처음배우는 딥러닝 챗봇

3. 토크나이징

* 문장을 일정한 의미가 있는 가장 작은 단어들로 나누어야 하는데 여기서 가장 기본이 되는 단어들을 “토큰”이라고 한다.
* 주어진 문장에서 토큰 단위로 정보를 나누는 작업을 토크나이징이라고 한다.

KoNLPy: 한국어 자연어 처리를 위한 파이썬 라이브러리

사용자 사전 구축

* 새로운 단어들을 추가할 수 있게

대부분의 형태소 분석기들은 사용자 사전을 추가할 수 있도록 구성되어 있다.

Kkma, Komoran, Okt가 한국어 자연어 처리를 위한 라이브러리이다.

각각 장단점이 있음

Kkma: 분석 품질이 좋음, 지원하는 품사 태그가 가장 많음

Komoran: 자소가 분리된 문장이나 오탈자에 강함 사용자 사전 관리 용이

Okt: 매우 빠른 분석 속도, 정규화 기능 지원

4. 임베딩

* 컴퓨터는 수치 연산만 가능해서 자연어를 숫자, 벡터 형태로 변환 해야한다.
* 이 과정을 자연어 처리 분야에서는 “임베딩”이라고 한다.
* 즉 임베딩은 단어나 문장을 수치화해 벡터 공간으로 표현하는 과정을 의미함
* 임베딩에는 문장 임베딩 단어 임베딩이 있다
* 문장 임베딩은 문장 전체를 벡터로 표현하는 방법- 전체 문장의 흐름을 파악해 벡터로 변환하기 때문에 문맥적 의미를 지니는 장점이 있음
* 단어 임베딩은 개별 단어를 벡터로 표현하는 방법- 동음이의어에 대한 구분을 하지 않기 때문에 의미가 다르더라도 단어의 형태가 같다면 동일한 벡터값으로 표현되는 단점이 있음   
  하지만 문장 임베딩보다 학습 방법이 간단해 많이 사용됨

원-핫 인코딩: 단어를 숫자 벡터로 변환하는 가장 기본적인 방법

* 단 하나의 값만 1이고 나머지 요솟값은 0인 인코딩을 의미
* 원-핫 인코딩을 하려면 단어 집합이라 불리는 사전을 먼저 만들어야 한다.
* 여기서 사전은 말뭉치에서 나오는 서로 다른 모든 단어의 집합을 의미함
* 사전이 구축되면 단어 순서대로 고유한 인덱스 번호를 부여함

희소 표현과 분산 표현:

* 원-핫 인코딩은 표현하는 단어의 인덱스 요소만 1이고 나머지는 0으로 표현되는 희소 벡터이다.
* 이처럼 단어가 희소 벡터로 표현되는 방식을 희소 표현이라 부른다.
* 희소 표현은 사람이 이해하기에 직관적인 장점이 있지만 크기가 커질수록 메모리 낭비와 계산복잡도가 커지는 단점이 있음
* 또한 단어들 간의 연관성이 없어 의미를 담을 수 없다.

**자연어 처리를 잘하기 위해서는 기본 토큰이 되는 단어의 의미와 주변 단어 간의 관계가 단어 임베딩에 표현되어야 한다.**

* 희소 표현의 경우 그러지 못하는데 이를 해결하기 위해 각 단어 간의 유사성을 잘 표현하면서도 벡터 공간을 절약할 수 있는 방법이 “**분산 표현**”이다.
* 분산 표현은 한 단어의 정보가 여러 차원에 분산되어 표현된다.
* 즉 하나의 차원에 다양한 정보를 가지고 있음
* 예시로 RGB(204,255,204) 형태로 분산 표현을 하는 것이다.
* 분산 표현 방식은 데이터를 원하는 차원에 밀집시킬 수 있어 “밀집 표현”이라 하며 밀집 표현으로 만들어진 벡터를 “밀집 벡터”라 한다.

분산 표현 장점

* 임베딩 벡터의 차원을 데이터 손실을 최소화하면서 압축 가능
* 임베딩 벡터에는 단어의 의미, 주변 단어간의 관계 등 많은 정보가 내포되어 있어 일반화 능력이 뛰어남
* 예시로 남자, 남성은 원-핫 인코딩으로 하면 전혀 다른 의미를 가진 단어지만  
  분산 표현에서는 비슷한 위치에 분포되어 있어서 비슷한 의미로 해석 가능하다.

Word2Vec

**챗봇의 경우 많은 단어를 처리하면서 단어 간 유사도를 계산할 수 있어야 좋은 성능을 낼 수 있기 때문에 원-핫 이코딩은 좋은 선택이 아니다.**

* 그래서 분산 표현을 사용해야한다. 그 중 대표적인 방법이 word2Vec이다.
* CBOW, skip-gram 두가지 모델로 제안됨
* CBOW 모델은 맥락이라 표현되는 주변 단어들을 타깃 단어를 예측하는 신경망 모델
* Skip-gram 모델은 cbow 모델과 반대로 하나의 타깃 단어를 이용해 주변 단어들을 예측하는신경망 모델 cobw 모델에 비해 예측해야 하는 맥락이 많아 더 우수함

CBOW는 주변의 한 단어를 찾는데 그 단어 앞 뒤 단어를 확인함

앞뒤로 몇 개의 단어를 확인할지 결정하는 범위를 윈도우(window)라고함

Window=2라고 하면 앞뒤 2개씩 총 4개를 보고 한 단어를 예측함

Skip-gram은 한 단어를 이용해 앞,뒤 단어를 예측함

Window는 cbow와 마찬가지임 window=2하면 앞 뒤 2개씩 총 4개를 예측함

한국어 말뭉치 데이터를 원하면 github.com/e9t/nsmc 에서 다운하기

5. 텍스트 유사도

- 두 문장 간의 유사도를 계산하기 위해서는 문장 내에 존재하는 단어들을 수치화해야 한다.

