
+ push_movie
+ push_history_update
+ push_movie_recmmend
+ push_tag_based_recommend
+ setupUI
Movie_dlg
+ movie_info :dictionary
+ processed_info : dictionary
+ db_manager
- tag_complete_msb : QMessageBox.Yes
- btnCancel :QPushButton
- movie_info_input_btn :QPushButton
- file_attach_btn:QPushButton
- tagging_btn : QPushButton
+ cancel_btn_clicked
+ input_btn_clicked
+ attach_btn_clicekd
+ tagging_btn_clicked
+ tag_complete_msb_clicked
+ setupUI
Movie_input_dlg
+ movie_info :dictionary
- title_le:QLineEdit
- year_le :QLineEdit
- title_label :QLabel
- year_label :QLabel
+ ok_btn_clicked
+ cancel_btn_clicked
+ initUI()
+ pushOKBtn
File_Attach_dlg
+ movie_info :dictionary
+ path_list : list
- attachBtn :QPushButton
- path_te :QTextEdit
+ showDialog()
+ set_path_list()
+ initUI()
MyMainWidget
+ db_m
+ movie_table: QTableView
+ movie_table_model: MovieTableModel
+ tag_table: QTableView
+ tag_table_model: TagTableModel
+title_search_LE: QLineEdit
+tag_search_LE: QLineEdit
+title_search_btn: QPushButton
+tag_search_btn: QPushButton
+ mov_info_revise_btn: QPushButton
+ initUI()
+ push_revise_movie
Movie_info_revise_dlg
+ movie_info : dictionary
+ db_manager
- title_LE :QLineEdit
- year_LE :QLineEdit
- title_label:QLabel
- year_label:QLabel
+ complete_btn_clicked
+ get_inform
+ initUI()
Tag_complete_dlg
+ processed_info :dictionary
- pixmap_label: QLabel
- tag_le :QLineEdit
- tag_label:QLabel
- keyframe_label:QLabel
+ initUI()
Key Frame Extraction
+ cap : cv2.VideoCapture
+ video_name: string
+ frame_count: int
+ frame_rate: int
+ window_size: int
+ hsv: list
+ hist: list
+ frames: list
+ st_boundary: list
+ hist_intersect: list
+ get_frame_count(videocap)
+ get_frame_rate(videocap)
+ get_video_object(path)
+ frame_compute_histogram(videocap, frame_array, histogram)
+ calculate_intersect(count, histogram)
+ shot_boundary_detection(count, intersected)
+ extract_key_frame(boundary, frame_array)
TagTableModel
+ data: list
+ setData
MovieTableModel
+ data: list
+ setData
Recommend_dlg
+ data: list
+ title str
+ table: QTableView
+ pushOKBtn
+ initUI()
SearchTagDlg
+ data: list
+ title str
+ table: QTableView
+ pushOKBtn
+ initUI()
SearchMovieDlg
+ data: list
+ title str
+ table: QTableView
+ pushOKBtn
+ initUI()
dbmanage
- __create_connection(path)
- __create_table(conn, create_table_sql)
- __table_initializer()
+ insert_movie_info(title, year)
+ cancel_movie_info(mid)
+ update_movie_info(mid, title, year)
+ insert_tags(mid, tags)
+ fetch_tagging_history()
+ fetch_tags()
+ fetch_tags_by_str(str)
+ fetch_movies_by_title(title)
+ fetch_movies_by_tags()
+ fetch_similar_movies()
tagpredictor
- model : tf.keras.Model
- __preprocess_frames(frames)
- __predict(x)
- __refine_tags(predictions, top_n, threshold)
+ load_weights(path)
+ predict_tags(frames, top_n, threshold)

기타 모듈 설명

KeyFrameExtraction

입력받은 동영상의 Key Frame들을 추출하는 모듈.
우선, 영상을 프레임 단위로 읽어들인 다음, 각 프레임을 HSV Color로 변환하고, Histogram을 계산한다.
HSV Color를 사용하는 이유는 HSV Color가 light variations에 강건할 뿐만 아니라, 사람이 인식하기에도 더 편하기 때문이다.
그 후, shot boundary를 찾기 위해 연속된 프레임에서 HSV 히스토그램의 교차점을 계산한다.
교차점 계산은 두 개의 HSV 히스토그램 사이의 유사성의 척도를 제공하기 때문이다.

