

이 프로그램이 입력과 출력은 각각 무엇이며

본 시스템의 입력은 사용자가 관계를 정의하고자 하는 트리플이고 출력은 링크드 데이터 형식으로 변환된 트리플입니다. 또한, 이러한 방식으로 나온 출력값을 다시 입력값으로 사용해서 다른 데이터와 어떻게 유기적으로 연결되었는지 확인할 수 있습니다. 이 때의 출력값은 데이터간의 연결관계도입니다.

이 프로그램의 용도/의도는 무엇이며

본 프로그램은 두 가지 의도를 가지고 제작되었습니다. 먼저 시맨틱 웹 환경에 진입장벽이 너무 높다는 문제의식으로 인하여 신규사용자들이 복잡한 문법에 대한 이해없이 링크드 데이터를 저작할 수 있도록 벤다이어그램 모델 preset을 이용한 저작방식을 도입하였습니다.

두 번째로 시맨틱 웹 환경에서 특정 entity와 멀리 떨어져있는 다른 entity간의 연결관계를 검색하는 방법에 있어서 연결되어있는 모든 entity를 조회하는 식의 기존의 접근은 너무 비효율적이라고 판단하여 개념간의 depth값을 저장하여 같은 predicate를 가진 entity끼리는 연결관계를 즉각적으로 판단하는 방법을 고안하였습니다.

이 프로그램이 왜/어떻게 도움이 되며

위에서 소개한 두가지 용도/의도는 다음과 같은 이유로 도움이 됩니다. 먼저 일반 유저가 어렵지 않게 링크드 데이터관계를 저작할 수 있기 때문에 사용자가 별개의 서비스에 각각 접근해서 여러번 수행해야 되는 일을 '기계가 스스로 이해하는 웹'에서는 한번에 처리 가능합니다. 특히나 현행 시맨틱웹에서는 연결관계 부재로 인하여 여러 서비스가 상호작용하지 못하는 상황이 자주 있는데 간단하게 연결관계를 정의하는 것으로 이러한 문제에 대한 해결방안을 모색할 수 있습니다.

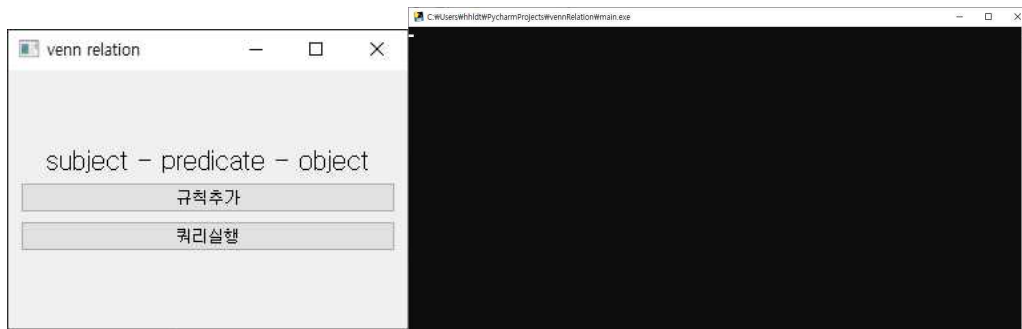
두번째로 본 시스템에서 정의된 relation의 포함관계는 서로에 대한 포함값을 벤다이어그램 모델에 따라 predicate으로서 저장하게 되기 때문에 entity간의 상하관계를 즉각적으로 판단할 수 있습니다. 본 시스템에서는 이러한 상하관계를 활용하여 해당계층 단계에 따라 relation을 검색할 때 기존방법에 비해 계산효율적으로 접근하는 방법을 모색하였습니다. 예를들어 포함관계를 선택할 때 entity a,b 는 상하관계에 있는 만큼 선택한 predicate에 depth index값을 부여하고 추후 entity b,c 등의 관계에도 같은 predicate가 사용된다면 통용되는 index값을 저장하여 직접적으로 연결되지않은 a,c도 index값을 통해 그 상하관계를 유추하는 방식을 골자로 합니다.

