

# 캡스톤 디자인 I

## 종합설계 프로젝트

프로젝트 명	WittyPhotos - 더 똑똑한 갤러리
팀 명	메모리즈
문서 제목	24조_중간보고서

Version	1.3
Date	2019-04-18


팀원	정찬영 (조장)
	서민호
	손민지
	이가빈
	장지은
지도교수	박하명 교수님

### CONFIDENTIALITY/SECURITY WARNING


이 문서에 포함되어 있는 정보는 국민대학교 전자정보통신대학 컴퓨터공학부 및 컴퓨터공학부 개설 교과목 캡스톤 디자인 수강 학생 중 프로젝트 “WittyPhotos”를 수행하는 팀 “메모리즈(24조)”의 팀원들의 자산입니다. 국민대학교 컴퓨터공학부 및 팀 “메모리즈(24조)”의 팀원들의 서면 허락없이 사용되거나, 재가공 될 수 없습니다.

## 문서 정보 / 수정 내역

Filename	중간보고서-WittyPhotos.doc
원안작성자	정찬영, 서민호, 손민지, 이가빈, 장지은
수정작업자	정찬영, 서민호, 손민지, 이가빈, 장지은

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	중간보고서		
	프로젝트 명	WittyPhotos	
	팀 명	메모리즈	
	Confidential Restricted	Version 1.3	2019-APR-18

수정날짜	대표수정자	Revision	추가/수정 항목	내 용
2019-04-15	전원	1.0	최초 작성	
2019-04-16	전원	1.1	내용 추가	
2019-04-17	전원	1.2	내용 추가	
2019-04-18	전원	1.3	내용 추가	

 <b>국민대학교 컴퓨터공학부 캡스톤 디자인 I</b>	<b>중간보고서</b>		
	<b>프로젝트 명</b>	WittyPhotos	
	<b>팀 명</b>	메모리즈	
	Confidential Restricted	Version 1.3	2019-APR-18

## 목 차

1. **프로젝트 목표**
  - 1.1 프로젝트 소개
  - 1.2 Use Case Diagram
  - 1.3 시나리오
  - 1.4 프로젝트 목표
2. **수행 내용 및 중간결과**
  - 2.1 계획서 상의 연구내용
    - 2.1.1 이미지 태그 기능 구현
    - 2.1.2 태그 카테고리화와 데이터베이스
    - 2.1.3 통계 및 네트워크 분석
    - 2.1.4 UI/UX 디자인 설계 및 안드로이드 어플리케이션 구현
    - 2.1.5 웹 서버와 웹 클라이언트 구축
  - 2.2 수행내용
    - 2.2.1 이미지 태그 기능 구현
    - 2.2.2 태그 카테고리화와 데이터베이스
    - 2.2.3 통계 및 네트워크 분석
    - 2.2.4 UI/UX 디자인 설계 및 안드로이드 어플리케이션 구현
    - 2.2.5 웹 서버와 웹 클라이언트 구축
3. **수정된 연구내용 및 추진 방향**
  - 3.1 수정사항
    - 3.1.1 이미지 태그 기능 구현
    - 3.1.2 태그 카테고리화와 데이터베이스
    - 3.1.3 통계 및 네트워크 분석
    - 3.1.4 UI/UX 디자인 설계 및 안드로이드 어플리케이션 구현
    - 3.1.5 웹 서버와 웹 클라이언트 구현
4. **향후 추진계획**
  - 4.1 향후 계획의 세부 내용
    - 4.1.1 이미지 태그 기능 구현
    - 4.1.2 태그 카테고리화와 데이터베이스
    - 4.1.3 통계 및 네트워크 분석
    - 4.1.4 UI/UX 디자인 설계 및 안드로이드 어플리케이션 구현
    - 4.1.5 웹 서버와 웹 클라이언트 구축
5. **고충 및 건의사항**

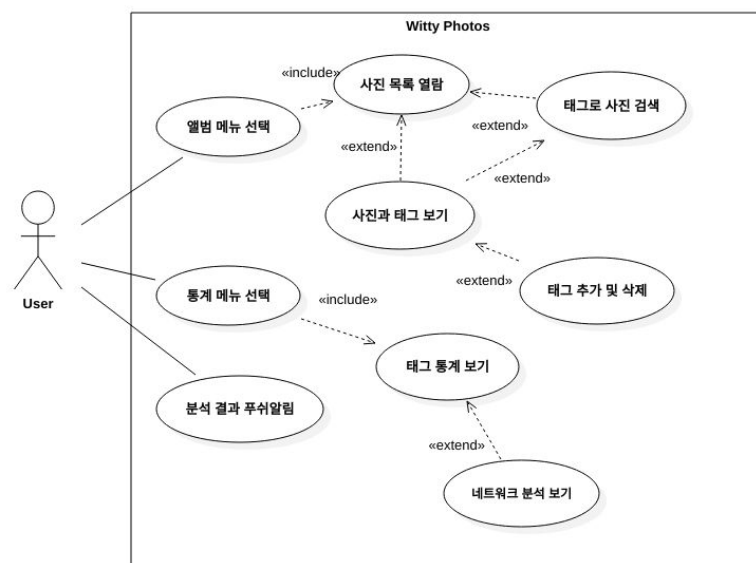
## 1. 프로젝트 목표

### 1.1. 프로젝트 소개


현대인들은 이전에 비해 훨씬 방대한 양의 이미지를 생산해 내고 있으며, 때때로 개인이 갖는 이미지가 지나치게 많아지기도 합니다. 개인이 소장하고 있는 이미지의 양이 많아짐에 따라 이미지 관리가 필요하지만 개인적으로 이미지 데이터 관리를 하는 것에 어려움을 겪고 있습니다.

본 프로젝트는 이러한 문제를 해결하기 위해 딥러닝을 이용한 **이미지 자동 태그 기능**을 제공하고 사용자가 직접 수동적으로 태그를 추가, 삭제 등 관리할 수 있게 함으로써 이미지 데이터 관리를 용이하게 합니다. 또한 태그 데이터를 분석하여 **사용자의 라이프 스타일과 휴먼 네트워크정보**를 재치있게 전달하는 안드로이드 어플리케이션을 제작합니다.

### 1.2. Use Case Diagram




[그림 1 - Use Case Diagram]

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	중간보고서		
	프로젝트 명	WittyPhotos	
	팀 명	메모리즈	
	Confidential Restricted	Version 1.3	2019-APR-18

### 1.3. 시나리오


NO.	자세한 시나리오 내용
S1	사용자가 WittyPhotos 어플리케이션을 실행한다.
S1-1	Splash 화면이 실행된다.
S1-2	사용자의 이미지들이 동기화되고 이미지마다 자동으로 태그가 지정된다.
S2	사용자가 앨범 메뉴와 통계 메뉴 중 선택할 수 있는 창이 뜬다.
S2-1	앨범 메뉴를 누를 경우 저장되어있는 모든 이미지들을 보여준다.
S3	사용자는 이미지를 선택하거나 검색창에 태그를 입력하여 이미지를 검색할 수 있다.
S4	이미지를 선택할 경우 이미지를 확대하여 보여주고 아래에 태그 리스트를 보여준다.
S4-1	‘+’ 버튼을 누르면 원하는 태그를 추가할 수 있다.
S4-2	‘-’ 버튼을 누르면 기존 태그를 삭제할 수 있다.
S4-3	삭제한 태그가 자동으로 지정된 태그인 경우 영구 삭제를 물어보는 창이 뜬다.
S4-4	영구 삭제를 할 경우 해당 태그는 모든(기존, 새로운) 이미지에 태그로 지정되지 않는다.
S4-5	인물 사진일 경우 이미지 상의 얼굴에 인식된 얼굴의 box가 그려지고 사용자가 box를 클릭하면 해당 인물의 이름을 태그로 지정할 수 있다.
S4-6	지정된 이름은 다음 번에 같은 인물이라고 프로그램이 인식하면 자동으로 이름 태그를 지정해준다.
S5	검색창에 태그를 검색할 경우 해당 태그가 지정된 이미지들을 보여준다.
S5-1	FR4와 마찬가지로 이미지를 선택하여 태그 리스트를 볼 수 있다.
S6	통계 메뉴를 누를 경우 태그들의 사용횟수로 통계를 낸 그래프가 나타난다.
S6-1	오른쪽으로 슬라이드하면 태그들을 네트워크 분석한 그래프(1)가 나타난다.
S6-2	또 오른쪽으로 슬라이드하면 네트워크 분석 그래프(2)가 나타난다.
S6-3	그래프 아래에 데이터를 분석한 재치있는 코멘트가 나타난다.
S7	코멘트들은 주기적으로 한 번씩 사용자에게 푸시 알림으로 띄워진다.

 <b>국민대학교 컴퓨터공학부 캡스톤 디자인 I</b>	<b>중간보고서</b>		
	<b>프로젝트 명</b>	WittyPhotos	
	<b>팀 명</b>	메모리즈	
	Confidential Restricted	Version 1.3	2019-APR-18

## 1.4. 프로젝트 목표

1.2의 시나리오에서 작성한 사항을 달성하기 위해 본 프로젝트의 어플리케이션은 다음과 같은 기능의 구현을 목표로 한다.

- 1) 안드로이드 어플리케이션 기본 기능 구축 및 UI/UX 디자인
- 2) 안드로이드 어플리케이션 내부 데이터베이스 구축
- 3) 클라이언트 구축
- 4) AWS 서버 구축
- 5) 이미지 동기화
- 6) 이미지를 인식하여 자동으로 태그 지정
- 7) 수동으로 태그 추가 삭제 기능
- 8) 태그 통계 및 네트워크 시각화
- 9) 태그 네트워크 커뮤니티 추출
- 10) 태그 네트워크 분석 및 정보 추출

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	중간보고서		
	프로젝트 명	WittyPhotos	
	팀 명	메모리즈	
	Confidential Restricted	Version 1.3	2019-APR-18

## 2. 수행 내용 및 중간결과

### 2.1. 계획서 상의 연구내용

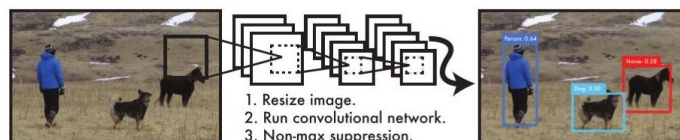
#### 2.1.1. 이미지 태그 기능 구현

##### ● 자동 이미지 태그 - 딥러닝을 기반으로 한 이미지 자동 인식

###### - 객체 인식(Object detection) 기술 사용

사용자의 스마트폰에 저장되어있는 이미지들이 어플리케이션에 동기화 된 후 바로 각 이미지 안의 객체들을 인식하여 자동으로 태그가 지정되어야한다. 이를 위해 객체 인식 기술을 사용하는데, 이는 이미지 또는 비디오 상의 객체를 식별하는 컴퓨터 비전 기술로 딥러닝 혹은 머신러닝 알고리즘을 통해 산출되는 기술이다.

