졸업작품 연구결과보고서

이모지톡 (지능형 채팅 서비스)

2020 년 12 월 03 일

조다운, 오하영

지도교수: 최민 (사인)

충북대학교 전자정보대학 정보통신공학부장 귀하

목 차

I. 연구 배경 및 필요성

1.1 연구 배경

최근 인터넷 보급과 발전으로 빅데이터, 인공지능 등의 개념이 등장하게 되었고, 방대한 데이터의 양과 인터넷 속도의 향상은 인공지능 기술의 기반 기술인 머신 러닝이 가능한 환경을 제공해 주었으며, 빅데이터의 등장으로 알고리즘 위주의 인공지능 방향이 데이터와 경험 중심의 사고로 바뀌면서 인공지능 기술은 급격히 발전해왔다. 추가로 오픈소스 소프트웨어가 활성화 되면서 소프트웨어 발전에 큰 기여를 하고 있다. 여기서 오픈소스 소프트웨어란 소스 코드를 공개하여 누구나 수정 및 재배포 할 수 있는 소프트웨어를 의미한다. 인공지능 기술의 요소 기술인 머신러닝, 빅 데이터, 클라우드 등 최신 기술들 역시 오픈소스 소프트웨어 특성을 통해 급격히 발전하고 있다. 인공 지능이 발전함에 따라 인공지능의 인간중심 디자인에 대한 중요성이 더욱 부각되고 있다. 모션 인식과 사물을 분별하는 기술에 대해서는 상용화 된 어플리케이션과 기기에 기본 내장된 어플리케이션을 통해 쉽게 접해 볼 수 있다. 하지만 카메라를 통해 사용자의 감정을 인식하고 인식한 내용에 따라 서비스를 제공하는 분야에 대해서는 아직 개발 진행 중인 상태이다. 사용자의 얼굴에는 위치, 각도, 성별, 인종 외에도 현재 심리상태, 생리학적 신호 등과 같은 더 많은 정보가 남아있다. 현재 인공지능 기술에서 더 인간중심적이고, 사용자를 이해할 수 있는 인공지능 기술을 위해서는 물리적인 정보 이외에도 생물학적인, 인문학적인 이해를 할 수 있는 기술로 발전해 나가는 것이 중요할 것이다.

그리고 메신저 프로그램의 등장으로 인해 과거 가장 중요한 커뮤니케이션 매체였던 전화의 사용률이 급격히 감소하고, 카카오톡의 월간 이용자수는 4,300만명을 웃돌 정도로 현대사회의 대부분의 사람들은 채팅으로 의사소통을 하고 있다. 현재 우리 삶에서 빼놓을 수 없는 어플리케이션 하나를 꼽자면 10명중 9명은 “카카오톡” 을 꼽을 정도로 메신저 프로그램은 우리 삶에서 빼 놓을 수 없는 중요한 커뮤니케이션 플랫폼이다. 메신저 응용 프로그램은 쉽고 간결하며, 실제로 목소리를 주고받지 않아도 실시간 의사소통의 가능하다는 장점이 있다. 이 메신저 프로그램에 인공지능을 결합하여 인간중심적이고 사용자를 이해할 수 있는 메신저 프로그램을 만든다면 어떨까 라고 생각을 하게 되어 감정 기반 채팅 어플리케이션을 선택하게 되었다.

1.2 연구의 필요성

기존 메신저 응용 프로그램은 쉽고 간결하며, 실제로 목소리를 주고받지 않아도 실시간 의사소통의 가능하다는 장점이 있지만, 채팅 사용 시 텍스트에 의존하여 대화를 하게 되고, 이 때 말투나 표정이 드러나지 않기 때문에 감정이 온전하게 전달되지 못하여 오해가 생기는 경우가 많다.

이러한 단점을 보완하고 좀 더 감정적이고 효과적인 대화를 위하여 사용자의 표정을 통해 감정을 분석하고 대화의 전체적인 내용 분석을 제공하는 지능적인 채팅 플랫폼 “이모지톡”(지능형 채팅 서비스)를 제안하였다.

“이모지톡”은 메신저 프로그램과 인공지능이 결합된 형태로 얼굴 인식 및 감정 분석, 텍스트 분석 등 전망 있는 차세대 기술들이 활용된 작품을 만드는 것에도 의의가 있습니다. 현재 상용화된 채팅 메신저의 기능을 포함하는 것에서 더 나아가 감정을 표현하는 새로운 세대의 채팅 프로그램을 목표로 하고 있으며, 추가적으로 대화 종료 후 해당 채팅의 내용을 이용하여 전체적인 감정을 분석하여 감정 빈도수 그래프 및 대화 내용을 한눈에 정리해주는 기능과 사용자에게 편리한 기능을 추가 제공할 예정이다. 이러한 과정을 통해 사용자에게 지능적이고 유능한 채팅 서비스를 제공할 수 있을 것으로 기대하고 있다. 이와 더불어 작품을 계획하고 연구하며 우리에게도 android, python, 데이터 분석 등의 새로운 지식이 생길 것으로 예상된다.

1.3 연구 목표

이번 종합설계 프로젝트 목표는 다중 채팅과 동시에 감정을 분석해 상대의 감정을 실시간으로 알 수 있는 채팅 플랫폼을 만드는 것이다. 이 채팅 플랫폼을 통하여 얼굴을 마주하지 않고 채팅을 하면서도 상대방의 감정을 파악할 수 있고 대화 종료 후 어떤 대화를 했는지 분석할 수 있다.

본 어플리케이션의 기능은 다음과 같다.

