

# CTA(Celeb's Tweet Analysis)

---

희소행렬 처리를 통한 트윗 감성 분석 프로그램

강의: Capstone 디자인 (001)

담당 교수: 안용학, 양효식, 박기호

팀 이름: 16s

팀장: 16010945 노수민 컴퓨터공학

팀원: 16010573 이아현 컴퓨터공학

16011041 홍주희 컴퓨터공학

16011189 양승주 디지털콘텐츠

# 목 차

1. 개요.....	1
1.1 개발 동기 .....	1
1.2 목표.....	1
1.3 기대 효과 .....	1
1.4 개발 일정 .....	2
2. 요구사항 조사.....	3
2.1 선행 조사 .....	3
2.1.1 시장성 조사.....	3
2.1.2 기술 조사.....	3
2.2 요구 사항 명세서 .....	5
2.2.1 기능별 요구사항.....	5
2.2.2 성능 요구사항 .....	11
2.2.3 인터페이스 요구사항 .....	12
2.2.4 제약조건.....	12
2.3 유스케이스 다이어그램 .....	13
2.3.1 유스케이스 기술서 .....	13
3. 시스템 분석 .....	18
3.1 시스템 구조.....	18
3.1.1 웹.....	18
3.1.2 웹 서버 .....	18
3.1.3 모델 .....	18
3.1.4 데이터베이스.....	19
3.2 클래스 다이어그램.....	19
3.3 시퀀스 다이어그램.....	20
4. 시스템 설계 .....	21
4.1 사용자 인터페이스 설계 .....	21
4.2 분석 모델 설계.....	22
4.2.1 기존 분석 모델.....	22
4.2.2 NZB Indexing 적용 분석 모델.....	22
4.3 데이터베이스 설계.....	23

5. 시스템 구현 .....	24
5.1 실제 코드 및 결과 .....	24
5.1.1 모델 빌드 .....	24
5.1.2 NZB Indexing .....	26
5.1.3 Convolution 연산 수정 .....	27
5.2 구현 화면 .....	28
5.2.1 웹 접속 화면 .....	28
5.2.2 ABOUT & CONTACT 화면 .....	29
5.2.3 분석 대상 검색 화면 .....	29
5.2.4 분석 결과 조회 화면 .....	30
6. 팀원 역할 분담 .....	31
7. 성능 테스트 .....	32
7.1 테스트 개요 .....	32
7.1.1 테스트 목적 .....	32
7.1.2 테스트 대상 .....	32
7.1.3 가정 및 제약사항 .....	32
7.2 테스트 단위 .....	33
7.2.1 정확도 테스트 .....	33
7.2.2 CPU 사용량 & Memory 사용량 .....	34
7.2.3 분석 수행 시간 .....	35
8. 결과 및 향후 계획 .....	36
9. 참고 문헌 및 참고 사이트 .....	37

CTA	희소행렬 처리를 통한 트윗 감성 분석 프로그램
	작성 날짜 : 2020년 11월 25일

## 1. 개요

### 1.1 개발 동기

모델의 weight pruning 을 통해 희소 행렬이 생성되면, 해당 행렬을 CSR 방식이 아닌 새로 제안된 NZB 인덱싱 기술을 통해 기존의 convolution 연산이 이뤄지는 모델의 연산 부분을 수정하려고 한다. 기존 모델과 성능 비교가 가능하며, 성능을 향상시킬 수 있는 프로그램 개발을 위해 조사했다. 실시간 반영이 가능한 트위터를 이용해 사용자의 언어습관을 분석하기로 했다. 해당 사용자가 평상시에 어떤 감성의 문장을 주로 사용하는지, 어떤 단어를 자주 사용하는지 분석해주는 프로그램을 개발한다.

### 1.2 목표

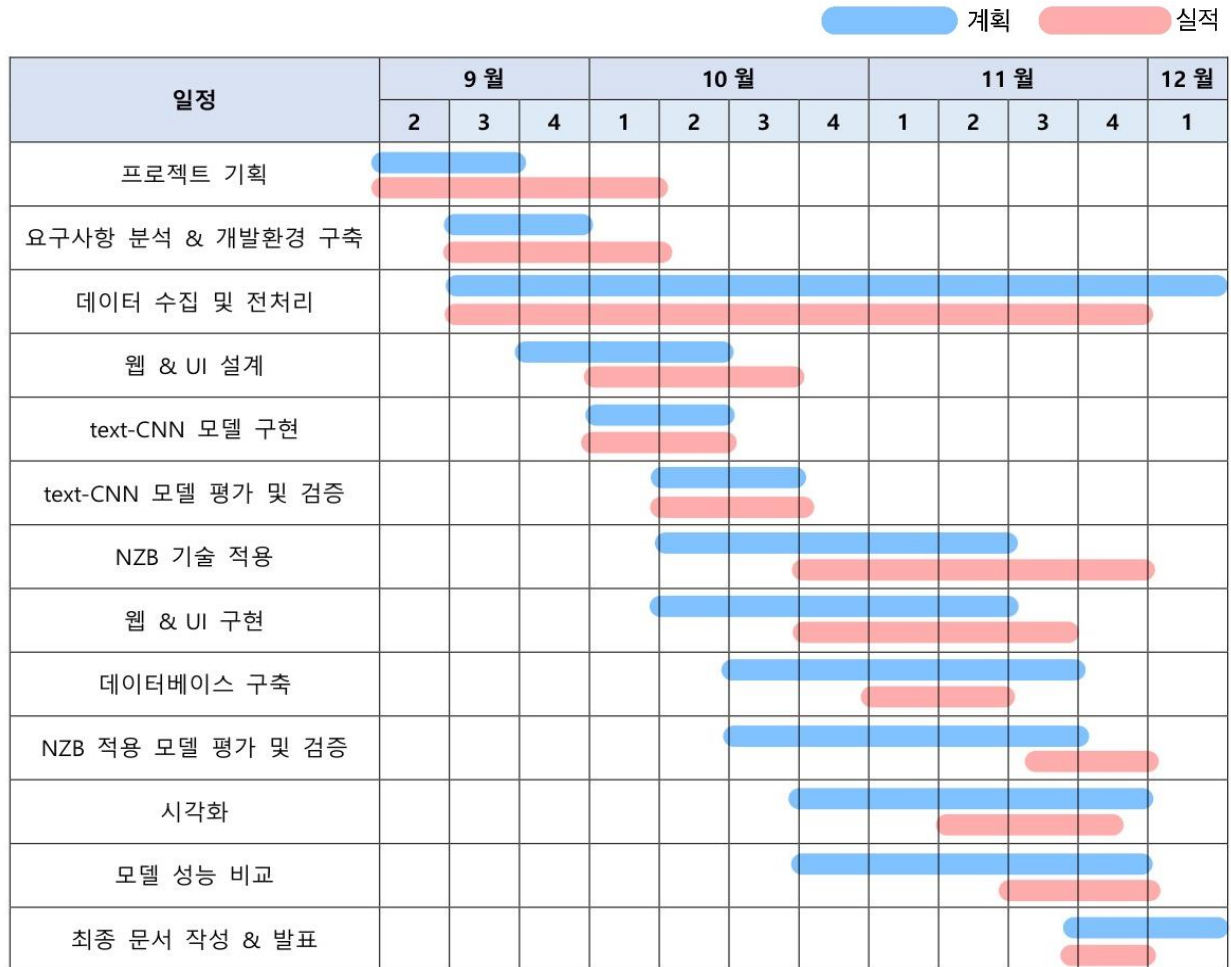
- 기존 모델의 행렬을 NZB 인덱싱 기술로 표현하여 convolution 연산 방식을 수정해 메모리 접근 효율을 증가시킨다.
- 트위터 API 를 통해 유명인의 트윗 데이터를 수집하고, 게시글 속 문장을 분석하여 긍정, 중립, 부정 3 가지의 감성으로 분류해 3 가지 감성의 비율과 각 감성에 해당하는 문장을 3 개씩 보여주며, 자주 사용하는 단어를 최대 5 개를 제공해 시각화 한다.
- 웹 기반 프로그램으로 누구나 접근이 가능하며, 쉬운 UI 구성으로 사용하기 편하고, 오락성과 정보성을 동시에 포함하는 프로그램을 개발한다.

### 1.3 기대 효과

- CNN, LSTM 같은 모델의 연산과정에는 희소 행렬, 밀집 행렬을 포함한 행렬을 통해 연산과정이 이뤄진다. 이 연산 과정에서 기존에 자주 사용하는 CSR 방식이 아닌, NZB 인덱싱 기법을 적용하면 전체 인덱스 사이즈와 메모리 접근 횟수 측면에서 높은 효율을 가질 것이다.
- 트위터 API 유료 버전을 사용할 경우 데이터를 데이터베이스에 저장하고 불러오는 방식이 아닌 사용자가 검색한 특정 인물의 트위터 게시글을 실시간으로 가져와 분석이 가능하다.

## 1.4 개발 일정

구체적인 개발 일정 진행사항은 아래 요구사항 명세서에 포함되어 있습니다.



CTA	희소행렬 처리를 통한 트윗 감성 분석 프로그램
	작성 날짜: 2020년 11월 25일

## 2. 요구사항 조사

### 2.1 선행 조사

#### 2.1.1 시장성 조사

- DeepMoji

영어 문장의 감성을 분석해 여러 이모지로 표현한다.

차이점: 하나의 문장을 여러가지 감정으로 표현하지만, 프로젝트에서 만들 웹 애플리케이션은 여러 문장을 분석하여, 긍정, 중립, 부정 문장으로 나누어 통계를 낸다.

- Watson Tone Analyzer

영어, 프랑스어 2 가지 언어의 트위터, 온라인 리뷰, 텍스트를 7 가지의 톤으로 분석한다.

차이점: 하나의 게시글을 가지고 분석을 해주지만, 프로젝트에서 만들 웹 프로그램은 게시글을 분석하여, 긍정, 중립, 부정 문장으로 나누어 통계를 낸다.

#### 2.1.2 기술 조사

- Keras

사용자가 손쉽게 딥 러닝을 구현할 수 있도록 도와주는 상위 레벨의 인터페이스다.

- ① Tokenizer()

토큰화와 정수 인코딩에 사용한다.