1차 수행 계획서에서 본 프로젝트 진행을 위한 객체 인식 기술을 선정하기 위해 속도와 정확성에 대한 tradeoff를 고려하여 CNN, Fast-RCNN, Google Cloud Vision 등 다양한 기술을 연구하였다. 학습시간을 줄일 수 있는 고성능 GPU가 있거나 개발 시간이 충분한 경우 RCNN 기술을 기반으로 좀 더 정확한 객체 인식 결과를 얻을 수 있지만 현실적 제한 요소를 고려하여 초고속으로 기술 실행이 가능한 YOLO v3를 활용하여 구현하기로 계획하였다.




**Figure 1: The YOLO Detection System.** Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to  $448 \times 448$ , (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.

[그림 2 - YOLO 객체 인식 시스템]

###### - YOLO(You Only Look Once)

YOLO는 대표적인 단일 단계 방식의 객체 인식 알고리즘이다. 단순한 처리로 속도가 매우 빠르고 직관적이기 때문에 기존 다른 기술과 비교할 때 2배 이상의 높은 성능을 보인다. 또한 소규모 프로젝트에서 다른 기술과 정확성의 차이는 거의 없다.

 <b>국민대학교 컴퓨터공학부 캡스톤 디자인 I</b>	<b>중간보고서</b>		
	<b>프로젝트 명</b>	WittyPhotos	
	<b>팀 명</b>	메모리즈	
	Confidential Restricted	Version 1.3	2019-APR-18

## ● 수동 이미지 태그

사용자는 자동으로 생성된 태그 이외에 자신의 취향과 의도에 맞게 직접 조작을 통해 태그를 추가 또는 삭제 할 수 있다.

이 때, 안드로이드 어플리케이션 내부의 데이터베이스(SQLite3)에서 태그가 추가 또는 삭제되며 그에 따른 기타 작업(태그의 사용 횟수 증가, 감소 등)도 수행된다.

1차 수행 계획서에서 추가와 삭제에 관한 제약은 다음과 같이 계획하였다.

- 태그 수동 추가
  - 태그의 삽입은 개수의 제한이 없다.
  - 한 이미지 내의 동일한 태그가 두개 이상 존재할 수 없다.
  - 영어, 한국어 두 개의 언어를 지원한다.
  - 이모티콘과 특수문자는 지원하지 않는다.
  - 태그의 길이는 최대 20자 이다.
  - 사용자에게 의해 생성된 태그는 데이터베이스에 저장되어 추후 사용자가 태그 삽입을 시도할 경우 사용자에게 추천 태그로 노출된다.
- 태그 수동 삭제
  - 자동으로 생성된 태그를 포함한 모든 태그는 삭제가 가능하다.
  - 삭제된 태그는 데이터베이스에서도 영구 삭제된다.


## ● 인물 태그 - face recognition (얼굴 인식)을 통한 태그

사용자의 이미지에서 얼굴을 인식하여 사용자 본인과 주위 인물을 태그할 수 있도록 하기위한 기능이다. 이 기능을 구현하기 위해 Python의 dlib(배포 라이브러리)의 face recognition 기능과 DBSCAN(Density-Based Spatial Clustering of Applications with Noise - 밀도기반 클러스터링) 알고리즘을 활용하기로 계획하였다.

먼저 Python의 dlib의 얼굴 인식 기능을 사용해서 구축된 face recognition 기능을 사용하여 미리 학습된 얼굴 인식 모델을 이용하여 이미지 속에 인물이 있을 경우 그 얼굴을 인식하고 128크기의 벡터로 수치화하고 같은 인물이 다음에 이미지에 노출 될 경우 자동으로 같은 태그를 붙일 있도록 Python의 scikit-learn 패키지를 활용하여 클러스터링한다. 분류해야 하는 사람의 수가 정해져있지 않으므로 DBSCAN 알고리즘을 사용한다.

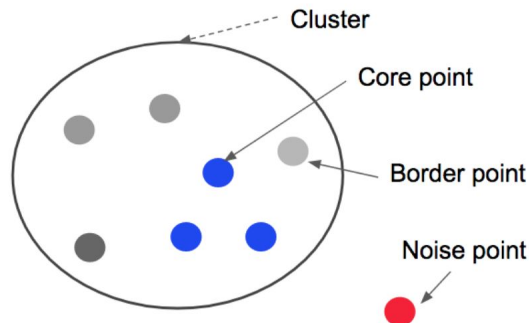
- 얼굴 인식을 위한 Python의 face\_recognition
  - i) 이미지의 단순화 버전을 만들어 주는 HOG알고리즘을 사용해 이미지를 인코딩. 단순화된 이미지에서 얼굴의 일반 HOG 인코딩(generic HOG encoding)과 가장 유사하게 보이는 부분을 찾아 얼굴 검출(face detection)한다.
  - ii) 얼굴의 주요 69개의 특징 포인트인 랜드마크(landmarks)를 찾아 얼굴의 포즈를 알아내어 눈과 입이 중앙에 오도록 이미지를 변형.
  - iii) 딥 컨볼루션 신경망(Deep Convolutional Neural Network)을 훈련시킨다. 얼굴에 대해 128개의 측정값을 저장하고 같은 인물에 대한 두 이미지의 측정값은 가깝게 서로 다른



 <b>국민대학교 컴퓨터공학부 캡스톤 디자인 I</b>	<b>중간보고서</b>		
	<b>프로젝트 명</b>	WittyPhotos	
	<b>팀 명</b>	메모리즈	
	Confidential Restricted	Version 1.3	2019-APR-18

인물에 대한 두 이미지의 측정값은 좀 더 멀어지도록 조정하는 단계를 반복하여 신경망이 128개 측정값을 신뢰성있게 배운다.

- DBSCAN 알고리즘으로 클러스터링 기술 구현



[그림 3 - DBSCAN 알고리즘]


점이 세밀하게 몰려 있어 밀도가 높은 부분을 클러스터링하는 밀도 기반의 클러스터링 기법으로 군집 개수를 사전에 설정하지 않아도 군집을 산출하는 알고리즘이다. 복잡한 형상도 찾을 수 있으며, 어떤 클래스에도 속하지 않는 포인트를 구분할 수 있다.

한 데이터 포인트에서 eps 거리(군집으로 인정하기 위한 반경) 안에 데이터가 min\_samples 개수 만큼 들어 있으면 이 데이터를 핵심 샘플로 분류한다. eps보다 가까운 핵심 샘플은 DBSCAN에 의해 동일한 클러스터로 합쳐지고 eps 거리 안에 있는 포인트 수가 min\_samples 보다 적다면 그 포인트는 어떤 클래스 속에도 속하지 않는 잡음으로 레이블한다.

## 2.1.2. 태그 카테고리화와 데이터베이스

사용자의 라이프 스타일을 분석하기 위해 모든 태그들은 특정 카테고리 안에 속해 있어야 한다. 태그들과 연결된 카테고리는 데이터베이스에서 관리한다. 카테고리는 인물, 동물, 식물, 사물, 음식, 색상, 장소, 기타 총 8개로 나누어져 있다.

- 자동 이미지 태그의 카테고리화  
본 프로젝트에서 이미지에서 태그를 자동으로 추출하기 위해 사용할 기술인 YOLO에서는 약 9000개의 단어 데이터를 제공한다. 자동으로 이미지가 태그됨과 동시에 태그를 분류하여 데이터베이스에 저장하기 위해 YOLO에서 제공하는 모든 단어 데이터들을 각 카테고리별로 나누어 데이터베이스에 저장해둔다.
- 수동 이미지 태그의 카테고리화  
사용자가 태그를 작성할 때 직접 카테고리를 고를 수 있게 하고 데이터베이스에 저장한다.
- 데이터베이스

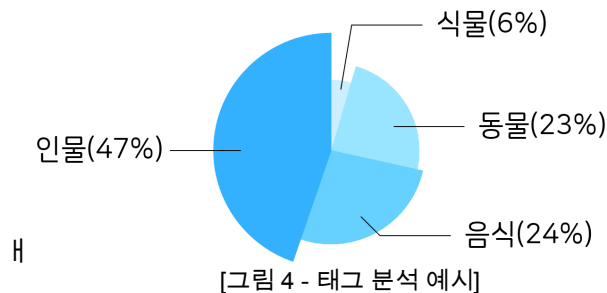
 <b>국민대학교 컴퓨터공학부 캡스톤 디자인 I</b>	<b>중간보고서</b>		
	<b>프로젝트 명</b>	WittyPhotos	
	<b>팀 명</b>	메모리즈	
	Confidential Restricted	Version 1.3	2019-APR-18

수 각 태그가 소속된 카테고리나 사용된 횟수 정보가 저장된다.  
기본키는 태그 단어이며 개체 무결성을 만족해야한다. 따라서 중복된 태그는 존재할  
없으며 Null값을 가질 수 없다.

### 2.1.3. 통계 및 네트워크 분석

데이터베이스에 저장되어있는 사용자의 태그 기록을 이용하여 사용자의 라이프 스타일과 휴먼 네트워크를 분석하여 시각적으로 제공한다. 데이터 분석은 두 가지 방법으로 이루어지며, 첫번째로 각 태그의 사용률을 분석하고 두번째로 네트워크 분석을 기반으로 네트워크 내의 응집력있는 태그를 이용해 다른 사용자와의 상관관계를 파악하여 분석한다.

- 태그의 사용률 통계 - 사용자의 라이프 스타일을 분석하고 코멘트 및 푸시 알림 제공




이미지에 사용된 태그의 개수를 기반으로 전체 혹은 각 카테고리 별(음식, 동물, 인물 등)로 어떠한 태그를 많이 사용하였는지 상위 항목을 추출하고 원형 그래프로 시각화하여 보여준다.

(User) 님은 2018.12/25~2019.2/10 동안  
치킨 을 가장 많이 드셨네요!