- 그 방법에는 통계를 이용하는 방법과 인공 신경망을 이용하는 방법으로 나 뉜다.

- 상황에 따라 신경망(word2vec ) 보다 통게적 방식이 더 적절할 수 있다.

n-gram 유사도

* N-gram은 주어진 문장에서 n개의 연속적인 단어 시퀀스(단어 나열)를 의미함
* 문장에서 n개의 단어를 토큰으로 사용함
* 이는 이웃한 단어의 출현 횟수를 통계적으로 표현해 텍스트의 유사도를 계산하는 방법
* 논문 인용, 도용 정도를 조사 가능
* N=1 이면 문장에서 단어 한 개씩
* N=2이면 문장에서 단어 2개 한 묶음 n=3이면 3개가 한 묶음 이렇게됨

코사인 유사도

* 단어나 문장을 벡터로 표현할 수 있다면 벡터 간 거리나 각도를 이용해 유사성을 파악할 수 있다.
* 벡터 간 거리를 구하는 방법중 하나가 코사인 유사도이다.
* 두 벡터 간 코사인 각도를 이용해 유사도 측정
* 벡터의 크기가 중요하지 않을 때 그 거리를 측정하기 위해 사용함
* 예로 동일한 단어가 많이 포함되어 있을수록 벡터의 크기가 커지는데 코사인 유사도는 벡터의 크기와 상관없이 결과가 안정적임
* -1~1 사이의 값을 가지며 동일한 경우 1, 반대 방향인 경우는 -1 두 벡터가 서로 직각을 이루면 0의 값을 가짐 방향이 같을수록 유사함

3,4,5장 정리(전처리 과정)

1. 3장에서 다룬 토크나이징을 통해 문장에서 최소한의 의미를 가지는 단어들을 토큰화시킨다.
2. 4장에서 다룬 임베딩 처리를 해야한다. 즉 컴퓨터가 계산 가능한 벡터 형태로 수치화 시키는 과정을 거쳐야한다.
3. 그러고 문장간 유사도를 계산한다. Word2vec 내장 함수 사용 또는 5장에서 구한 유사도로 계산하면된다.

6. 챗봇 엔진에 필요한 딥러닝 모델

챗봇 문답 데이터 감정 분류 모데 구현은 colab에서 진행

챗봇 예제 사이트

Github.com/songys/Chatbot\_data

* CNN의 경우 이미지 외에 임베딩 품질만 괜찮다면 자연어 분류에도 좋은 성능을 낸다.

6.3 개체명 인식을 위한 양방향 LSTM 모델

RNN

- RNN에서 은닉층 노드는 이전 시점(t-1)의 상탯값을 저장하는 메모리 역할을 수행하기 때문에 셀 or 메모리 셀 이라 부름

- 은닉층의 메모리 셀의 출력 벡터는 출력층과 다음 시점(t+1)의 메모리 셀에 전달되는데 이를 은닉 상태라 함

- RNN은 현재 시점의 메모리 셀은 이전 시점의 은닉 상탯값에 영향을 받고 있으며 완전 연결 계층 구조를 가진다.

- many -to -one 여러 개의 input 하나의 output ex)스팸메일

- one – to – many 하나의 input 여러 개의 output ex)사진하나에 여러 개의 물체 인식해서 텍스트로

- many – to – many ex) 단어 시퀀스를 입력으로 받아 각 시퀀스가 의미하는 개체명을 출력하는 구조, 한국어를 입력받아 영어로 출력하는 번역기로도 사용가능 등등

- 은닉층h(t) = tanh(w(x)\*x(t) + w(h)\*h(t-1) + b(h)

- 출력층y(t) = w(t)\*h(t) + b(y)

LSTM(Long Short Term Memory)

* RNN은 시퀀스의 시점이(time step) 길어질수록 앞쪽 데이터가 뒤로 전달되지 않는 문제가 발생함
* RNN을 다층 구조로 쌓으면 입력과 출력 데이터 사이의 연관 관계가 줄어들어 장기 의존성(long term dependency) 문제가 생김
* 그래서 나온 것이 LSTM
* LSTM은 셀 상태가 추가되고 input gate, forget gate, output gate 3개로 구성됨
* Input gate(입력 게이트)는 현재 정보를 얼마나 기억할지 결정하는 게이트
* 현재 입력되는 정보를 엄마만큼 다음 시점의 셀 상태값으로 전달할지 결정
* Forget gate(삭제 게이트)는 이전 시점의 셀 상태값을 삭제하기 위해 사용됨
* 삭제게이트의 결과값으로 기억된 정보(이전 셀 상태)를 얼마나 삭제할지 결정함
* 입력,삭제 게이트의 결과값으로 현재 셀 상태를 계산함
* 즉 삭제게이트는 이전 시점의 입력을 얼마나 반영할지, 입력 게이트는 현재 시점의 입력을 얼마나 반영할지 결정
* 출력 게이트의 결과값은 현재 시점의 은닉 상태(h(t))를 결정하는데 사용되며 해당 값은 전달되는 메모리 셀이 많아질수록 정보 유실이 크기 때문에 단기 상태라고 부른다.
* 단기상태(은닉 상태)는 장기 상태(셀 상태값)에 영향을 받는 구조

