2020 졸업프로젝트 :

축소된 지식기반 데이터 베이스를 이용한 가벼운 한국어 KBQA 프레임워크 제안

201410349 서원준

201411310 장승원

201610099 이수민

졸업 프로젝트

001 개요 및 목표

한국어 KBQA Simple Question

일반 사용자도 쉽게 접근할 수 있도록 자연어 문장을 SPARQL 쿼리문으로 변환해 질의하는 방식을 사용한다.
요구 컴퓨팅 파워를 최소화한 KBDB 기반 Simple QA 프레임워크를 제안한다.
한국어로 지식 기반 데이터 관련 논문이 존재하지만 Question Answer이라는 주제에 대해서는 적으며 한국어 KBQA에 관련된 데이터셋도 존재하지 않아 이를 연구 주제로 잡게 되었다.

연구 목표

1. Subject-Predicate-Object 형태의 자료를 담는 한국어 트리플 데이터베이스를 구축
2. 요구 컴퓨팅 파워를 최소화한 KBDB 기반 Simple QA 프레임워크 제안
3. Simple Question 형태의 자연어 한국어 문장을 SPARQL 쿼리문으로 변환해 데이터에 접근
4. 입력 받은 자연어 질문에 대하 정답을 80% 이상 반환

002 요구사항 분석

기능 요구사항

1. 한국어 트리플 데이터베이스 구축

1.1 한국어 DBPedia 데이터 수집

1.1.1 시맨틱 트리플 데이터 형태의 한국어 데이터 베이스 덤프를 수집

1.2 충분한 수의 데이터 확보

1.2.1 데이터 수집 후 가공 결과 집계된 트리플 데이터 엔트리의 수가 30만개 이상

1.2.2 데이터 수집 후 가공 결과 집계된 Subject와 Object의 종류가 각각 10만개 이상

1.2.3 데이터 수집 후 가공 결과 집계된 Predicate의 종류가 200개 이상

2. 한국어 Simple Question의 SPARQL 변환

2.1 자연어 입력 문장의 SPARQL 변환을 위한 AI 모델 구축

2.1.1 입력 문장에서 Subject를 추출하기 위한 BiLstm 모델을 구축 및 학습

2.1.2 입력 문장에서 Relation를 추출하기 위한 CNN 모델을 구축 및 학습

2.2 구축된 AI 모델을 통한 자연어 문장의 Subject와 Relation 추출

2.2.1 입력 학습시킨 BiLstm 도델과 CNN 모델을 사용해, 입력받은 한국어 Simple Question으로 부터 Subject, Relation를 올바르게 추출