(User) 님은 2018.1/3~2019.1/15 동안  
(이름) 님을 가장 많이 만나셨네요!



[그림 5 - 라이프스타일 분석 코멘트와 푸시 예시]

 <b>국민대학교 컴퓨터공학부 캡스톤 디자인 I</b>	<b>중간보고서</b>		
	<b>프로젝트 명</b>	WittyPhotos	
	<b>팀 명</b>	메모리즈	
	Confidential Restricted	Version 1.3	2019-APR-18

또한, 그룹 별로 기본 문장 구성(structure)을 만들어두고 태그가 속한 카테고리에 맞게 문장을 재생성하여 사용자에게 코멘트를 제공한다.


## ● 태그 네트워크 분석 - 사용자의 휴먼 네트워크 등의 파악

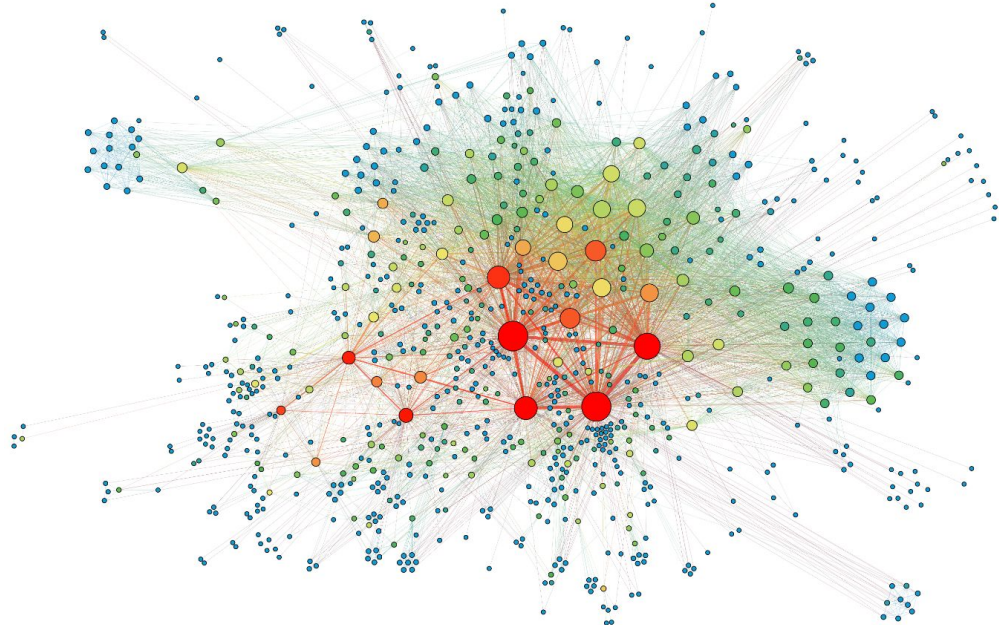
사용자의 이미지에 등장한 인물 및 사물, 그리고 사용자 사이의 연관 관계를 그래프의 형태로 표현하고, 클러스터링 (network clustering), 중심성 분석 (centrality analysis), 그래프 시각화 (graph visualization) 등의 그래프 분석 기법들을 활용하여 사용자의 주변 환경 및 라이프 스타일을 분석한다.

- **Network Clustering**  
주어진 데이터의 특징을 파악하여 여러개의 군집으로 분리하는 방법이다. 사물 및 사람 네트워크에서 network clustering을 하여 사물과 사람 사이의 상호 연관관계를 찾고, 이를 통해 각 이미지의 context를 분석한다.
- **Network Centrality**  
사진으로부터 추출한 인물 및 사물관계 네트워크에서 network centrality 분석을 함으로써 사용자의 주변에서 가장 중요한 사물이나 인물을 찾는다. 이를 기반으로 사용자가 최근에 자주 만난 사람이나 자주 먹은 음식, 자주 간 장소 등을 파악한다.

Network Centrality에는 다음과 같은 종류들이 있다.

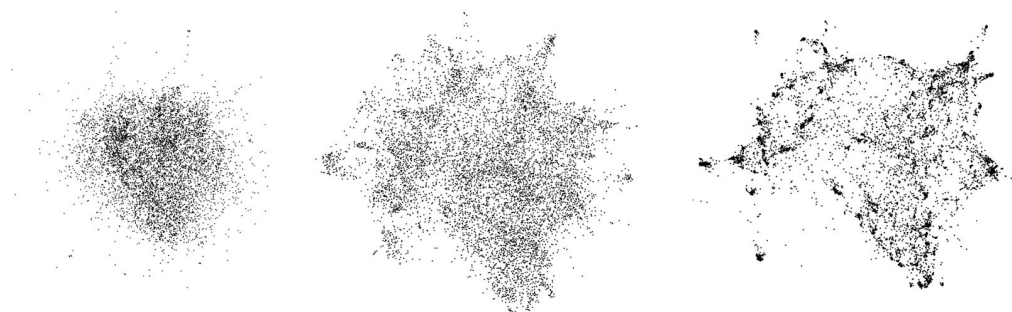
노드에 간선이 몇개 연결되어 있는 지를 보고 연결되어있는 간선이 많을 수록 중요성이 높다고 평가하는 연결중심성(Degree Centrality), 한 노드에서 다른 노드까지 가는 경로가 짧을 수록 중요한 노드라고 판단하는 근접 중심성(Closeness Centrality), 특정 노드를 제외한 다른 노드 간의 경로에 있는 특정 노드가 많이 거쳐갈 수록 중심성이 높다고 판단하는 매개 중심성(Betweenness Centrality), 중요한 노드들과 연결되어 있는 노드가 핵심적인 노드 고유벡터 중심성(Eigenvector Centrality)

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	중간보고서		
	프로젝트 명	WittyPhotos	
	팀 명	메모리즈	
	Confidential Restricted	Version 1.3	2019-APR-18



[그림 6 - Network Centrality 분석 예제]


- Network Visualization  
태그 네트워크와 네트워크 분석 결과를 시각화하여 네트워크의 구조와 분석 결과에 대한 직관적인 이해를 돕는다. 네트워크 시각화를 위해 그래폴로지(graphology) 알고리즘을 활용한다. 대표적인 그래폴로지 알고리즘으로는 ForceAtlas2가 있다. ForceAtlas2는 그래프 노드의 위상을 노드간 연결성에 기반하여 반복적으로 재정렬한다.



[그림 7 - ForceAtlas2에 의한 그래프 노드의 위상 변화 예제]

#### 2.1.4. UI/UX 디자인 설계 및 안드로이드 어플리케이션 구현

안드로이드 스튜디오 베이스로 사용자에게 보다 직관적이고 깔끔한 어플리케이션을 제공한다. 앱 내 기능은 크게 두 가지 방법으로 세분화되며, 첫번째로 앨범 카테고리 내에서

 <b>국민대학교 컴퓨터공학부 캡스톤 디자인 I</b>	<b>중간보고서</b>		
	<b>프로젝트 명</b>	WittyPhotos	
	<b>팀 명</b>	메모리즈	
	Confidential Restricted	Version 1.3	2019-APR-18

태그 관리를 통한 이미지 데이터 정리 기능과 두번째로 태깅되어 있는 데이터를 기반으로 분석된 다양한 그래프(워드 클라우드, 네트워크 그래프, 군집 그래프)를 visualising하는 기능이다.

## ● 모바일 User Interface와 작동 순서

1. 구글플레이에서 어플리케이션 다운로드
2. WittyPhotos - 더 똑똑한 갤러리 어플리케이션 실행
3. 메인 페이지에서 사용자가 원하는 카테고리 선택 (Tagging or Graph)
4. 사용자의 갤러리와 연동 (연동 후 이미지마다 자동적으로 태그 생성)

## ● Tag 항목 선택


- Tag 항목 선택시 다음과 같은 기능을 수행하여야한다.
- 검색 툴을 이용해 태그 검색
- 유저의 안드로이드 기기에 저장되어있는 이미지 노출
- 태그 수정(삭제/추가)이 필요한 이미지 클릭
- 이미지 하단에 노출된 태그 리스트에서 삭제할 태그 선택 후 삭제
- 최하단의 '+' 버튼을 사용해 새로운 태그 생성

## ● Chart 항목 선택

- Chart 항목 선택시 다음과 같은 기능을 수행하여야한다.
- Chart 메뉴 내 Graph 혹은 Pattern 카테고리 선택
- 그래프 분석하고자 하는 특정한 기간을 선택 (전체앨범 또한 가능)
- 생성된 이미지 태그를 기반으로 전체/일정 기간 동안의 통계, 관계 그래프 생성
- 또한, 통계 결과를 기반으로 사용자의 라이프 스타일을 분석하여 재밌는 코멘트와 푸시(push) 알림을 제공한다.

## 2.1.5. 웹 서버와 웹 클라이언트 구축

객체 인식과 얼굴 인식, 네트워크 분석 기능은 파이썬으로 작성될 예정이다. 하지만 안드로이드에서 파이썬으로 만든 프로그램을 구동시키는 것은 아직 어렵기 때문에 서버에서 파이썬 프로그램을 구동시켜 결과물을 안드로이드에 전송하는 것으로 계획하였다. 웹 서버는 안정적인 클라우드 서비스인 AWS(Amazon Web Server)를 사용하여 구축하고 Amazon RDS(Relational Database Service)로 웹 서버 데이터베이스를 구축하여 백업 용도로 사용한다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	중간보고서		
	프로젝트 명	WittyPhotos	
	팀 명	메모리즈	
	Confidential Restricted	Version 1.3	2019-APR-18

## ● AWS를 사용하여 클라우드 서버 구축

- AWS EC2를 사용하여 웹 클라우드 서버를 구축한다.
- Android Application에서 이미지를 전송받아 객체 및 얼굴인식 프로세스 후 자동태그를 생성하여 application에 전송한다.

## ● Amazon RDS를 사용하여 서버 데이터베이스 구축

- Amazon RDS를 구축하여 사용자의 정보와 이미지 파일, 태그 데이터 등을 백업하는 용도로 사용한다.

## ● 안드로이드 어플리케이션에 웹 클라이언트 구축

- 안드로이드 어플리케이션에 API를 사용하여 서버와 연동할 웹 클라이언트를 구축한다.