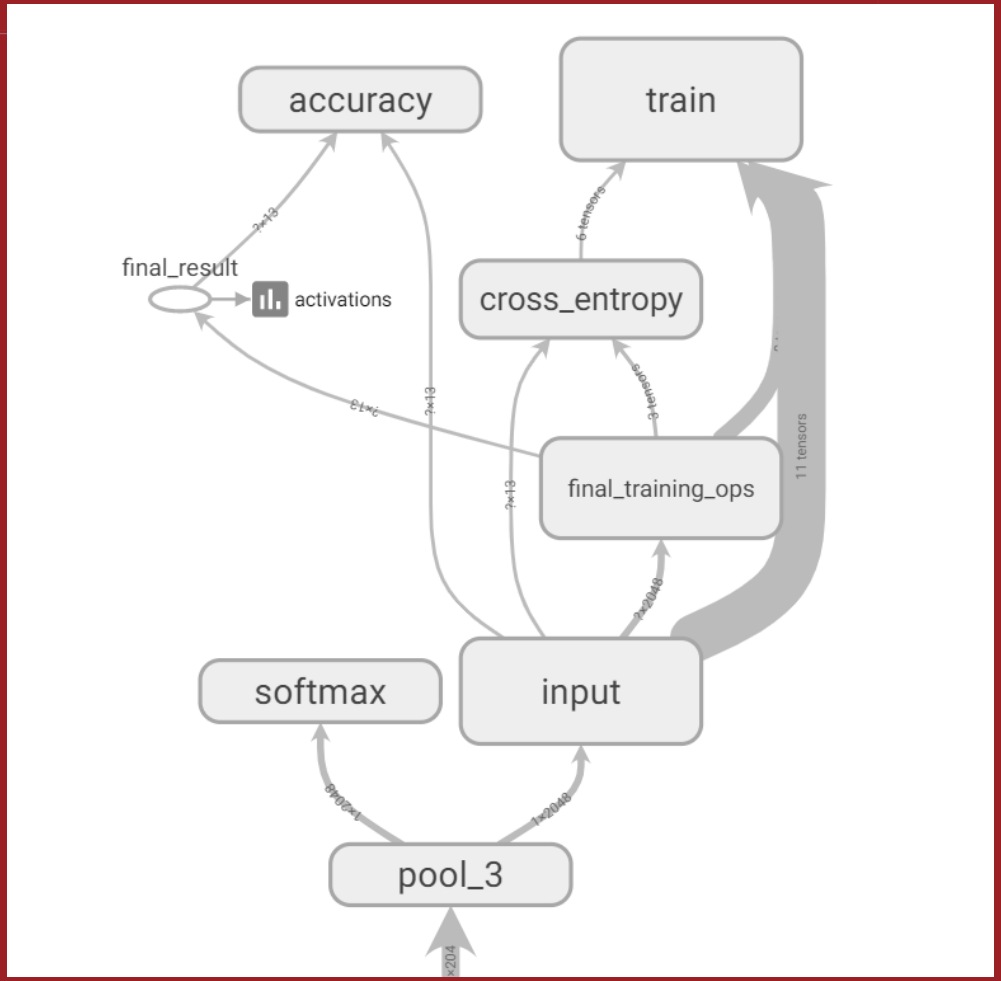
$$S(i, i+1) = m_1$$
$$S(i, i+1) \leq \alpha m_2$$
$$\mu \geq \alpha m_1$$

그 다음, 계산된 교차값이 있는 배열에 현재 프레임 i를 가운데에 두는 sliding window를 사용한다.
window 내에서 가장 작은 교차값을 m1, 2번째 최소값을 m2, 임의의 값 **α**를 1.1, window 내의 평균을 **μ**로 정하였으며, 아래의 등식을 모두 만족한다면 shot boundary으로 판단하도록 하였다.
이를 통해 shot boundary를 구하였으며, boundary의 중간에 있는 Frame을 Key Frame으로 선정하여 추출하였다.

기타 모듈 설명

Inception V3 Transfer Learning

사전 학습된 모델인 Inception-V3를 사용하여, final layer를 수정하는 transfer learning을 통해 training 이미지들을 태그 어휘와 맞도록 retraining 하는 모듈.
overfitting을 방지하기 위해 Inception 모델을 통해 나온 값인 bottleneck value에 Dropout을 50%로 하였다.
아래 그래프에서 pool_3 layer 이후에 적용되었다. 그리고, 실제 Tag 확률 분포와 차이를 보기 위해 cross_entropy를 사용하였다.
수정된 final layer의 그래프는 다음과 같다.



기타 모듈 설명

Tags Extraction

앞서 transfer learning을 통해 생성된 그래프를 이용하여 키 프레임 이미지의 태그를 추출하는 모듈.
저장된 그래프를 불러오고, 그래프의 최종 결과 tensor를 가져와 거기에 key frame을 넣어 실행시키면 각 frame에 적합한 태그 & 확률이 결과로 나온다.

기타 모듈 설명

Computation of CNN features

태그 추출에 사용되는 영상을 영화 예고편으로 가정하고 있기에, 영화 예고편에 자주 등장하는 암전된 화면, 제작사 로고 화면, 영화 제목 화면 등, 정보량이 낮은 화면들을 키 프레임 목록에서 제외시킬 필요가 있다.
이를 위해 정보 엔트로피를 이용하였다.
우선 키 프레임을 YCbCr 색 공간 내로 변환하고 각 luminance/chrominance 채널별로 히스토그램을 생성한다.
히스토그램을 이용해 채널의 여러 값들의 확률을 구한 뒤 이를 이용해 정보 엔트로피를 구하며, 이때 식은 다음과 같다.

$$-\sum_{i=0}^{n-1} p_i \log_2 p_i$$

여기서 n은 gray level들의 수, pi는 픽셀이 gray level i를 가질 확률을 나타낸다.
세 채널의 엔트로피의 합이 3.0이 되지 않는 프레임들을 목록에서 제거한다.
이렇게 추려진 키 프레임들은 별도로 생성된 폴더에 JPEG 파일 형식으로 저장되며, 이후 태그 추출 과정에서 다시 로드 된다.

기타 모듈 설명

Refinement

모델을 거쳐 출력된, 키 프레임들의 태그별 확률 분포들을 이용해 최종 영화 태그를 결정하는 과정이다.
먼저 키 프레임별로 확률이 가장 높은 태그들을 3개씩 선택한다. 이들 태그들을 모아 각 태그별로 다음의 식을 이용해 가중치를 구한다.
여기서 ni는 모아진 태그들 내에서 i번째 태그의 등장 횟수, N은 모아진 태그들의 전체 수, 그리고 $\sum_{j=0}^{N} P_{ij}$ 는 i번째 태그의 모든 등장에서의 확률들의 합이다.

$$W_i = \frac{n_i}{N} \sum_{j=0}^{N} P_{ij}$$

태그별로 가중치가 구해지면 이들에 대해 최소-최대 정규화를 거친 후, 최종적으로 특정 임계값보다 높은 가중치를 갖는 태그만을 영화의 태그들로 선정한다.
선정된 태그들은 영화 정보와 함께 DB 내에 저장된다.