2.3 추출된 Subject와 Relation을 조합하여 SPARQL 쿼리문 작성 및 QA

2.3.1 Subject와 Relation을 을 기반으로 Object를 찾는 SPARQL 쿼리 스트링이 완성

2.3.2 작성된 SPARQL을 사전 구축한 DB에 질의해 올바른 Object 반환

002 요구사항 분석

비기능 요구사항

3. Simple Question 정확도 80% 달성

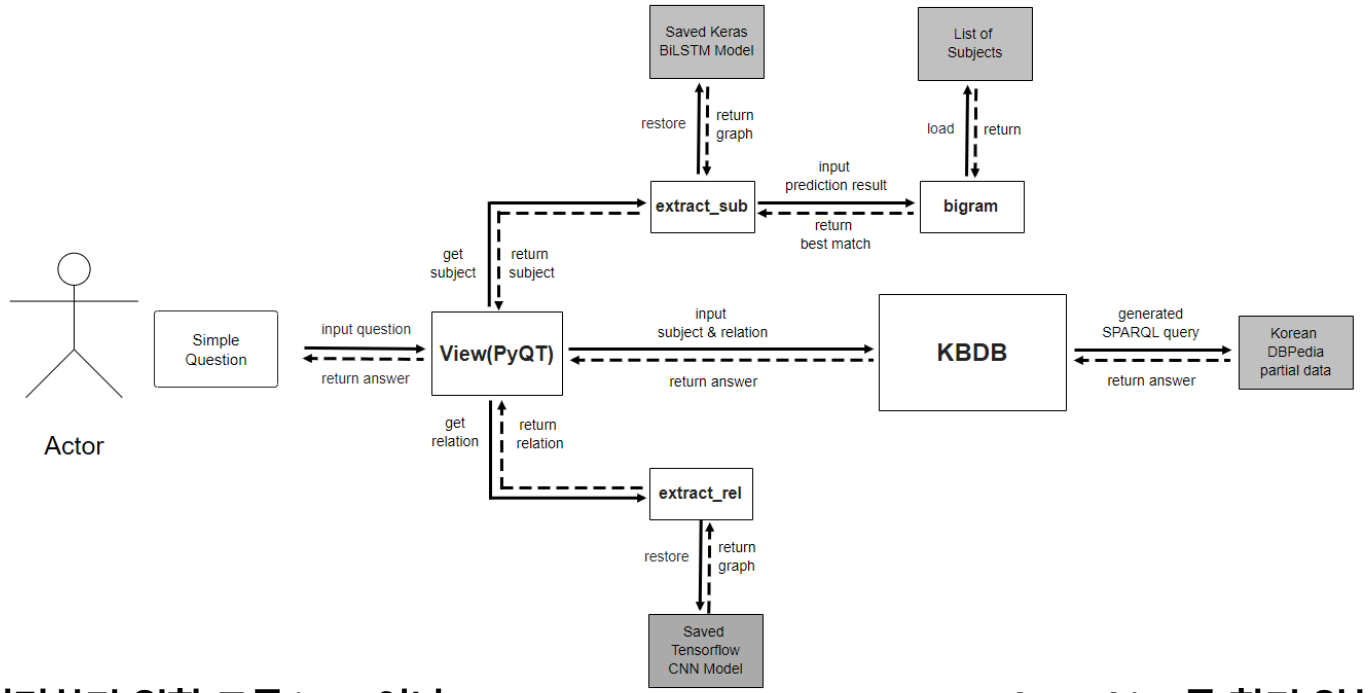
3.1 NL2SPARQL과 SPARQL 질의 결과 정답률 향상

3.1.1 저희가 연구 진행 초반에 success criteria를 세우는 과정에서 다른 논문들을 참조했는데, 그 중 하나가 전북대학교 논문'이었다.

3.1.2 NL2SPARQL과 SPARQL 질의 결과의 정확도를 각각 80%이상 끌어올린다.

3.2 달성하지 못할 경우 최적의 방법 탐색

003 상위 디자인
아키택쳐 설계



View - UI적 요소를 처리하기 위한 모듈(input이나 exposure)

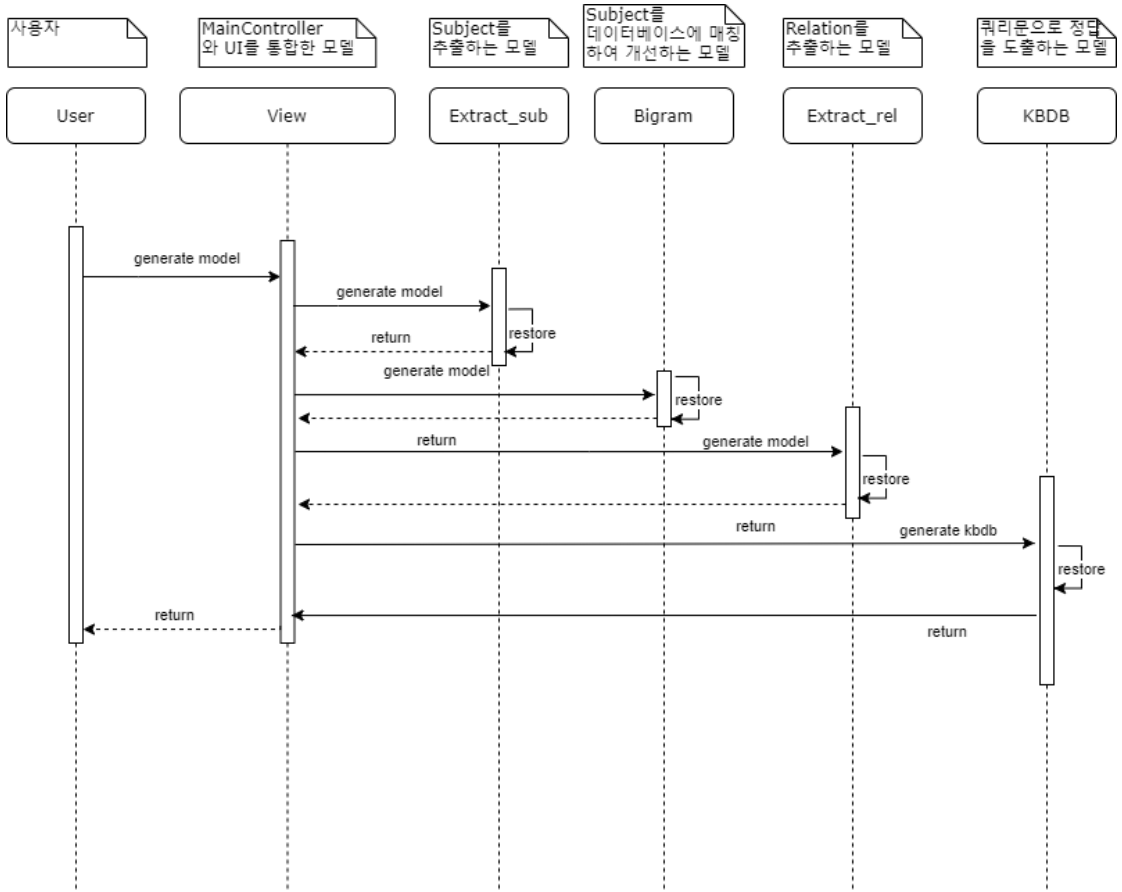
KBDB– Subject와 Relation을 통해 SPARQL 쿼리 작성 및 Object 추출