## 2.2. 수행내용

### 2.2.1. 이미지 태그 기능 구현

## ● 자동 이미지 태그 - 딥러닝을 기반으로 한 이미지 자동 인식

- 객체 인식 기술 연구  
YOLO v3 기술을 사용하여 이미지에서 객체를 인식하고 자동 태그를 추출하고자 계획하였지만 좀 더 나은 모델을 찾아 보는것이 좋을 것 같아 본격적인 기능 구현 전 여러 다른 기술들을 비교하여 분석하였다.

사용자가 어플리케이션을 최초 실행할 때 사용자의 모든 이미지들은 어플리케이션에 동기화되고 바로 서버에서 객체 인식 프로세서를 거친 후 자동으로 태그가 입력된다. 보통 일반인들의 스마트폰 속 이미지의 수가 수천 장이 넘는다는 것을 고려할 때, 객체 인식 프로세서는 최대한 짧은 시간안에 기능을 수행해야 한다. 또한, 객체 인식의 정확도가 떨어질 경우 사용자에게 제대로 된 정보를 전달하지 못하므로 정확도도 고려해야한다.

또한, 학습시간을 줄일 수 있는 고성능 GPU가 있거나 개발 시간이 충분한 경우 직접 학습 데이터셋을 마련하여 딥러닝을 시킬 수 있지만 현실적 제약 사항을 고려하여 이미 학습된 러닝 모델을 사용하도록 한다.

### 1. Azure Computer vision API를 사용한 이미지 분석

Azure Computer vision API는 Microsoft에서 제공하는 이미지 분석 API 서비스이다. YOLO와 비교하였을때 더욱 높은 정확성을 보여주며 다양한 태그를 지원해 주기 때문에 본 프로젝트에 적절한 모델인지 연구해보았다.



하지만 연구 결과 이용하기 위해서는 구독키를 필요로 하며 트랜잭션 및 속도에 제한이 생기기 때문에 여러 명이 동시에 이용해야 하는 어플리케이션의 기능으로 사용하기에는 적절치 않다고 판단하였다.

## 2. 다양한 딥러닝 모델 비교

다음은 다양한 딥러닝 모델을 비교한 그래프이다. [그림 8]은 각 모델의 속도와 정확도를 분석한 그래프이고 [그림 9]는 이미지 규모별 각 모델의 정확도 차이를 비교한 그래프이다.



[그림 8 - Fast RCNN, Faster RCNN, SSD, YOLO의 속도와 정확도 분석]

Real-Time Detectors	Train	mAP	FPS
100Hz DPM [31]	2007	16.0	100
30Hz DPM [31]	2007	26.1	30
Fast YOLO	2007+2012	52.7	<b>155</b>
YOLO	2007+2012	<b>63.4</b>	45
Less Than Real-Time			
Fastest DPM [38]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[28]	2007+2012	73.2	7
Faster R-CNN ZF [28]	2007+2012	62.1	18
YOLO VGG-16	2007+2012	66.4	21


**Table 1: Real-Time Systems on PASCAL VOC 2007.** Comparing the performance and speed of fast detectors. Fast YOLO is the fastest detector on record for PASCAL VOC detection and is still twice as accurate as any other real-time detector. YOLO is 10 mAP more accurate than the fast version while still well above real-time in speed.

[그림 9 - 실시간 객체 인식 모델의 성능과 정확도 비교]

[그림 8]의 그래프를 보면 정확도는 Faster RCNN이 가장 높고 속도는 YOLO가 가장 빠르다는 것을 알 수 있다. 그리고 SSD가 그 중간에 위치한다.

[그림 9]의 표를 보면 실시간 객체 인식 모델 중 정확도는 Faster RCNN이 Fast YOLO에 비해 약 10% 정도 높지만 속도는 Fast YOLO가 20배 이상 빠르다.

<sup>1</sup> Learn Machine Learning, AI & Computer vision, <https://cv-tricks.com/>

 <b>국민대학교 컴퓨터공학부 캡스톤 디자인 I</b>	<b>중간보고서</b>		
	<b>프로젝트 명</b>	WittyPhotos	
	<b>팀 명</b>	메모리즈	
	Confidential Restricted	Version 1.3	2019-APR-18

속도와 정확도의 tradeoff를 고민해본 결과 정확도의 차이는 수정 태그 기능으로 보완할 수 있으므로 사용자에게 빠른 속도의 이미지 인식을 제공하는 것이 낫다는 판단을 하였다.

따라서 YOLO의 다양한 모델 중 적합한 모델을 찾는 연구를 진행하였다.

#### - YOLO 모델 비교

아래 모델들은 모두 직접 실행해보고 속도와 정확도를 비교해보았다. 또한 코드를 분석해서 어떠한 방식으로 진행되는지 연구하였다.

### 1. YOLO v3 darknet

#### 1) 속도

YOLO darknet을 사용하여 CPU 환경에서 이미지 한 장을 분석하는 데 약 40~50초가 소요되었다. 이는 사용자의 입장에서 지나치게 긴 시간이므로 본 프로젝트에 적합하지 않다고 판단하였다.

#### 2) 정확도

YOLO darknet의 정확도는 mAP 수치상 50mAP라고 한다. 80개의 단어 데이터셋(coco.names)을 사용하여 10장의 이미지를 인식시키고 예상했던 결과값과 비교했을 때 정확도는 약 70% 정도가 되었다.

#### 3) 기타 사항

YOLO darknet은 C언어 기반으로 작성되어 본 프로젝트에서 Python으로 기존에 작성된 다른 프로세스와 호환성에 어려움을 겪었다.  
C언어로 작성된 darknet을 object 파일로 컴파일 한 후 Python에서 구동시키는 방향을 연구하였다.

### 2. YOLO v3 darkflow

#### 1) 속도

YOLO darkflow는 Python언어로 작성된 YOLO darknet이다. 이미지 인식 속도 면에서는 큰 차이를 보이지 못했다. 하지만 다른 Python 프로세스와 호환이 빨라져 입/출력이 수월하였다.

#### 2) 정확도


YOLO darkflow도 darknet과 같은 정확도를 보여주었다. 같은 YOLO v3의 가중치 값을 이용한다.

#### 3) 기타 사항

Python언어로 작성된 YOLO darkflow에서는 태그 키워드 추출이 간편하였다. json형태의 결과값에서 태그 정보만 추출하는 연구를 진행하였다.

### 3. tiny YOLO



 <b>국민대학교 컴퓨터공학부 캡스톤 디자인 I</b>	<b>중간보고서</b>		
	<b>프로젝트 명</b>	WittyPhotos	
	<b>팀 명</b>	메모리즈	
	Confidential Restricted	Version 1.3	2019-APR-18

- 1) 속도  
tiny YOLO는 이미지 한 장을 분석하는 데 약 1초의 시간이 소요되었다. 가장 빠른 속도를 보인 모델이었다.
- 2) 정확도  
tiny YOLO의 정확도는 가장 좋지 않았다. 거의 대부분의 이미지 객체를 인식하지 못하였기 때문에 속도가 가장 빠르지만 본 프로젝트에 적합하지 않다고 판단하였다.


#### 4. YOLO v3 with OpenCV (Python)

- 1) 속도  
OpenCV로 구현된 YOLO는 기본 darknet 모델보다 10배 이상 빠른 속도를 보였다. 이미지 한 장당 객체 인식하는데 약 2초가 소요되었다.
- 2) 정확도  
YOLO v3의 weight를 가져와 사용하므로 정확도는 기본 모델과 별다른 차이가 없었다.
- 3) 기타 사항  
정확도가 크게 떨어지지 않는 선에서 가장 빠른 모델이고 프로그램이 무겁지 않으며 Python언어를 사용하여 다른 프로세스와 연결도 수월하기 때문에 본 프로젝트에 적합한 모델이라 판단하여 최종 선정하였다.

#### - YOLO with OpenCV

OS	Framework	CPU/GPU	Time(ms)/Frame
Linux 16.04	Darknet	12x Intel Core i7-6850K CPU @ 3.60GHz	9370
Linux 16.04	Darknet + OpenMP	12x Intel Core i7-6850K CPU @ 3.60GHz	1942
Linux 16.04	OpenCV [CPU]	12x Intel Core i7-6850K CPU @ 3.60GHz	220
Linux 16.04	Darknet	NVIDIA GeForce 1080 Ti GPU	23
macOS	DarkNet	2.5 GHz Intel Core i7 CPU	7260
macOS	OpenCV [CPU]	2.5 GHz Intel Core i7 CPU	400

[그림 10 - 다양한 환경에서 YOLO 프레임워크들의 성능 비교]

 <b>국민대학교 컴퓨터공학부 캡스톤 디자인 I</b>	<b>중간보고서</b>		
	<b>프로젝트 명</b>	WittyPhotos	
	<b>팀 명</b>	메모리즈	
	Confidential Restricted	Version 1.3	2019-APR-18

최종적으로 본 프로젝트에서는 이미지 객체 인식을 위해 DNN 모듈을 사용한 OpenCV를 사용하여 CPU환경에서 YOLO를 구현한다.  
[그림 10]의 표에 나와있는 것과 같이 OpenCV는 같은 환경에서 다른 YOLO 프레임워크 보다 훨씬 빠른 속도를 보여준다.<sup>2</sup>

이후 다른 프로세스와 연결하기 위해 입/출력 코드를 수정하여 사용하였다.

- 계획서 상의 진도와 비교 분석

중간 발표 전까지 객체 인식 기능을 구현한다는 기존 계획에 맞게 진행되었다.

## ● 수동 이미지 태그

- 태그 수동 추가 기능 구현

Python언어로 SQLite3를 사용하여 사용자가 이미지에 태그를 수동으로 추가하고 데이터베이스에 해당 태그의 tagCount 값 증가 및 tag log 추가와 같은 기능을 구현하는 코드를 완성하였다.

- 태그 수동 삭제 기능 구현

Python언어로 SQLite3를 사용하여 사용자가 이미지에 지정된 태그를 삭제하고 데이터베이스에 해당 태그의 tagCount 값 감소 또는 태그 영구삭제, tag log 삭제와 같은 기능을 구현하는 코드를 완성하였다.

- 계획서 상의 진도와 비교 분석


계획 상의 진도와 동일하게 진행되었다.

