Progress5(~6/5)

2조 김명곤 배상환 류호 이한결

1. 이번주에 구현한 behavior 목록 및 주요 사항

이번주에는 친구 요청 및 친구 추가 기능, 이미지 검열 기능, 포스트 및 채팅방에 이미지 업로드, 네비게이션바 상의 유저 검색기능을 구현하였고, 친구 기능 추가에 따른 타임라인/담벼락에 보이는 글의 permission 수정하였다. 또한 브라우저에서 위치정보를 활용하고자 시도하고 있으나 https 설정 문제를 파악한 상태이다. 또한 일정관리 페이지를 구성하고 있다. 전반적으로 구현해야 하는 기능이 많았고, 서비스의 방향이 수정되어 다음 회의에서 일정 배분 및 세부 계획을 조정할 필요가 있다.

- 주요 behavior상세 설명

1)이미지 검열 기능

프로필 이미지 검열을 위해 CNN 모델을 구현했다. 학습에는 22000여 장의 학습 데이터를 사용하였으며 이중 not safe for work에 해당하는 이미지는 8000여 장이다. Safe for work 이미지는 easy와 difficult로 나누어서 모았으며 easy는 동식물, 사물과 같이 선정적 요소가 전혀 없는 사진들로 구성하였고, difficult는 비키니 사진이나 상의를 탈의한 남자 사진과 같이 살색이 많이 포함되나 성인 이미지는 아닌 사진으로 구성하였다.

데이터의 양이 많은 것이 아니므로 IMAGENET에서 2014년 뛰어난 성적을 보이며 2등을 한 VGGNet을 transfer learning 방식으로 학습시켰다. 이는 IMAGENET에 사용되는 데이터의 양이 매우 많고 다양하여 VGGNet의 Convolution 레이어들이 보편적인 필터 역할을 해낼 것으로 생각되었기 때문이다. Convolution 레이어들은 본래 VGGNet의 파라미터를 그대로 사용하였고 뒤의 fully connected 레이어는 제거하여 새로 추가했다. 또한 Batch Normalization을 사용하는 것이 성능 향상에 기여하는 것으로 확인되어 이를 사용하였다. 이 모델은 224\*224 사이즈의 RGB 이미지를 input으로 하며 not safe for work / safe for work 두 개의 클래스에 대한 확률값을 아웃풋으로 한다. 학습에 필요한 코드는 모두 빼고 학습된 파라미터와 모델만 추가하여 인풋 이미지에 대해 결과값을 낼 수 있도록 했다. 개발 환경으로서 Keras를 프레임워크로 사용했으며 Theano를 backend로 사용했다.

VGGNet 자체의 규모가 매우 큰 편이어서 파라미터만 400MB에 육박했다. Aws 서버의 메모리는 1GB이기 때문에 서버를 돌리면서 VGGNet 모델을 메모리에 동시에 올리는 것은 불가능했다. 따라서 swap을 통해 1GB의 disk를 메모리 대신 사용하도록 하였다. 또한 본래는 포스트, 채팅 메세지 등을 통해 업로드 되는 모든 이미지에 대하여 이미지 검열을 할 계획이었으나 메모리의 한계로 우선은 프로필 이미지에 대해서만 검열을 하도록 했다.

2)친구 기능

친구 기능의 경우 생각보다 고려할 점이 많아 세부적으로 구현하는 데 어려움을 겪었다. 특히 친구 요청 단계와 친구 상태인 단계가 구분되고, 각 단계에서 요청 또는 친구 삭제를 할 수 있어야 하는 점을 고려했다.