- ② pad\_sequence()

모델의 입력을 사용하기 위해 모든 샘플의 길이를 동일하게 맞추어 사용할 때 사용한다. 정해진 길이보다 길 경우 샘플을 자르고, 길이가 짧은 경우 0으로 채운다.

- ③ Embedding()

단어를 밀집 벡터로 만드는 역할을 한다. 정수 인코딩이 된 단어들을 입력 받아 임베딩을 수행한다. 총 단어의 개수, 임베딩 벡터의 출력 차원, 입력 시퀀스의 길이를 함수 인자로 받는다.

## ④ Sequential()

model로 선언한 뒤에 model.add()라는 코드를 통해 층을 단계적으로 추가한다.

층을 입력층, 은닉층, 출력층으로 구성할 수 있다.

## ⑤ Dense()

전결합층을 추가한다. 출력 뉴런의 수, 입력의 차원, 활성화 함수 종류를 함수의 인자로 받는다.

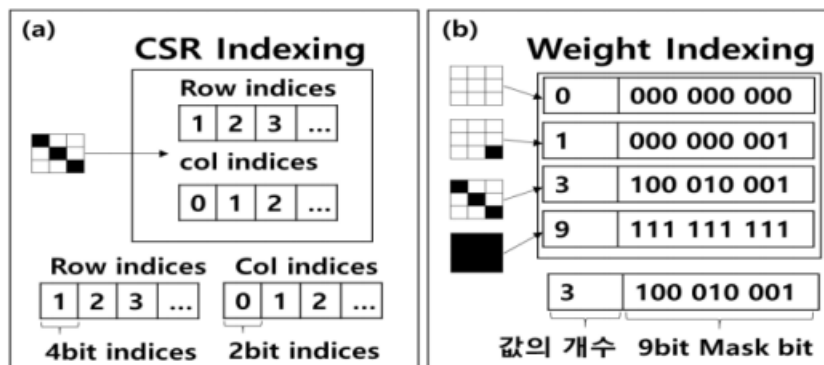
## - CSR indexing

희소행렬 처리 기법 중 하나로, 행렬 내의 0이 아닌 값에 행과 열로 표현되는 인덱스를 부여하여 해당 값의 행렬 내 위치정보를 알려주는 방식이다. Convolution 연산에서 주로 파라미터 압축을 수행하는 용도로 사용하고 있다.

CSR indexing에서는 가중치 행렬 내부에서 0인 원소를 2bit, 0이 아닌 원소를 4bit로 표현한다.

## - NZB indexing

가중치 행렬 내부의 0이 아닌 원소의 개수 정보와 행렬 내 원소의 값이 0의 값인지를 나타내는 bitmap 정보로 인덱싱을 수행한다.



CTA	희소행렬 처리를 통한 트윗 감성 분석 프로그램
	작성 날짜: 2020년 11월 25일

## 2.2 요구 사항 명세서

### 2.2.1 기능별 요구사항

- 웹 화면 ■■■■■ 계획 ■■■■■ 실적

일정	9 월	10 월				11 월			
	4	1	2	3	4	1	2	3	4
웹 & UI 설계	■■■■■	■■■■■	■■■■■	■■■■■					
웹 서버 구현									
분석 가능한 인물 프로필 조회			■■■■■	■■■■■	■■■■■	■■■■■	■■■■■		
분석 요청 화면			■■■■■	■■■■■	■■■■■	■■■■■	■■■■■		
분석 결과 조회 화면			■■■■■	■■■■■	■■■■■	■■■■■	■■■■■	■■■■■	
모델 배포 및 결과 제공									
NZB 기술 적용 모델 배포						■■■■■	■■■■■	■■■■■	■■■■■
분석 결과 시각화			■■■■■	■■■■■	■■■■■	■■■■■	■■■■■	■■■■■	■■■■■

요구사항명	데이터베이스내 유명인 목록 표시	ID	R-F-01	우선순위	중
요구사항설명	사용자가 분석 가능한 유명인들을 웹에서 확인한다.				
해결방안	분석 가능한 50명의 유명인들 프로필을 표시하고, 클릭하면 분석이 가능하다. 프로필에는 사진, 이름, 직업, 트위터 아이디를 표기한다.				
위험요소	사용자가 프로필을 실수로 클릭할 경우를 위해, 선택된 인물의 분석을 진행할지 확인하는 안내창을 보인다.				
설계 시 고려사항	50명의 프로필을 화면에 나타낼 경우, 가독성에 지장이 생길 수 있다.				
관련요구사항	분석 결과 조회				
시나리오	1. 사용자가 웹페이지에 존재하는 분석 가능한 유명인의 프로필을 확인한다. 2. 분석을 하고 싶은 유명인의 프로필을 클릭한다. 3. 해당 유명인의 트윗 분석을 진행할지 확인하는 안내창이 뜬다.				



CTA	희소행렬 처리를 통한 트윗 감성 분석 프로그램
	작성 날짜: 2020년 11월 25일

요구사항명	유명인 이름 검색	ID	R-F-02	우선순위	하
요구사항설명	사용자가 프로그램에서 분석하고 싶은 유명인 이름을 검색한다.				
해결방안	검색창에 유명인의 이름을 입력할 경우, 자동완성 기능으로 검색을 돕는다.				
위험요소	데이터베이스에 존재하지 않는 경우, 분석이 불가능하다고 안내한다.				
설계 시 고려사항	유명인의 이름, 예명을 정확하게 사용해야 한다.				
관련요구사항	데이터베이스내 유명인 목록 표시				
시나리오	1. 사용자가 웹페이지 중앙의 검색창에 유명인의 이름을 입력한다. 2. 해당 유명인이 존재할 경우 자동완성 기능으로 입력을 돕는다.				

요구사항명	트윗 분석 요청	ID	R-F-03	우선순위	상
요구사항설명	사용자가 프로그램에서 유명인의 트윗 분석을 요청한다.				
해결방안					
위험요소	해당 유명인의 트윗 데이터(영어)를 수집한 후 진행한다는 점을 공지한다. ( 2020.09.22 부터 매주 화요일마다 txt 파일로 저장 )				
설계 시 고려사항					
관련요구사항	데이터베이스내 유명인 목록 표시, 유명인 이름 검색				
시나리오	1. 사용자가 '데이터베이스내 유명인 목록 표시', '유명인 이름 검색'을 통해 분석할 인물을 선택한다. 2. 인물을 선택 후, 분석을 진행할지 확인하는 안내창을 통해 분석을 요청한다.				

CTA	희소행렬 처리를 통한 트윗 감성 분석 프로그램
	작성 날짜: 2020년 11월 25일



요구사항명	트윗 분석 결과 조회	ID	R-F-04	우선순위	상
요구사항설명	사용자가 확인하고 싶은 유명인의 트윗 분석 결과를 화면에 표시한다.				
해결방안	데이터 분석 결과를 통계 제출 후, 시각화를 통해 화면에 표시한다. 1. 긍정 / 중립 / 부정 문장의 비율과 해당 감성의 문장을 최대 3개 제공한다. 2. 자주 사용하는 단어 5개를 내림차순으로 나열한다.				
위험요소	사용자가 긍정 / 중립 / 부정 의 판단 기준을 궁금해 할 수 있다.				
설계 시 고려사항	긍정 / 중립 / 부정 의 기준을 잘 설정해야 한다. 데이터 수집 기간을 정확히 표시해야 한다.				
관련요구사항	트윗 분석 요청				
시나리오	1. 사용자가 트윗 분석을 요청해, 유명인 트윗 분석을 한다. 2. 분석 결과를 화면에 표시한다. 3. 사용자는 유명인의 트윗 게시물 내 문장의 감성 비율과 자주 사용하는 단어를 확인한다.				

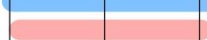
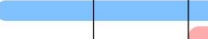
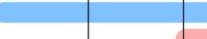
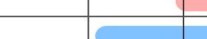
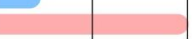


요구사항명	트윗 분석 결과 공유	ID	R-F-05	우선순위	하
요구사항설명	유명인의 트윗 분석 결과를 외부로 공유한다.				
해결방안	분석 결과를 조회할 때, 공유하기 버튼을 추가한다. 분석 결과를 트위터로 공유를 가능하게 한다.				
위험요소	공유 가능한 범위를 지정한다.				
설계 시 고려사항	많은 사람들이 사용하는 사이트의 공유 기능은 필수로 제공한다.				
관련요구사항	트윗 분석 결과 조회				
시나리오	1. 사용자가 트윗 분석 결과를 조회한다. 2. 사용자가 웹 외부로 분석 결과를 공유한다.				

CTA	희소행렬 처리를 통한 트윗 감성 분석 프로그램
	작성 날짜: 2020년 11월 25일

요구사항명	프로그램 이용 안내 확인	ID	R-F-06	우선순위	중
요구사항설명	웹 프로그램의 이용 안내를 확인한다.				
해결방안	웹 상단의 about 카테고리를 통해 이용 안내를 확인한다.				
위험요소					
설계 시 고려사항	웹의 어느 페이지에서도 접근 가능하도록 한다. 사용자가 사용하기 쉬운 UI로 구성한다.				
관련요구사항					
시나리오	1. 사용자가 웹 상단의 about을 클릭한다. 2. 사용자는 개발 의도, 서비스 설명(분석 방법), 개발자 소개를 확인한다.				

- 분석 모델

 계획  실적

일정	10 월				11 월			
	1	2	3	4	1	2	3	4
text-CNN 기존 모델								
text-CNN 모델 구현								
text-CNN 모델 학습								
text-CNN 모델 평가								
NZB 인덱싱 기술 적용 모델								
Weight Pruning 비교 및 적용								
NZB Indexing 기술 적용								
Convolution 연산부 수정								
NZB 적용 모델 평가								
기존 모델, 적용 모델 성능 비교								



CTA	희소행렬 처리를 통한 트윗 감성 분석 프로그램
	작성 날짜: 2020년 11월 25일

요구사항명	문장의 감성 분석	ID	R-F-07	우선순위	상
요구상황설명	트윗 문장의 감성 분석을 한다.				
해결방안	문장의 감성을 긍정 / 중립 / 부정 3가지로 분류한다.				
위험요소	문장 속 사전에 학습되지 않은 단어가 존재할 경우 분류의 정확도가 낮다.				
설계 시 고려사항	문장 속 이모지의 감성을 나타낸다.				
관련요구사항	트윗 분석 프로그램: 트윗 분석 요청, 데이터베이스: 분석할 데이터 제공				
시나리오	1. 데이터베이스에서 분석할 유명인의 트윗 데이터(txt)를 불러온다. 2. 트윗을 문장 단위로 분석해 문장의 긍정, 중립, 부정의 비율을 계산 후, 해당 감성의 문장 3개를 제공한다.				