기타 모듈 설명

DB

영화 정보와 함께 추출된 태그가 저장되는 곳으로서 SQLite로 구현되었다.
저장되는 정보는 제목, 연도, 태그 목록의 형식을 취한다.
UI 클래스와의 상호작용을 위해 여러 함수들을 제공하는데, 이러한 함수들은 영화 정보의 추가와 갱신, 태그명 검색, 영화 제목을 이용한 영화 검색, 주어진 태그들을 태그 목록에 포함하는 영화 검색, 주어진 영화와 유사한 태그 목록을 갖는 영화 검색과 같은 기능들을 제공한다.

프로젝트 성과에 대한 기대 효과

영화의 태그 분류를 자동화함으로써 사용자로 하여금 시청할 영화를 결정하는 데에 들어가는 시간과 노력 등의 자원을 획기적으로 줄이고, 보다 취향에 맞는 영화를 선택하는 데 도움을 준다.

활용 가능성 및 확장 가능성

넷플릭스, 왓챠, 유튜브 등 동영상 스트리밍 서비스에 대한 수요가 날로 증가하고 있다.
이들 서비스는 태그 기능을 지원하지 않거나, 투표를 기반으로 태그를 부여하고 있는데, 충분한 투표수가 확보되지 않는 콘텐츠의 태그 분류가 미흡한 문제가 있다.
이들 플랫폼과의 연계를 통한 서비스의 활용 가능성이 충분하다.
더불어 현재는 영화를 기반으로 한 태깅만 진행하였으나 영화 이외의 동영상 소스를 학습할 수 있다면, 다양한 동영상에 대해 태그 기능을 확장할 수 있을 것이다.

시스템 시험 항목 결과

#	System Test Case	P/F
1.1.1.a	영화 기본정보 입력 버튼을 누른다.	P
1.1.2.a	영화 정보인 영화 제목, 제작 년도를 입력받는다.	P
1.1.2.b	영화 제목으로 입력받은 데이터 값을 확인한다.	P
1.1.2.c	제작년도로 입력받은 데이터 값을 확인한다.	P
1.2.1.a	사용자가 '파일 첨부'를 누르면 파일 위치 선택 창이 뜬다.	P
1.2.1.b	사용자가 파일을 첨부한다.(동영상 확장자가 아닌 확장자)	P
1.2.1.c	사용자가 파일을 첨부한다.(0개 또는 2개 이상의 파일을 첨부) → 사용자는 2개 이상의 선택 자체가 불가능하다. 요구사항 명세서와같이 2개를 첨부해서 에러메시지가 뜨지 않고 애초에 하나씩만 첨부가 가능하다	P
1.3.1.a	입력받은 영상의 shot boundary를 찾는다.	P
1.3.1.b	shot boundary의 가운데에 위치한 프레임들을 키 프레임으로 추출한다.	P
1.3.1.c	특정 임계값 이하의 키 프레임들은 특징 계산을 위한 후보에서 탈락시킨다.	P
1.3.1.d	태깅이 끝난 후 UI의 출력 화면에서 프레임 3개가 나온다.	P
1.3.2.a	키 프레임의 크기 조정	P
1.3.3.a	전처리된 키 프레임들이 CNN 모델에 입력된다.	P
1.3.4.a	각 키 프레임에 대해 계산된 태그 값 중에서 가장 score가 높은 태그 3개씩을 선택한다.	P

시스템 시험 항목 결과

1.3.5.a	추출된 태그들의 확률에 대해, 추출된 태그들 내의 등장 빈도에 따라 가중치를 부여한다.	P
1.3.5.b	계산된 확률들을 정규화하고, 특정 임계값 이하의 확률을 갖는 태그들은 최종 태그 목록에서 누락시킨다.	P
1.3.6.a	시스템은 최종 태그 목록을 DB로 전송해야 한다.	P
1.3.6.b	시스템은 최종 태그 목록을 UI의 출력 화면에 나타낸다.	P
2.1.1.a	메인 화면에 (현재 DB에 저장되어 있는 영화 목록을 보여주는)테이블에서, 원하는 영화를 하나 선택한다.	P
2.1.2.a	툴바에 있는 검색하기 아이콘을 눌렀을 때 뜨는 검색창에서 문자열을 입력한다.	P
2.1.2.b	검색으로 새롭게 뜬 메인메뉴의 테이블에서, 원하는 영화제목이 있는 열을 하나 선택한다.	P
2.1.3.a	영화 목록에서 영화를 이미 하나 선택한 상태에서 영화 목록에서 다른 영화를 선택한다.	P
2.1.3.b	영화 목록에서 여러개의 영화를 선택 → 하나의 영화만 선택돼야 한다	P
2.1.3.c	영화목록에서 영화를 하나 선택한 상태에서, 검색결과로 나온 영화목록의 영화를 선택한다.	P
2.1.3.d	검색결과 목록에서 영화를 선택 후, History버튼을 눌러서 나오는 영화 목록에서 다시 영화를 선택 한다.	F
2.2.1.a	영화 목록 table에서 영화 하나를 선택한 상태로, 수정하기 버튼을 클릭한다.	P
2.2.1.b	메인화면의 테이블에서 영화가 선택되어 있지 않은 상태에서, 사용자가 수정하기 버튼을 클릭한다.	P
2.3.1.a	해당 영화의 기존정보(영화제목, 영화년도)가 입력되어 있는 Edit창에서 사용자는 해당 정보를 수정한다.	P
2.3.2.a	수정 정보 창에서 사용자가 수정완료 버튼을 누른다.	P