이 프로그램은 어떤 플랫폼 (윈도우 10 및 어떤 환경인지) 에서 수행되는지

- 본 프로그램은 <https://github.com/kellysolow/vennRelation> 에서 다운로드 가능하며 exe 배포용 파일을 구성하였기 때문에 대부분의 윈도우 환경에서 무리없이 작동하리라 생

각합니다. 다만 다루는 파일의 용량이 큰 편이기 때문에 컴퓨터의 램 크기가 4기가 이하일 때 문제가 발생할 수 있습니다.

## 시스템 사용방법



0) 본 시스템은 실행시 위의 그림과같이 규칙추가, 쿼리실행 버튼을 가진 **ui**와 출력내용을 표시하는 **콘솔창**이 함께 등장합니다.

콘솔창은 출력내용을 확인하는 데에 사용하고 실질적으로는 좌측의 규칙추가/쿼리실행 두가지 기능이 존재합니다.

### 1) 규칙추가

규칙추가 버튼은 사용자가 두 개의 entity를 선택하여 relation을 구성하는 기능입니다.

여기에서 entity란 주어나 목적어 자리에 들어갈 수 있는 하나의 개체를 의미합니다. 가장 기초적인 단위가 되며 시맨틱 웹이 수많은 node와 edge로 이뤄진 graph라고 할 때, entity는 node에 해당합니다. 규칙이란 두 개의 entity가 서로 어떻게 이어지는지에 대한 관계(relation)을 의미하며 graph에서는 edge역할을 합니다. 규칙의 형식은 주어-서술어-목적어(subject-predicate-object)로 이루어지며 이 3요소를 트리플(triple)이라고 합니다.

시맨틱 웹에서는 무수히 많은 entity가 규칙들을 통해서 서로 어떠한 상호작용을 하고 어떠한 관계가 있는지 명시되어 있으며 이러한 규칙들을 이용해서 기계가 스스로 이해하는 웹(시맨틱 웹)이 구현될 수 있습니다.

다만 본 프로그램에서는 이러한 규칙을 세부적으로 다루지 않고 가장 기초적인 연결단계의 성립만 다룹니다. 즉 subject -predicate - object 형식으로 트리플을 서술하면, 'subject와 object가 predicate로 연결되었다.' 를 의미합니다.

해당 기능을 수행하기 위해서 사용자는

- 1-1) entity 선정
  - 1-2) 벤다이어그램 선택
  - 1-3) relation을 triple 형태로 입력
- 의 과정을 거치게 됩니다.

#### 1-1) entity 선정

사용자는 자신이 관계(relation)를 구축하고 싶은 아무 두 개의 단어를 선택하여 relation을 생성할 수 있습니다. 시맨틱 웹의 특징을 파악하기 위한 최소단위로서, 두 개의 단어를 선택하고 해당 단어간의 관계가 무엇인지 작성하게 됩니다.

여기에 단어는 제약이 없으나 relation은 주어-서술어-목적어의 트리플 구조를 따라야 합니다.

주어, 목적어에 들어가게 되는 entity는 기존에 있는 것을 사용할 수 있는데 기존에 있는 entity의 목록은 함께 첨부된 entitiesCount.xlsx 엑셀파일을 통해 확인할 수 있습니다.

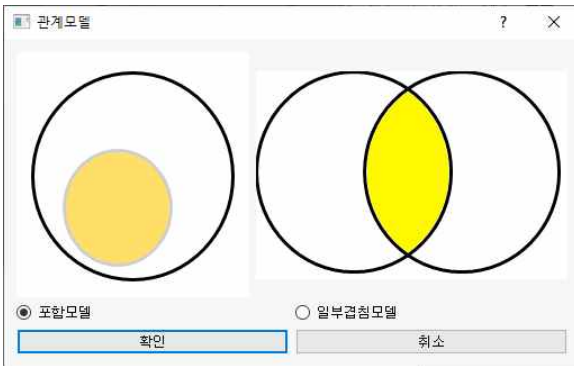
#### 1-2) 벤다이어그램 선택

트리플 구조로 선택한 relation을 Linked Data 형식으로 만드는 과정입니다. 기존의 시맨틱 웹에 접근하고자 하는 사람들은 여기에 필요한 문법형식 등에 대한 지식이 필요했지만 본 프로그램에서는 벤다이어그램을 선택하는 것으로 이 과정을 대체할 수 있습니다.