양방향 LSTM

* 정방향 lstm은 기존과 동일하게 입력 문장을 왼, 오른쪽으로 처리하며 역방향 LSTM은 입력 문장의 단어 순서를 반대로 처리함
* 문장이 길어질수록 성능이 저하되는 것을 LSTM만으로는 해결하지 못하여 나온 것이 양방향 LSTM이다.

개체명 인식(NER)

* 문장 내에 포함된 어떤 단어가 인물, 장소, 날짜 등을 의미하는 단어인지 인식하는 것이 개체명 인식
* 개체명 인식은 챗봇에서 문장을 정확하게 해석하기 위해 반드시 해야하는 전처리 과정

Ex)

|  |
| --- |
| 입력 문장: 내일 부산 날씨 알려줘  문장 의도 분류: 날씨 요청  개체명 인식 결과  내일 – 날짜(개체명)  부산 – 지역(개체명) |

* 단순한 질문 형태라면 개체명사전을 구축해 해당 단어들과 매핑되는 개체명을 찾을 수 있다.
* 문장 구조가 복잡하거나 문맥에 따라 단어의 의미가 바뀐다면 딥러닝 모델을 활용해야함
* 개체명 인식 모델을 만들기 위해선 BIO표기법을 알아야함

|  |
| --- |
| BIO란: Beginning Inside Outside의 약자로 각 토큰마다 태그를 붙이기 위해 사용함  B는 개체명이 시작되는 단어에 B-개체명으로 태그되며  I는 B-개체명과 연결되는 단어일 때 I-개체명으로 태그됨  O는 개체명 이외의 모든 토큰에 태그됨  Ex)  오늘부터 샤닐 길동은 삼성전자에 근무합니다.  오늘 / B-data 샤닐 / B-person 삼성 / B-company 근무 / O  부터 / O 길동 / I – person 전자 / I-company 합니다. / O  은 / O 에 / O  ‘오늘’은 날짜의 시작 개체이기 때문에 B-data로 태그됨  ‘오늘’과 연결되는 단어가 없어서 B-data만 존재  샤닐 길동은 두개의 토큰이 합쳐져서 사람 객체를 나타냄  O는 관련이 없는 토큰들  Github.com/machinereading/KoreanNERCorpus  여기서 한국어 개체명 인식 말뭉치 제공  ;로 시작하는 문장 라인은 원본 문장  $로 시작하는 문장 라인은 해당 문장에서 NER 처리된 결과  그다음 라인 부터는 토큰 번호, 단어 토큰, 품사 태그, BIO 태그로 구성된 열이 존재 |

- 개체명 인식에 사용되는 성능 평가는 F1 스코어를 계산하는 방법을 사용해야함

- F1 스코어를 계산하기 위해서는 정밀도와 재현율을 사용해야함

    - 정확도(accuracy): 실제 정답과 얼마나 유사한지 나타냄

    - 정밀도(precision): 정밀도가 높다고 해서 정확하다는 의미는 아님 정밀도가 높으면 결과값이 일정하게 분포되어 있는 것

    - 재현율(recall): 실제 정답인 것들 중 예측 모델이 정답이라 예측한 것의 비율

- F1 스코어란 정밀도와 재현율의 조화 평균을 의미함

6장은 의도 분류와 개체명 인식에 필요한 딥러닝 모델을 배움

Mysql(데이터베이스 연동)

|  |
| --- |
| Host = ‘127.0.0.1’호스트 주소  User= root 데이터베이스 로그인 유저  Password= 데이터베이스 비번  Db= 데이터베이스명  Charset=’utf8’ 인코딩 |

7. 챗봇 학습툴 만들기

목표1. 학습 데이터를 DB에 저장했을 때 실시간으로 챗봇 시스템에 적용될 수 있도록 하기

우리의 챗봇 엔진은 크게 2가지 과정을 거쳐서 답변을 출력하게 된다.