시스템 시험 항목 결과

2.3.3.a	사용자가 메인화면의 History버튼을 누른다.	F
3.1.1.a	3개의 태그가 이미 선택된 전체 태그 목록에서 5개의 태그들을 임의로 선택한다.	P
3.1.1.b	전체 태그 목록에서 태그를 하나 선택한 후, 검색창에 해당 태그명의 일부 문자열을 입력한다.	P
3.1.2.a	검색창에 특정 문자열을 입력한다.	P
3.1.2.b	검색창에 61자 길이의 문자열을 입력한다.	P
3.1.2.c	검색창에 특정 문자열을 입력하여 검색 결과 목록이 출력되도록 한다. 검색창에서 해당 문자열을 다시 지운다.	P
3.1.3.a	태그 3개 이상이 출력되는 검색 결과 목록에서 태그 하나를 선택한다. 그 다음 3개의 태그들을 다시 임의로 선택한다.	P
3.1.3.b	검색창에 특정 문자열을 입력하여 출력된 검색 결과 목록에서 한 태그를 선택한 뒤 다시 전체 태그 목록으로 돌아간다.	P
3.1.3.c	전체 태그 목록에서 한 태그를 선택한 후, 검색창에 해당 태그명의 일부 문자열을 입력할 때 출력되는 목록에서 해당 태그를 다시 한번 선택한다.	P
3.1.4.a	태그 목록에서 어떤 태그도 선택하지 않고 '추천하기' 버튼을 누른다.	P
3.1.4.b	태그 목록에서 하나의 태그를 선택하고 '추천하기' 버튼을 누른다.	P
3.1.4.c	전체 태그 목록에서 모든 태그들을 선택하고 '추천하기' 버튼을 누른다.	P
3.2.1.a	이전 단계에서 하나의 태그를 선택하고 추천을 요청했다.	P
3.2.1.b	이전 단계에서 모든 태그들을 선택하고 추천을 요청했다.	P
4.1.1.a	태깅이 완료 된 영화 리스트에서 1개의 영화를 임의로 선택한다.	P
4.1.2.a	검색창에 특정 문자열을 입력한다.	P