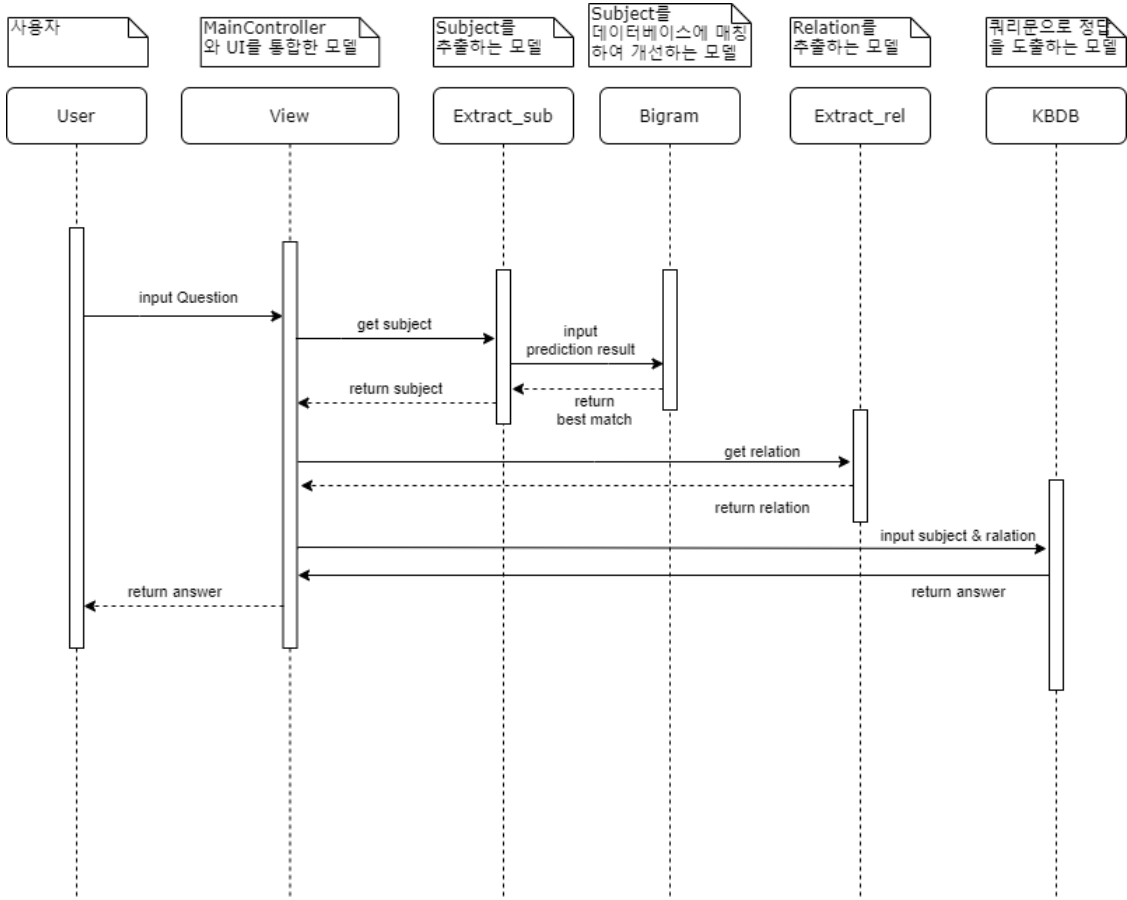
Extract_sub - Subject를 찾기 위하여 생성된 AI 모듈

Extract_rel - Predicated를 찾기 위하여 생성된 AI 모듈

DB_Connecter – SQL과 DB를 연결하기 위한 모듈

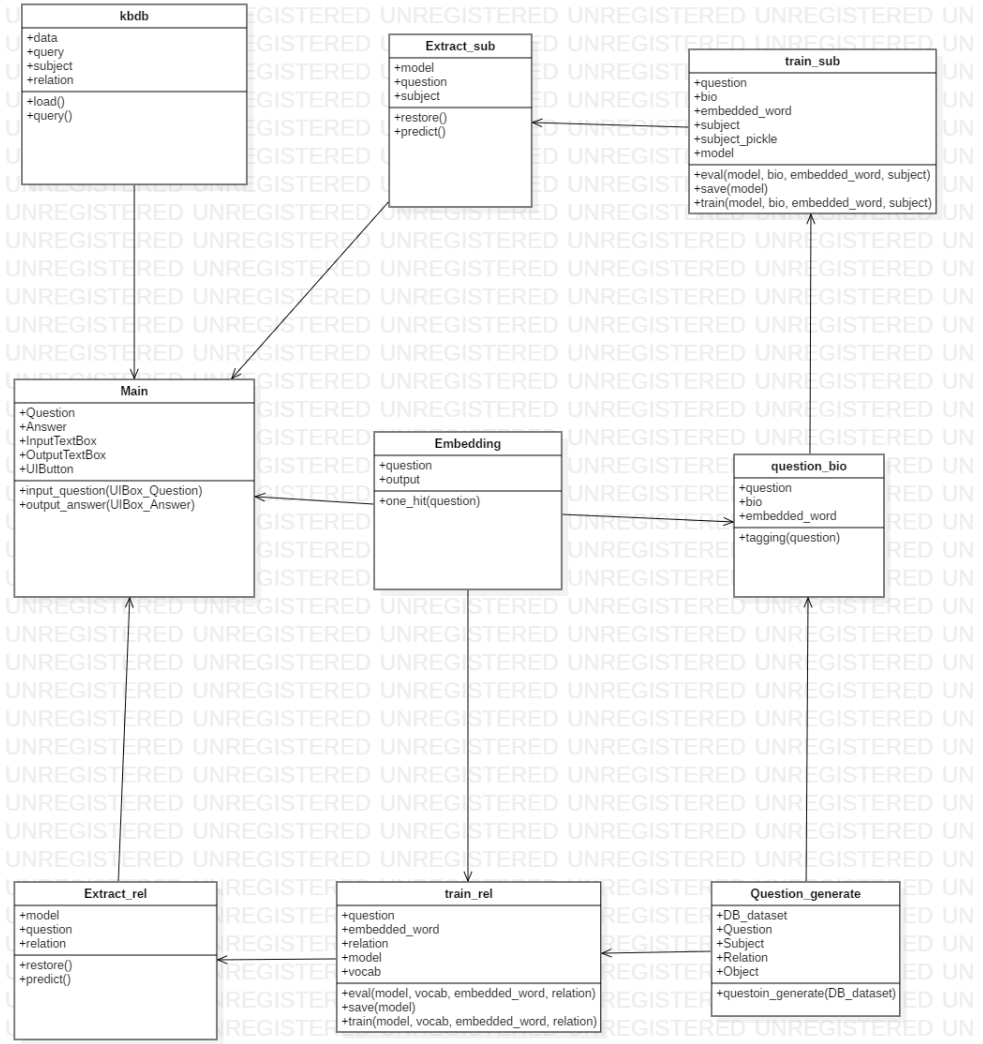
003 상위 디자인
시퀀스 다이어그램





003 상위 디자인

클래스 다이어그램 설계



004 초기 모델

1. 한국어 DBPedia 질문 데이터 생성

1.1 DBPedia 한국어 버전으로부터 Triple(Subject, Relation, Object) 데이터 확보

1.1.1 2016년도 DBPedia dump 의 일부를 사용

1.1.2 triple(Subject, Relation, Object) 구조 데이터를 pandas의 dataFrame 구조로 추출

예시) Subject - <http://ko.dbpedia.org/resource/ 야나미_조지>

Relation - <http://dbpedia.org/ontology/birthPlace>

Object - <http://ko.dbpedia.org/resource/도쿄_시>

1.2 Triple Data로부터 Simple Question Dataset 생성

1.2.1 실제 사람이 질문하는 어투와 유사하도록 조사와 어미 등을 고려하여 질문을 작성

1.2.2 데이터셋 중 괄호나 특수기호가 포함된 데이터는 그 특수 기호를 삭제

1.2.3 현재 약 100개의 Relation을 사용하여 Relation마다 5-10개의 Question Data를 구축하였음

예시) birthPlace - ~'의 출생지는 어디인가?', ~이/가 ' 태어난 곳이 어디에요?',

~이/가 ' 태어난 곳이 어디입니까?'