첫 번째는 입력되는 문장을 자연어 처리하여 해당 문장의 의도, 개체명, 키워드 정보 등을 추출

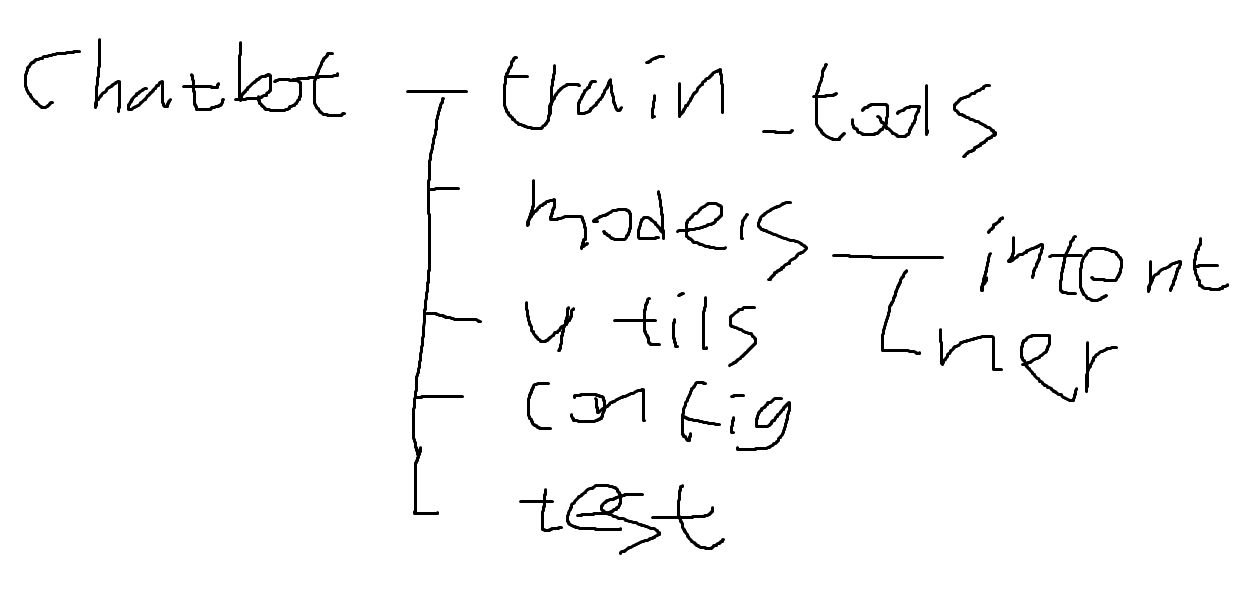
|  |
| --- |
| Ex)챗봇 엔진 입력 처리 과정  문장 입력: ‘내일 오전 10시에 탕수육 주문 가능할까요?’  -> 챗본엔진  -> 엔진 해석결과:  의도: 음식 주문  개체명: 내일:date / 오전 10시:time / 탕수육:food  키워드: 내일, 오전, 10시, 탕수육, 주문 |

두 번째는 엔진에서 해석한 결과를 이용해 학습 DB 내용을 검색. 이때 해석 결과(의도, 개체명)에 매칭되는 답변 정보가 DB에 존재하면 데이터를 불러와 사용자에게 답변으로 제공

|  |
| --- |
| Ex) 챗봇 엔진 답변 처리 과정  답변 출력:”주문해주셔서 감사합니다.  <-  챗봇 엔진 ->(답변검색{의도, 개체명}) <-(검색된 답변 데이터)학습 db |

7.3.1 프로젝트 구조

챗봇 프로젝트 디렉터리 구조는 다음과 같음



Train\_tools: 챗봇 학습툴 관련 파일

Models: 챗봇 엔진에서 사용하는 딥런이 모델 관련 파일

Inent: 의도 분류 모델 관련 파일

Ner: 개체 인식 모델 관련 파일

Utils: 챗봇 개발에 필요한 유틸리티 라이브러리

Config: 챗봇 개발에 필요한 설정

Test: 챗봇 개발에 필요한 테스트 코드

7.3.2 학습용 데이터베이스 설계 및 데이터 테이블 생성

챗봇 엔진 답변 처리 과정에서 필요한 DB 구조를 설계해보고 학습할 수 있는 툴 만들기

Column: id, intent, ner, query, answer, anser\_image

Db 서버 접속 정보를 /config 디렉터리에서 따로 관리함

DatabaseConfig.py 파일을 생성하여 관리

챗봇 데이터 학습용 테이블을 생성하는 코드는 /train\_tools/qna내에 생성

create\_train\_data\_table.py로 생성

파이썬 패키지 module 에러 뜨면 해결법

<https://wikidocs.net/1418>

하위폴더, 상위폴더 모듈 사용법…

<https://freedata.tistory.com/70>

7장에서는 config.DatabaseConfig.py(db 접속정보) / train\_tools.qna.create\_train\_data\_table.py(챗봇 학습 데이터 테이블 생성) / train\_tools.qna.load\_train\_data.py(엑셀 파일로 챗봇 학습 데이터 불러오기)