1) 로그인, 친구목록, 프로필 수정, 사진 및 파일 전송 등의 기능을 지닌 채팅 어플리케이션

2) 채팅 방 접근 후 전면 카메라로 사용자 얼굴 인식

3) 실시간 감정 분석 후 이모티콘으로 변환하여 대화방 안의 사용자들에게 공유

감정은 다음과 같이 6가지로 분류된다. (무표정, 기쁨, 화남, 슬픔, 놀람, 경멸)

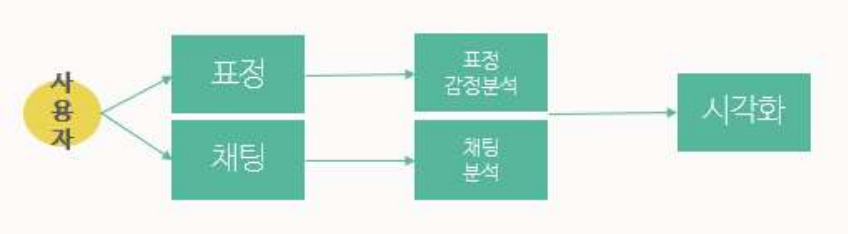
4) 사용자가 원하는 경우 버튼을 눌러 감정 분석을 중지하거나 재개할 수 있다.

5) 채팅 분석 버튼 클릭 시 해당 채팅방의 대화내용을 분석한다.

분석 결과는 다음과 같이 4개이다.

wordcloud, 사용자 별 빈도수, 요일 별 빈도수, 사용자들의 감정 빈도수

6) 저장된 이미지를 웹페이지에 파싱하여 안드로이드 어플리케이션에 연결한다.



해당 채팅 어플리케이션을 실행하게 되면 먼저 로그인을 통해서 자신의 정보를 등록할 수 있고, 채팅 서비스를 이용할 수 있는 UI로 넘어갈 수 있다. 그리고 채팅방을 개설하여 채팅에 참여할 수 있고 해당 채팅을 통해 텍스트뿐만 아니라 사진, 파일 등의 추가적인 데이터도 전송할 수 있다. 음성인식을 적용하여 채팅을 음성인식으로 키보드뿐만 아니라 채팅을 입력할 수 있게 한다.

그리고 채팅 시 카메라를 통해 사용자의 얼굴 표정을 분석하여 사용자들의 감정을 이모티콘으로 공유한다. 이는 채팅방 안의 구성원들이 모두 확인할 수 있다. 이 때 감정 공개를 원하지 않는 사용자는 감정 가리기 기능을 사용할 수 있다.

대화가 끝난 뒤에는 채팅 내용을 분석하여 주제별 빈도수 그래프를 만들고 핵심 단어들을 정리한다. 해당 채팅방에서 어떤 대화가 오고 갔는지, 중심 내용이 무엇이었는지 한눈에 볼 수 있게 요약하여 제공한다.

Ⅱ. 연구 내용

2.1 관련 이론

얼굴의 표정은 사람의 감정 뿐 만 아니라 마음의 상태, 사회적 상호작용, 생리학적 신호 등

과 같은 다양한 정보를 반영한다. 얼굴 표정 인식에 관한 연구의 중요성이 중대되고 있는 이유는 컴퓨터 성능의 향상에 따라 빠른 처리가 가능할 뿐만 아니라 얼굴 검출, 추적, 인식 등과 같은 영역에서의 연구와 밀접한 연관성이 존재하여 연구 수행을 향상시키기 때문이다. 얼굴 표정의 동적 변화를 실시간으로 분석하기 위해서는 얼굴 동작의 시간적 변이를 반영할 수 있는 최적의 표정 정보의 추출과 실시간 추적이 필요하며 특정 표정 사이의 변화를 능동적으로 설명할 수 있는 표정변화 모델에 기반한 얼굴 표정의 해석 방법이 필요하다. 현재 다양한 얼굴 표정 인식을 위한 방법들이 존재하고 있다. 이러한 방법들은 독자적으로 쓰이거나 여러 가지 방법들이 혼용되어 사용되기도 한다. 얼굴 표정 검출 알고리즘의 성능이 향상됨으로써 실시간 처리가 필수적인 고급 응용 분야에까지 널리 활용되고 있는 것으로 판단된다. 특히 표정 인식 기술은 시시각각으로 변화하는 표정의 실시간 인식을 위해서는 고속의 얼굴 검출이 필수적이라고 판단된다. 이번 프로젝트에서 표정인식 기능은 Affectiva의 Affdex SDK를 사용하였다. 이 SDK는 가장 널리 사용되는 얼굴 분석 시스템 중 하나이다.

본 채팅 어플리케이션은 기본적으로 Google의 Firebase 실시간 데이터베이스를 이용한다. Google의 Fireabase Realtime Database는 일반적인 Http 요청이 아닌 동기화를 사용하여 연결된 모든 기기가 데이터가 업데이트 되었을 때 변경 내용을 받아올 수 있다. 기기가 오프라인 상태일 때도 데이터 업데이트에 대한 이벤트는 실시간으로 누적 발생하며 기기가 네트워크에 연결되었을 때 데이터가 동기화되고 충돌이 자동 해결된다.

Realtime Database는 실시간 이벤트와 동기화를 제공하지만 해당 기능은 모두 asynchronous하게 진행된다. 즉 Firebase의 변경 사항이 기기에 전달되어 기기의 로컬 데이터가 변경될 때까지 기기는 block상태가 되지 않고 별도의 작업을 수행할 수 있다. 이는 기기가 Database에 데이터를 변경하는 write의 경우 문제가 되지 않고 오히려 더 좋은 성능을 보이지만 데이터를 받아오는 read의 경우 문제가 발생한다.

어플리케이션에서는 채팅 분석 기능을 추가로 제공하는데, 이를 위해서 서버에서 전송받은 데이터에서 날짜, 채팅데이터, 이용자 등의 정보를 추출하여야 한다. 이 과정에서 정규표현식을 사용하였다. 정규표현식은 텍스트로부터 특정한 형태나 규칙을 가진 문자열을 찾기 위해 그 형태나 규칙을 나타내는 패턴을 정의한다.