벤다이어그램은 집단간의 관계를 시각적으로 표현한 수학적 모형으로, 일반적으로 시맨틱 웹 relation 등에서 자주 활용되는 모형은 아닙니다. 본 연구에서는 벤다이어그램이 relation을 만드는데에 집단간의 관계를 시각적으로 잘표현하고, 일반 사용자들이 자주 접해본 간단한 모델이라고 여겨 도입하였습니다.

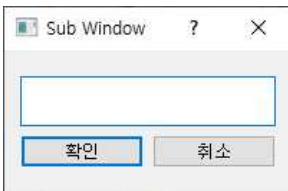
제시된 벤다이어그램 모델중 정의하고자 하는 관계와 비슷한 것을 선택하면 시맨틱 웹의 데이터 형식으로 정의한 관계가 자동 변환되어 저장되며, 사용자는 이 과정에서 시맨틱 웹의 데이터형식과 문법에 대해 학습할 필요가 없습니다.

ex) 주어-서술어- 목적어 관계의 예시로 한국-소속하다-서울 의 관계를 가정해보겠습니다.

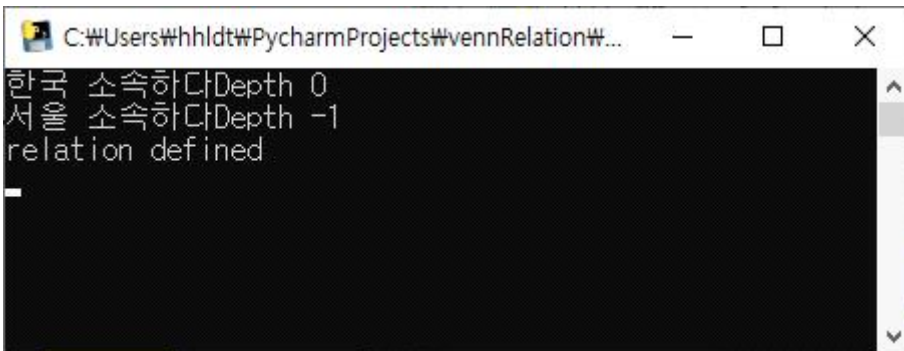


정의하고자 하는 관계가 양쪽의 모델중 어느 것에 부합하는지를 선택하면 됩니다.  
 여기에서 좌측의 모델은 큰원- subject, 작은 색칠된 원-object에 해당하는 관계이며  
 우측의 모델은 좌측원-subject, 색칠된부분 -object, 우측원-predicate에 대응하는 object  
 집단에 해당합니다.

### 1-3) relation을 트리플 형태로 입력



모델을 선택하고 확인 버튼을 누르면 위와같은 입력창이 등장합니다.  
 여기에 정의하고자 하는 관계를 subject predicate object 로 적으면 됩니다.  
 위의 예시에서는 한국 소속하다 서울 을 입력하면 됩니다.



해당 과정이 잘 수행될 경우 위와같은 콘솔창을 확인할수 있습니다.

## 2) 쿼리실행

쿼리실행에 해당하는 관계에서는 주어진 data dump에서 두 개의 entity간의 연결관계가 존재하는지에 대한 검색을 수행할 수 있습니다.

해당 기능을 통해 사용자는 1)규칙추가를 통해서 성립시킨 entity의 관계와 기존에 존재하던 data들의 관계까지도 검색할 수 있습니다.

### 2-1) entity 선정

사용자는 연결관계가 궁금한 두 개의 entity를 선택합니다.

여기에서 선택하는 entity는 1)규칙추가에서 직접 만들어넣은 단어가 될수도 있고

기존 entitiesCount.xlsx 파일에 존재하는 단어일 수도 있습니다.