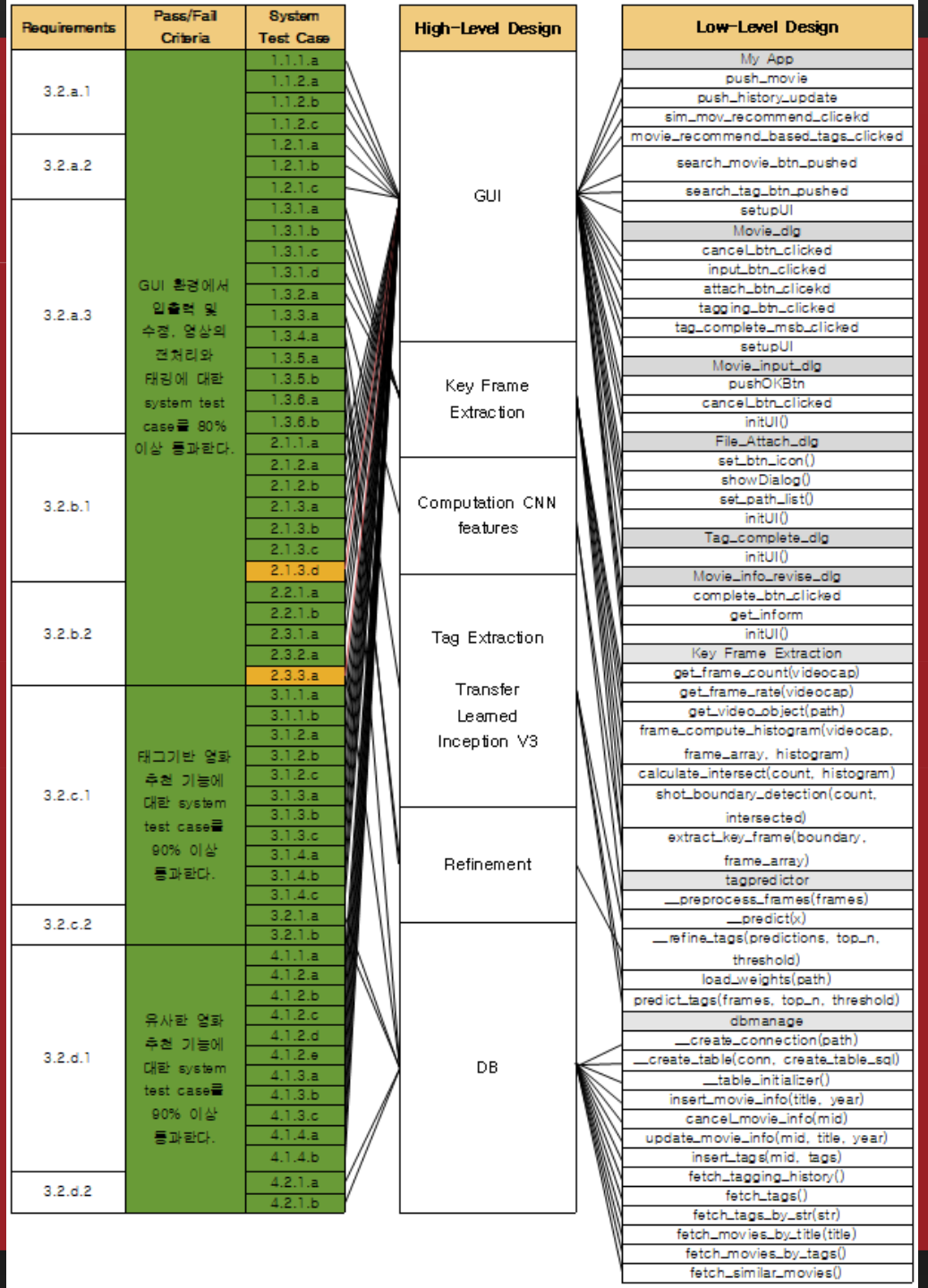
시스템 시험 항목 결과

4.1.2.b	영화 검색 결과가 출력된 상태에서, 검색창에 특정 문자열을 입력한다.	P
4.1.2.c	영화가 선택된 상태에서 검색창에 특정 문자열을 입력한다.	P
4.1.2.d	검색창에 입력된 문자열을 조건으로 포함하는 질의를 DB로 보낸다.	P
4.1.2.e	DB로부터 결과값을 전달받는다.	P
4.1.3.a	검색에 사용된 문자열을 포함하는 검색 결과 목록에서 영화를 하나 선택한다.	P
4.1.3.b	검색창에 특정 문자열을 입력하여 출력된 검색 결과 목록에서 영화를 하나 선택한 뒤 다시 전체 영화 리스트로 돌아간다.	P
4.1.3.c	전체 영화 리스트에서 한 영화를 선택한 후, 검색창에 해당 영화제목의 일부 문자열을 입력할 때 출력되는 목록에서 해당 영화를 다시 한 번 선택한다.	P
4.1.4.a	영화 리스트에서 어떤 영화도 선택하지 않고 '추천하기' 버튼을 누른다.	P
4.1.4.b	영화 리스트에서 하나의 영화를 선택하고 '추천하기' 버튼을 누른다.	P
4.2.1.a	선택된 영화의 태그를 조건으로 포함하는 질의를 DB로 보낸다.	P
4.2.1.b	이전 단계에서 하나의 영화를 선택하고 추천을 요청했다.	P

통과 기준 결과

#	Pass/Fail Criteria	P/F
1	GUI 환경에서 입출력 및 수정, 영상의 전처리와 태깅에 대한 system test case 를 80% 이상 통과한다.	P
2	태그기반 영화 추천 기능에 대한 system test case를 90% 이상 통과한다.	P
3	유사한 영화 추천 기능에 대한 system test case를 90% 이상 통과한다.	P