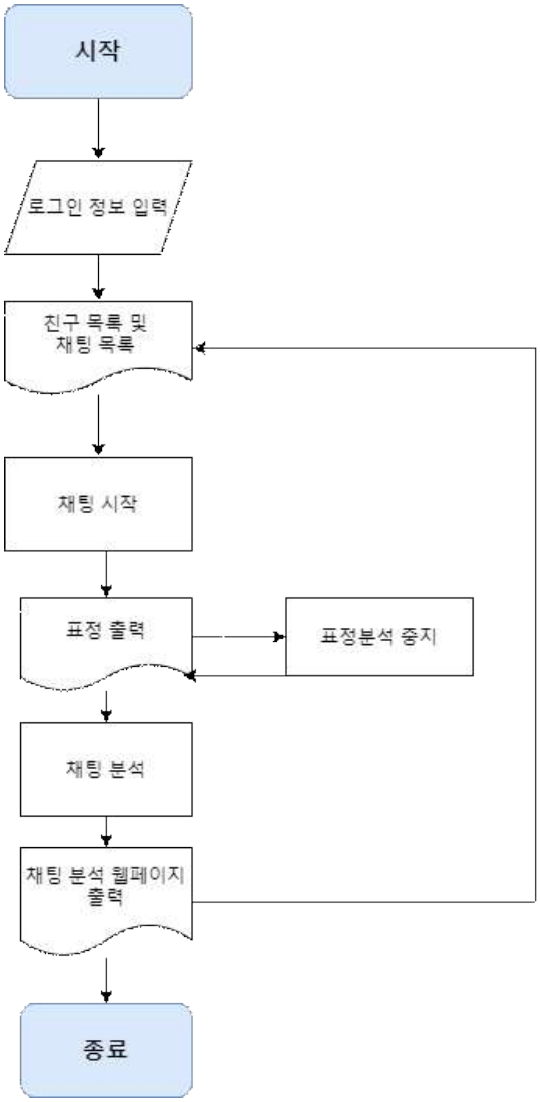
Python에서 정규표현식을 사용하기 위해서는 Regex를 위한 모듈인 re 모듈을 사용한다. re 모듈의 compile 함수는 정규식 패턴을 입력으로 받아들여 정규식 객체를 return 한다.

채팅 데이터 분석을 위해서는 Pandas를 사용하였다. Pandas는 Python에서 사용하는 데이터 분석 라이브러리로, 행과 열로 이루어진 데이터 객체를 만들어 다룰 수 있게 되며 보다 안정적으로 대용량의 데이터들을 처리하기 위하여 편리하게 이용되는 도구이다.
Dataframe은 2차원 자료구조로 행(index)과 열(column)의 구조를 가지고 있는 대표적인 pandas 객체이다. pandas를 통해 Dataframe 형식으로 변환된 데이터의 행과 열에 직접 접근 할 수 있으며 csv, txt, excel, SQL database, HDF5 포맷 등 다양한 외부 리소스에 데이터를 읽고 쓸 수 있는 기능을 제공한다.

어플리케이션의 채팅 분석에서는 채팅방의 주제나 키워드를 알 수 있도록 wordcloud를 제공한다. 채팅 데이터에서 wordcloud를 생성하기 위해서는 전처리 과정이 필수적이다. 그를 위해 사용하는 것은 KoNLPy로, 한국어 텍스트를 이용하여 NLP(Natural Language Processing, 자연어 처리)작업을 수행할 수 있게 하는 Python 패키지이다. KoNLPy는 Hannanun, Kkma, Komoran, Mecab, Open Korean Text와 같은 다양한 형태소 분석, 태깅 라이브러리를 가지고 있다. 이 클래스들은 공통적으로 nouns(명사 추출), morphs(형태소 추출), pos(품사 부착) 메소드를 제공한다. 본 프로젝트에서는 Kkma 라이브러리를 채택하여 형태소 분석을 진행하였다.

분석이 마친 뒤 생성된 그래프들을 사용자에게 제공할 때에는 Flask를 사용하여 웹페이지에 이미지를 파싱하였다. Flask는 Python의 Micro framework를 기반으로 단순하고 매우 가벼운 web framework이다. URL 라우팅, Template, Cookie, Debugger 및 개발 서버 등 기본적인 기능만을 제공하며, 구조는 크게 WSGI용 Liberary인 Werkzeug와 HTML 렌더링 엔진인 Jinja2 template 으로 구성되어 있다. Flask는 기본 기능 제공에 다양한 확장 모듈을 이용할 수 있는 구조로 짧은 코드로 간단하게 웹 서버를 구동할 수 있지만, 데이터베이스 연결, 양식 처리, 보안, 인증 등 개발자가 처리해야 하는 부분이 많다는 점이 문제가 된다. 본 프로젝트에서는 웹페이지를 단순히 분석 내용 출력을 위해 사용하므로 flask 사용이 적절하다고 판단했다.

2.2 전체 시스템 블록 다이어그램



2.3 동작 원리

2.2절에는 본 어플리케이션에 간한 간략한 순서도를 제시하였다. 우선 어플리케이션 실행하고 회원가입 / 로그인 정보를 입력할 수 있다.

회원가입 / 로그인 데이터는 Google의 Firebase 데이터베이스에 저장 후 접근 가능하다. 데이터는 JSON으로 저장되며 연결된 모든 클라이언트에 실시간으로 동기화된다. 채팅은 Firebase의 인스턴스에 접근하여 메시지를 디바이스 간 송수신하여 이루어진다.

대화방 접근 시 사용자의 전면 카메라에 접근하고 Google의 FaceDetector로 사용자의 얼굴을

인식한다. 표정 인식은 affdex detector sdk를 사용하였고 face의 emotions과 expressions 중 Joy, Anger, Sadness, Surprise, Contempt, Attention, Valance를 get하여 가장 큰 emotion을 사용자의 표정으로 결정하였다. 결정된 표정은 이모티콘으로 변환하여 사용자에게 제공한다. 채팅 데이터는 키보드 또는 음성으로 입력할 수 있다. 음성 인식 기능은 Google Speech Recognition API를 사용하였다.