요구사항명	자주 사용하는 단어 통계	ID	R-F-08	우선순위	중
요구상황설명	트윗 속 자주 사용하는 단어를 통계 낸다.				
해결방안	자주 사용하는 단어를 시각화해서 제공한다.				
위험요소					
설계 시 고려사항	특수 문자, 이모지를 사용하는 트윗이 많아 이모지도 같이 통계를 낸다.				
관련요구사항	트윗 분석 프로그램 : 트윗 분석 요청, 데이터베이스 : 분석할 데이터 제공				
시나리오	1. 데이터베이스에서 분석할 유명인의 트윗 데이터(txt)를 불러온다. 2. 자주 사용하는 단어의 통계를 낸 후, 5개의 단어를 제공한다. 3. 추가적으로, 상위 단어 100개를 워드 클라우드로 제공한다.				

CTA	희소행렬 처리를 통한 트윗 감성 분석 프로그램
	작성 날짜: 2020년 11월 25일

- 데이터 베이스

 계획  실적

일정	9 월			10 월				11 월			
	2	3	4	1	2	3	4	1	2	3	4
분석할 유명인 자료 조사											
트윗 데이터 수집											
트윗 데이터 전처리 (잘린 문장 복구)											
데이터베이스 구축											

요구사항명	수집한 데이터 업데이트	ID	R-F-09	우선순위	상
요구사항설명	트위터 API 라이선스 문제로 데이터 관리자가 수집한 데이터를 업데이트 한다.				
해결방안	트위터 API 라이선스 문제로 트윗 데이터를 실시간으로 가져올 수 없다. 데이터 관리자가 매주 화요일마다 50명의 트윗 데이터를 수집해 txt파일로 저장한다.				
위험요소	많은 시간이 소요된다.				
설계 시 고려사항	문장을 단위로 데이터를 수집한다.				
관련요구사항					
시나리오	1. 데이터 관리자는 50명의 트윗을 트위터 API 무료 버전을 통해 수집한다. 2. 글자 수 제한으로 잘린 문장을 확인 후 실제 문장으로 수정한다.				

CTA	희소행렬 처리를 통한 트윗 감성 분석 프로그램
	작성 날짜: 2020년 11월 25일

요구사항명	분석할 데이터 제공	ID	R-F-10	우선순위	중
요구사항설명	트윗 분석을 진행할 데이터를 데이터 분석 모델에 제공한다.				
해결방안	저장된 트윗 데이터 파일(txt)을 모델에 제공한다.				
위험요소					
설계 시 고려사항	유명인의 트위터 계정과, 이름, 트윗 데이터(txt파일)를 저장한다.				
관련요구사항	데이터 분석 모델 문장의 감성 분석, 자주 사용하는 단어 통계				
시나리오	1. 데이터 분석 모델에서 분석을 위해 데이터를 요청한다. 2. 데이터베이스에 미리 저장해둔 데이터를 제공한다.				

### 2.2.2 성능 요구사항

번호	요구사항 내용
R-P-01	기존 모델의 convolution 연산에 NZB 인덱싱 기법을 적용한 모델을 개발한다.
R-P-02	기존 연산 모델과 NZB 인덱싱 기법을 적용한 모델의 성능을 측정하고 비교한다. 비교 항목: 연산 처리 속도, 메모리 사용량, CPU 사용량, 정확도
R-P-03	실시간 처리를 통해 신속하고 정확하게 화면에 분석 결과 출력한다. -> 분석 시간 최장 30초 이내 처리한다.

CTA	희소행렬 처리를 통한 트윗 감성 분석 프로그램
	작성 날짜: 2020년 11월 25일

2.2.3 인터페이스 요구사항

번호	요구사항 내용
R-I-01	트위터 API 라이선스 문제로 유명한 50 명의 트윗을 데이터 관리자가 매주 화요일마다 데이터를 txt 파일로 수집 후, 글자 수 제한으로 잘린 문장을 확인해서 데이터베이스에 업데이트한다.
R-I-02	트위터 API 라이선스 문제로 데이터베이스에 저장된 50 명의 유명한 트윗만 분석이 가능하다.
R-I-03	데이터 분석 모델 훈련 시 영어 데이터를 사용해, 영어가 아닌 다른 언어의 분석은 불가능하다.
R-I-04	PC/모바일에 따라 별도의 화면을 제공한다. 브라우저에 제한을 받지 않는다.

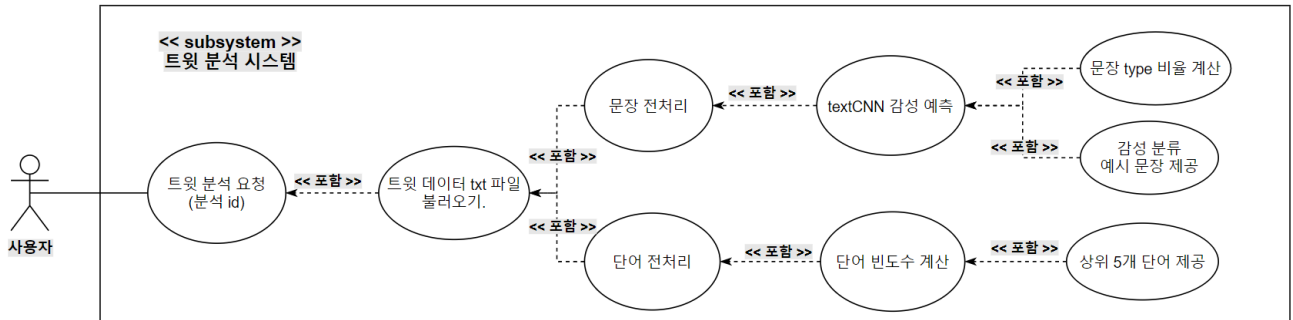
2.2.4 제약조건

소프트웨어 환경		하드웨어 환경		네트워크 환경	
Language	JavaScript, python, C++	CPU	Intel core i5	Cloud computing service	AWS
DBMS	MongoDB	RAM	8GB	VM	EC2
WAS	Node.js	O/S	Windows10	AMI	Ubuntu Server
Editor	Jupyter Notebook, Colab, PyCharm, VS Code			Server region	Seoul

CTA	희소행렬 처리를 통한 트윗 감성 분석 프로그램
	작성 날짜: 2020년 11월 25일

## 2.3 유스케이스 다이어그램

### 2.3.1 유스케이스 기술서



유스케이스명	단어 전처리
액터명	트윗 분석 시스템
개요	자주 사용하는 단어 5개를 제공하기 위해 문장을 전처리 한다.
사전 조건	사용자가 트윗 분석할 인물을 선택해야한다. 데이터베이스에 해당 인물의 데이터가 존재해야 한다. 데이터베이스와 API 서버가 잘 연결되어 있어야 한다. 데이터베이스에서 분석에 사용할 트윗 데이터 txt 파일을 가져온다.
정상 흐름	1. 트윗 분석 시스템은 데이터베이스에서 분석에 사용할 트윗 데이터 txt 파일을 가져온다. 2. 트윗 분석 시스템은 가져온 txt 파일을 string형 변수에 할당한다. 3. 트윗 분석 시스템은 string 데이터를 소문자로 변환 후, 구두점, URL, 숫자, 아이디, 이메일을 제거한다. 4. 트윗 분석 시스템은 단어를 토큰화해 불용어를 제거하면서 리스트로 바꾼다. 5. 트윗 분석 시스템은 단어 리스트에서 단어의 어휘 추출을 진행한다. 6. 트윗 분석 시스템은 데이터 문장을 단어 단위로 전처리를 했다.
사후 조건	트윗 분석 시스템은 전처리 된 단어 리스트를 가진다. 트윗 분석 시스템은 전처리 된 단어 리스트를 가지고 단어의 통계를 낸다.



CTA	희소행렬 처리를 통한 트윗 감성 분석 프로그램
	작성 날짜: 2020년 11월 25일

유스케이스명	단어 빈도수 계산
액터명	트윗 분석 시스템
개요	전처리 된 데이터 단어 리스트에서 자주 사용하는 단어 5개를 제공하기 위해 단어들의 빈도수를 계산한다.
사전 조건	트윗 분석 시스템은 "단어 전처리" 유스케이스를 실행한 후 전처리 된 단어 리스트를 가져온다.
정상 흐름	1. 트윗 분석 시스템은 데이터 문장을 단어 단위로 전처리를 했다. 2. 트윗 분석 시스템은 단어 리스트의 빈도수 계산을 한다.
사후 조건	트윗 분석 시스템은 단어 리스트의 빈도수를 계산했다. 트윗 분석 시스템은 자주 사용하는 단어 순으로 정렬했다.

유스케이스명	문장 전처리
액터명	트윗 분석 시스템
개요	문장의 감성을 예측하기 위해 데이터를 전처리 한다.
사전 조건	사용자가 트윗 분석할 인물을 선택해야한다. 데이터베이스에 해당 인물의 데이터가 존재해야 한다. 데이터베이스와 API 서버가 잘 연결되어 있어야 한다.
정상 흐름	1. 트윗 분석 시스템은 데이터베이스에서 분석에 사용할 트윗 데이터 txt 파일을 가져온다. 2. 트윗 분석 시스템은 가져온 txt 파일을 string형 변수에 할당한다. 3. 트윗 분석 시스템은 string형 트윗을 dataframe 형 변수에 할당한다. 4. 트윗 분석 시스템은 문장에 있는 이모지를 단어로 변환한다. 5. 트윗 분석 시스템은 문장의 url, hashtag 등을 특정 단어로 전처리 한다.
사후 조건	트윗 분석 시스템은 전처리 된 문장들을 dataframe 형태로 가진다. 트윗 분석 시스템은 전처리 된 문장들을 가지고 감성을 예측한다.