1.2.4 정확도 개선을 위하여 더 많은 Relation과 그에 따른 Question Dataset이 필요할 것임

004 초기 모델

2. 한국어 Question에서 Subject를 추출

2.1 Question BIO 처리

2.1.1 Question에서 Subject에 해당하는 부분을 한 글자(음절) B, I로 처리하며 나머지 부분을 O로 처리
기존 NLP의 BIO tag 방식은 space 단위로 처리하지만 이지만
한국어는 조사 등 복잡성이 있어 한 글자(음절단위)씩 처리

2.1.2 출력할 경우, 질문 문장이 BIO로 tag된 문장이 출력
예시) Input Qusetion– 김대중이 태어난 곳은 어디인가?
Input Subject – 김대중
output = [김, B], [대, I], [중, I] , [이, O] ……[가,. O]. [? ,O]

2.2 Bilstm으로 글자 tag를 뽑는 AI 구축

2.2.1 keras sequential model 내의 Bilstm을 이용

2.2.2 각 글자의 토큰화 0(Pad)을 제외하고 train data에서 많이 사용되는 글자 순으로 토큰화

2.2.3 출력 되는 데이터는 글자 당 BIO로 tag된 데이터

2.2.4 BIO 태깅에서 B,I만 선택하여 predicted 단어 생성

004 초기 모델

3. 한국어 Question에서 relation을 추출

3.1 DBPedia relation 정보 토큰화

3.1.1 DBPedia의 triple 구조에서 relation을 전부 모아 class 파일로 변경

3.1.2 relation은 entity와 관계없이 양이 매우 적으므로 많이 나오는 relation값부터 토큰화

3.2 CNN을 이용하여 입력된 문장 relation으로 추출

3.2.1 text CNN 모델을 통하여 단어 단위로 분리하여 Embedding