채팅 분석 버튼 클릭 시 로딩 화면이 출력되고 사용자들이 주고받은 대화 내용이 서버로 전송 된다. 로딩 화면이 출력되는 동안 AsyncTask를 이용하여 Background에서 채팅 분석 웹페이지의 URL에 계속 접근하여 get ResponseCode한다. ResponseCode가 ok이면 채팅 분석 웹페이지가 구현이 다 되었다는 뜻이므로 doInBackgroud의 while문을 종료한다. onPostExecute에 채팅 분석 웹페이지 intent를 startActivity하여 통신이 완료되면 채팅 분석 결과 웹페이지로 넘어가도록 한다.

또한 대화가 분석되는 동안 로딩 화면을 구현하기 위해 Glide를 이용하여 로딩 이미지를 화면에 띄우도록 하였다.

Android와 Python 서버 간 TCP/IP Socket 통신을 구현하였고 Python 서버에서는 Multi Thread로 Android의 채팅 데이터를 수신한다. 서버에서는 수신된 String 형식의 Android 채팅 데이터를 정규표현식으로 패턴을 정의하여 채팅 분석을 위한 데이터를 수집한다. 수집한 데이터를 pandas 모듈을 사용하여 dataframe 형식으로 가공하였으며 채팅방 분석을 위하여 timestamp를 날짜로 변경하였고, 요일, 시, 분의 칼럼을 추가하였다.

채팅방의 주제와 대화 흐름을 알기 쉽게 하기 위하여 wordcloud를 생성하였다. 한글 형태소 분석 모듈 KoNLPy에서 제공하는 kkma라는 패키지를 사용하여 진행하였다. 이렇게 형태소 단위로 쪼개지는 과정이 끝난 데이터들은 kkma에서 제공하는 함수 kkma.nouns()를 통하여 명사만 추출하여 따로 저장한다. 추출된 명사의 횟수는 pandas를 통하여 특정 칼럼을 그룹으로 하여 집계한다. 집계된 명사들을 빈도수가 많은 순서대로 정렬한 뒤 wordcloud를 생성할 수 있는데, 빈도수가 많은 순서대로 글씨가 크게, 빈도수가 적을수록 글씨 크기가 작게 표현했다.

채팅방이 가장 활발한 요일, 사용자별 대화 수, 채팅방 분위기를 파악하기 위해 요일, 사용자, 감정을 빈도수를 기반으로 집계하였으며, Python의 matplotlib를 기반으로 한 시각화 라이브러리 seaborn을 사용하여 분석 결과를 도출하였다.

이렇게 도출된 그래프 이미지는 flask 서버로 전송되어 웹페이지에 띄우도록 설계하였다.

Ⅲ. 연구 환경

3.1 연구진

표 3.1 업무 분담표

팀원	업무 분담 업무
조다운	- 표정 분석 안드로이드 어플리케이션 개발 - android-python 서버 간 데이터 송수신 구현 - 프로그램 실행 및 성능 평가 - 졸업작품전시회 출품 및 결과 발표
오하영	- python 데이터 분석 서버 개발 - 분석 웹페이지 구현 - 프로그램 실행 및 성능 평가 - 졸업작품전시회 출품 및 결과 발표

3.2 연구 추진 일정

월	수행 내용	비고
12월	주제 선정, 계획서 작성	
3월	계획 재정비, SW툴 공부	
7월	기본 채팅 어플리케이션 구형 표정 분석 기능 구현	
8월	서버 구현, 어플리케이션 UI 디자인,	
9월	서버 AWS로 이전, 채팅 분석 기능 구현	
10월	성능 테스트, 버그 수정	
11월	캡스톤 디자인 출품 준비, 시연 동영상 제작	

3.3 연구 기자재 및 실험 재료

3.3.1 연구 기자재

- PC (Window10)
- 스마트폰 (android Galaxy S8, Galaxy Note 5)