CTA	희소행렬 처리를 통한 트윗 감성 분석 프로그램
	작성 날짜: 2020년 11월 25일

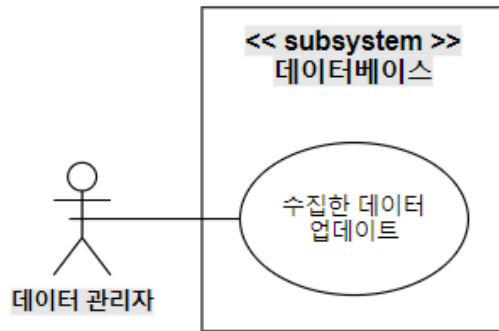
유스케이스명	상위 5개 단어 제공
액터명	트윗 분석 시스템
개요	빈도수로 정렬된 단어 리스트에서 상위 5개의 단어를 제공한다.
사전 조건	트윗 분석 시스템은 "단어 빈도수 계산" 유스케이스를 실행한 후 단어 리스트를 빈도수로 정렬을 했다.
정상 흐름	1. 트윗 분석 시스템은 단어 리스트의 빈도수 계산을 한다. 2. 트윗 분석 시스템은 단어 리스트에서 제일 많이 사용된 단어 5개를 제공한다.
사후 조건	트윗 분석 시스템은 단어 리스트에서 제일 많이 사용된 단어 5개를 제공받는다. 추가적으로 트윗 분석 시스템은 상위 단어 100개의 단어를 워드 클라우드로 제공받는다.

유스케이스명	textCNN 감성 예측
액터명	트윗 분석 시스템
개요	전처리된 문장들의 감성을 예측한다.
사전 조건	트윗 분석 시스템은 "문장 전처리" 유스케이스를 실행한 후 전처리 된 문장들을 가진다.
정상 흐름	1. 트윗 분석 시스템은 데이터 문장들을 전처리 했다. 2. 트윗 분석 시스템은 미리 저장해둔 감성 분석 모델을 가져온다. 3. 트윗 분석 시스템은 라벨 인코더를 통해 문장을 숫자로 변환한다. 4. 문장들의 감성을 문장 단위로 감성 분석을 한다. 감성 분석은 0 - 1 실수 값으로 수치화 한 후, 0.35 미만: negative / 0.7 초과: positive / 그 외의 값: neutral 3가지의 타입으로 분류한다.
사후 조건	트윗 분석 시스템은 문장들의 감성을 예측했다.

CTA	희소행렬 처리를 통한 트윗 감성 분석 프로그램
	작성 날짜: 2020년 11월 25일

유스케이스명	문장 type 비율 계산
액터명	트윗 분석 시스템
개요	문장들의 3가지 감성 비율을 계산한다.
사전 조건	트윗 분석 시스템은 "textCNN 감성 예측" 유스케이스를 실행한 후 문장들의 감성을 예측했다.
정상 흐름	1. 트윗 분석 시스템은 데이터 문장을 전처리 했다. 2. 트윗 분석 시스템은 문장들의 감성을 예측했다. 3. 트윗 분석 시스템은 3가지 감성의 비율을 계산한다.
사후 조건	트윗 분석 시스템은 문장들의 3가지 감성 비율을 계산했다.

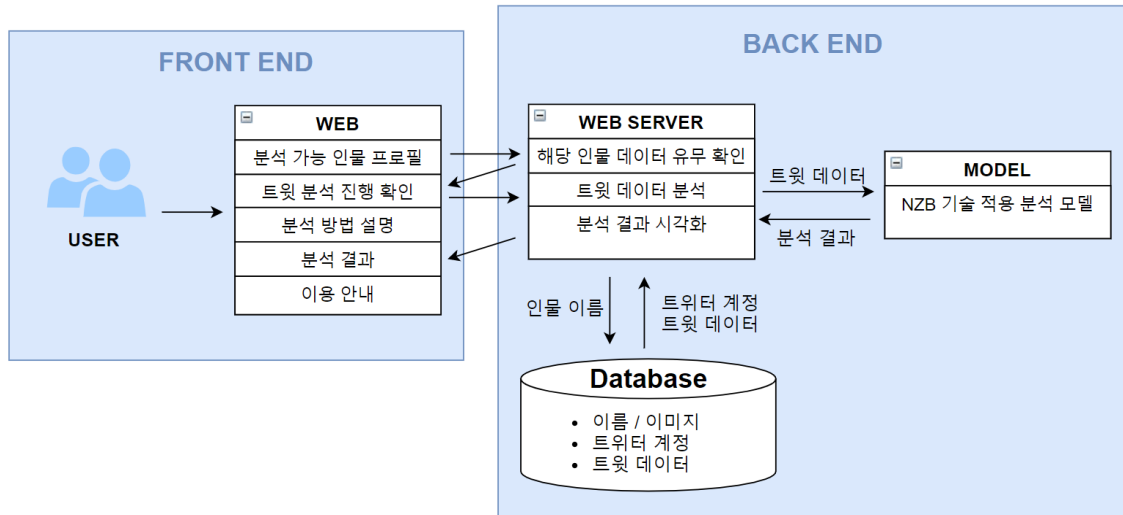
유스케이스명	감성 분류 예시 문장 제공
액터명	트윗 분석 시스템
개요	3가지 감성으로 분류된 실제 문장을 최대 3개까지 제공한다.
사전 조건	트윗 분석 시스템은 "textCNN 감성 예측" 유스케이스를 실행한 후 문장들의 감성을 예측했다.
정상 흐름	1. 트윗 분석 시스템은 데이터 문장을 전처리 했다. 2. 트윗 분석 시스템은 문장들의 감성을 예측했다. 3. 트윗 분석 시스템은 3가지 감성에 해당하는 문장을 최대 3개까지 제공한다. positive: 제일 높은 점수의 문장 neutral: 점수에서 - 0.5의 절댓값의 크기가 제일 작은 문장 negative: 제일 낮은 점수의 문장
사후 조건	트윗 분석 시스템은 3가지 감성에 해당하는 문장을 최대 3개까지 제공했다.



유스케이스명	수집한 데이터 업데이트
액터명	주액터: 데이터 관리자, 부액터: 데이터베이스
개요	분석에 사용할 데이터를 직접 수집한 후 업데이트한다.
사전 조건	데이터베이스에 기존에 저장된 50명의 데이터가 있어야한다. 데이터는 인물의 이름(str), 트위터 계정(str), 직업(str), 이미지(png), 트윗 내용(txt)으로 구성된다.
정상 흐름	1. 데이터 관리자는 매주 화요일마다 트위터 API를 이용해 50명의 트윗을 txt파일로 저장한다. 2. 트위터 API 제약 조건으로 잘린 내용을 데이터 관리자는 직접 확인 후 수정한다. 3. 데이터 관리자는 확인을 마친 데이터를 데이터베이스에 업데이트한다.
사후 조건	데이터베이스에 매주 월요일까지의 트윗 데이터가 업데이트 되었다.

### 3. 시스템 분석

#### 3.1 시스템 구조



##### 3.1.1 웹

- ABOUT 에서 분석 알고리즘 확인, CONTACT 에서 개발자에게 연락 가능하다.
- 분석 가능한 인물의 프로필을 확인 후, 트위터 분석을 요청할 수 있다.

##### 3.1.2 웹 서버

- 사용자가 웹에서 분석 요청한 인물의 데이터가 존재하는지 데이터베이스에서 확인을 한다.
- predict 엔드 포인트로 요청이 들어온 경우 텍스트 문서를 전달받아 전처리 함수와 단어 카운트함수, text-CNN 모델을 수행하는 함수 실행한다.
- 분석 모델에서 제공받은 분석 결과를 시각화 한다.