위의 예시에서 사용한 한국과 서울의 관계가 존재하는지를 검색해 보도록 하겠습니다.

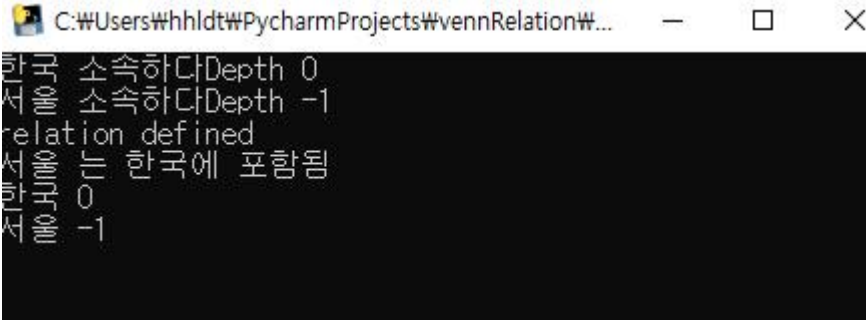
### 2-2)쿼리를 triple 형태로 입력

어떤 관계가 존재하는지를 검색하기 때문에 여기에서 사용하는 triple은

위에서 사용하던 주어-서술어-목적어 형식에서 서술어 대신 얼마나 정밀하게 검색할지를 정하는 depth를 사용합니다.

즉 주어 depth 목적어 형식으로 검색하게 되며 약 10의 depth승만큼 주어진 data dump를 스캔하게 되므로 너무 높은 숫자를 적게되면 오류가 발생할 수 있습니다. (5이하 권장)

한국 2 서울 의 검색결과는 아래 사진과 같습니다.



```
C:\Users\hhldt\PycharmProjects\vennRelation#\...
한국 소속하다Depth 0
서울 소속하다Depth -1
relation defined
서울 는 한국에 포함됨
한국 0
서울 -1
```

본 시스템을 통해 정의한 (한국 - 서울)의 경우 depth를 같이 기록하기 때문에 훨씬 빠르게 검색되며 entitiesCount.xlsx에 존재하는 기존 entities 들은 검색에 시간이 소요되지 않습니다.

## Q. 사용자가 왜 linkeddata를 만들고 연결시켜보아야 하는가

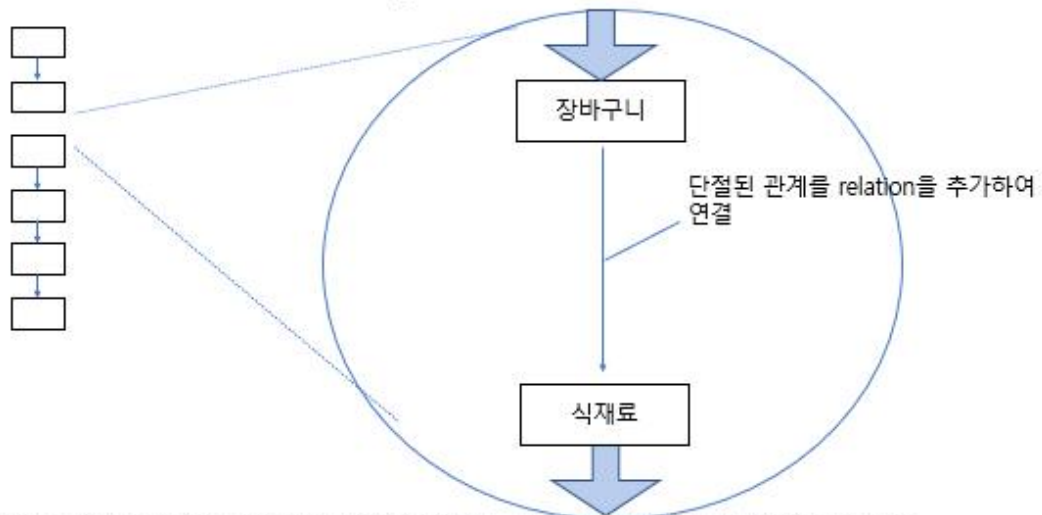
시맨틱 웹은 문서 단위로 연결되어 고안된 목적에 따른만큼의 작동을 하는 기존웹과 다르게 각각의 서비스에 접근하지 않고 기계가 유기적으로 처리해주는 혜택을 누릴수 있음.