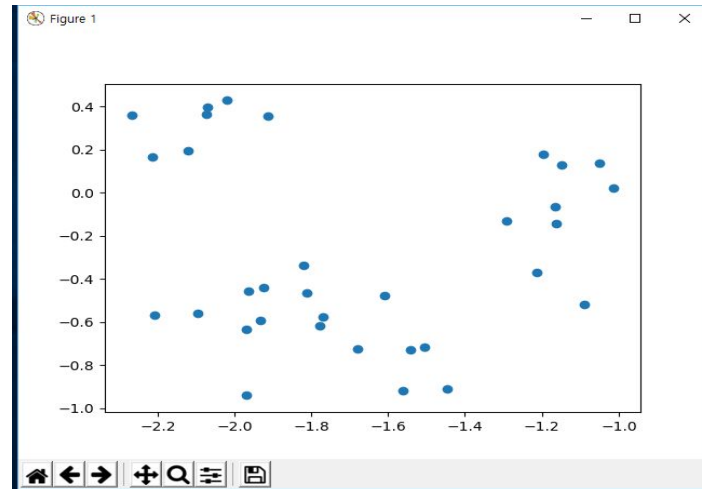
## ● 인물 태그 - face recognition (얼굴 인식)을 통한 태그

- 이미지에서 얼굴을 인식하고 같은 얼굴 끼리 구분하기 위해 다음과 같은 기능을 구현하였다.

1. Python의 face\_recognition package를 이용하여 face encoding 과정을 거쳐 각 이미지에 대한 얼굴의 128개의 벡터와 얼굴의 영역 박스를 추출하여 이미지 파일명과 박스의 위치, 128개의 벡터(BLOB형으로 전달)를 데이터베이스에 저장한다.
2. 추출한 벡터값이 적절한 값인지 확인하기 위해 인식된 모든 얼굴의 128개의 벡터값을 t-SNE를 이용하여 시각화하였다.  
(t-SNE : 고차원의 데이터를 저차원 데이터로 맵핑하여 시각화하는 알고리즘)

<sup>2</sup> <https://www.learnopencv.com/deep-learning-based-object-detection-using-yolov3-with-opencv-python-c/>

 <div> <p>국민대학교 컴퓨터공학부 캡스톤 디자인 I</p> </div>	중간보고서		
	프로젝트 명	WittyPhotos	
	팀 명	메모리즈	
	Confidential Restricted	Version 1.3	2019-APR-18



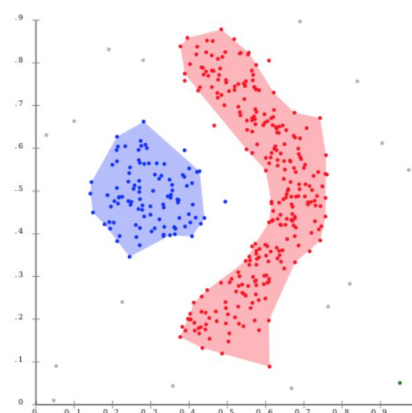
[그림 11 - 얼굴들의 벡터값을 t-SNE를 이용하여 시각화]

위 사진은 여러장의 사진에서 3명의 인물에 한하는 얼굴을 인식한 33개의 벡터를 나타낸 t-SNE이다.

### 3. 128개의 벡터값을 DBSCAN을 이용하여 클러스터링

#### DBSCAN


- 몇명의 인물에 대한 군집화가 필요할지 알 수 없기때문에 군집의 수를 정할 필요가 없는 DBSCAN을 사용하였다. 군집의 밀도에 따라서 서로 연결하기 때문에 기하학적인 모양을 갖는 군집도 잘 찾을 수 있다.
- 라벨이 0부터 각 다른 인물로서 분류되며 noise로 구분하는 경우는 라벨이 -1로 분류된다.

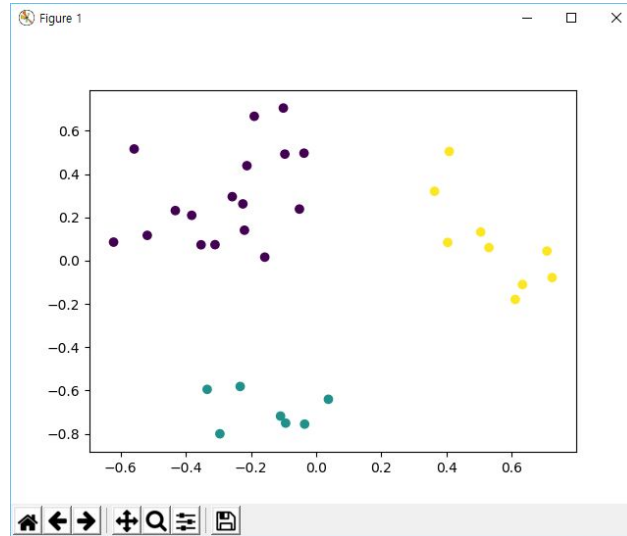


[그림 12 - DBSCAN으로 클러스터링한 결과 그래프]

### 4. DBSCAN에서 설정한 eps값과 min\_samples 값에 의해 각 얼굴에 추출되는 라벨이 변화한다.

적절한 eps 와 min\_samples의 값을 찾기 위해서 t-SNE를 사용하여 그래프를 확인해가면서 적절한 값을 찾았다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	중간보고서		
	프로젝트 명	WittyPhotos	
	팀 명	메모리즈	
	Confidential Restricted	Version 1.3	2019-APR-18



[그림 13 - eps와 min\_samples 값 수정후 t-sne 확인한 그래프]

- 계획서 상의 진도와 비교 분석  
계획한 진도와 동일하게 진행되었다.

## 2.2.2. 태그 카테고리화와 데이터베이스

- 태그 카테고리화
  - 카테고리는 총 9개이며 인물, 사물, 동물, 식물, 음식, 지역, 장소, 색깔, 기타가 포함되어있다.
  - 자동 태그들은 먼저 카테고리를 지정하여 데이터베이스에 저장하였다.
  - 사용자가 태그를 추가할 때 카테고리를 반드시 설정하도록 코드를 작성하였다.

categoryID	category_name
필터	필터
1	인물
2	사물
3	동물
4	식물
5	음식
6	지역
7	장소
8	색깔
9	기타

[그림 14 - 데이터베이스에 저장된 카테고리 테이블]


tagID	EnglishTag	KoreanTag	tagCount	tagState	categoryID
필터	필터	필터	필터	필터	필터
n00000001	person	사람	0	1	1
n00000002	bicycle	자전거	0	1	2
n00000003	car	자동차	0	1	2
n00000004	motorbike	오토바이	0	1	2
n00000005	aeroplane	비행기	0	1	2
n00000006	bus	버스	0	1	2
n00000007	train	기차	0	1	2
n00000008	truck	트럭	0	1	2
n00000009	boat	보트	0	1	2

[그림 15 - 자동 태그 테이블. categoryID가 지정되어있다.]

- 데이터베이스
- 데이터베이스 설계안은 다음과 같다.

### 1. [Category] 테이블

Category 테이블에는 태그들이 속할 9개의 카테고리가 저장된다.  
필요한 기능 및 요구사항은 다음과 같다.

 <b>국민대학교 컴퓨터공학부 캡스톤 디자인 I</b>	<b>중간보고서</b>		
	<b>프로젝트 명</b>	WittyPhotos	
	<b>팀 명</b>	메모리즈	
	Confidential Restricted	Version 1.3	2019-APR-18

1) Category 테이블에는 category name 애트리뷰트가 있으며 인물, 사물, 동물, 식물, 음식, 지역, 장소, 색깔, 기타 총 9개가 입력될 수 있다. (TEXT형)

2) 통계 및 분석 등의 기능을 구현할 때 불필요한 실수를 줄이고자 category ID 애트리뷰트를 만든다.  
(key) category ID 는 1부터 9까지의 자연수로 나타낸다. (INTEGER형)

3) Category 엔티티는 [Object\_Auto\_tag], [Object\_manual\_tag], [Face\_tag] 테이블과 category ID를 통해 관계를 맺는다.

## 2. [Object\_Auto\_tag] 테이블

Object\_Auto\_tag 엔티티는 이미지속의 사물을 자동으로 인식하여 추출한 태그를 다룬다.  
필요한 기능 및 요구사항은 다음과 같다.

1) 자동 태그는 YOLO를 사용하여 구현하는데, YOLO는 wordnet에서 가져온 9419개의 (영어)단어를 사용하고 각 단어에는 n00000000 (명사를 뜻하는 n과 숫자 8개)와 같은 번호가 붙는다.  
이 번호를 그대로 tag ID로 사용한다. (Key) (TEXT형)

2) English Tag 애트리뷰트에 자동 태그로 입력될 수 있는 9419개의 영어 단어와 각 단어가 속한 카테고리(category id)를 먼저 저장한다.  
중복된 단어가 있으므로 중복을 허용하고 id로 구별한다.


3) 사용자에게 한글 태그를 보여줄 것이므로 Korean Tag 애트리뷰트를 만들어서 YOLO에서 추출된 영어 단어를 데이터베이스의 English Tag 애트리뷰트에서 검색하고 같은 tag ID를 가진 한글 단어(Korean Tag)를 찾아서 보여준다.

4) 사용자의 이미지에 태그가 입력될 때마다 tagCount애트리뷰트 값이 하나 증가하고 사용자가 기존에 입력했던 태그를 삭제할 때 tagCount 애트리뷰트 값이 하나 감소한다. default값은 0이다.

5) 자동으로 입력되는 태그가 최대한 사용자의 의도와 일치되도록 9419개의 단어 중 사용하지 않을 단어는 tagState 값을 0으로 갖고 사용하고 있는 단어는 tagState값을 1로 갖는다. 먼저 개발자의 판단에 따라 불필요한 단어는 default값을 0, 필요하다 생각되는 단어(약 3000개)는 default값을 1로 한다.  
이후 사용자가 불필요하다 생각되는 단어는 태그되지 않게 하는 기능(자동 태그되는 단어 삭제 메뉴)을 만들어 그 단어의 tagState값을 0으로 바꾼다.  
그 중 다시 사용하고 싶은 단어는 tagState값을 1로 되돌릴 수 있게 한다.  
따라서 이미지 인식 결과 default값이 0인 단어가 추출될 경우 그 트리 노드를 따라서 상위 단어중 tagState값이 1인 가장 가까운 단어를 리턴한다. tagState값이 1인 상위 단어가 트리의 level 2까지 나오지 않을 경우 아무것도 리턴하지 않는다.  
tagState값이 0인 태그는 태그 통계 및 네트워크 분석 기능에 포함되지 않는다.  
(BOOLEAN 형)

6) [Category] 엔티티의 category ID를 참조한다. (NULL값)

## 3. [Object\_manual\_tag] 테이블

 <b>국민대학교 컴퓨터공학부 캡스톤 디자인 I</b>	<b>중간보고서</b>		
	<b>프로젝트 명</b>	WittyPhotos	
	<b>팀 명</b>	메모리즈	
	Confidential Restricted	Version 1.3	2019-APR-18

Object\_manual\_tag 엔티티는 사용자가 직접 입력한 태그를 다룬다. 필요한 기능 및 요구사항은 다음과 같다.