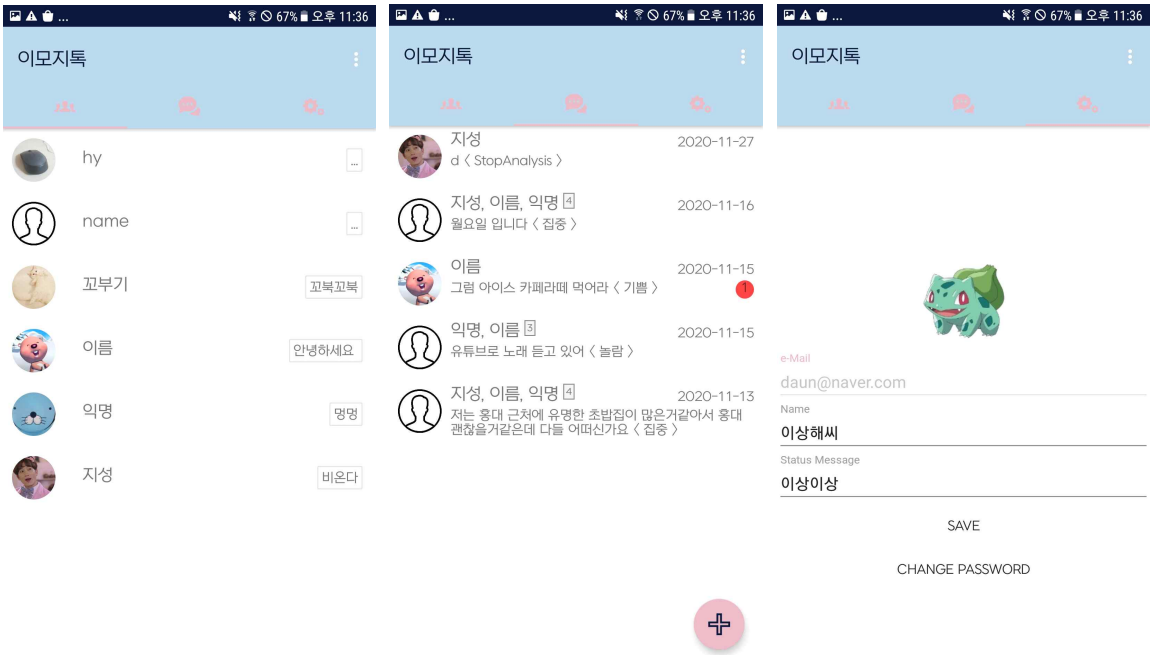
3.4 소프트웨어 및 사용 툴

- 개발언어 :Java, Python
- 개발도구 : Android Studio, Spyder, Putty
- 데이터베이스 : Firebase (Cloud Firestore)

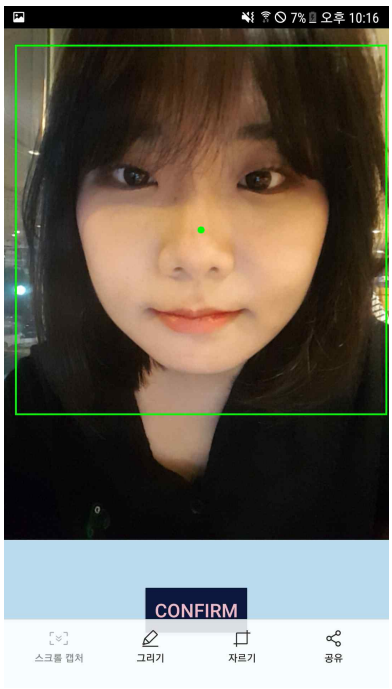
Ⅳ. 연구 결과

4.1 결과물 외형

1. 어플 실행 시 초기 화면 (로그인/회원가입)

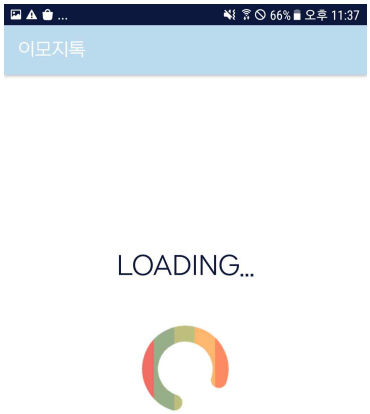


2. 메인 화면 (친구목록/채팅목록/프로필수정)



3. 채팅방 진입(카메라 접근 허용)

4. 채팅방 구현



5. 채팅 분석 버튼 클릭 (로딩)