3.2.2 text CNN 모델의 CNN 모델을 이용하여 relation을 출력

004 초기 모델

4. 정확도 분석

4.1 Subject 추출

- 4.1.1 전체 데이터를 8 : 2로 나누어 각각 train, test 데이터로 사용
- 4.1.2 단순 BIO 태깅 결과 정확도 91%
- 4.1.3 But! Subject의 정확도는 35.8% (e.g. [조지_워] != [조지_워싱턴])

4.2 Relation 추출

- 4.2.1 Data의 15%를 무작위로 추출하여 test data로 사용
- 4.2.2 relation 추출의 정확도는? 76.6%

4.3 종합 평가

- 4.3.1 subject의 space 처리는 모두 _로 처리
- 4.3.2 subject와 relation의 정확도는 26.3%

005 개선 모델

1. Question Dataset 개수 변경

1. 1 Relation class 증가

1. 1. 1. Relation의 개수가 기존에는 약 100개였으나 2배정도인 약 200개(238개)를 추가

1. 2 Qusetion 개수 증가

1. 2. 1. 기존의 Relation 개수에서 추가된 만큼 Question의 개수도 추가

1. 2. 2. 새로 추가된 Relation은 Dataset에서 적은 데이터, 따라서 기존 Relation의 강화도 추가적으로 진행

1. 2. 3. 기존 Question의 개수는 800개였으나 약 1000개 가량 추가하여 총 데이터가 약 2000개로 증가

005 개선 모델

2. Predict Subject model 변경

2. 1 BIO tagging 변경

2. 1. 1. B가 여러 개 나오는 것을 방지시키기 위하여 문장일 집어 넣기 전 /S 라는 추가적인 tag를 집어 넣음(바꾼 후 중복되는 대부분의 B가 사라졌음을 확인)