- 1) 수동 태그도 태그ID가 필요하므로 m00000000 (수동을 뜻하는 m과 숫자 8개)와 같이 만든다. (Key)
- 2) 사용자가 직접 태그를 입력할 경우 영어와 한글 태그가 가능하다. 자동 태그와 다르게 영어 태그와 한글 태그 애트리뷰트를 따로 만들지 않고 tag 애트리뷰트만 만든다. (중복 불가, TEXT형)
- 3) 사용자가 태그를 입력할 때 카테고리도 반드시 지정할 수 있게한다. [Category] 엔티티의 category ID를 참조한다.
- 4) 사용자의 이미지에 태그가 입력될 때마다 해당 태그의 tagCount애트리뷰트 값이 하나 증가하고 사용자가 기존에 입력했던 태그를 삭제할 때 tagCount 애트리뷰트 값이 하나 감소한다. default값은 1이다.

#### 4. [Face\_tag] 테이블


Face\_tag 엔티티는 자동으로 이미지 속에서 사람의 얼굴을 찾은 후 사용자가 입력한 사람의 이름 태그를 다룬다.

- 1) 얼굴 태그의 tagID는 f00000000 (얼굴을 뜻하는 f와 숫자 8개)와 같이 만든다. (key)
- 2) 사용자가 직접 태그를 입력할 경우 영어와 한글 태그가 가능하다. 자동 태그와 다르게 영어 태그와 한글 태그 애트리뷰트를 따로 만들지 않고 name 애트리뷰트만 만든다. (중복 불가 동명이인은 나중에 처리)
- 3) 모든 태그의 카테고리는 인물이다. 따라서 따로 사용자에게 카테고리를 입력 받을 필요가 없이 categoryID 애트리뷰트는 인물로 지정한다. [category]엔티티의 category ID를 참조하고 default 값을 인물로 지정한다.
- 3) 수동 태그와 마찬가지로 사용자의 이미지에 태그가 입력될 때마다 해당 태그의 tagCount애트리뷰트 값이 하나 증가하고 사용자가 기존에 입력했던 태그를 삭제할 때 tagCount 애트리뷰트 값이 하나 감소한다. default값은 1이다.

#### 5. [Face\_vector\_value] 테이블

Face\_vector\_value 엔티티는 face recognize 프로세스를 거쳐 추출된 128개의 vector 값을 다룬다.

- 1) vector id는 INTEGER형이며 1부터 입력된 순서대로 증가한다. (key)
- 2) imageFileName 애트리뷰트에 얼굴이 인식된 해당 이미지 파일의 파일 명을 저장한다. (TEXT 형)
- 3) imageLeft, imageTop, imageRight, imageBottom 애트리뷰트에 얼굴이 인식된

 <b>국민대학교 컴퓨터공학부 캡스톤 디자인 I</b>	<b>중간보고서</b>		
	<b>프로젝트 명</b>	WittyPhotos	
	<b>팀 명</b>	메모리즈	
	Confidential Restricted	Version 1.3	2019-APR-18

이미지 안의 box값을 저장한다. (INTEGER형)

4) vector 값은 vector value 애트리뷰트에 저장된다. (BLOB형)

5) faceName 애트리뷰트에 [Face\_tag] 엔티티의 face ID 애트리뷰트를 참조하여 face clustering 프로세스를 거쳐 추출된 사람의 이름을 저장한다. 없을 경우는 null값을 허용한다.

## 6. [Images] 테이블

Images 엔티티는 사용자가 갖고있는 모든 이미지들과 이미지들의 정보를 다룬다. 이미지를 그대로 저장할 수도 있지만 속도가 느리고 db용량이 커지기 때문에 로컬 파일 시스템에 저장하고 경로를 db에 저장한다.

1) imageID 애트리뷰트는 이미지 파일의 id값을 저장한다.

2) imageFileName 애트리뷰트에 이미지 파일명과 형식을 저장한다 (ex. img1.jpg) (key, TEXT형)

2) imageFilePath 애트리뷰트에 경로를 저장한다. (key, TEXT형)

3) date, time, gpsLat, gpsLon 애트리뷰트에 이미지의 시간과 위치 정보를 저장한다. Null값을 허용한다.

## 7. [Tag\_log] 테이블

Tag\_log엔티티는 각 이미지에 달린 태그들을 저장한다.

1) imageID 애트리뷰트에 이미지 파일 id값을 저장한다. [images]엔티티의 imageID 애트리뷰트를 참조한다.


2) Tag NO 애트리뷰트는 각 이미지 파일에 담긴 태그의 번호를 저장한다. INTEGER형이며 이미지 파일 하나 당 1부터 20까지 입력된 순서대로 증가한다. Tag NO가 20이 되면 더 이상 태그를 저장하지 못한다. imageID와 함 다중키로 사용된다.

3) autoTagID, manualTagID, faceTagID 애트리뷰트가 있으며 자동 태그로 지정된 태그 id는 autoTagID에, 수동 태그로 지정된 태그 id는 manualTagID에, 얼굴 태그로 지정된 태그 id는 faceTagID에 저장되고 각각은 [Object\_auto\_tag] [Object\_manual\_tag] [face\_tag] 엔티티를 참조한다.

사용자의 라이프 스타일을 분석하기 위해 모든 태그들은 특정 카테고리 안에 속해 있어야 한다. 태그들과 연결된 카테고리는 데이터베이스에서 관리한다. 카테고리는 인물, 동물, 식물, 사물, 음식, 색상, 장소, 기타 총 8개로 나누어져 있다.

- 계획서 상의 진도와 비교



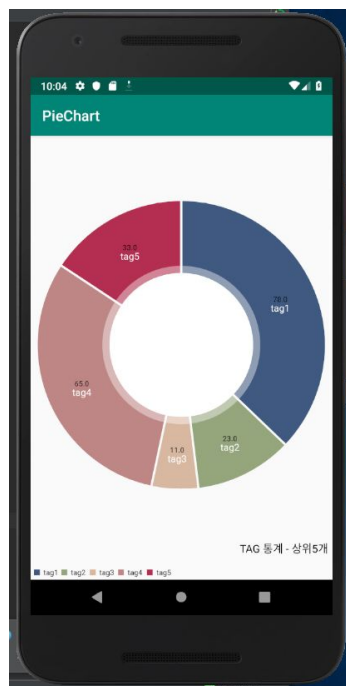
 국민대학교 컴퓨터공학부 캡스톤 디자인 I	중간보고서		
	프로젝트 명	WittyPhotos	
	팀 명	메모리즈	
	Confidential Restricted	Version 1.3	2019-APR-18

계획서 상의 진도와 동일하게 진행하였다.

### 2.2.3. 통계 및 네트워크 분석


#### ● 태그 통계

많이 사용된 태그의 통계 데이터를 추출하고 추출한 데이터를 파이 그래프 모양으로 시각화 하기 위해 MPAndroidChart 라이브러리를 이용하여 코드를 작성하였다. 그리고 작성한 코드를 안드로이드 어플리케이션에서 구동하는 작업을 하였다.



[그림 16 - pie chart]

#### ● 네트워크 분석

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	중간보고서		
	프로젝트 명	WittyPhotos	
	팀 명	메모리즈	
	Confidential Restricted	Version 1.3	2019-APR-18

## 1. 전체 네트워크에서 군집화

네트워크에서 군집을 추출하여 시각화하기 위해 알고리즘에 대한 조사를 하였다.

### - divisive algorithm

네트워크가 분열되도록 군집간의 링크를 잘라나가는 방법이다. inter-community link를 추정하는 대표적인 방법이 링크의 betweenness 중심성을 조사하는 방법으로, Girvan-newman algorithm이 이와같은 방법을 사용한다.

### - agglomerative algorithm

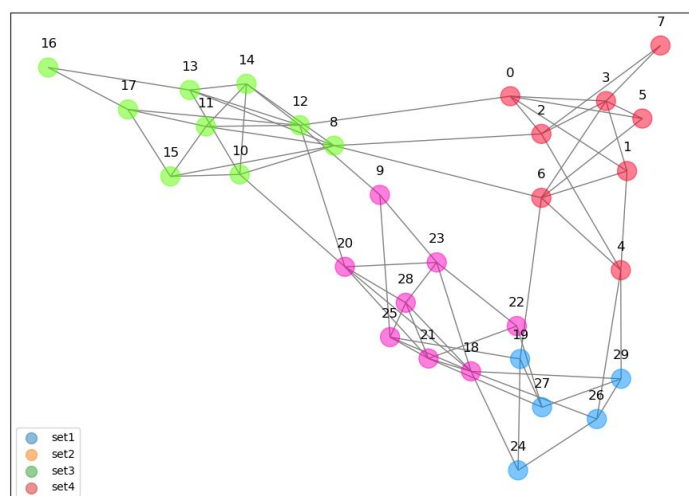
비슷한 노드, 군집을 점진적으로 합쳐나가는 방법이다. 주로 클러스터링 방법을 이용하며 link community같은 방법이 있다.

### - Louvain algorithm


한개의 노드로부터 주변 노드를 흡수하여 군집을 생성해가는 것은 greedy algorithm과 비슷하지만 위의 두가지 방법에 비해 modularity 계산법과 네트워크의 계층적 구조를 이용하는 개선책을 적용하고 있다.

## 2. python networkx라이브러리를 이용하여 군집 분류를 확인하고 어떠한 식으로 나타나는지 코드를 작성하여 테스트 하였다.

Girvan Newman algorithm : 복잡하게 구성되어 있는 전체 커뮤니티(강하게 연결되어 있는 노드들)를 분석하는데 유용하게 사용되는 알고리즘이다. betweenness centrality 기반으로 병목지점에 위치하고 있어 소통량이 많은 링크를 잘라 커뮤니티를 나눈다.