즉 챗봇 엔진이 적절한 답변을 출력할 수 있도록 필요한 학습 데이터베이스를 구축하고 학습 데이터 엑셀 파일로 관리할 수 있는 툴을 만듦

사용자가 요청한 질의와 연관된 답변을 찾을 수 있도록 학습 데이터를 DB에 저장하는 것을 의미

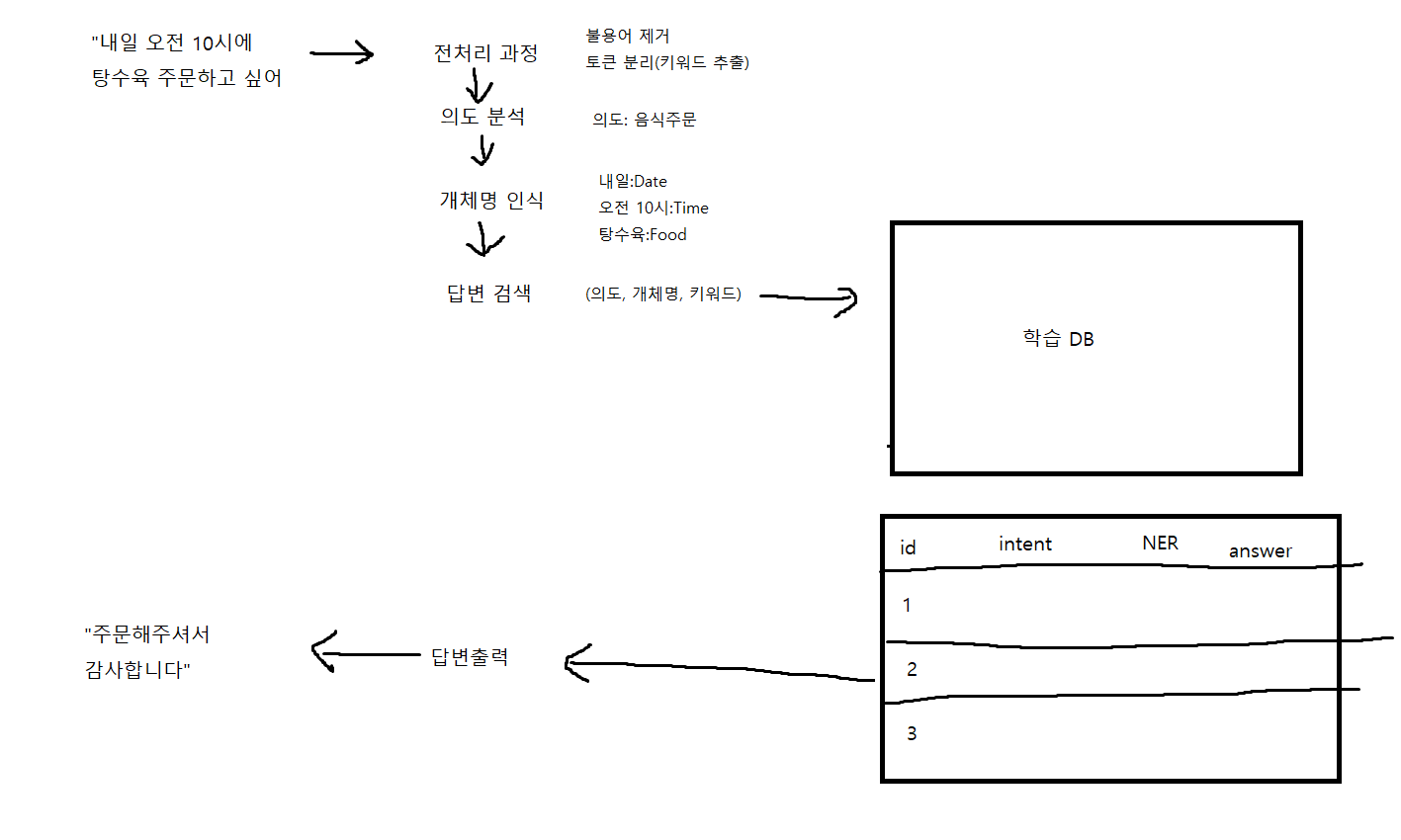
8. 챗봇 엔진 만들기

챗봇 엔진이란? 챗봇에서 핵심 기능을 하는 모듈이며, 화자의 질문을 이해하고 알맞은 답변을 출력하는 역할을 함. 한마디로 자연어 처리 모듈이다.

8.2챗봇 엔진 구조

|  |  |
| --- | --- |
| 핵심 기능 | 설명 |
| 질문 의도  분류 | 화자의 질문 의도를 파악합니다. 이는 해당 질문을 의도 분류 모델을 이용해 의도 클래스를 예측하는 문제이다. |
| 개체명 인식 | 화자의 질문에서 단어 토큰별 개체명을 인식합니다. 이는 단어 토큰에 맞는 개체명을 예측하는 문제 |
| 핵심 키워드  추출 | 화자의 질문 의미에서 핵심이 되 만한 단어 토큰을 추출합니다. 형태소 분석기를 이용해 핵심 키워드가 되는 명사나 동사를 추출 |
| 답변 검색 | 해당 질문의 의도, 개체명, 핵심 키워드 등을 기반으로 답변을 학습 DB에 검색 |
| 소켓 서버 | 다양한 종류(카톡, 네이버톡톡)의 챗봇 클라이언트에서 요청하는 질문을 처리하기 위해 소켓 서버 프로그램 역할을 함. 따라서 여기서의 챗봇 엔진 서버 프로그램이라고 부르겠음 |

우리가 목표하고 있는 챗봇 엔진 처리 과정



순서

1. 화자의 질의 문장이 입력되면 챗봇 엔진은 먼저 전처리를 진행함
2. 형태소 분석기를 이용해 단어 토큰(키워드)을 추출한 뒤 명사나 동사 등 필요한 품사만 남기고 불용어는 제거
3. 의도 분석과 개체명 인식을 완료한 후 결과값을 이용해 적절한 답변을 학습 DB에서 검색해 화자에게 답변을 출력

* 해당 챗봇 엔진에는 자연어 처리를 위해 2가지(의도 분석, 개체명 인식) 딥러닝 모델을 사용함
* 데이터셋을 많이 보유하고 있다면 우수한 챗봇 엔진 개발이 가능하지만 그렇지 않음
* 그래서 부족한 모델 성능을 보완하기 위해 룰 베이스와 딥러닝 모델을 같이 사용하는 방법으로 챗봇 엔진을 만듦
* 자주 반복되고 빈도가 높은 문제에 대해서는 작은 학습 데이터셋만 가지고도 딥러닝 모델보다 품질이 우수한 결과를 보여줌

8.3 전처리 과정

- 우리 챗봇 엔진의 전처리 과정은 형태소 분석기로 토크나이징 작업을 하고, 문장 해석에 의미 있는 정보만 남기고 나머지 불용어들은 제거하는 작업을 한다.

