2020 졸업프로젝트 :

DBPedia 트리플 데이터를 이용한 한국어 자연어 질의응답 시스템

- 001 구현 상황
- 002 한계
- 003 시스템 테스트 케이스
- 004 Demo 영상

201410349 서원준

201411310 장승원

201610099 이수민

001 구현 상황

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 데이터셋 중 괄호나 특수기호가 포함된 데이터는 제외

1.1.3 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 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?'

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

Copyrightⓒ. Saebyeol Yu. All Rights Reserved.

001 구현 상황

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

Copyrightⓒ. Saebyeol Yu. All Rights Reserved.

001 구현 상황

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 relation 값과 관계없이 dataset에 나오는 순서대로 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

Copyrightⓒ. Saebyeol Yu. All Rights Reserved.

001 구현 상황

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 35.8% (e.g. [조지_워] != [조지_워싱턴])

4.1.3 Bigram Feature 적용 시 정확도 91%

4.2 Relation 추출

4.2.1 Data의 15%를 무작위로 추출하여 test data로 사용

4.2.2 relation 추출의 정확도는? 76.6%

Copyrightⓒ. Saebyeol Yu. All Rights Reserved.

002 개선 사항

4. Predict Subject model 변경

1. 1 BIO tagging 변경

1. 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)

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

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

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

1. 2 Bigram 방식 이용

1. 2. 1. Dataset의 Subject의 모음을 저장

1. 2. 2. Predicted Subject의 문장을 2글자 씩 나눠 pair로 저장

1. 2. 3. pair가 Dataset 내부의 Subject에 포함이 되어 있다면 해당 Subject에 점수를 부여

1. 2. 4. 점수가 가장 높은 Subject를 Predicted Subject로 변경

1. 2. 5. 점수가 가장 높은 Subject를 Predicted Subject로 변경

1. 2. 6. 기존 질문들에 대해서는 Sparql 데이터 형식으로 처리 되지 않았으나 이를 이용하여 Sparql 데이터 형식으로 변경이 가능했다.(95% 정확도)

Copyrightⓒ. Saebyeol Yu. All Rights Reserved.

002 개선 사항

5. Predict Relation model 변경

2. 1 Train 및 test data 제작 변경

2. 1. 1. Train data에 train 되지 않는 relation이 발견되어 강제적으로 relation 1개당 question data들이 train data에 적어도 1개 포함될 수 있도록 변경

2. 1. 2. Train data와 test data의 학습 데이터가 relation 순서대로 적용이 되어 있어 shuffle을 하여 학습

2. 2 Predict Relation 변경

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

1. 2. 2. 새로 학습되는 모델에 더 가중치를 두기 위하여 probability의 유지도 변경(일정 수치 제외하고 0으로 변경)

1. 2. 3. Predicted Relation의 정확도가 85%이상으로 증가하였다.

1. 2. 4. 특정 학습 시점에서는 90%이상의 정확도를 확인 할 수 있음

Copyrightⓒ. Saebyeol Yu. All Rights Reserved.

002 개선 사항

6. 종합 평가

3. 1 Predicted Subject와 Predicted Relation을 함께 평가

3. 1. 1. 한 질문에서 subject 와 relation이 출력될 경우를 평가

3. 1. 2. 정확도는 ?%가 나옴

3. 2 UI적 요소 추가하여 질문을 집어넣을 경우 결과가 나오도록 출력

3. 2. 1. UI로 질문을 입력란과 질문을 받았을 경우 답변하는 출력을 보여줌

3. 2. 2. Train 데이터를 토대로 만들어진 모델 추가

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

3. 2. 4. 실제 Object와 비교하여 정확도 측정 :

Copyrightⓒ. Saebyeol Yu. All Rights Reserved.