[그림 17 - girvan\_newman algorithm으로 군집을 나누어 시각화한 그래프]

 <b>국민대학교 컴퓨터공학부 캡스톤 디자인 I</b>	<b>중간보고서</b>		
	<b>프로젝트 명</b>	WittyPhotos	
	<b>팀 명</b>	메모리즈	
	Confidential Restricted	Version 1.3	2019-APR-18

### 3. 군집화 내의 네트워크

노드의 중요도에 따라 크기를 다르게 표현하여 사용자가 중요한 태그를 쉽게 파악할 수 있게 하기 위해 python networkx 라이브러리 이용, 네가지 알고리즘으로 각 노드에 대한 centrality를 출력하고 그래프 시각화를 테스트했다.

- 1) degree : 한 노드에 직접 연결된 이웃 노드의 수가 많을 수록 중요한 노드
- 2) closeness : 한 노드에서 다른 노드로 가는 경로가 짧을 수록 중요한 노드
- 3) betweenness : 한 노드를 제외한 다른 노드들이 서로 이동할때 그 노드를 많이 거쳐가면 중요한 노드


- 계획서 상의 진도와 비교 분석

태그 통계 그래프 형성까지가 중간평가까지 계획한 진도였기때문에 비슷하게 진행되었다.

## 2.2.4. UI/UX 디자인 설계 및 안드로이드 어플리케이션 구현

### ● 안드로이드 레이아웃 디자인(Xml)


- 스플래쉬 화면
  1. 자체 디자인한 WittyPhotos의 로고를 사용해 1.5초 스플래쉬 화면을 구성한다.
- 메인 태그/그래프 선택하는 화면
  1. WittyPhotos의 로고가 담겨있는 메인 이미지와 함께 자체 디자인한 세부 카테고리 버튼을 vertical로 표시한다.  
: 한 손 작동을 선호하는 유저의 수가 증가한다는 점과 왼손을 사용하는 유저를 고려하여 Horizontal 보다는 Vertical 레이아웃을 사용하기로 했다.
- 앨범 카테고리 선택 후 앨범 카테고리 및 이미지 노출
  1. 기기 내부에 저장되어 있는 앨범 카테고리를 RecyclerView를 통해 보다 효율적인 ListView를 보여준다.  
: 앨범 내 카테고리가 아주 많을 수 있다는 점을 고려하여 스크롤 이동시 자동적으로 화면 재활용이 가능한 RecyclerView를 사용하기로 했다.
  2. 세부 앨범 디렉토리 선택 후 디렉토리 내 이미지를 노출한다.  
: 가장 많이 사용하고 있는 레이아웃인 그리드 방식을 사용하며 행당 노출되는 이미지의 개수는 3장이며 추후 수정가능하다.
- 그래프 카테고리 선택 후 그래프 노출
  1. 시각화된 그래프를 유저에게 직관적으로 전달한다.

 <b>국민대학교 컴퓨터공학부 캡스톤 디자인 I</b>	<b>중간보고서</b>		
	<b>프로젝트 명</b>	WittyPhotos	
	<b>팀 명</b>	메모리즈	
	Confidential Restricted	Version 1.3	2019-APR-18

: 다양한 그래프의 시각화를 구상하고 있기 때문에 다수의 그래프가 생성될 것이다. 다른 타입의 그래프를 보기 위해 일련의 프로세스를 반복할 필요없이 View Pager를 통해 시각화된 그래프를 간단한 슬라이드 동작을 통해 유저에게 전달한다.

## ● 안드로이드 동작 구현(Java)


- 스플래쉬 화면이 뜨고 메인 화면으로 넘어가는 동작
  1. onCreate() 메소드 내 run() 함수를 구현하여 activity\_splash화면을 시작하고 splash화면이 다시 호출되지 않도록 finish()한다.
- 메인 화면에서 이미지 버튼을 통해 activity로 이동하는 동작
  1. startActivity()를 통해 현재 activity와 이동할 activity를 설정해준다.
- 앨범 메뉴가 실행되면 기기내의 이미지를 동기화하는 동작
  1. ImagePickerDirectoryActivity.class : 실제 앨범 카테고리가 보여지는 기능을 구현한 클래스이다.
    - ImagePickerAdapter 를 이용해 이미지 데이터 값을 ArrayList로 받아온다.
    - onCreateViewHolder 와 onBindViewHolder 를 이용해 앨범 카테고리 버킷을 받아오고 레이아웃에 표시한다. onBindViewHolder를 사용해서 버킷화 되어있는 카테고리의 dirName과 카테고리별 이미지 개수를 보여주는 dirCounter를 표시한다.
  2. PickerDir.class : 기기의 이미지를 가져 오기 위한 element들을 정의하고 받아오는 기능을 구현한 클래스로 dirName, dirId, imgResId, imgPath, count로 구성
  3. ImgPickerDirController.class : ImagePickerDirectoryActivity 클래스가 작동하기 위해서 화면 background에서 돌아가는 동작을 구현한 클래스이다.
    - LoadImageDirList 클래스
      - AsyncTask - doInBackground 를 이용해 동작한다.
      - MediaStore.Images.Media 에서 DATA, \_ID, BUCKET\_DISPLAY\_NAME, BUCKET\_ID를 받아와서 실질적인 이미지 경로를 저장한다.
      - Cursor를 이용해 이미지 데이터를 움직이면서 이미지 세부의 카테고리화를 시작한다. 이 부분에서 각 이미지의 개수를 명시하는 dirCounter의 개수가 저장된다.
      - 이미지를 HashMap으로 받아 이미지 서치에 용이하게 배열한다.
    - onPreExecute() : 실행 전 준비동작하는 기본적인 함수이다.
    - onPostExecute(): 클래스 내의 모든 함수가 작동하고 끝내는 onComplete함수를 불러들인다.

 <b>국민대학교 컴퓨터공학부 캡스톤 디자인 I</b>	<b>중간보고서</b>		
	<b>프로젝트 명</b>	WittyPhotos	
	<b>팀 명</b>	메모리즈	
	Confidential Restricted	Version 1.3	2019-APR-18

4. ImagePickerDetailActivty.class : 위의 ImgPickerDirectoryActivity의 함수와 기능을 같이 한다. ImagePickerDetailActivity는 디렉토리가 아닌 이미지 한 장 한 장을 보여줄 수 있는 동작을 구현한 클래스이다.
    - init()을 이용해 실질적인 이미지 아이템 데이터 값을 표시한다.
  5. PickerImage.class : 이미지 마다 가지고 있는 imgPath를 Uri값으로 받고 저장하는 동작을 한다. boolean을 이용해 imgPath가 올바른지 더블 체크한다.
  6. ImgPickerDetailController.class : ImgPickerDirectoryController와 동일한 동작을 하고 추가적으로 각 이미지의 path값을 사용한다.
- 세부 이미지를 클릭한 화면에서 태그 추가 버튼을 눌러서 태그 추가 창이 뜨는 동작
    1. image\_tag\_list 화면에서 onClick() 메소드 내 Intent() 함수를 사용하여 기존 화면에서 태그 추가하는 창이 뜨도록 한다.
  - 태그 추가 '확인' 버튼을 누르면 태그가 추가되는 동작
    1. 태그 추가 창에서 태그명을 넣고 '확인' 버튼을 누르면 getText()로 태그 문자열을 받고 창을 닫는다.
    2. 문자열은 setText()로 태그로 추가된다.

## 2.2.5. 웹 서버와 웹 클라이언트 구축

- 웹 서버
  - Django를 사용하여 웹 프레임워크 생성
  - PythonAnywhere를 사용하여 테스트용 웹 클라우드 서버 구축
  - AWS를 사용하여 실제 배포용 웹 클라우드 서버 구축
- 웹 클라이언트
  - HTTP 통신에서 어떤 자원에 대한 CRUD 요청을 Resource와 Method로 표현하여 특정한 형태로 전달하는 REST 방식을 사용하여 웹 클라이언트 구축
  - 가장 기본적인 방법이자 HTTP 표준을 준수하고 확장하여 다양한 request방식을 만들기 용이한 HttpURLConnection를 사용하여 클라이언트 구축

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	중간보고서		
	프로젝트 명	WittyPhotos	
	팀 명	메모리즈	
	Confidential Restricted	Version 1.3	2019-APR-18

### 3. 수정된 연구내용 및 추진 방향

#### 3.1. 수정사항

##### 3.1.1. 이미지 태그 기능 구현

- 변경 사항

- [보완 사항] OpenCV로 YOLO v3 구동

제안서에 작성한 YOLO V3는 darkflow, tiny YOLO 등 여러가지 방법으로 사용될 수 있다. 그 중 최대한 속도를 높이면서 정확도를 유지할 수 있는 모델을 구현하기 위해 OpenCV로 YOLO V3를 구현하였다.

- 추진 방향

- 자동 이미지 태그

현재 구현된 이미지 객체 인식 기능은 80개의 단어 데이터셋으로 태그를 추출하기 때문에 다양한 태그를 제공하지 못한다. 따라서 향후에 단어 데이터셋을 늘리면서도 정확도를 유지시키는 작업이 필요하다.

- 수동 이미지 태그

안드로이드 어플리케이션과 연결되기 전 기능 테스트를 위해 Python언어로 작성된 태그의 추가와 삭제 기능을 자바로 수정하여 안드로이드 어플리케이션에서 구동한다.

- 인물 태그


테스트용 이미지로 얼굴 라벨이 정확히 분류되었는지 확인하여 인물 태그 기능의 정확도를 향상시킨다.

##### 3.1.2. 태그 카테고리화와 데이터베이스

- 변경 사항

- [추가 사항] 지역 카테고리 추가

이미지의 GPS정보를 구글맵 API를 사용하여 국가와 도시 태그를 추출하여 저장하기 때문에 데이터베이스의 Category 테이블에 지역 카테고리가 추가로 필요하였다.

 <b>국민대학교 컴퓨터공학부 캡스톤 디자인 I</b>	<b>중간보고서</b>		
	<b>프로젝트 명</b>	WittyPhotos	
	<b>팀 명</b>	메모리즈	
	Confidential Restricted	Version 1.3	2019-APR-18

- [보완 사항] Python언어로 SQLite3를 사용하여 테스트용 데이터베이스 구축

완성된 프로젝트에서는 안드로이드 어플리케이션에서 사용자의 입력을 받아 어플리케이션 내부 데이터베이스에 추가하는 기능을 구현해야 하지만 어플리케이션과 Python으로 작성된 Back-end의 기능이 연결 되기 전에 Python언어로 SQLite3를 사용하여 테스트용 데이터베이스를 구축하여 기능이 제대로 동작하는지 테스트하였다.