추적성 분석표 결과



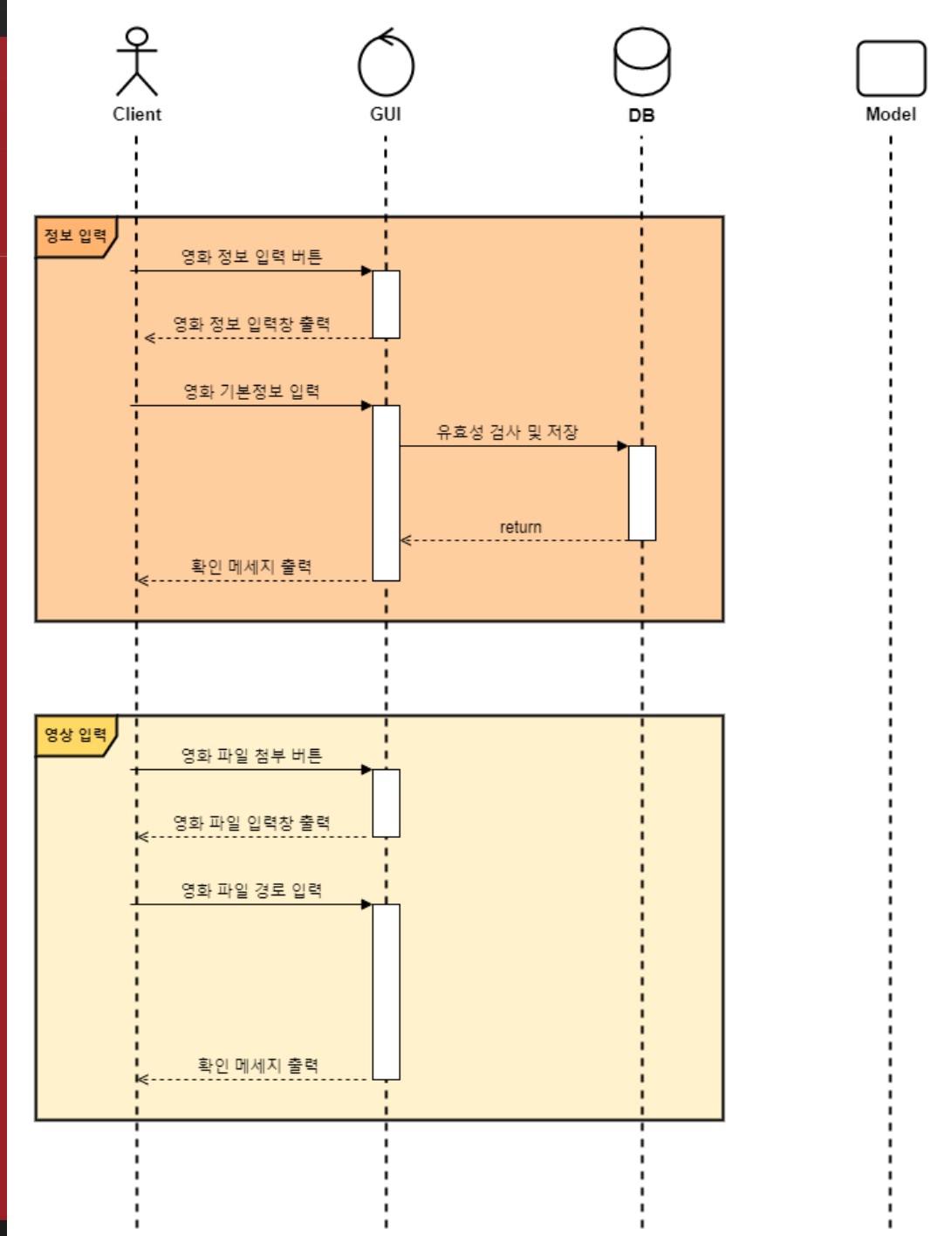
Requirements	Pass/Fail Criteria	System Test Case
3.2.a.1	GUI 환경에서 입출력 및 수정, 영상의 전처리와 태깅에 대한 system test case를 80% 이상 통과한다.	1.1.1.a
		1.1.2.a
		1.1.2.b
		1.1.2.c
3.2.a.2		1.2.1.a
		1.2.1.b
		1.2.1.c
3.2.a.3		1.3.1.a
		1.3.1.b
		1.3.1.c
		1.3.1.d
		1.3.2.a
		1.3.3.a
		1.3.4.a
		1.3.5.a
		1.3.5.b
		1.3.6.a
		1.3.6.b
3.2.b.1		2.1.1.a
		2.1.2.a
		2.1.2.b
		2.1.3.a
		2.1.3.b
		2.1.3.c
		2.1.3.d
3.2.b.2		2.2.1.a
		2.2.1.b
		2.3.1.a
		2.3.2.a
		2.3.3.a
3.2.c.1	태그기반 영화 추천 기능에 대한 system test case를 90% 이상 통과한다.	3.1.1.a
		3.1.1.b
		3.1.2.a
		3.1.2.b
		3.1.2.c
		3.1.3.a
		3.1.3.b
		3.1.3.c
		3.1.4.a
		3.1.4.b
		3.1.4.c
3.2.c.2		3.2.1.a
		3.2.1.b
3.2.d.1	유사한 영화 추천 기능에 대한 system test case를 90% 이상 통과한다.	4.1.1.a
		4.1.2.a
		4.1.2.b
		4.1.2.c
		4.1.2.d
		4.1.2.e
		4.1.3.a
		4.1.3.b
		4.1.3.c
		4.1.4.a
		4.1.4.b
3.2.d.2		4.2.1.a
		4.2.1.b

High-Level Design
GUI
Key Frame Extraction
Computation CNN features
Tag Extraction Transfer Learned Inception V3
Refinement
DB

Low-Level Design
My App
push_movie
push_history_update
sim_mov_recommend_clicekd
movie_recommend_based_tags_clicked
search_movie_btn_pushed
search_tag_btn_pushed
setupUI
Movie_dlg
cancel_btn_clicked
input_btn_clicked
attach_btn_clicekd
tagging_btn_clicked
tag_complete_msb_clicked
setupUI
Movie_input_dlg
pushOKBtn
cancel_btn_clicked
initUI()
File_Attach_dlg
set_btn_icon()
showDialog()
set_path_list()
initUI()
Tag_complete_dlg
initUI()
Movie_info_revise_dlg
complete_btn_clicked
get_inform
initUI()
Key Frame Extraction
get_frame_count(videocap)
get_frame_rate(videocap)
get_video_object(path)
frame_compute_histogram(videocap, frame_array, histogram)
calculate_intersect(count, histogram)
shot_boundary_detection(count, intersected)
extract_key_frame(boundary, frame_array)
tagpredictor
__preprocess_frames(frames)
__predict(x)
__refine_tags(predictions, top_n, threshold)
load_weights(path)
predict_tags(frames, top_n, threshold)
dbmanage
__create_connection(path)
__create_table(conn, create_table_sql)
__table_initializer()
insert_movie_info(title, year)
cancel_movie_info(mid)
update_movie_info(mid, title, year)
insert_tags(mid, tags)
fetch_tagging_history()
fetch_tags()
fetch_tags_by_str(str)
fetch_movies_by_title(title)
fetch_movies_by_tags()
fetch_similar_movies()