- 챗봇 개발에 필요한 전처리 담당 모듈은 유틸리티 라이브러리이기 때문에 /utils에 Preprocess.py로 생성하기

- 전처리 모듈의 경우 챗봇 엔진 내에서 자주 사용하기 때문에 클래스로 정의하기

- /test 디렉터리에 preprocess\_test.py 파일을 생성하고 preprocess 클래스 동작 테스트

8.4 단어 사전 구축 및 시퀀스 생성

- 여기 채봇 엔진에서 의도 분류 및 개체명 인식 모델의 학습을 하려면 단어 사전을 구축해야 한다.

- 단어 사전 구축에 필요한 말뭉치 데이터(corpus.txt)는 /train\_tools/dict/ 디렉터리에 있음

- 단어 사전을 생성하는 소스 파일명 create\_dict.py 생성

8.5 의도 분류 모델

- 챗봇 엔진에 화자의 질의가 입력되었을 때 전처리 과정을 거친 후 해당 문장의 의도를 분류한다.

- 이때 문장을 의도 클래스별로 분류하기 위해 6장에서 배운 CNN 모델을 사용한다.

- 인사, 욕설, 주문, 예약, 기타 5가지 의도로만 분류할 수 있도록 구현

- 모델학습 모듈을 만들기 전에 챗봇 엔진 소스 전역에서 사용할 파라미터 정보를 /config 디렉터리에 GlobalParams.py에서 관리

8.5.1 의도 분류 모델 학습

- 챗봇 엔진의 의도 분류 모듈을 만들기 전에 해당 모델의 설계 및 학습을 진행해야한다.

- /models/intent에 total\_train\_data.csv파일 학습 데이터셋 사용 의도 분류용 학습 데이터셋임

- 음식점 주문과 예약을 위한 챗봇에 특화되어 있으며, 클래스별 샘플 텍스트가 다양하지 않아 특정 의도 클래스의 경우 예측 품질이 떨어질 수 있음

- train\_model.py 생성

8.5.2 의도 분류 모듈 생성

- 이 모듈은 앞서 학습한 의도 분류 모델 파일을 활용해 입력되는 텍스트의 의도 클래스를 예측하는 기능을 가지고 있음

- 해당 모듈은 딥러닝이여서 models/intent에 IntentModel.py로 생성

8.6 개체명 인식 모델 학습

- 챗봇 엔진에 입력된 문장의 의도가 분류된 후 문장 내 개체명 인식을 진행함

- 양방향 LSTM 모델 사용

- 음식 / 날짜,시간 / 사람 / 조직,회사 / 지역 개체명 인식 가능

8.6.1 개체명 인식 모델 학습

- 양방향 LSTM 모델 사용

- /models/ner에 ner\_train.txt파일을 학습 데이터셋으로 이용

- ner\_train.txt를 이용해 ner 모델을 생성하고 학습하는 코드는 train\_model.py로 만들기

8.6.2 개체명 인식 모듈 생성

- 챗봇 엔진의 개체명 인식 모듈 만들기

- 이 모듈은 앞서 학습한 개체명 인식 모델 파일을 활용해 입력한 문장 내부의 개체명을 인식하는 기능을 가지고 있다.