- 기타 프로젝트 방향 관련

또한 SNS 서비스의 방향을 일정관리가 가능한 스케쥴형 SNS로 설정하여 추후 계획을 수정하기로 했다. 즉, 메인이 되는 기능은 스케쥴 관리인데, 친구들과 잡는 약속이나 이벤트를 수락 시 자신의 달력에 추가할 수 있는 것이다. 현재 카카오톡의 경우 투표 기능이 있으나, 이를 자신이 별도로 다른 앱을 이용해 캘린더에 입력해야 한다. 또한 페이스북의 경우 캘린더 기능과 친구 기능이 있으나, 캘린더는 자신이 참가하기로 한 이벤트나 생일이 표시되고, 주 기능으로 사용되지 못한다. 따라서 이를 통합한 그룹형 스케쥴관리 SNS가 구현 목표이다.

- Behavior배분 내용

|  |  |  |
| --- | --- | --- |
| 역할 분담 | 기능 | 상세 |
| 김명곤 | 친구 요청 | 백앤드 구현(친구요청 model 별도로 생성) |
| 해당 테스트 추가 |
| 친구 추가 | 백앤드 구현(친구 요청과 연동되는 모델 생성) |
| 해당 테스트 추가 |
| 위치 기능 | -브라우저 위치 정보(위도/경도)  -Google API를 이용해 지명으로 변환  \*localhost상 구현 , aws세팅 https변환필요 |
| 일정 관리 기능 | 외부 캘린더 컴포넌트 api 확인중 |
| 배상환 | 프로필이미지검열 | 프로필 이미지 검열 CNN 모델 구현 |
| 프로필 이미지 검열 백엔드/프론트엔드 구현 |
| 프로필 이미지 사이즈 고정 |
| 해당 테스트 추가 |
| 이미지 업로드 | 사진 포스팅 백엔드/프론트엔드 구현 |
| 일대일 채팅 사진 메세지 백엔드/프론트엔드 구현 |
| 그룹 채팅 사진 메세지 백엔드/프론트엔드 구현 |
| 해당 테스트 추가 |
| 류호 | 친구 검색 기능 | 친구 검색 frontend 구현 |
| 해당 테스트 추가 |
| 담벼락 | 기존에 timeline에 쓰여진 글이 담벼락에는 안보이는 현상 수정 |
| 친구기능 추가에 따른 접근권한 수정 | 타임라인에서는 본인 글 및 친구의 글에만 접근권한 부여 |
| 담벼락에서는 본인 및 친구에게만 post를 쓰거나 열람할 권한 부여 |
| 채팅은 1:1과 그룹채팅 모두 친구들과만 가능 |
| 해당 테스트 추가 |
| 이한결 | 친구 기능  프론트앤드 | 친구 요청 및 취소 (타인의 담벼락) /  버튼의 요청 대기 상태 표시 |
| 받은 친구 요청 승낙 및 취소 (개인 담벼락) |
| 본인의 친구 요청 취소(개인 담벼락) |
| 친구 목록 보기 및 친구 끊기(개인 담벼락) |

2. backend 새로 구현한 api 스펙 및 테스트 코드

2.1. 신규 api 스펙

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| url | GET | POST | PUT | | DELETE |
| friendrequest/ | 자신이 주고 받은 친구요청 반환 | [친구요청보내기]  User 포스트시 user\_from에 자신, user\_to에 상대 user로 요청이 등록됨 |  | |  |
| friendrequest  /<pk> |  |  | [친구요청 수락]  PATCH : status=1을 보내 친구 요청 수락 | | [친구요청 취소]  취소하고자 하는 친구요청 삭제(pk) |
| myfriend/ | 자신과 친구인 모든 유저 반환 |  |  | |  |
| myfriend/<pk> |  |  | [친구 삭제]  PATCH:  자신의 친구 목록에서 특정 친구 삭제 | | |
| userphotos/ | 모든 유저의 프로필 이미지 리턴 | 프로필 이미지 검열 후 등록 / 변경  -검열 통과시  status=HTTP\_200\_OK  -검열 불통과시  status=HTTP\_403\_FORBIDDEN과 CNN 모델이 결과값으로 내놓은 확률값을 전달 |  |  | |
| friends/ | 친구를 userlist 형태로 가져옴 |  |  |  | |