2. 1. 2. BIO tag된 list 데이터를 거꾸로 집어넣어 학습

2. 1. 3. BIO tagging의 정확도는 96%로 5% 증가함

2. 1. 4. Predicted Subject의 정확도는 70.2%로 2배 가량 증가함

2. 2 Bigram 방식 이용

2. 2. 1. Dataset의 Subject의 모음을 저장

2. 2. 2. Predicted Subject의 문장을 2글자 씩 나눠 pair로 저장

2. 2. 3. pair가 Dataset 내부의 Subject에 포함이 되어 있다면 해당 Subject에 점수를 부여

2. 2. 4. 점수가 가장 높은 Subject를 Predicted Subject로 변경

2. 2. 5. 기존 질문들에 대해서는 Sparql 데이터 형식으로 처리 되지 않았으나
이를 이용하여 Sparql 데이터 형식으로 변경이 가능했다.(85.6% 정확도)

005 개선 모델

3. Predict Relation model 변경

3. 1 Train 및 test data 제작 변경

3. 1. 1. Word Embedding을 진행할 때, 어절 단위로 처리가 되어 있어 다른 방식을 채택

3. 1. 2. 형태소 분석으로 하였으나 고유명사에 대하여 형태소 분석이 제대로 이루어지지 않음

3. 1. 3. 음절 단위로 Word Embedding을 하여 고유명사도 더 정확하게 파악하도록 채택

3. 2 Predict Relation 변경

3. 2. 1. 정확한 학습을 하기 위하여 batch size, filtering data 변경(정확도 결과 변경 값이 미미)

3. 2. 2. 새로 학습되는 모델에 더 가중치를 두기 위하여 probability의 유지도 변경(일정 수치 제외하고 0으로 변경)

3. 2. 3. Predicted Relation의 정확도가 85%이상으로 증가하였다.

3. 2. 4. 특정 학습 시점에서는 90%이상의 정확도를 확인 할 수 있음

3. 2. 5. 최종 테스트 결과 92.9%의 정확도를 얻음

005 개선 모델

4. 종합 평가

4. 1 Predicted Subject와 Predicted Relation을 함께 평가

4. 1. 1. 한 질문에서 subject 와 relation이 출력될 경우를 평가

4. 1. 2. 정확도는 84.9%가 나옴

4. 2 UI적 요소 추가하여 질문을 집어넣을 경우 결과가 나오도록 출력

4. 2. 1. UI로 질문을 입력란과 질문을 받았을 경우 답변하는 출력을 보여줌

4. 2. 2. Train 데이터를 토대로 만들어진 모델 추가

4. 2. 3. Triple 구조를 통해 질문하고자 하는 Subject와 Relation을 받게 되면 Object의 출력을 확인

4. 2. 4. 실제 Object와 비교하여 정확도 측정을 하고자 했으나 Subject, Relation에 해당하는 정답이 여러 개 인 경우가 존재. 따라서 dataset의 object가 predict object 내에 있으면 정확하다고 판단. 위와 같은 84.9% 결과가 나온다.

005 개선 모송강호는 어디서 태어났는가?델

4. 종합 평가

4.3 Predicted Subject 성능 개선 표

모델	Bilstm + BIO	기존 모델 + BIO trans model	기존 모델 + BIO trans model + Bigram model
정확도	35.8	70.2	85.6%