- 해당 모듈은 딥러닝이여서 /models/ner에 NerModel.py로 생성

8.7 답변 검색

- 입력된 문장이 전처리, 의도 분류, 개체명 인식 과정을 거쳐 해석된 데이터를 기반으로 적절한 답변을 학습 DB로부터 검색하는 방법을 알아보자

- 챗봇 엔진이 자연어 처리를 통해 해석한 문장을 토대로 유사한 답변을 검색하는 일은 매우 중요하고 기술적으로 난이도 있는 분야

- 여기서는 SQL 구문을 이용해 룰 베이스 기반으로 답변을 검색하는 방법을 사용

8.7.1 데이터베이스 제어 모듈 생성

- 답변 검색 모듈을 만들기 전에 데이터베이스 제어를 쉽게 할 수 있는 모듈을 먼저 생성하기

- /utils Database.py로 생성

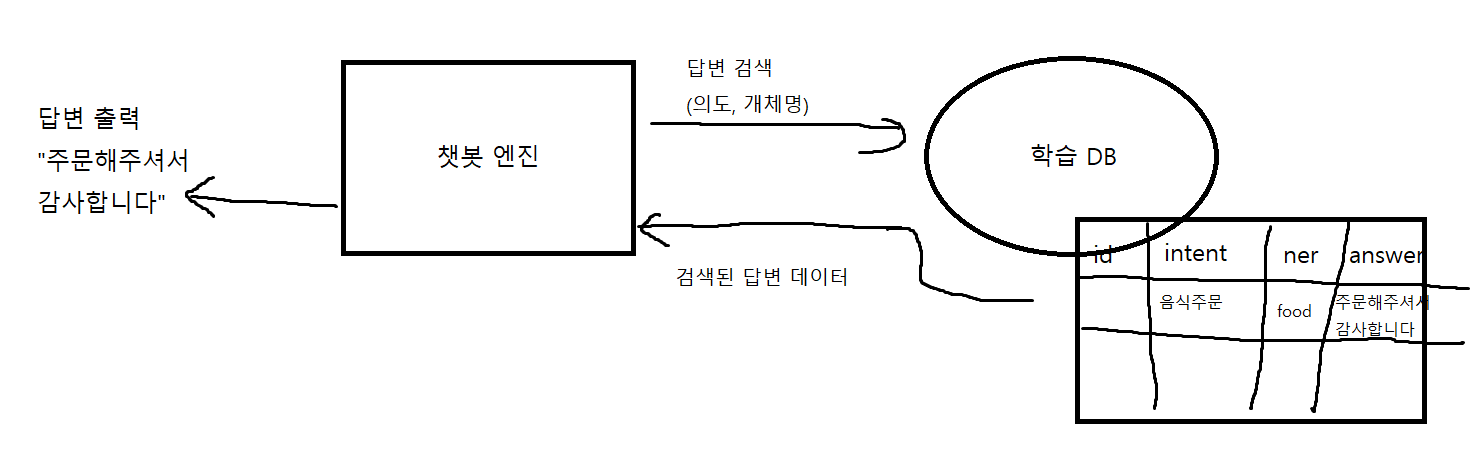
8.7.2 답변 검색 모듈 생성

- 챗봇 엔진은 입력되는 문장을 전처리, 의도 분류, 개체명 인식 과정을 거쳐서 나온 자연어 해석 결과를 이용해 학습 db에서 적절한 답변을 검색한다.

- 해석된 결과 항목에 따라 학습 db에서 어떤 방식으로 답변을 검색할지 결정하는 일은 개발자의 몫

- 여기서는 단순하게 **의도명과 개체명 2가지 항목만 가지고 답변을 검색해** client에게 제공하도록 한다.

챗봇 엔진 답변 검색 과정

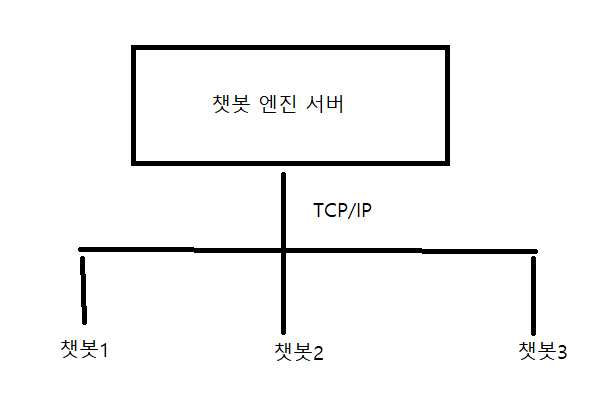


* 답변 검색 모듈 즉 답변 검색에 필요한 기능을 제공하는 클래스를 /utils FindAnswer.py로 생성
* Test에 chatbot\_test.py로 동작 확인

8.8 챗봇 엔진 서버 개발

- 챗봇 엔진 서버 프로그램은 다수의 챗봇 서비스가 접속해 client의 질의에 대한 응답을 제공할 수 있도록 구현함

챗봇 엔진 서버 프로그램 구성



8.8.1 통신 프로토콜 정의

- 프로토콜이란 서버와 클라이언트 간에 통신을 하기 위해 필요한 규약(약속) 이다.

- 최근에는 server 와 client간에 JSON문자열 형태로 통신을 많이 한다.

- json은 key/value 쌍으로 이루어진 데이터 객체를 전달하기 위해 인간이 읽을 수 있는 텍스트 형태를 사용하는 개방형 표준 포맷

- 양방향 통신이기 때문에 두 가지 형태의 프로토콜의 정의해야 한다.

|  |
| --- |
| Ex)  Client -> server  {  “Query”: “자장면 주문”,  “botType”: “kakao”  }  Server -> client  {  “Query”: “자장면 주문”,  “intent”:”주문”,  “ner”:”[(‘자장면’, ‘b\_food’), (‘주문’,’o’)]”,  “Answer”: “자장면 주문 처리 감사”,  “answerimageurl”:””  }  즉  챗봇 클라이언트 -> 요청 서버로  챗봇 엔진 서버 -> 응답 챗봇 클라이언트로 |

8.8.2 다중 접속을 위한 TCP 소켓 서버

- 스레드란 프로그램 내에서 실행되는 단위 즉 어떤 하나의 함수를 호출하면 하나의 스레드가 생겨서 실행됨

- 챗봇 엔진 서버는 챗봇 클라이언트가 연결 요청을 할 때마다 챗봇 엔진 처리 스레드를 생성해 일련의 과정을 거쳐 요청한 질의의 답변을 클라이언트 쪽으로 전송함