003 한계

1. 전체적인 모델의 한계

1. 1 학습 데이터 부족(Predict Relation의 한계)

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

1. 1. 2. 위를 해결하기 위하여 각 relation당 5~10개의 템플릿을 제작하여 질문을 생성

1. 1. 3. 더 많은 학습 데이터를 추가하여 해결 필요

2. 2 이음동의어 처리

2. 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

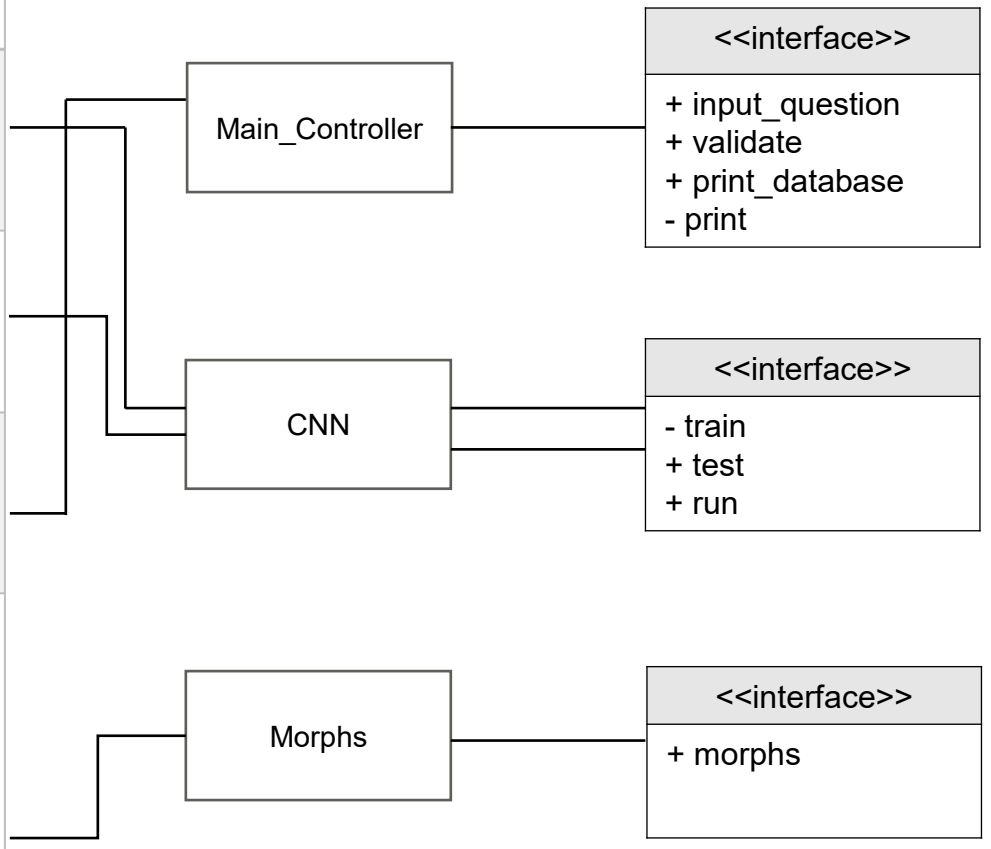
2. 2. 2. 예시로 한국은 대한민국으로 처리가 불가

2. 2. 3. Subject 데이터 자체를 Word2Vector를 이용가능한 방식을 사용하여 해결 필요

Copyrightⓒ. Saebyeol Yu. All Rights Reserved.