기존 웹에서 '병원의 필수영양소에 대한 처방을 받은 시나리오' 상황을 생각해보면

1. 병원에서 진찰받은 영양소 기록
2. 영양소에 해당하는 식단 검색
3. 그중 환자가 선호하는 식사와 식사의 재료 판단
4. 해당 식사재료를 쇼핑몰 사이트 환자의 장바구니에 수록함

등의 각각의 손이가는 시나리오를 한단계로 축약가능함.

## 문제 접근 방안



Semantic Web 환경상에서 주로 발생하는 위와 같은 Linked Data 단절에 대한 연결관계 쉽게 구성

위의 시나리오가 이론처럼 이루어지지 않는 이유로는 성립된 Linked Data 단절로 인한 요인이 있음.

Linked Data에 대한 relation을 구성해보면서 사용자는 이러한 단절을 링크드 데이터에 대한 세부지식없이 극복가능해짐.

## 시나리오

### 1. Michigan- Illinois 관계 정의

11	Michigan	1902
12	Town	1730
13	Mountain_Time_Zone	1599
14	Illinois	1441

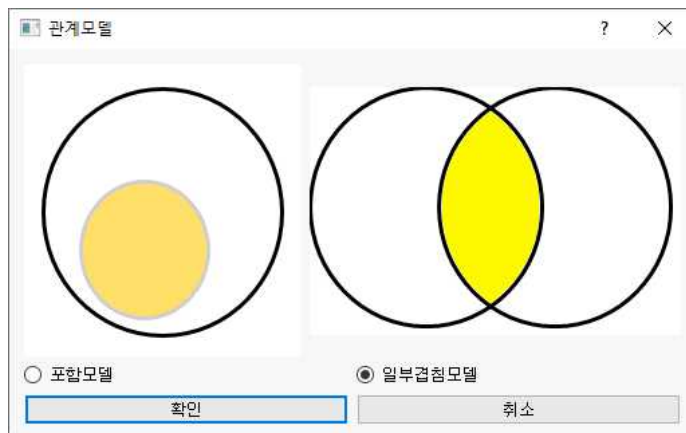
entitiesCount.xlsx에서 관계를 정의하고 싶은 entity를 확인하던 도중 Michigan과 Illinois의 존재를 확인하였다.

Michigan과 Illinois는 둘다 미국의 주로서 이러한 관계는 동등한 급의 entity임을 암시하는 '일부겹침' 모델을 통해 정의하고자 한다.

해당 정의는

Michigan StatesOfAmerica Illinois로 수행하였으며

일부겹침모델에서 좌측원-Michigan 우측원-StatesOfAmerica 색칠부분-Illinois에 해당한다.



결과는 아래와 같다.

```
C:\Users\Whhldt\PycharmProjects\ven...
Michigan StatesOfAmericaDepth 0
Illinois StatesOfAmericaDepth 0
relation defined
```

## 2. 서울시 - 한국의하위행정구역 - 회기동 관계정의 및 검색

본 시스템에서 정의한 triple은 depth라는 기준을 통해서 추후 검색에 용이하게끔 기록을 남기며 1 depth란 하나의 subject에 연결된 object까지의 거리를 의미한다.

즉 subject -predicate -object 까지가 1depth이며  
subject - predicate -object(subject로 다시 작용) -predicate - object가 2 depth이다.