2.2 세부 수정 api 스펙

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| posts/ | 모든 포스트를 리턴  (permission 변경)  본인과 친구의 글 가져옴 | 텍스트 또는 이미지를 포스팅  둘 중 하나가 없어도 됨 | 포스트의 텍스트를 수정 | 포스트를 삭제 |
| rooms/<pk>  /history/ | <pk>를 id로 갖는 1:1 채팅방의 모든 메세지를 리턴 | <pk>를 id로 하는 1:1 채팅방에 새로운 텍스트 또는 이미지를 메세지로 추가  텍스트 또는 이미지 중 하나가 없어도 됨 |  |  |
| rooms/group/  <pk>/history/ | <pk>를 id로 갖는 그룹 채팅방의 모든 메세지를 리턴 | <pk>를 id로 하는 그룹 채팅방에 새로운 텍스트 또는 이미지를 메세지로 추가  텍스트 또는 이미지 중 하나가 없어도 됨 |  |  |
| wall/<username> | 퍼미션 변경(친구/본인) | 퍼미션 변경(친구/본인) |  |  |

2.3. 테스트코드

테스트 코드는 기존의 테스트 코드는 활용하여 새로운 테스트는 새롭게 파일을 만들어 구현했다. 따라서, 이전 테스트 모드는 past에 넣어두었고 추가한 테스트 코드만 시행하면 된다. 이번에 추가한 각각의 테스트 코드와 실행 커맨드는 다음과 같다.

1. $python3 friendTest.py
   1. /friendrequest
      1. 유저 3명 생성 -> test0이 test1~9에게 친구 요청(POST)
         1. 모든 유저의 친구요청 상태 확인(GET)
         2. test1~3 친구요청 취소(DELETE)
         3. test4~9 친구요청 수락(PATCH)
   2. /myfriend
      1. test0, test4~9 친구 리스트 확인(GET)
      2. test4~6 친구 리스트에서 삭제(PATCH)
2. $python3 getFriendTest.py

friends/ 로부터 GET으로 친구를 불러와 친구가 아닌 유저가 없는지/친구인 유저가 있는지 확인

1. $python3 timelineTest.py

posts/ 에 POST를 한 뒤 GET으로 글을 불러와 post들에 친구가 아닌 유저의 글이 없는지/친구의 글이 있는지 확인

1. $python3 wallTest.py

wall에 친구가 아닌 유저가 POST 및 GET 불가능한지/친구인 유저가 POST 및 GET 가능한지 확인

1. $python3 imageCensorTest.py

회원가입 – safe 이미지로 프로필 이미지 변경 -> unsafe 이미지로 프로필 이미지 변경시도

3. frontend 테스트코드

이전 테스트 코드들은 Past 디렉토리에 넣어두었고 새로이 작성된 프론트엔드 테스트는 다음과 같다. 또한 현재 https port 문제가 있는 gps와 구현 중인 calendar는 구현이 된 후 테스트를 올리도록 한다.

1. $python3 friend\_test.py: 아래 각각의 경우에 대해서 테스트
2. 친구 요청 승낙(test1 -> test2)
3. 친구 요청 거절(test3 -> test2)
4. 친구 요청 취소(test4 -> test2, 타인의 담벼락에서)
5. 친구 요청 취소(test4 -> test2, 본인의 담벼락에서)
6. 친구 끊기 (test1 -:- test2)
7. $python3 search\_test.py