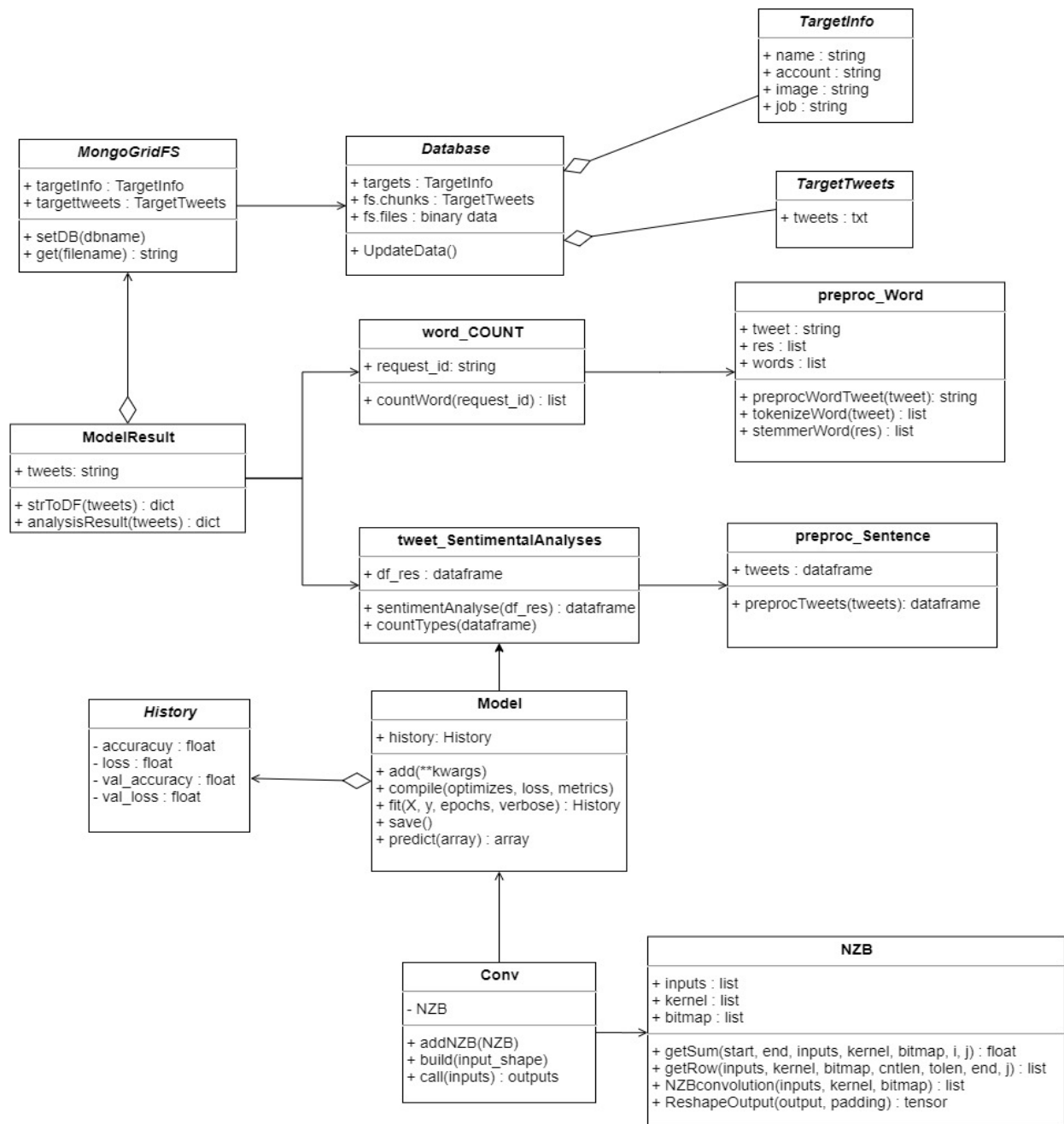
##### 3.1.3 모델

- text-CNN 모델에 NZB Indexing 기술을 적용해 convolution 연산을 수행한다.
- 긍정/중립/부정 3 가지 타입의 결과로 문장의 감성분석을 진행 후, 타입 별 비율을 계산 후, 해당하는 3 가지 타입의 문장 결과를 3 개씩 제공한다.
- 자주 사용하는 단어를 5 개까지 제공한다.

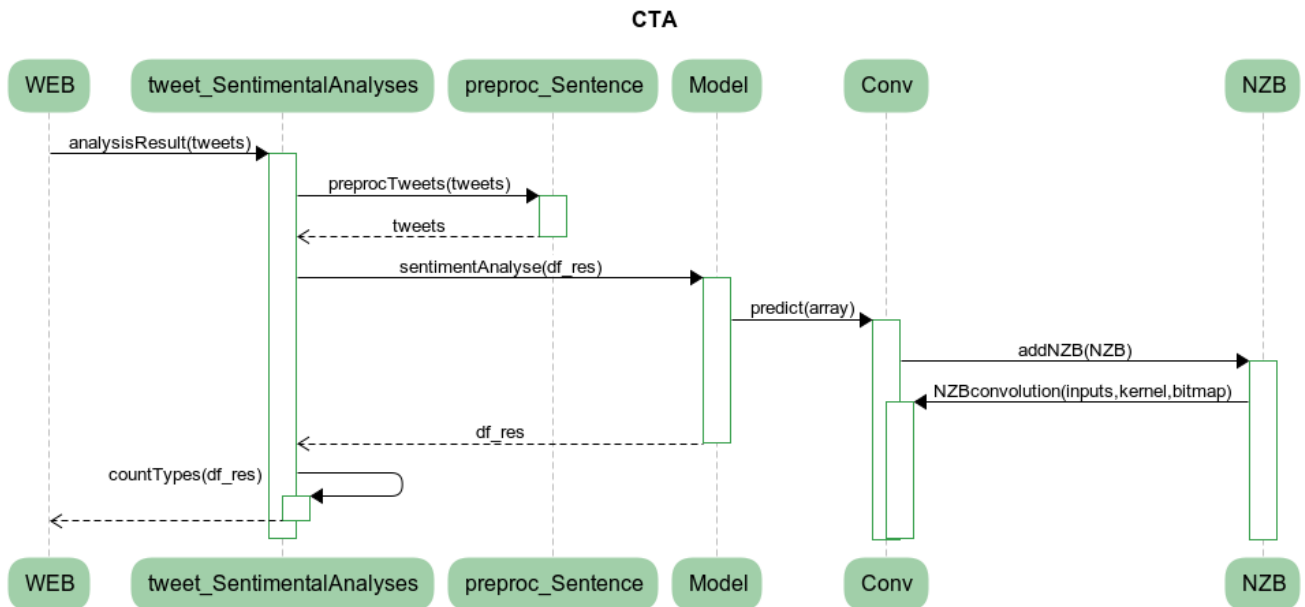
### 3.1.4 데이터베이스

- 50 명의 데이터는 "이름/트위터 계정/이미지/트윗 데이터" 구성으로 저장되어 있다.