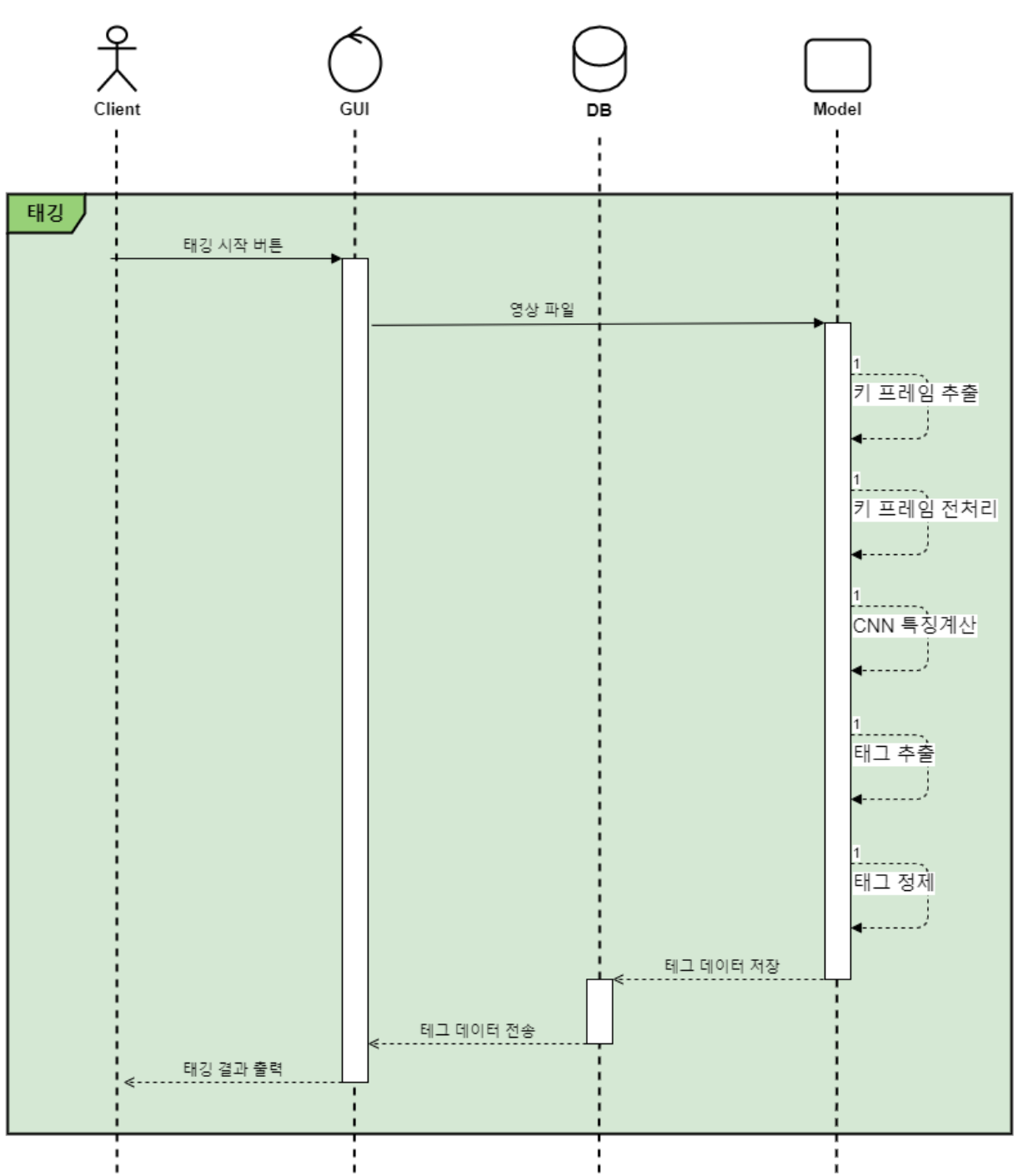
데모 시나리오 #1

영화의 기본정보와 영화 파일을 입력한다.