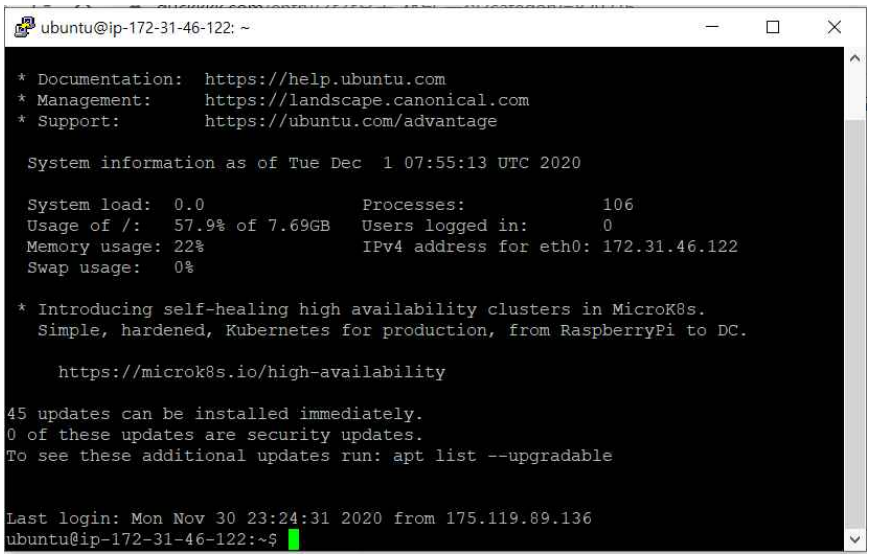
6. 채팅 분석 결과

	A	B	C	D	E	F	G
1	Date	Speaker	timetype	time	contents	emotions	token
2	######	이름	AM	9:50:00	강남에사는기쁨		['강남', '에', '살', '는', '이름', '이', 'ㅂ니다', '^^', '초밥', '좋', '네요', '첫', '모임', '은', '초밥', '먹', '으러', '갈', '나요', '?']
3	######	이름	AM	11:38:55	왕십리에져기쁨		['왕십리', '에', '제일', '곱창', '이제', '최', '애', '음식점', '이', '에요']
4	######	지성	AM	11:33:34	맛있어보여놀람		['맛있', '어', '보이', '어요']
5	######	이상해씨	AM	11:50:27	홍대에도지놀람		['홍', '대', '에', '도', '지점', '이', '있', '네요', '레몬', '마들렌', '이랑', '휘', '낭시', '에', '꼭', '먹', '읍시다', '2', '차', '로', '디저트', '는', '필수', '죠']
6	######	이상해씨	AM	11:37:11	와전부맛조놀람		['오', '아', '전부', '맛', '집', '이', '네요']
7	######	이름	AM	11:00:51	좋아요!!홍놀람		['좋', '아요', '!!', '홍', '대', '에', '알', '는', '맛', '집', '많', '은데', '앞', '으로', '기대되', '네요']
8	######	익명	AM	11:35:03	좋아요ㅎㅎ기쁨		['좋', '아요', '다', '를', '해산물', '좋아하', '시', '는', '군', '멀']
9	######	이름	AM	11:42:39	아앗그러시슬픔		['아', '아', '앗', '그러', '시', '군요', '...', '곱창', '못', '드시', '는', '분', '들', '중', '종계', '시', '더라', '구', '요', '.', '.', '곱창', '은', '따로', '먹', '을게요']
10	######	이상해씨	AM	9:27:18	여러분안녕집중		['여러분', '안녕', '하세', '요', '!!']
11	######	이상해씨	AM	11:21:08	첫번째모임놀람		['첫', '번째', '모임', '토요일', '어떠', '시', '느가요']
12	######	이상해씨	AM	1:32:30	훠궈,.',.마라,탕,.,.마라,'샹궈',.',.'중식','좋','아요','마라','탕','에','꾸','어','바로','우',...,'삥','홍차',...']
13	######	지성	AM	1:40:26	세상엔맛의StopAnalys		['세상', '에', '는', '맛있', '는', '것', '이', '너무', '맛', '아요']
14	######	이상해씨	AM	9:51:39	저도초밥좋기쁨		['저', '도', '초밥', '좋아하', '어요', '첫', '모임', '초밥', '좋', '아요', '!', '홍', '대', '에', '맛있', '는', '초밥', '집', '알', 'ㅂ니다', '같이', '가요']
15	######	이상해씨	AM	11:52:04	그럼내일4.기쁨		['그럼', '내일', '4', '시', '에', '홍', '대', '입구', '3', '번', '출구', '에서', '만', '나', '요']
16	######	익명	AM	11:40:14	아곱창.저경멸		['아', '아', '곱창', '...', '저', '는', '못', '먹', '어요', '무슨', '맛', '으로', '먹', '는지', '모르', '겠', '어요']
17	######	지성	AM	9:47:02	반가워요조놀람		['반갑', '어', '요조', '밥', '좋아하', '는', '지성', '이', 'ㅂ니다', '~', '빨리', '맛', '집', '찾', '으러', '다니', '고', '싶', '네요', '^^']
18	######	익명	AM	11:46:09	다들못먹는집중		['다', '들', '못', '먹', '는', '음식', '하나쯤', '은', '있', '지', '않', '나요', '저', '도', '밀가루', '좋아하', '어요', '^^']
19	######	이상해씨	AM	11:42:55	개인취향존회남		['개인', '취향', '존중', '합', '시다', '^^', '~']
20	######	익명	AM	11:35:51	맛있어보여놀람		['맛있', '어', '보여', '요', '!!', '어디', '이', '느가요']
21	######	이름	AM	11:30:41	제인생초빕기쁨		['제인', '생초', '밥집', '이', '에요']
22	######	이름	AM	11:26:50	토요일좋이집중		['토요일', '좋', '아요', '!', '장소', '는', '홍', '대나', '강', '남쪽', '어떨', '느가요', '?']
23	######	지성	AM	11:41:21	곱창싫어하화남		['곱창', '싫어하', '는', '분', '이랑', '겸상', '안해', '요']
24	######	익명	AM	9:41:24	저는익명이기쁨		['절', '는', '익명', '이', '라고', '합', '니', '다', '~', '잘', '부탁드리', '어요']
25	######	이름	AM	11:47:41	다음모임언기쁨		['다음', '모임', '에', '는', '식사', '하', '고', '디저트', '까지', '부시', '러', '갈', '시', '죠', '!']
26	######	이상해씨	AM	1:27:13	두번째모임집중		['두', '번째', '모임', '은', '저번', '에', '말하', 'ㄴ', '대로', '강남', '역', '이', 'ㅂ니다']
27	######	지성	AM	11:54:40	알겠습니다기쁨		['알', '겠', '습니다', '맛', '집', '탐방', '기대되', '어요', '~~~']
28	######	이상해씨	PM	1:18:59	네맞아요11기쁨		['네', '맞', '아요', '10', '번', '출구', '에서', '만', '나', '요']
29	######	이상해씨	AM	9:42:03	저는맛집탐기쁨		['절', '는', '맛', '집', '탐방', '동아리', '회장', '이', '상해', '씨', '이', 'ㅂ니다']
30	######	이름	AM	11:38:30	다들인생맛집중		['다', '들', 'ㄹ', '인생', '맛', '집', '이', '어디', '이', 'ㄴ가요', '??', '곱창', '좋아하', '시', '나요']
31	######	이름	AM	11:47:58	커피는제가기쁨		['커피', '는', '제가', '쏘', 'ㄹ게요']