- TCP 소켓 서버를 관리하는 모듈 즉 서버에 접속하는 클라이언트 소켓을 생성하고 처리하는 모듈을 /utils BotServer.py로 생성

- main에 bot.py파일을 생성해 실행해보자

8.8.3 챗봇 테스트 클라이언트 프로그램

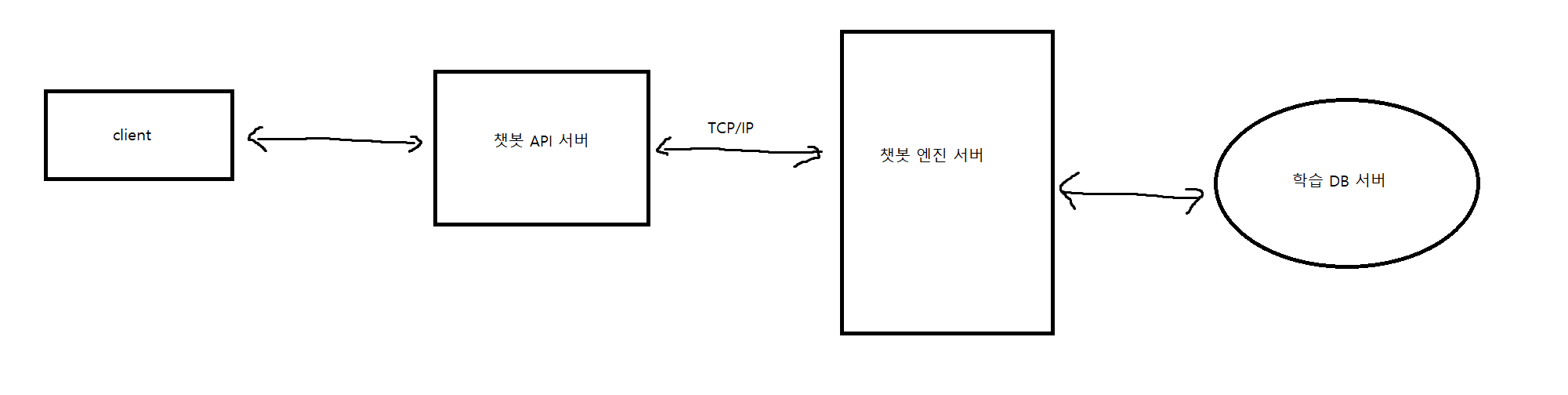
- 챗봇 테스트 클라이언트 프로그램은 콘솔 화면에 질문을 입력하면 챗봇 엔진 서버와 통신해 결과를 출력함 /test에 chatbot\_client\_test.py로 생성

8.9 마지막

- 우리의 챗봇 엔진은 개체명 인식 모델을 학습할 때 음식 이름이 2개 이상 포함된 문장 유형이 없었기 때문에 ‘자장면’과 ‘탕수육’을 동시에 인식할 수 없다.

- 또한 주문 수량 역시 개체명을 인식할 수 있는 데이터가 없기에 정확한 수량 정보를 파악할 수 없다.

9. 챗봇 API 만들기



* 챗봇 API 서버는 다양한 메신저 플랫폼과 챗봇 엔진 사이에서 서로 데이터를 주고받을 수 있도록 인터페이스 역할을 한다.
* 챗봇 기능을 지원하는 메신저 플랫폼과 통신하기 위해서는 REST API방식으로 챗봇 서버를 구현해야 한다.
  + REST API란? 웹(HTTP) 메서드 (get, post, put, delete)를 통해 해당 자원에 대한 CRUD동작을 처리하는 API 호출 방식이다.
  + http 인프라를 그대로 사용하므로 구축하는 데 시간과 노력이 적게 들어감
  + 또 http 표준 프로토콜을 최대한 활용하므로 웹을 지원하는 모든 플롯폼에서 사용 가능한 장점이 있다.

9.2 파이썬 Flask

* Rest api는 웹(http)상에서 호출할 수 있도록 웹 애플리케이션 형태로 만들어야한다.
* 파이썬 웹 애플리케이션 프레임워크로 Flask를 사용함
* Flask는 파이썬 기반 경량화된 프레임워크로 구현이 간단하고 웹 서비스 구현에 있어 자유도가 높아 rest api 서버 개발에 많이 사용됨

9.2.2 URI 동적 변수

- Rest api는 http 메서드(get, post 등)에 따라 RUI를 호출한다.

- 이때 필요한 기능에 따라 URI에 동적으로 변수가 들어갈 수도 있다.

-

9.2.3 기본적인 REST API 서비스 구현

- rest api는 기능에 따라 get, post, delete, put http 메서드를 사용한다.

- client로부터 요청이 들어왔을 때 http 메서드별로 뷰 함수를 정의하는 방법

REST API에서 CRUD 동작이 어떤 http 메서드와 매핑되어 있는지 설명하는 표

|  |  |  |
| --- | --- | --- |
| HTTP 메서드 | CRUD동작 | 설명 |
| POST | Create | 서버 리소스를 **생성**할 때 사용 |
| GET | Read | 서버 리소스를 **읽어**올 때 사용 |
| PUT | Update | 서버 리소스를 **수정**할 때 사용 |
| DELETE | Delete | 서버 리소스를 **삭제**할 때 사용 |

9.3 챗봇 api 서버 구현

- rest api호출 시 챗봇 엔진 서버에 소켓 통신으로 접속해 질의에 대한 답변을 받아오는 api 서버를 만들기

- /chatbot\_api에 app.py 생성