4.4 Predicted Relation 성능 개선 표

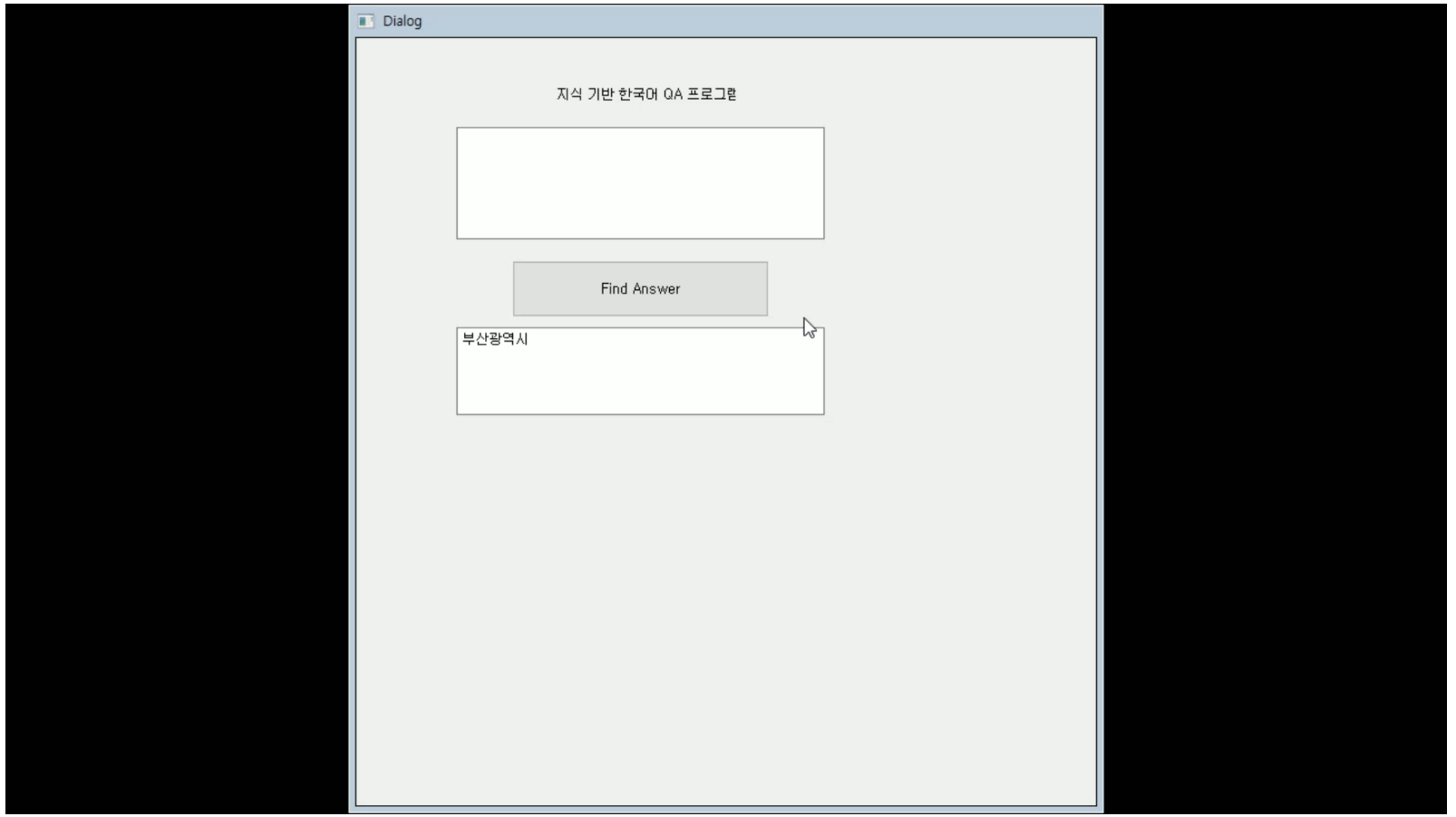
모델	기존 모델	기존 모델 + 어절 단위의 embedding	변경 모델	변경 모델 + 어절 단위의 embedding
정확도	76.8	92%	87.5	92.9%

006 System Test Cases

Case No.	테스트케이스 목표	입력 상황	예상 결과	실행 결과
TEST 1.1.1 (기능)	한국어로 된 시멘틱 트리플 데이터 데이터베이스를 구축해야 한다.	https://wiki.dbpedia.org/downloads-2016-10 로부터 다운받은 데이터를 기반으로 한국어 데이터베이스를 구축함	야나미_조지, birthPlace, 도쿄_시	성공 야나미_조지, birthPlace,도쿄_시
Test 1.2.1 (기능)	집계된 트리플 데이터 엔트리의 수가 30 만개 이상이어야 한다.	data_file = pd.read_csv('data0607/concat_mappingbased_objects_ko.csv') print(data_file.count())	300000 300000 300000	성공 523319 523319 523319
Test 1.2.2 (기능)	집계된 Subject 와 Object 종류가 각각 10 만개 이상이어야 한다.	data = pd.read_csv('data/concat_mappingbased_objects_ko.csv') data_subject = data.Subject data_object = data.Object subject_list = data_subject.unique().tolist() object_list = data_object.unique().tolist() print(len(subject_list)) print(len(object_list))	100000 100000	성공 113695 114485
Test 1.2.3 (기능)	집계된 Predicate 종류가 200 개 이상이어야 한다.	data = pd.read_csv('data/concat_mappingbased_objects_ko.csv') data_relation = data.Relation relation_list = data_relation.unique().tolist() print(len(relation_list))	200	성공 244