6. 형태소 분석 적용

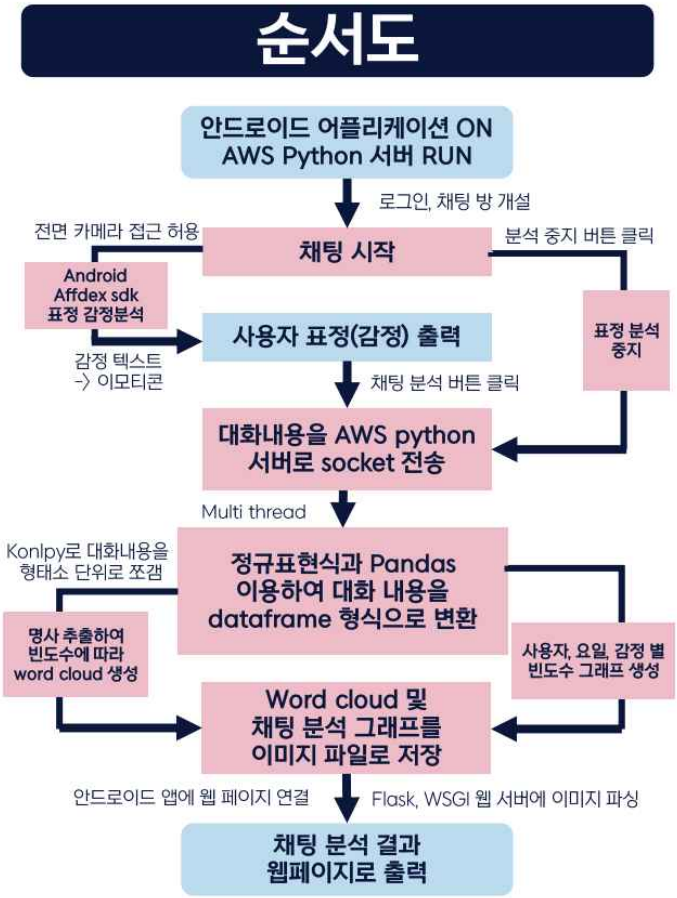
	Date	Speaker	timetype	time	contents	emotions	token
2	######	이름	AM	9:50:00	강남에사는	기쁨	['강남', '이름', '초밥', '모임', '초밥']
3	######	이름	AM	11:38:55	왕십리에저	기쁨	['왕십리', '제일', '곱창', '이제', '최', '애', '음식점']
4	######	지성	AM	11:33:34	맛있어보여	놀람	[]
5	######	이상해씨	AM	11:50:27	홍대에도지	놀람	['홍', '대', '지점', '레몬', '마들렌', '낭시', '디저트', '필수']
6	######	이상해씨	AM	11:37:11	와전부맛집	놀람	['전부', '맛', '집']
7	######	이름	AM	11:00:51	좋아요!!홍	놀람	['홍', '대', '맛', '집', '앞']
8	######	익명	AM	11:35:03	좋아요ㅎㅎ	기쁨	['다', '해산물', '군']
9	######	이름	AM	11:42:39	아앗그러시	슬픔	['곱창', '분', '종', '종계', '곱창']
10	######	이상해씨	AM	9:27:18	여러분안녕	집중	['안녕', '하세']
11	######	이상해씨	AM	11:21:08	첫번째모임	놀람	['모임', '토요일']
12	######	이상해씨	AM	1:32:30	훠궈,마라탕	기쁨	['마라', '탕', '마라', '중식', '마라', '탕', '우', '홍차']
13	######	지성	AM	1:40:26	세상엔맛있	StopAnalys	['세상']
14	######	이상해씨	AM	9:51:39	저도초밥좋	기쁨	['초밥', '모임', '초밥', '홍', '대', '초밥', '집', '가요']
15	######	이상해씨	AM	11:52:04	그럼내일4	기쁨	['내일', '홍', '대', '입구', '출구']
16	######	익명	AM	11:40:14	아곱창..저	경멸	['곱창', '맛']
17	######	지성	AM	9:47:02	반가워요초	놀람	['요초', '밥', '지성', '맛', '집']
18	######	익명	AM	11:46:09	다들못먹는	집중	['다', '음식', '하나쯤', '밀가루']
19	######	이상해씨	AM	11:42:55	개인취향존	화남	['개인', '취향', '존중', '합', '시다']
20	######	익명	AM	11:35:51	맛있어보여	놀람	['보여']
21	######	이름	AM	11:30:41	제인생초빋	기쁨	['제인', '생초', '밥집']
22	######	이름	AM	11:26:50	토요일좋아	집중	['토요일', '장소', '홍', '대나', '강', '남쪽']
23	######	지성	AM	11:41:21	곱창싫어하	화남	['곱창', '겸상', '안해']
24	######	익명	AM	9:41:24	저는익명이	기쁨	['익명', '합']
25	######	이름	AM	11:47:41	다음모임언	기쁨	['다음', '모임', '식사', '디저트']
26	######	이상해씨	AM	1:27:13	두번째모임	집중	['모임', '저번', '강남', '역']
27	######	지성	AM	11:54:40	알겠습니다	기쁨	['맛', '집', '탐방']
28	######	이상해씨	PM	1:18:59	네맞아요1(	기쁨	['출구']
29	######	이상해씨	AM	9:42:03	저는맛집탐	기쁨	['맛', '집', '탐방', '동아리', '회장', '상해']
30	######	이름	AM	11:38:30	다들인생맛	집중	['인생', '맛', '집', '곱창']
31	######	이름	AM	11:47:58	커피는제가	기쁨	['커피', '제가']

7. 형태소 분석 후 명사만 추출



8. 원격 접속한 파이썬 서버

4.2 작동 과정



4.3 성능 분석

채팅 중 실시간으로 전면 카메라를 통해 사용자의 표정을 인식하여 이모티콘으로 출력됨을 확인하였고, 서버에서 채팅 데이터를 수신하여 word cloud 및 사용자, 요일, 감정 별 빈도수 그래프 생성 후 웹 서버에 파싱됨을 확인함.

word cloud를 생성하기 위한 형태소 분석에서 최소 단위는 사전에 포함되지 않은 단어, 띄어쓰기가 불분명한 경우, 문장에서의 위치나 문맥에 따라 달라지므로 명사를 추출했을 때 어느 정도의 오차가 발생. 그러나 채팅방의 대화 주제를 확인함에는 무리가 없음.

4.4 문제점 및 개선방안

채팅 분석 버튼 클릭 시 웹 페이지가 뜰 때 까지 delay 시간이 존재한다. 채팅 방의 데이터 양에 따라 짧게는 10여초에서 길게는 40여초까지 delay가 존재하는 것으로 확인되었다. 이를 개선하기 위해 데이터 송수신을 재배치하여 시간을 단축시켜 보았으나 Socket을 이용하여 채팅 데이터를 보내고, Python 서버에서 채팅 분석 그래프를 만드는데 최소한의 시간이 필요하므로 어느 정도의 delay는 감수해야 한다고 판단했다.