### 3.2 클래스 다이어그램

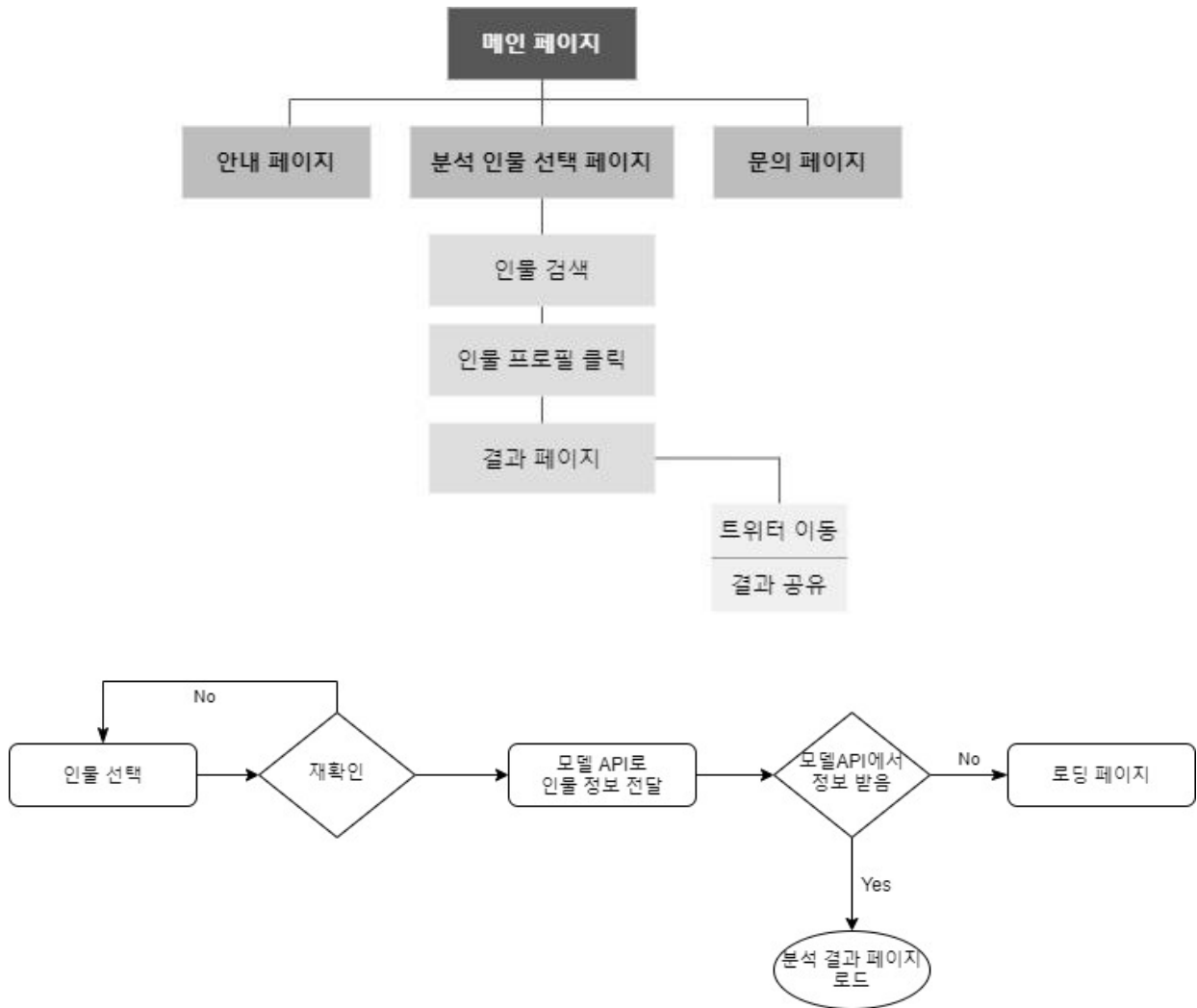


## 3.3 시퀀스 다이어그램



#### 4. 시스템 설계

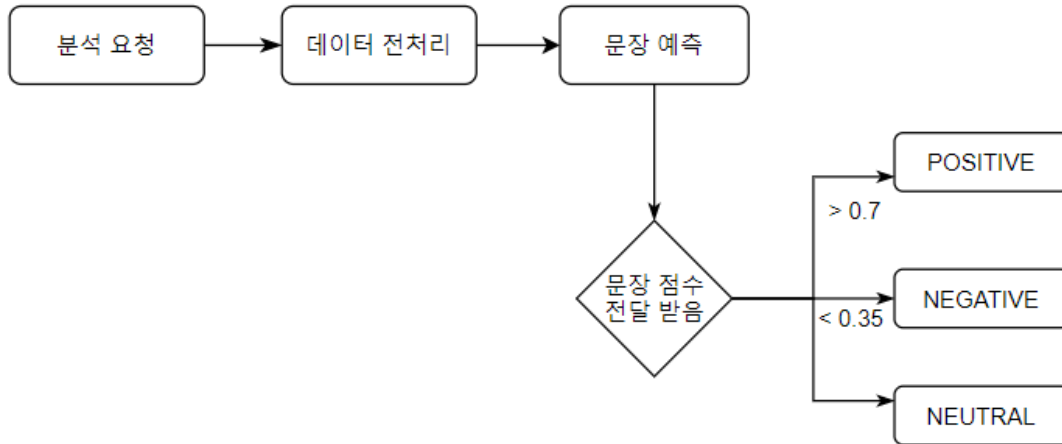
##### 4.1 사용자 인터페이스 설계



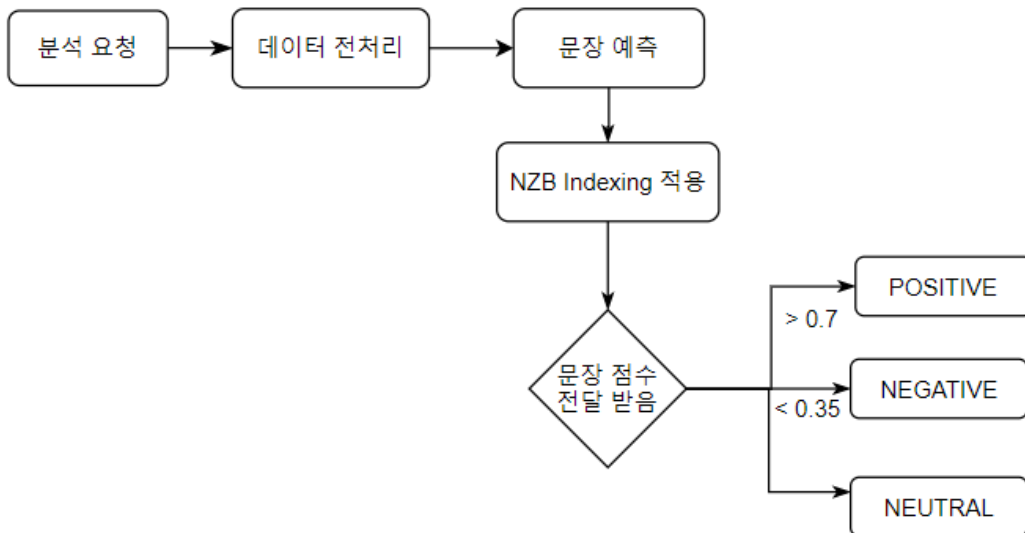


## 4.2 분석 모델 설계

### 4.2.1 기존 분석 모델



### 4.2.2 NZB Indexing 적용 분석 모델



### 4.3 데이터베이스 설계

Targets Document

```
{
  _id : <ObjectId>,
  name : {
    type: String,
    maxlength:50
  },
  account:{
    type:String,
    trim:true
  },
  image:String,
  job:{
    type: String
  }
}
```

fs.files Document

```
{
  _id : ObjectId,
  length : Int64,
  chunkSize : Int32,
  uploadDate : date,
  filename : String,
  metadata : Object
}
```

fs.chunks Document

```
{
  _id : ObjectId,
  files_id : ObjectId,
  n : Int32,
  data : Binary
}
```

## 5. 시스템 구현

### 5.1 실제 코드 및 결과

#### 5.1.1 모델 빌드

- base text-CNN 모델

```

1 model = Sequential()
2
3 model.add(Embedding(vocab_size, 32, input_length=max_len))
4
5 model.add(Conv1D(filters=128, kernel_size=5, padding='same', activation='relu'))
6 model.add(MaxPooling1D(pool_size=2))
7 model.add(Dropout(0.2))
8
9 model.add(Conv1D(filters=64, kernel_size=6, padding='same', activation='relu'))
10 model.add(MaxPooling1D(pool_size=2))
11 model.add(Dropout(0.2))
12
13 model.add(Conv1D(filters=32, kernel_size=7, padding='same', activation='relu'))
14 model.add(MaxPooling1D(pool_size=2))
15 model.add(Dropout(0.2))
16
17 model.add(Conv1D(filters=32, kernel_size=8, padding='same', activation='relu'))
18 model.add(MaxPooling1D(pool_size=2))
19 model.add(Dropout(0.2))
20
21 model.add(Flatten())
22 model.add(Dense(1, input_shape=(1,)))
23 model.compile('SGD', 'mse', metrics=['accuracy'])
24 model.summary()
25
26 model.fit(X_train, y_train, epochs=8, verbose=1)

```

```

1 weight = model.get_weights()
2 weight[1]

```

```

array([[[-0.08483198,  0.10879445, -0.0392587 , ..., -0.06532168,
        -0.01460742,  0.10556462],
        [-0.08496561, -0.00843406,  0.01690301, ..., -0.07297008,
         0.08761676, -0.12777297],
        [ 0.04901376, -0.05214268, -0.07659314, ...,  0.00484493,
        -0.02682856,  0.00608644],
        ...,
        [ 0.0069919 , -0.01444536, -0.05929992, ..., -0.03798576,
         0.04171865, -0.17185484],
        [-0.00214236,  0.03407577,  0.0150112 , ..., -0.0470692 ,
         0.01037525,  0.14364514],
        [ 0.04413214, -0.01114238,  0.07179853, ..., -0.0252963 ,
         0.02178209,  0.05814373]],

```

- prune text-CNN 모델 (코드는 85%의 pruning, weight 사진은 80%의 pruning 결과)

```

1 base_model = load_model(my_path+'text-CNN.h5') # basic model
2
3 ConstantSparsity = pruning_schedule.ConstantSparsity
4
5 pruning_params = {
6     'pruning_schedule': ConstantSparsity(0.85, begin_step=2000, frequency=100)
7 }
8
9 def apply_pruning_to_dense(layer):
10     if isinstance(layer, tf.keras.layers.Conv1D):
11         return tfmot.sparsity.keras.prune_low_magnitude(layer, **pruning_params)
12     return layer
13
14 model_for_pruning2 = tf.keras.models.clone_model(
15     base_model,
16     clone_function=apply_pruning_to_dense, )
17
18 model_for_pruning2.compile('SGD', 'mse', metrics=['accuracy'])
19 model_for_pruning2.summary()
20
21 log_dir = tempfile.mkdtemp()
22 callbacks = [
23     tfmot.sparsity.keras.UpdatePruningStep(),
24     # Log sparsity and other metrics in Tensorboard.
25     tfmot.sparsity.keras.PruningSummaries(log_dir=log_dir)
26 ]
27 model_for_pruning2.compile('SGD', 'mse', metrics=['accuracy'])
28 model_for_pruning2.fit(X_train, y_train, callbacks = callbacks, epochs = 8, verbose = 1)

```

```

1 weight = model.get_weights()
2 weight[1]

```

```

array([[[-0.08490542,  0.13946635,  0.          , ..., -0.          ,
          0.          ,  0.14509021],
        [-0.08522931,  0.          ,  0.          , ..., -0.          ,
          0.10017765, -0.18178467],
        [-0.          , -0.          , -0.08196158, ..., -0.          ,
          0.          , -0.          ]],
       ...,
        [ 0.          , -0.          ,  0.          , ...,  0.          ,
         -0.          , -0.19687262],
        [ 0.          ,  0.          , -0.          , ..., -0.          ,
         -0.          ,  0.18011148],
        [-0.          ,  0.          , -0.          , ...,  0.          ,
         -0.          ,  0.          ]],

```

## 5.1.2 NZB Indexing

```

1 def NZBIndexing(array):
2     a = array.shape
3     indexnum = 1
4
5     for i in range(0, len(a) - 1):
6         indexnum *= a[i] #총 원소 개수
7
8     sizelen = len(bin(indexnum)) - 2 # 원소의 길이
9     non_zero = 0
10    totalbitmap = []
11    ind_size = a[1]
12
13    for k in range(0, a[2]):
14        bitmap = [0 for i in range(sizelen + indexnum)] #bitmap 길이 설정
15        for i in range(0, a[0]):
16            for j in range(0, a[1]):
17                if (array[i][j][k] != 0):
18                    non_zero += 1
19                    bitmap[sizelen + i * ind_size + j] = 1
20
21        #0이 아닌 원소 개수 bit array 로 반환
22        for i in range(sizelen - 1, -1, -1):
23            bitmap[sizelen - 1 - i] = (int)(non_zero / pow(2, i))
24            non_zero -= bitmap[sizelen - 1 - i] * pow(2, i)
25
26        totalbitmap.insert(len(totalbitmap), bitmap)
27
28    totalbitmap = np.array(totalbitmap, dtype = np.uint8)
29
30    return totalbitmap

```

```
1 NZBList[0][0]
```

```

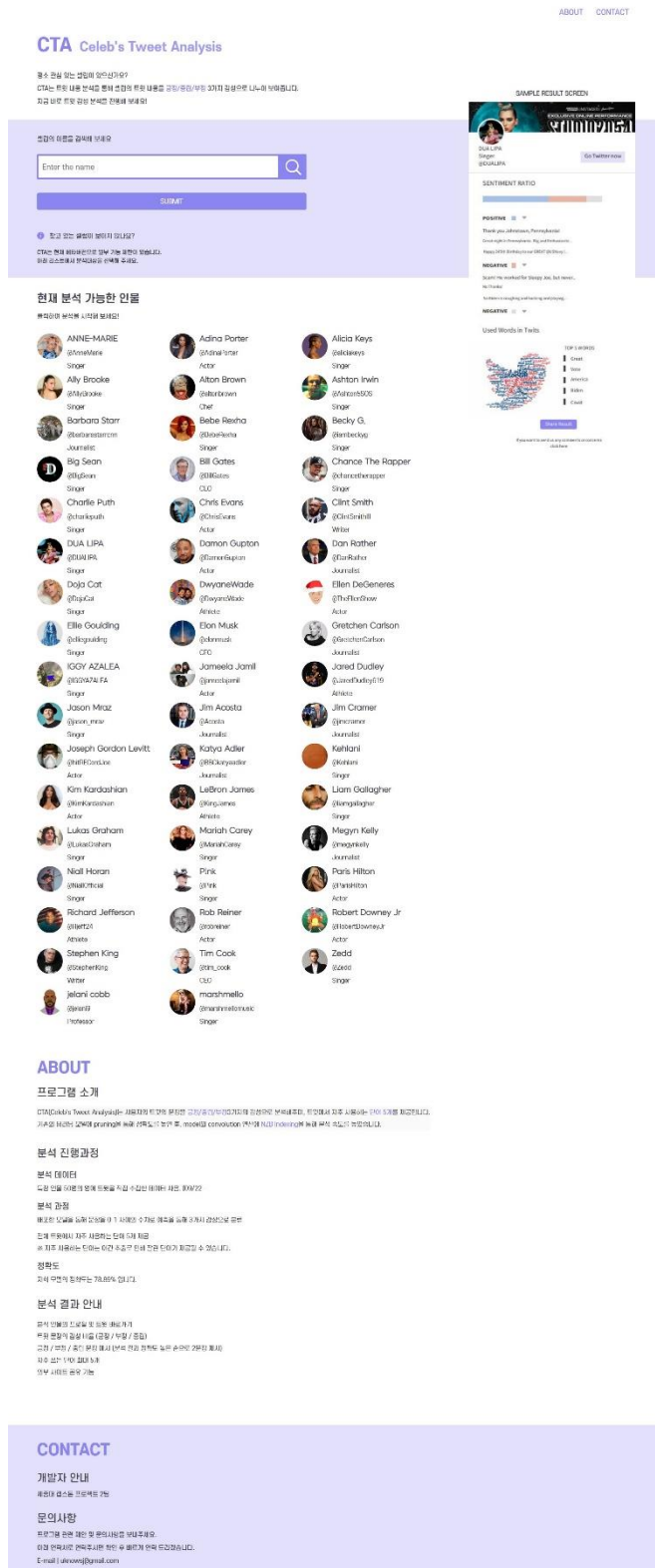
array([0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0,
       1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=uint8)