004 System Test Cases

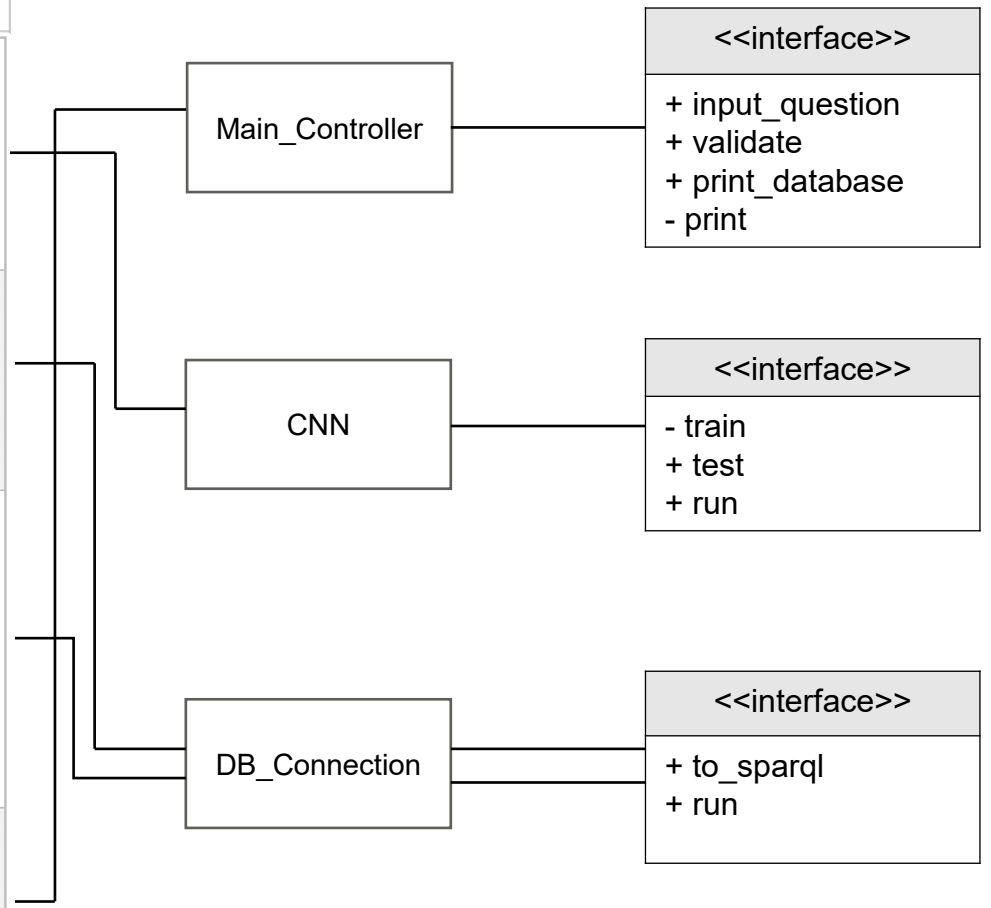
Case No.	테스트케이스 목표	입력 상황	예상 결과	실행 결과
TEST 1.1.1 (기능)	입력 쿼리에서 Subject 를 정확히 추출하는 AI 가 구축되어야 한다.	야나미 조지가 태어난 곳은 어디인가?	"야나미 조지"라는 데이터가 도출됨	성공 36%의 정확도
Test 1.1.2 (기능)	입력 쿼리에서 Relation 을 정확히 추출하는 AI 가 구축되어야 한다.	야나미 조지가 태어난 곳은 어디인가?	"birthPlace"이라는 데이터가 도출됨	성공 76.6%의 정확도
TEST 1.2.1 (기능)	사용자가 등록한 Simple question 이 프로그램에 정확히 입력되어야 한다.	사용자 입력: 야나미 조지가 태어난 곳은 어디인가?	"야나미 조지가 태어난 곳은 어디인가?"라는 문장을 입력받음	성공 - '야나미 조지가 태어난 곳은 어디인가?' 가 인풋 문장으로 입력된다.
TEST 1.2.2 (기능)	입력받은 문장을 형태소로 정확히 분해할 수 있어야 한다.	야나미 조지가 태어난 곳은 어디인가?	(야/나미/조지/가/ 태어나/ㄴ/곳/은/ 어디/ 이/ㄴ가/?/)	성공 - [('야', 'NNG'), ('나미', 'NNG'), ('조지', 'NNG'), ('가', 'JKS'), ('태어나', 'VV'), ('ㄴ', 'ETD'), ('곳', 'NNG'), ('은', 'JX'), ('어디', 'NP'), ('이', 'VCP'), ('ㄴ가', 'EFQ'), ('?', 'SF')]



Copyrightⓒ. Saebyeol Yu. All Rights Reserved.