Case No.	테스트케이스 목표	입력 상황	예상 결과	실행 결과
TEST 2.1.1 (기능)	입력 쿼리에서 Subject 를 정확히 추출하는 Bi-LSTM 기반 모델이 구축되어야 한다.	야나미 조지가 태어난 곳은 어디인가?	"야나미 조지"라는 데이터가 도출됨	성공 출력값 : "야나미_조지" 85.6%의 정확도
Test 2.1.2 (기능)	입력 쿼리에서 Relation 을 정확히 추출하는 CNN 기반 모델이 구축되어야 한다.	야나미 조지가 태어난 곳은 어디인가?	"birthPlace"이라는 데이터가 도출됨	성공 출력값 : "birthPlace" 92.9%의 정확도
TEST 2.2.1 (기능)	학습시킨 모델이 입력된 Simple Question 으로부터 정확히 Subject 와 Relation 을 추출해야 한다.	사용자 입력: 야나미 조지가 태어난 곳은 어디인가?	Subject – 야나미 조지 Relation – birthPlace	성공 Subject – 야나미 조지 Relation – birthPlace
TEST 2.3.1 (기능)	Subject 와 Relation 을 기반으로 Object 를 찾는 SPARQL 쿼리 스트링이 완성되어야 한다.	Subject – 야나미 조지 Predicated - 태어난 곳	Select birthplace Where 야나미 조지	성공 – SELECT ?b WHERE { <http://ko.dbpedia.org/resource/야나미_조지><http://dbpedia.org/ontology/birthPlace> ?b .}
TEST 2.3.2 (기능)	사전 구축한 데이터베이스를 참조하여 올바른 Object 값이 반환되어야 한다.	SELECT ?b WHERE { <http://ko.dbpedia.org/resource/야나미_조지><http://dbpedia.org/ontology/birthPlace> ?b .}	도쿄시	성공 출력값 : "도쿄시" 84.9%의 정확도
Test 3.1 (비기능)	NL to SPARQL 과 SPARQL 질의 응답 정답률을 각각 80% 이상 달성한다.	Simple Question 의 Test Dataset 을 모델에 입력	테스트 결과 각각 80% 이상의 정확도 달성	성공 NL to SPARQL 정확도: Subject- 85.6% Relation- 92.9% SPARQL 질의응답 정확도: 84.9%

007 데모 영상



008 추후 연구 과제

1. 전체적인 모델의 한계

1. 1 학습 데이터 부족

1. 1. 1. Train data가 한국어 질문 데이터가 없기 때문에 학습 시키기 위한 데이터를 손수 제작해야 함.

1. 1. 2. Dataset에 존재하지 않는 내용을 찾을 경우에는 찾을 수가 없음

1. 1. 3. Train data를 제작(또는 툴을 제작)하여 추가적으로 KBQA에 알맞은 데이터셋 구축 필요.

1. 2 이음동의어 처리

1. 2. 1. word Embedding을 할 때, 존재하지 않는 단어 처리(DB에 있으나 word2vector에 없는 단어들 다수 존재)를 위하여 word2Vector를 하지 않고 다른 Embedding 방법을 했기 때문에 이음동의어 처리가 불가능

1. 2. 2. 예시로 한국은 대한민국으로 처리가 불가

1. 2. 3. FastText라는 word2Vector라는 모델을 적용 필요

008 추후 연구 과제

2. 추가적인 연구 방향

2. 1 Simple Question에서 복잡한 질문의 QA 처리

2. 1. 1. Simple Question은 triple 구조 중 Subject와 Relation만 있을 경우만 처리.

2. 1. 2. 복잡한 질문이란 Subject 또는 Relation이 Triple 데이터베이스에 존재하지 않으며 어떠한 연관성에 따라 새로이 triple 구조를 생성해 낼 수 있는 질문

2. 1. 3. 예를 들어 서울 시장의 나이는 몇 살인가? 라는 질문에 대하여 서울 시장이라는 인물의 데이터를 찾고, 그 인물의 나이를 찾는 방식

2. 2 정확도 및 컴퓨팅 파워 개선

2. 2. 1. 정확도 개선을 위하여 사용한 방법이 기존 모델을 유지하면서 수치 또는 모델을 약간씩 변경하며 정확도를 높이는 방법 선택. 따라서 다른 모델을 이용하여 정확도 개선이 가능