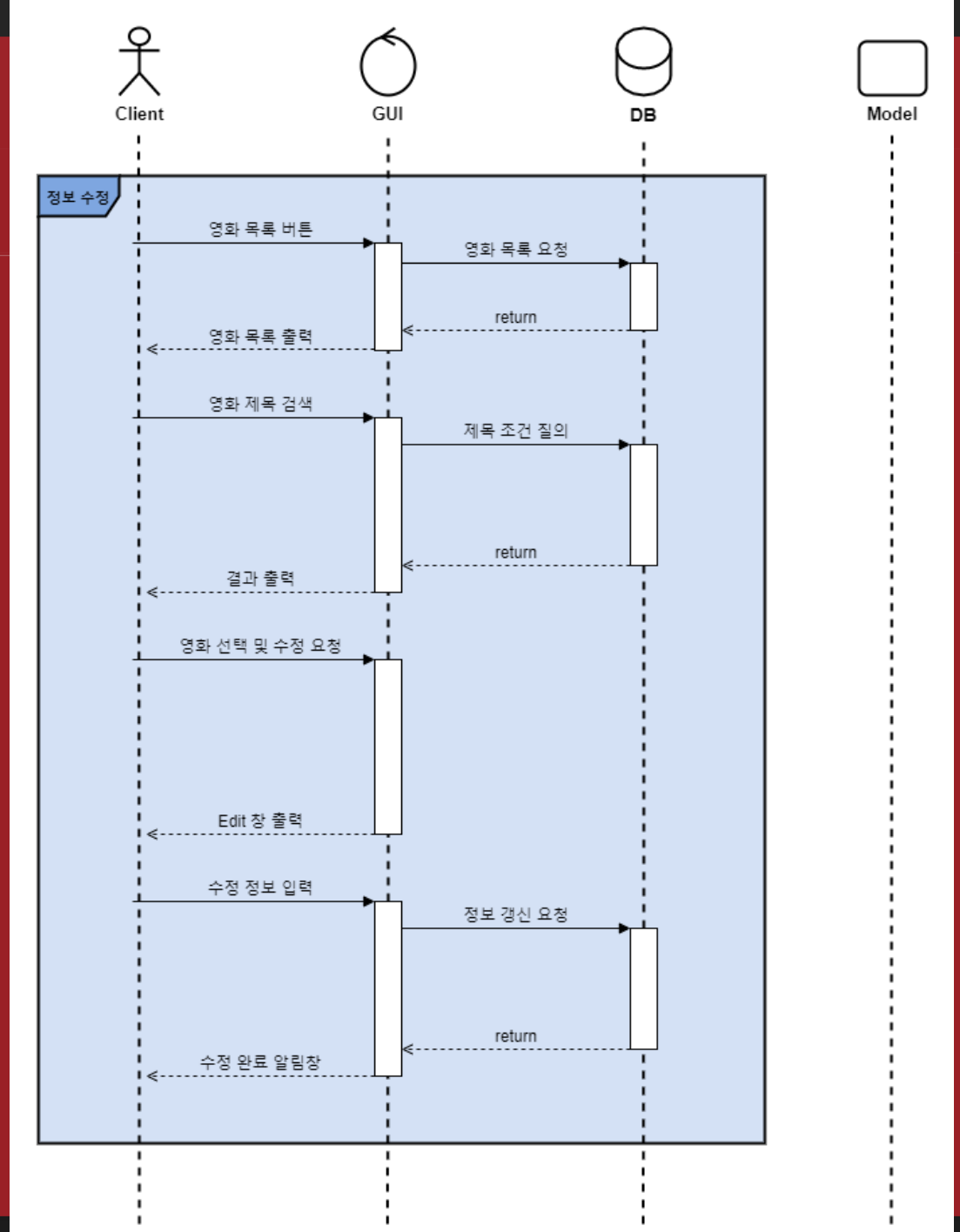
데모 시나리오 #2

입력 받은 영화 파일에서 태그를 도출해낸다.



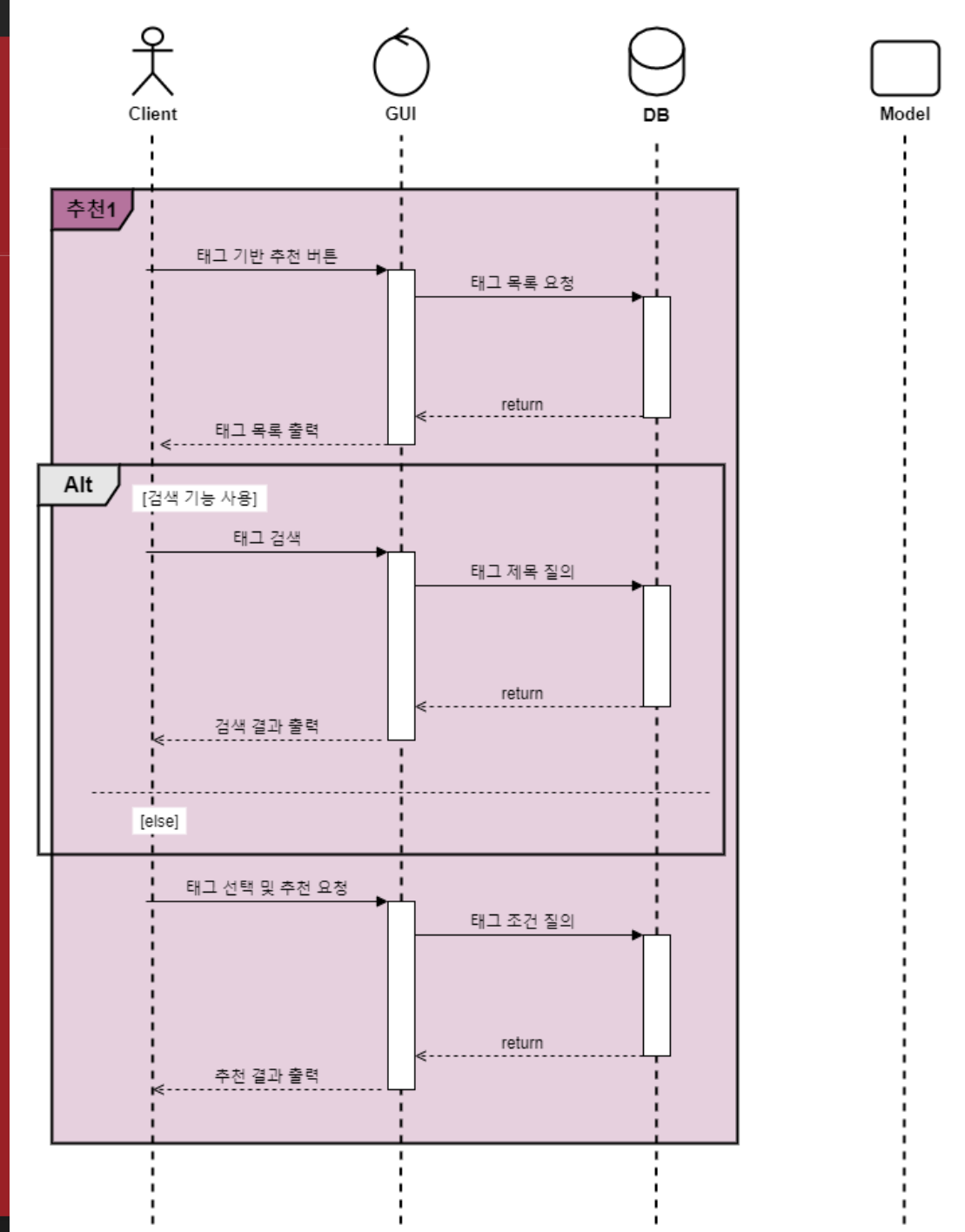
데모 시나리오 #3

이미 입력한 영화의 기본 정보를 수정한다.



데모 시나리오 #4

태그 기반 영화 추천 기능을 사용한다.



데모 시나리오 #5

유사한 영화 추천 기능을 사용한다.