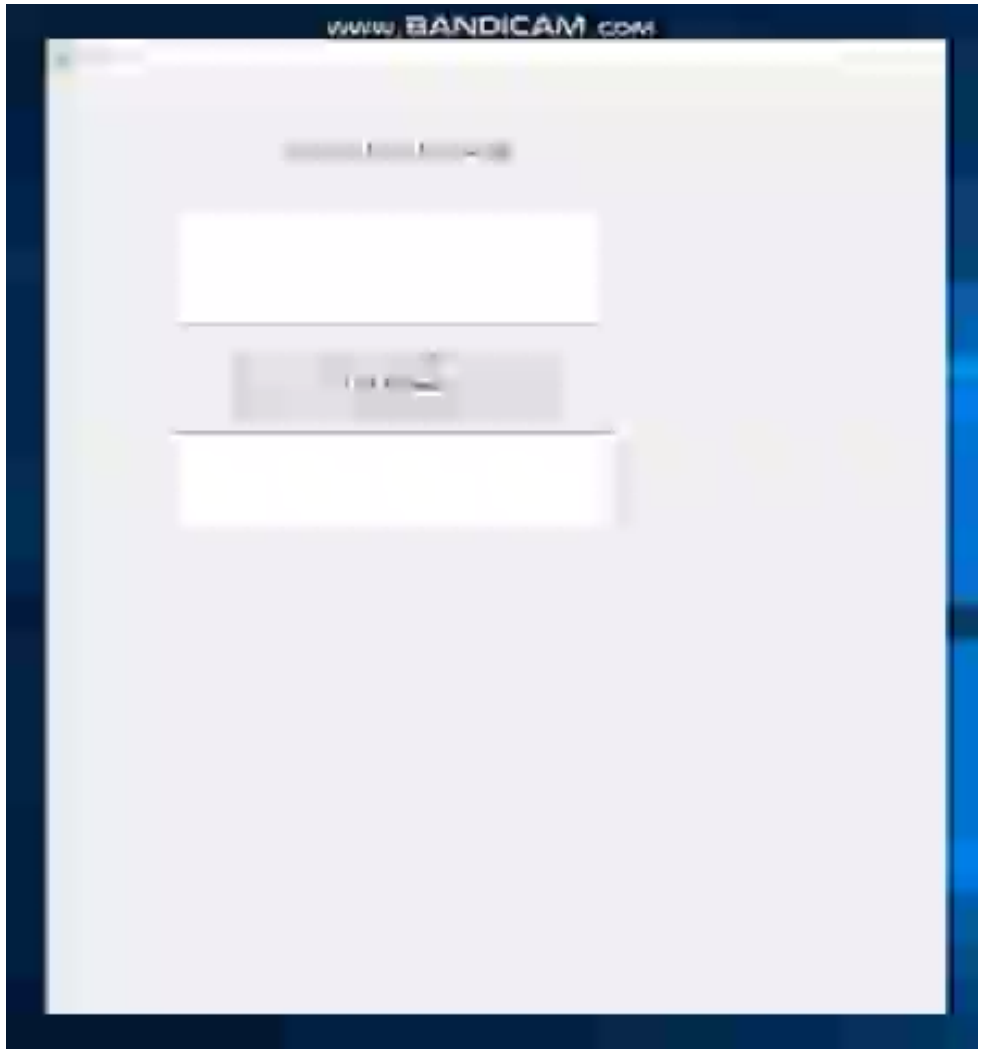
004 System Test Cases

Case No.	테스트케이스 목표	입력 상황	예상 결과	실행 결과
TEST 1.3.1 (기능)	입력된 쿼리로부터 정확히 Subject 와 Relation 을 추출해야 한다.	야나미 조지가 태어난 곳은 어디인가?	Subject – 야나미 조지 Relation – birthPlace	성공
TEST 1.3.2 (기능)	입력된 쿼리로부터 정확하게 Sparql 이 만들어져야 한다.	Subject – 야나미 조지 Predicated - 태어난 곳	Select 태어난 곳 Where 야나미 조지	성공 – SELECT ?b WHERE { <http://ko.dbpedia.org/resource/야나미_조지> <http://dbpedia.org/ontology/birthPlace> ?b .}
TEST 1.3.3 (기능)	DBPedia 를 참조하여 답변이 정확하게 나와야 한다.	SELECT ?b WHERE { <http://ko.dbpedia.org/resource/야나미_조지> <http://dbpedia.org/ontology/birthPlace> ?b .}	도쿄시	성공 -'도쿄시' 결과가 나온다.
Test 2 (비기능)	Simple Question 을 집어넣어 정답률 70% 이상을 달성해야한다.	Simple Question 의 Test Dataset 을 모델에 입력	테스트 결과 70%이상	



Copyrightⓒ. Saebyeol Yu. All Rights Reserved.

005 데모 영상



Copyrightⓒ. Saebyeol Yu. All Rights Reserved.