```

## 5.1.3 Convolution 연산 수정

```
59 @njit(parallel=True, fastmath=True)
60 def getSum(start, end, inputs, kernel, bitmap, i, j):
61     sum = 0
62     length = input.shape[2]
63     for k in prange(start, end):
64         if (bitmap[k] == 1):
65             r = (int)((k - start) / length)
66             c = (k - start) % length
67             sum += input[0][i + r][c] * kernel[r][c][j]
68     return sum
69
70 @njit(parallel=True)
71 def getRow(inputs, kernel, bitmap, cntlen, tolen, end, j):
72     # parallel되어야 할 부분
73     rowoutput = [getSum(cntlen, tolen, inputs, kernel, bitmap, i, j) for i in prange(0, end)]
74     return rowoutput
75
76 @njit(parallel=True)
77 def NZBconvolution(inputs, kernel, bitmap):
78     # output 초기화
79     output_shape = (kernel.shape[2], inputs.shape[1] - kernel.shape[0] + 1)
80     output = np.empty(shape=output_shape, dtype=np.float32) # padding 전 shape
81
82     # 변수 설정
83     dim = inputs.shape[1]
84     filter_size = kernel.shape[0]
85     cntlen = (len(bitmap[0]) - kernel.shape[0] * inputs.shape[2]) # 0 아닌 비트 개수 길이
86     tolen = (len(bitmap[0])) # bitmap 총 길이
87     end = inputs.shape[1] - kernel.shape[0] + 1
88
89     for j in prange(0, kernel.shape[2]):
90         output[j] = getRow(inputs, kernel, bitmap[j], cntlen, tolen, end, j)
91
92     return output
93
94 def ReshapeOutput(output, padding):
95     output = output.reshape(output.shape[1], output.shape[0])
96     output = np.pad(output, ((0, padding), (0, 0)), 'constant', constant_values=0)
97     output.tolist()
98     output = [output]
99     return output
```

### 5.2.1 웹 접속 화면



## CTA

희소행렬 처리를 통한 트윗 감성 분석 프로그램

작성 날짜: 2020년 11월 25일

### 5.2.2 ABOUT & CONTACT 화면

#### ABOUT

##### 프로그램 소개

CTA(Celeb's Tweet Analysis)는 사용자의 트윗의 문장을 긍정/중립/부정 3가지의 감성으로 분석해주며, 트윗에서 자주 사용하는 단어 5개를 제공합니다. 기존의 딥러닝 모델에 pruning을 통해 정확도를 높인 후, model의 convolution 연산에 NZB Indexing을 통해 분석 속도를 높였습니다.

##### 분석 진행과정

###### 분석 데이터

특정 인물 50명의 영어 트윗을 직접 수집한 데이터 사용. (09/22 -

###### 분석 과정

배포한 모델을 통해 문장을 0-1 사이의 수치로 예측을 통해 3가지 감성으로 분류

전체 트윗에서 자주 사용하는 단어 5개 제공

※ 자주 사용하는 단어는 어간 추출로 인해 잘린 단어가 제공될 수 있습니다.

###### 정확도

자희 모델의 정확도는 78.89% 입니다.

##### 분석 결과 안내

분석 인물의 프로필 및 트윗 바로가기

트윗 문장의 감성 비율 (긍정 / 부정 / 중립)

긍정 / 부정 / 중립 문장 예시 (분석 결과 정확도 높은 순으로 2문장 제시)

자주 쓰는 단어 최대 5개

외부 사이트 공유 가능

#### CONTACT

##### 개발자 안내

새종대 캠퍼스 프로젝트 2팀

##### 문의사항

프로그램 관련 제안 및 문의사항을 보내주세요.

아래 연락처로 연락주시면 확인 후 빠르게 연락 드리겠습니다.

E-mail | uknows@gmail.com

### 5.2.3 분석 대상 검색 화면

CTA는 트윗 내용 분석을 통해 셀럽의 트윗 내용을 긍정/중립/부정 3가지 감성으로 나누어 보여줍니다.

지금 바로 트윗 감성 분석을 진행해 보세요!

셀럽의 이름을 검색해 보세요

ANNE-MARIE

SUBMIT

🔍 찾고 있는 셀럽이 보이지 않나요?

CTA는 현재 베타버전으로 일부 기능 제한이 있습니다.  
아래 리스트에서 분석대상을 선택해 주세요.

##### 현재 분석 가능한 인물

클릭하여 분석을 시작해 보세요!



ANNE-MARIE  
@AnneMarie  
Singer

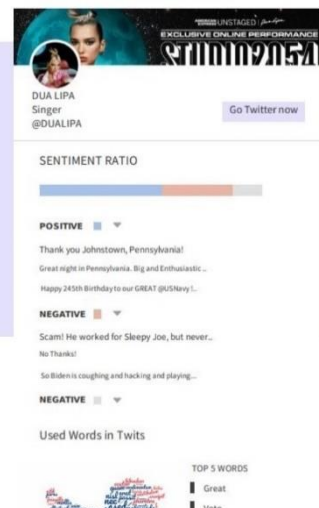


Adina Porter  
@AdinaPorter  
Actor



Alicia Keys  
@aliciakeys  
Singer

SAMPLE RESULT SCREEN





## 5.2.4 분석 결과 조회 화면

[ABOUT](#) [CONTACT](#)

## CTA Celeb's Tweet Analysis

## 트위터 감성 분석 결과



Analysis Target | Bill Gates

Job | CEO

Twitter Account | @BillGates

[Go Twitter Now](#)

## 트위터 문장 감성 비율



## POSITIVE

Congratulations to President-Elect Biden and Vice President-Elect Harris.

I really appreciated the thoughtful conversation.

Thank you @MohamedBinZayed for your continued commitment to polio eradication.

## NEGATIVE

To prevent the worst effects of climate change, we need innovation across all sectors—especially in the hardest to decarbonize sectors—to get us on a viable path to net-zero emissions.

I've been working for some time on a book about what we need to do over the next decade to avoid a climate disaster.

The COVID-19 pandemic has set back efforts to end poverty, hunger, and malnutrition.

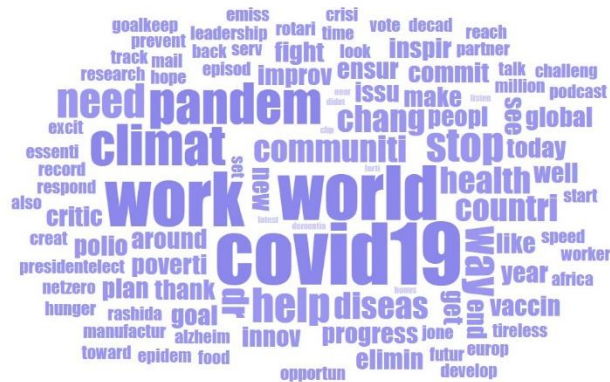
## NEUTRAL

More than 7 million community health workers serve their neighbors around the world, improving access to primary healthcare for their communities.

I'm inspired by the thousands of people in the organization who have committed their lives to fighting hunger.

What we need now are practical plans to reach those goals.

## Used Words in Twits



## TOP 5 WORDS

- covid19
- world
- work
- pandem
- climat

[SHARE RESULT](#)

프로그램 관련 제안 및 문의사항을 보내주세요.  
문의하기

CTA	희소행렬 처리를 통한 트윗 감성 분석 프로그램
	작성 날짜: 2020년 11월 25일

6. 팀원 역할 분담

Task 목록	진행자
PM 및 문서 작성	노수민
회의록 작성	홍주희
PPT 제작 및 대본 초안 작성	양승주
분석 가능한 특정 유명인 조사	노수민, 홍주희
데이터 수집 및 전처리 (잘린 문장 복구)	노수민, 홍주희
워드 임베딩 진행	양승주
Text-CNN 기존 모델 구현	이아현
학습 데이터 조사 및 수집	노수민, 양승주, 이아현, 홍주희
이미지 데이터셋 제작	홍주희
모델 학습	노수민, 양승주, 이아현, 홍주희
분석 데이터 전처리 과정 구현	노수민, 홍주희
문장 감성 분석 결과 제공 구현	노수민, 홍주희
자주 사용하는 단어 카운트 구현	노수민
웹 설계 및 웹 서버 구현	양승주
NZB 인덱싱 기술 자료 조사	이아현, 노수민(SUB), 홍주희(SUB)
NZB 인덱싱 기술 구현	노수민, 이아현, 홍주희(SUB)
데이터베이스 구축	양승주
모델 평가 및 검증	노수민, 이아현, 홍주희
분석 결과 시각화	양승주

CTA	희소행렬 처리를 통한 트윗 감성 분석 프로그램
	작성 날짜: 2020년 11월 25일

## 7. 성능 테스트

### 7.1 테스트 개요

#### 7.1.1 테스트 목적

- NZB Indexing 을 사용하기 위해 희소행렬을 만들어 모델의 pruning 을 진행한다.
- 기존 모델의 가중치 행렬을 NZB Indexing 기술로 표현했을 때, 메모리 접근 횟수 과 연산 수행 시간의 단축을 통해 전반적인 성능을 향상시킨다.