- 추진 방향

- 안드로이드 어플리케이션 내부 데이터베이스를 구축한다.
- 안드로이드 어플리케이션에서 태그 검색, 태그 추가, 태그 삭제 기능이
- 데이터베이스와 연결될 수 있게 한다.
- 데이터베이스에 Network 테이블을 만들어 네트워크 분석에서 사용할 정보를 제공할 수 있도록 한다.
- Network테이블은 태그들의 동시 출현 빈도를 저장해두는 기능을 한다.

### 3.1.3. 통계 및 네트워크 분석

- 변경 사항

- [보완 사항] 서버 의존도를 낮추기 위해 통계와 네트워크 분석 프로세스는 Java언어를 사용하여 안드로이드 어플리케이션에서 실행하도록 변경하였다.

- 추진 방향

- python 코드로 작성하여 테스트한 네트워크 분석 코드를 Java로 작성한다.
- 네트워크 분석을 위한 알고리즘들을 더 연구하여 비교 분석한 후 사용자에게 좀 더 유익한 정보를 제공할 수 있도록 한다.

### 3.1.4. UI/UX 디자인 설계 및 안드로이드 어플리케이션 구현

- 변경 사항

- [보완 사항] Viewpager 사용


 <b>국민대학교 컴퓨터공학부 캡스톤 디자인 I</b>	<b>중간보고서</b>		
	<b>프로젝트 명</b>	WittyPhotos	
	<b>팀 명</b>	메모리즈	
	Confidential Restricted	Version 1.3	2019-APR-18

Chart 항목을 선택하였을 때, Graph 혹은 Pattern 카테고리를 선택하는 것에서 한 화면에서 모든 그래프를 다 보여주는 것으로 변경

- [보완 사항] 최하단에 태그 삭제 버튼 추가

태그 리스트에서 삭제할 태그 선택 후 삭제하는 것에서 태그 삭제 버튼을 누른 뒤 지우고 싶은 태그를 선택하는 것으로 변경

- 추진 방향

- 안드로이드 스튜디오에서 xml파일로 앱 레이아웃을 만들고 Java 코딩을 통해 레이아웃 간 동작을 구현한다.

### 3.1.5. 웹 서버와 웹 클라이언트 구축

- 변경 사항

- [추가 사항] Django

제안서에는 파이썬 프로그램을 서버에서 구동할 구체적인 방법을 제시하지 않았으나 파이썬 프레임워크 중 가장 범용성이 높은 Django를 사용하여 웹 서버를 개발한다.

- [삭제 사항] AWS RDS

초기 제안서에 기술되었던 AWS 서버 데이터베이스는 백업 용도로 사용하기 위해 구축하기로 하였다. 하지만 수많은 이미지와 태그를 관리하는 본 프로젝트의 특성상 데이터베이스의 업데이트가 매우 잦기 때문에 동기화하는데 시간 소모가 클 것으로 예상되므로 전체 프로세스 시간을 줄이기 위해 불필요한 작업을 줄이고자 내부 데이터베이스만 구축하여 사용하기로 계획을 변경하였다.


- [보완 사항] PythonAnywhere 사용

실제로 안드로이드 어플리케이션과 연결할 배포용 AWS 서버를 사용하기 전에 파이썬 개발자들이 많이 사용하는 PythonAnywhere 클라우드 서버를 활용하여 테스트용 웹 서버를 구축하기로 계획을 변경하였다.

- 추진 방향

PythonAnywhere 클라우드 서버에서 테스트한 경험을 토대로 AWS에 파이썬 프로그램을 구동시키고 안드로이드 클라이언트에서 이미지 데이터를 받아 태그를 전송하는 코드를 작성한다.



 국민대학교 컴퓨터공학부 캡스톤 디자인 I	중간보고서		
	프로젝트 명	WittyPhotos	
	팀 명	메모리즈	
	Confidential Restricted	Version 1.3	2019-APR-18

## 4. 향후 추진계획

### 4.1. 향후 계획의 세부 내용

#### 4.1.1. 이미지 태그 기능 구현

- 자동 이미지 태그

단어 데이터셋을 최소 300개 까지 증가시켜 사용자에게 더 많은 자동 태그를 제공한다. 새롭게 단어를 학습 시키는 방식이 아닌 Image.net 등의 데이터셋을 이용함으로써 이미 딥러닝으로 학습된 단어들의 가중치를 추가하는 방식으로 진행한다.

데이터셋이 많아지면 정확도가 오히려 낮아지는 경우가 있으므로 Pascal VOC를 활용하여 정확도를 체크하며 데이터셋을 증가 시킴으로써 정확도를 유지 또는 향상시킨다.

- 수동 이미지 태그

Python으로 작성한 수동 태그 추가, 삭제 코드를 자바로 변환하여 안드로이드 어플리케이션에서 구동할 수 있게 한다.


- 인물 태그

DBSCAN의 적절한 eps, min\_samples 값을 찾아내기 위해 최대한 데이터를 많이 수집한 후 t-SNE 을 사용하여 얼굴 라벨이 적절히 분류되었는지 그래프를 확인해 보는 방식으로 가장 적합한 eps, min\_samples 값을 찾아내어 정확도를 향상시키는 작업이 필요하다.

#### 4.1.2. 데이터베이스

- 데이터베이스

- SQLite3을 사용하여 안드로이드 어플리케이션에 데이터베이스를 구축한다.
- Python언어를 사용하여 만들었던 태그 검색, 태그 추가, 태그 삭제 등의 함수를 Java언어로 바꾸어 안드로이드 어플리케이션에서 구동한다.
- Network 테이블을 만들어 네트워크 분석 기능에 사용될 수 있게한다. Network 테이블은 태그들의 동시 출현 빈도를 저장하는 테이블이다.

 <b>국민대학교 컴퓨터공학부 캡스톤 디자인 I</b>	<b>중간보고서</b>		
	<b>프로젝트 명</b>	WittyPhotos	
	<b>팀 명</b>	메모리즈	
	Confidential Restricted	Version 1.3	2019-APR-18

#### 4.1.3. 통계 및 네트워크 분석

- 통계 그래프

안드로이드와 데이터베이스가 연결되면 데이터베이스에서 태그 출현 빈도를 받아 오는 코드를 추가 작성한다.

- 네트워크 그래프

##### 1) 전체 네트워크 군집

어플 내에서 데이터베이스 접근을 위해 자바로 코드 작성

Girvan\_newman algorithm, Link community, Louvain algorithm

세 가지 커뮤니티 추출에 대해 더 연구하여 속도와 모듈성, 어떤 정보를 효과적으로 전달할 수 있을지 비교 분석하여 최종적으로 사용할 알고리즘을 선택한다.

군집을 찾아내고 각 노드가 속해 있는 군집 ID를 데이터베이스로 넘긴다.

##### 2) 군집에 내에서의 네트워크

같은 군집 ID를 갖는 노드들의 centrality 계산 코드 자바로 작성

##### 3) 안드로이드에서 그래프 시각화

GraphStream 라이브러리를 이용하여 그래프 생성한다.

전체 네트워크에서는 분류된 커뮤니티에 따라 색상을 달리한다.

centrality에 따라 노드의 크기를 다르게 하고 weight에 따라 색상이나 굵기를 다르게하여 영향력 있는 태그와 연결성을 한 눈에 파악할 수 있도록 한다.

- 통계와 네트워크 분석을 통한 코멘트 제공

통계와 네트워크 분석 코드가 완성되면 결과로 나온 정보를 사용하여 정해진 format으로 코멘트를 만들어 사용자에게 제공한다.


#### 4.1.4. UI/UX 디자인 설계 및 안드로이드 어플리케이션 구현

- 안드로이드 애플리케이션 완성

##### 1) 서버를 통한 이미지 자동 태그 데이터 GET / POST 완료

- 연결된 서버를 통한 태그 데이터 추가/삭제 기능 및 실시간으로 수정된 태그 데이터 리스트 노출

##### 2) 네트워크 및 태그 통계 데이터 베이스 그래프 시각화


 <b>국민대학교 컴퓨터공학부 캡스톤 디자인 I</b>	<b>중간보고서</b>		
	<b>프로젝트 명</b>	WittyPhotos	
	<b>팀 명</b>	메모리즈	
	Confidential Restricted	Version 1.3	2019-APR-18

- MPAndroidChart 와 플랜비의 라이브러리 리서치 필요
- 3) 보다 편리한 UX 추가 리서치
  - 한 손 작동, 왼손잡이를 고려한 UX 연구
- 4) 사용자 인터페이스 구축 완료
  - 컬러 팔레트 완성 및 동작 구현 된 액티비티에 레이아웃 추가 완성
- 5) 푸시알림을 위한 코딩
  - firebase를 활용한 푸시알림
- 6) 서버
  - 기기 내의 새로운 이미지를 웹 서버에 전송하고 웹 서버로부터 태그 데이터값을 받아오는 웹 클라이언트 구축
- 7) db
  - 안드로이드 내부 db 구축
- 8) 안드로이드와 파이썬 연결

#### 4.1.5. 웹 서버와 웹 클라이언트 구축

현재까지 PythonAnywhere으로 구축한 웹 클라우드 서버와 안드로이드 클라이언트 간의 통신 연결이 완료되었고 향후 개발 내용은 다음과 같다.

- 웹 서버
  - PythonAnywhere 클라우드 서버에서 테스트한 내용을 토대로 AWS(Amazon Web Server)에서 Django를 사용하여 파이썬 프로그램을 실행시킨다.
  - 이미지 데이터를 클라이언트에서 전달받아 파이썬으로 작성된 객체 인식과 얼굴 인식 프로그램에 전달하는 프로그램을 작성한다.
  - 이미지의 객체를 인식한 후 추출된 태그 데이터를 클라이언트에 전송하는 프로그램을 작성한다.
  - 이미지에서 얼굴을 인식하여 추출된 128개의 벡터 값을 클라이언트에 전송하는 프로그램을 작성한다.
  - 클라이언트와의 통신이 제대로 작동하는지 테스트한다.
- 웹 클라이언트
  - 어플리케이션에 새롭게 동기화된 이미지들을 서버에 전달하는 코드를 작성한다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	중간보고서		
	프로젝트 명	WittyPhotos	
	팀 명	메모리즈	
	Confidential Restricted	Version 1.3	2019-APR-18

- 서버에서 전송 받은 데이터를 안드로이드 내부 데이터베이스에 저장하는 코드를 작성한다.

## 5. 고충 및 건의사항