그래서 사용자가 채팅 분석 버튼 클릭 후 결과가 나올 때 까지 대기하는 동안 분석이 되고 있음을 시각적으로 표현하기 위해 AsyncTask를 사용하여 Background로 로딩바가 계속 돌아가게 만들었다.

또한 추가적인 delay를 최소화하기 위해 기존에 단순히 sleep하여 채팅 분석 웹페이지 intent로 넘어가던 것을 개선하여 로딩 화면이 출력되는 동안 AsyncTask를 이용하여 Background에서 채팅 분석 웹페이지의 URL에 계속 접근하여 Response Code를 계속 get한다. Response Code가 ok이면 채팅 분석 웹페이지가 구현이 다 되었다는 뜻이므로 Backgroud의 while문을 종료한다. onPostExecute에 채팅 분석 웹페이지 intent를 startActivity하여 통신이 완료되면 빠르게 채팅 분석 결과 웹페이지로 넘어가도록 했다.

채팅 분석이 wordcloud와 Seaborn의 Count Plot을 사용한 빈도수 그래프만 존재한다. 원래는 채팅 분석 결과의 다양성을 위해 Density Plot으로 시간대별 채팅 비율 그래프를 함께 사용하려고 하였으나, 그래프를 plot 하는 도중 categorical error 가 지속적으로 발생하여 아쉽지만 사용할 수 없었다.

V. 참고문헌

[1] 강민구, “인공지능 기반의 비주얼 인식 오픈소스 비교연구 : 인물 감정 분석 중심으로”, 2017년 12월

[2] Android Developers, Docs, Reference, “FaceDetector”, https://developer.android.com/reference/android/media/FaceDetector

[3] Firebase 실시간 데이터베이스 가이드 문서, https://firebase.google.com/docs/android/setup?hl=ko

[4] A messenger app for Android based on Firebase, https://github.com/gujc71/DirectTalk9

[5] 허경무, “얼굴 표정 인식 기술”, 2014년

[6] 한예림, 한국통신학회, “최신 정보통신기술 Google에서 제공하는 Firebase의 실시간 데이터베이스 이용에 관한 연구”, 2017년

[7] Pandas documentation, User Guide, https://pandas.pydata.org/docs/index.html

[8] KakaoTalk 대화 형태소 분석과 워드 클라우드, https://github.com/ssooni/data_mining_practice

[9] 박은정, 조성준, “KoNLPy: 쉽고 간결한 한국어 정보처리 파이썬 패키지“, 제 26회 한글 및 한국어 정보처리 학술대회 논문집, 2014

부록 A. Python Server 소스 코드

```
def apply_kko_regex(msg_list):

    kko_pattern = re.compile("([\S\s]+):([\S\s]+)<([\S\s]+)>/Timestamp\(seconds=([0-9]+)")

    emoji_pattern = re.compile("["u"\U0001F600-\U0001F64F"  # emoticons
                               u"\U0001F300-\U0001F5FF"  # symbols & pictographs
                               u"\U0001F680-\U0001F6FF"  # transport & map
symbols
                               u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
                               "]+", flags=re.UNICODE)

    kko_parse_result = list()
    cur_datetime= ""
    cur_date = ""
    cur_datetype = ""
    cur_time= ""

    for msg in msg_list:
        kko_pattern_result = kko_pattern.findall(msg)
        print(kko_pattern_result)
        if len(kko_pattern_result) > 0:
            tokens = list(kko_pattern_result[0])
            cur_datetime = dt.datetime.strptime(convert_datetime(tokens[-1]),
"%Y-%m-%d %H:%M:%S" )
            cur_date = cur_datetime.strftime("%Y-%m-%d")
            cur_datetype = cur_datetime.strftime("%p")
            cur_time = cur_datetime.strftime("%I:%M:%S")
            tokens.insert(0, cur_date)
```

```
            tokens.insert(2, cur_datetype)
            tokens.insert(3, cur_time)
            tokens.pop(6)
            print(tokens)
            #tokens[1] = tokens[1][1:]
            for i in tokens:

                if tokens[1] not in username:

                    username.append(tokens[1])

            kko_parse_result.append(tokens)

    kko_parse_result = pd.DataFrame(kko_parse_result, columns=["Date", "Speaker",
"timetype", "time", "contents", "emotions"])
    kko_parse_result.to_csv("./result/kko_regex.csv", index=False)

    return kko_parse_result
```

```
def draw_wordcloud(kkma_result):
    # List로 되어있는 열을 Row 단위로 분리
    tokens = pd.DataFrame(kkma_result["token"].apply(lambda x: ast.literal_eval(x)).tolist())

    tokens["Date"] = kkma_result["Date"]
    tokens["Speaker"] = kkma_result["Speaker"]
    tokens["timetype"] = kkma_result["timetype"]
    tokens["time"] = kkma_result["time"]
    tokens["contents"] = kkma_result["contents"]
```

```
    tokens[“emotions“] = kkma_result[“emotions“]

    tokens = tokens.set_index([“Date“, “Speaker“, “timetype“, “time“, “contents“,
“emotions“])
    tokens = tokens.T.unstack().dropna().reset_index()

    tokens.columns = [“Date“, “Person“, “time_type“, “time“, “sntc“, “emotions“,
“index“, “token“]
    print(tokens.head())

    # 빈도수 집계
    summary = tokens.groupby([“token“])[“index“].count().reset_index()
    summary = summary.sort_values([“index“], ascending=[False]).reset_index(drop=True)

    # 워드클라우드 생성
    wc = WordCloud(font_path='./font/NanumGothic.ttf', background_color='white',
width=800, height=600).generate(“ “.join(summary[“token“]))
    plt.imshow(wc)
    plt.axis(“off“)
    plt.show()
    plt.savefig('./static/'+name1+'.png')
    # plt.savefig('./static/wordcloud.png')
```

주요 코드만 첨부

부록 B. Android Application 소스 코드

https://github.com/JDUOHY/capstone_emojiTALK