#### 7.1.2 테스트 대상

- pruning 단계별 모델의 정확도 비교
- pruning 모델의 NZB Indexing 적용 유무 비교

#### 7.1.3 가정 및 제약사항

- NZB Indexing 기술은 model 에 존재하는 4 개 층의 convolution layer 중 3 개 층에만 적용이 가능하다.
- GPU 연산을 처리 결과를 확인이 불가능 해, CPU 연산으로 진행을 한다.
- 성능 비교는 1 명의 데이터로 비교를 진행한다.

## 7.2 테스트 단위

### 7.2.1 정확도 테스트

(기존 모델 vs. 가지치기 모델)

pruning (%)	X	70%	75%	80%	85%
fit(epochs = 8))	79.08%	81.09%	81.01%	80.94%	80.85%
evaluate()	78.48%	78.37%	78.76%	78.89%	78.81%

<p>basic</p> <pre>[29] 1 %time 2 score = model.evaluate(X_test, y_test, batch_size = 64)  5000/5000 [=====] - 47s 9ms/step - loss: 0.1487 - accuracy: 0.7848 CPU times: user 1min 13s, sys: 2.72 s, total: 1min 15s Wall time: 47.2 s</pre> <p>pruned 70</p> <pre>[30] 1 %time 2 model1.compile('SGD', 'mse', metrics=['accuracy']) 3 score = model1.evaluate(X_test, y_test, batch_size = 64)  5000/5000 [=====] - 46s 9ms/step - loss: 0.1494 - accuracy: 0.7837 CPU times: user 1min 12s, sys: 2.79 s, total: 1min 15s Wall time: 46.1 s</pre> <p>pruned 75</p> <pre>[31] 1 %time 2 model2.compile('SGD', 'mse', metrics=['accuracy']) 3 score = model2.evaluate(X_test, y_test, batch_size = 64)  5000/5000 [=====] - 45s 9ms/step - loss: 0.1473 - accuracy: 0.7876 CPU times: user 1min 12s, sys: 2.65 s, total: 1min 14s Wall time: 45.6 s</pre> <p>pruned 80</p> <pre>[32] 1 %time 2 model3.compile('SGD', 'mse', metrics=['accuracy']) 3 score = model3.evaluate(X_test, y_test, batch_size = 64)  5000/5000 [=====] - 47s 9ms/step - loss: 0.1470 - accuracy: 0.7889 CPU times: user 1min 13s, sys: 3.5 s, total: 1min 16s Wall time: 47.1 s</pre> <p>pruned 85</p> <pre>[68] 1 %time 2 model4.compile('SGD', 'mse', metrics=['accuracy']) 3 score = model4.evaluate(X_test, y_test, batch_size = 64)  5000/5000 [=====] - 47s 9ms/step - loss: 0.1473 - accuracy: 0.7881 CPU times: user 1min 12s, sys: 4.95 s, total: 1min 17s Wall time: 47.3 s</pre>	<p>basic</p> <pre>[21] 1 %time 2 score = model.evaluate(X_test, y_test, batch_size = BATCH_SIZE)  313/313 [=====] - 33s 106ms/step - loss: 0.1487 - accuracy: 0.7848 CPU times: user 1min 1s, sys: 691 ms, total: 1min 1s Wall time: 33.7 s</pre> <p>pruned 70</p> <pre>[22] 1 %time 2 model1.compile('SGD', 'mse', metrics=['accuracy']) 3 score = model1.evaluate(X_test, y_test, batch_size = BATCH_SIZE)  313/313 [=====] - 34s 107ms/step - loss: 0.1494 - accuracy: 0.7837 CPU times: user 1min 2s, sys: 669 ms, total: 1min 2s Wall time: 34 s</pre> <p>pruned 75</p> <pre>[23] 1 %time 2 model2.compile('SGD', 'mse', metrics=['accuracy']) 3 score = model2.evaluate(X_test, y_test, batch_size = BATCH_SIZE)  313/313 [=====] - 34s 107ms/step - loss: 0.1473 - accuracy: 0.7876 CPU times: user 1min 1s, sys: 781 ms, total: 1min 2s Wall time: 34 s</pre> <p>pruned 80</p> <pre>[24] 1 %time 2 model3.compile('SGD', 'mse', metrics=['accuracy']) 3 score = model3.evaluate(X_test, y_test, batch_size = BATCH_SIZE)  313/313 [=====] - 34s 110ms/step - loss: 0.1470 - accuracy: 0.7889 CPU times: user 1min 3s, sys: 763 ms, total: 1min 4s Wall time: 34.7 s</pre> <p>pruned 85</p> <pre>[70] 1 %time 2 model4.compile('SGD', 'mse', metrics=['accuracy']) 3 score = model4.evaluate(X_test, y_test, batch_size = BATCH_SIZE)  313/313 [=====] - 34s 109ms/step - loss: 0.1473 - accuracy: 0.7881 CPU times: user 1min 2s, sys: 908 ms, total: 1min 3s Wall time: 34.5 s</pre>
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

=> weight pruning 을 통해 모델의 정확도가 향상됐다.

## 7.2.2 CPU 사용량 &amp; Memory 사용량

## @elonmusk 분석 결과

- NZB Indexing 적용 전

```
트윗 문장 감성 비율
POSITIVE 51.66%
NEGATIVE 24.32%
NEUTRAL 24.02%
Name: label, dtype: object

      text      label      score  elapsed_time
389  Congratulations!  POSITIVE  0.970972      0.051141
303  Congratulations SpaceX Team!  POSITIVE  0.969335      0.045336
434  Glad Jen is safe!  POSITIVE  0.960235      0.043845

      text  ... elapsed_time
91  Mild sniffles & cough & slight fever past few ...  0.045404
509  Too bad that place got torn down. ...  0.045176
380  🙄 ...  0.052922

[3 rows x 4 columns]

      text  ...  cal
213  Even the small details matter. ...  0.001797
425  Will release order configurator probably in Jan ...  0.001939
378  You never know ...  0.002596

[3 rows x 5 columns]

자주 사용하는 단어 TOP5
[('test', 30), ('product', 27), ('tesla', 26), ('great', 26), ('need', 23)]

cpu usage      : 53.90.0 %
memory usage    : 0.61 %
```

- NZB Indexing 적용 후

적용 층	결과
Layer 1	cpu usage : 30.4 % memory usage : 0.69 %
Layer 1, 2	cpu usage : 44.7 % memory usage : 0.66 %
Layer 1, 2, 3	cpu usage : 56.4 % memory usage : 0.66 %

=> NZB Indexing 적용을 통해 CPU 사용량, Memory 사용량이 전반적으로 증가했다.

## 7.2.3 분석 수행 시간

## @elonmusk 분석 결과

## - NZB Indexing 적용 전

```

트윗 문장 감성 비율
POSITIVE 51.66%
NEGATIVE 24.32%
NEUTRAL 24.02%
Name: label, dtype: object

      text      label      score  elapsed_time
389  Congratulations!  POSITIVE  0.970972      0.046137
303  Congratulations SpaceX Team!  POSITIVE  0.969335      0.043981
434  Glad Jen is safe!  POSITIVE  0.960235      0.047925

      text ... elapsed_time
91  Mild sniffles & cough & slight fever past few ... ... 0.036287
509  Too bad that place got torn down. ... 0.050213
380  🙄 ... 0.047230

[3 rows x 4 columns]

      text ... cal
213  Even the small details matter. ... 0.001797
425  Will release order configurator probably in Jan ... 0.001939
378  You never know ... 0.002596

[3 rows x 5 columns]

자주 사용하는 단어 TOP5
[('test', 30), ('product', 27), ('tesla', 26), ('great', 26), ('need', 23)]

소요시간 32.002538204193115

```

## - NZB Indexing 적용 후

적용 층	결과
Layer 1	소요시간 32.600881814956665
Layer 1, 2	소요시간 27.096725463867188
Layer 1, 2, 3	소요시간 26.729569911956787

=> NZB Indexing 적용을 통해 convolution 수행 시간이 줄어들었다.

CTA	희소행렬 처리를 통한 트윗 감성 분석 프로그램
	작성 날짜: 2020년 11월 25일

## 8. 결과 및 향후 계획

결과	향후 계획
이모지 학습 데이터셋을 제작했다.	이모지 데이터셋의 모델 추가 학습을 통해 정확도를 향상시킨다.
NZB Indexing 을 원소 단위로 적용했다.	NZB Indexing 을 filter 단위로 적용한다.
NZB Indexing 을 Convolution layer 3 개 층에만 적용했다.	NZB Indexing 을 Convolution layer 4 개 층 전체에 적용한다.
NZB Indexing 기술을 CPU 연산부에 적용했다.	GPU 연산으로 수정을 진행해 성능을 더 향상시킨다.
분석 결과를 트위터로 공유 가능하다.	카카오톡, 페이스북으로 공유 범위를 늘린다.
분석을 요청하면 로딩 화면이 뜬다.	로딩 30 초 이상 지연되는 경우 오류 메시지를 띄운다.
자주 사용하는 5 개 단어를 제공한다.	단어의 감성 분석도 진행을 해 결과를 제공한다.
분석 결과를 수치로 제공한다.	몇 가지 유형의 타이틀을 통해 분석 결과를 제공한다.

## 9. 참고 문헌 및 참고 사이트

[1] 한치원, 기민관, 박기호. "Non-Zero Bitmap (NZN) 인덱싱: CNN 모델 등 희소행렬과 밀집행렬이 혼재된 모델을 위한 효율적인 행렬 표현 방식", 한국정보과학회 학술발표논문집, 2020.

[2] tensorflow. "Pruning in Keras example",

[https://www.tensorflow.org/model\\_optimization/guide/pruning/pruning\\_with\\_keras](https://www.tensorflow.org/model_optimization/guide/pruning/pruning_with_keras), 2020

[3] github. "keras convolutional.py",

<https://github.com/tensorflow/tensorflow/blob/master/tensorflow/python/keras/layers/convolutional.py>, 2020

[4] sentiment140, "For Academics, training data",

<http://help.sentiment140.com/for-students>, 2020

[5] Emoji sentiment ranking, "Emoji Sentiment Ranking v1.0",

[http://kt.ijs.si/data/Emoji\\_sentiment\\_ranking/](http://kt.ijs.si/data/Emoji_sentiment_ranking/), 2020