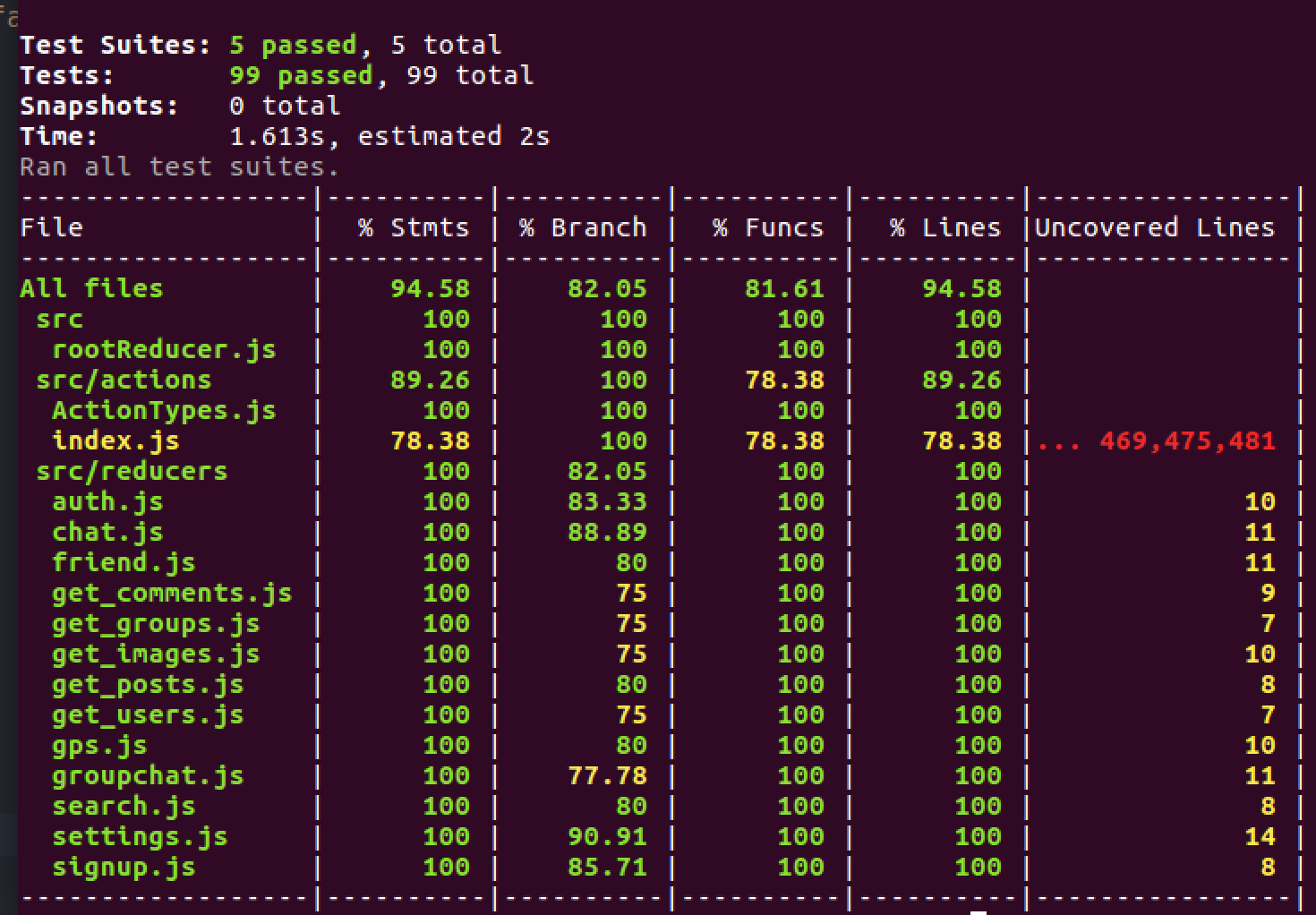
user가 잘 검색되는지/user를 눌러 해당 user의 wall로 이동하는지 확인

1. $python3 imageCensorTest.py

회원가입 – 로그인 – 유저 세팅 페이지 - safe이미지로 프로필 이미지 변경, unsafe이미지로 프로필 이미지 변경시도

4. jest 테스트

sns\_frontend 폴더에서 npm test 실행시 jest –coverage 코드가 실행되며, 이는 bitbucket pipeline에서 또한 확인가능하다. npm test 실행 결과는 다음과 같고, 80%이상의 커버리지 테스트 결과를 보여준다.



5. aws 링크

Frontend: <http://34.208.93.214:3000/>

Backend: <http://34.208.93.214:8000/>