위에서 정의하고자 하는 트리플은

1. 서울시 - 한국의하위행정구역 - 동대문구
2. 동대문구 - 한국의하위행정구역 - 회기동 이다.

```
서울시 한국의하위행정구역Depth 0
동대문구 한국의하위행정구역Depth -1
relation defined
동대문구 한국의하위행정구역Depth -1
회기동 한국의하위행정구역Depth -2
relation defined
```

이후 정의한 내용대로 쿼리실행버튼을 누르고

서울시 2 회기동

을 검색하면 서울시 동대문구 회기동에 대한 결과를 얻을수 있다.

```
회기동 는 서울시에 포함됨
서울시 0
회기동 -2
```

Depth가 포함되지않은 기존의 데이터덤프의 값과의 연결관계는 검색하는데에 10의 depth승만큼의 계산시간이 필요하지만 본문에서 정의한 값은 depth를 비교하여 결과를 얻으므로 빠르게 포함관계여부를 파악할 수 있다.



### 3. 기존에 정의되어 있는 England - London에 Big\_Ben 이어붙이기

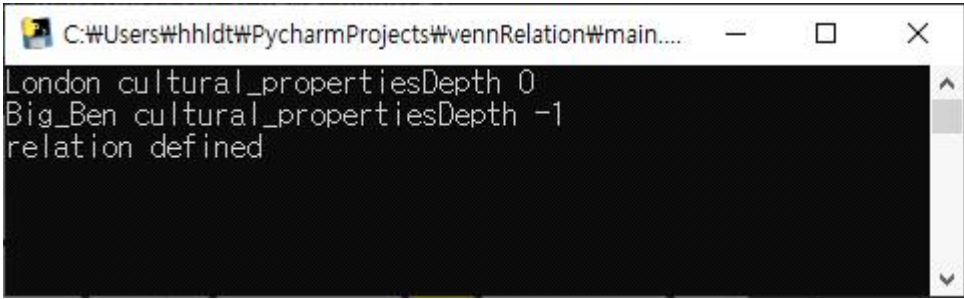
37	England	597
38	Kentucky	592
39	Alabama	572
40	Australia	566
41	France	554
42	0.0^^<http://www.w3.org/2001/XMLSchema#float>	525
43	Alaska	518
44	English_language	467
45	1035995.2441344^^<http://www.w3.org/2001/XMLSchema#double>	458
46	London	452

기존 dump에는 England - capital(서술어) - London 에 대한 정보가 존재한다.

본 시나리오 에서는 여기에 Big\_Ben이라는 런던에 존재하는 세계에서 가장 유명한 시계탑을 이어붙이고자 한다.

규칙추가-포함모델선택

London cultural\_properties Big\_Ben



```

C:\Users\hhldt\PycharmProjects\vennRelation\main...
London cultural_propertiesDepth 0
Big_Ben cultural_propertiesDepth -1
relation defined
  
```

위의 정의로 인하여

England - capital - London - cultural\_properties - Big\_Ben

관계가 형성되었다.

그러나 England의 경우 직접 정의하지 않은 상태로 depth값이 지정되지않아 시나리오 1,2와는 다르게 depth에 따라 직접 subject에 연결된 모든 object를 조회하는식으로 검색이 진행 된다.

England 2 Big\_Ben의 결과물은 다음과 같다.

```
depth 1 searching...

using subject : "2011"^^<http://www.w3.org/2001/XMLSchema#gYear>

using subject : Devolution

using subject : Pound_sterling

using subject : London

Found the relation!
```

+Depth가 활용되는 예시 London - leader- Boris\_Johnson

Big\_Ben을 추가한후 검색하는것처럼 정의하지 않고도 이미 존재하는 entity에 대해서 depth에 따른 검색을 수행할 수 있다.

data dump에 London-leader-Boris\_Johnson이 존재하는데 이를 활용해서 **England** - capital -London - leader -**Boris\_Johnson**을 검색하면 다음과 같다.

쿼리실행

England 2 Boris\_Johnson 입력



```
C:\Users\hhldt\PycharmProjects\vennRelation\main.exe
depth 1 searching...

using subject : "2011"^^<http://www.w3.org/2001/XMLSchema#gYear>

using subject : Devolution

using subject : Pound_sterling

using subject : London

Found the relation!
```

-