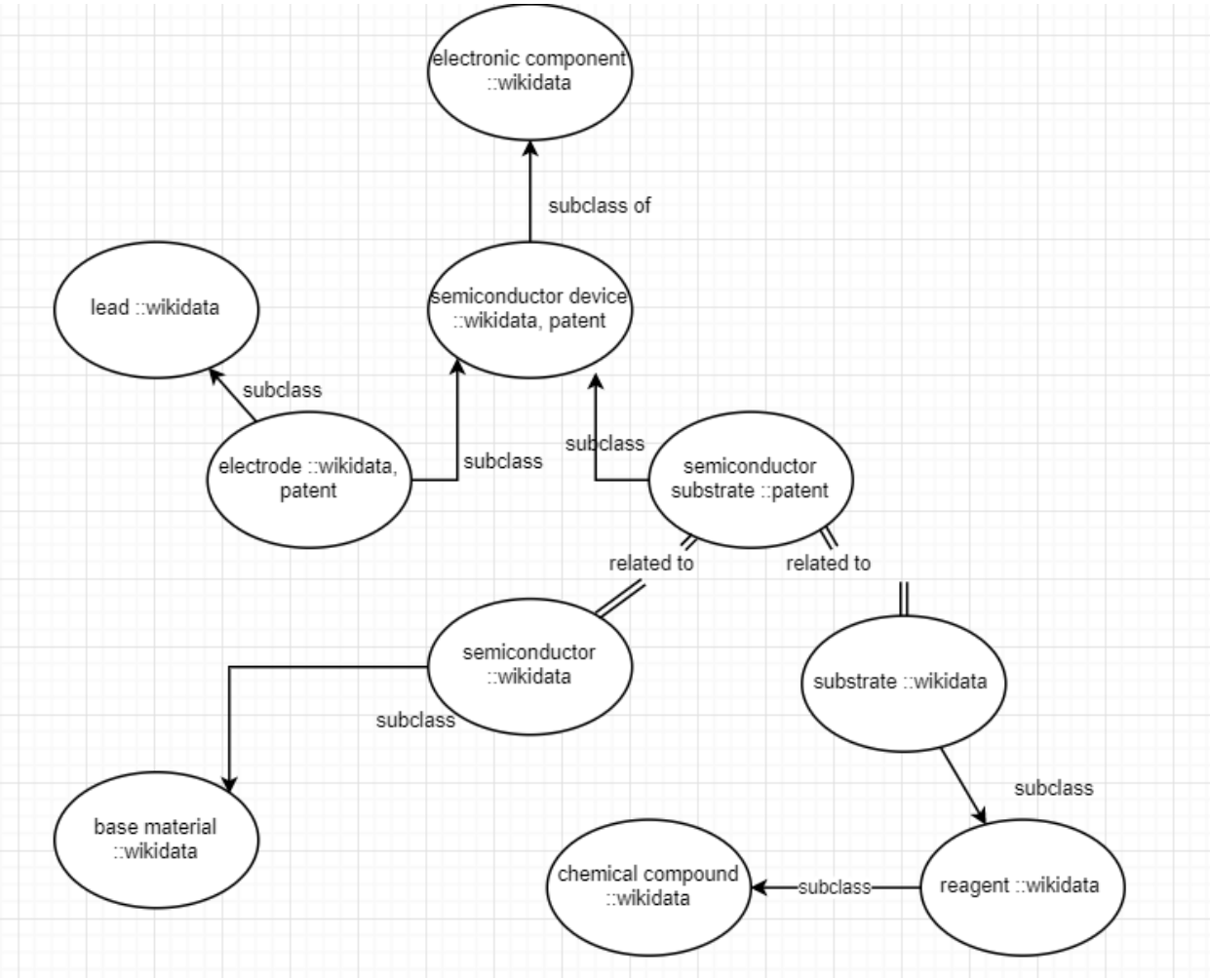
Knowledge Graph Crawling from Wiki

A – Purposes

With all kind of relations, we have a hypothesis that our system could understand if a new element A (not in our training data) is an instance of X, subclass of Y, part of Z must be the same of element B which is existed in our product, labeled as S

More over, using KG (knowledge graph) we can know the detail level of a concept from metaclass which is very useful for **GENERAL SCORE computation**. For example: entity (a kind of meta class) is a parent of many concepts used in patent data.



B – Methods

Step 0: Find name mention in description or short name of an entity in the wikidata

Ex: trifluoromethyl Q2302144

Method 1:

1. Step 1: Find direct parent of each entity for upper level 1

2. Step 2: Find parent intersection from all parents of each entity to find a root node

```
"""SELECT ?item ?itemLabel WHERE { wd:"""+entity_id1+""" wdt:P279+ ?item .
?item wdt:P279+ wd:"""+entity_id2+"""
. SERVICE wikibase:label { bd:serviceParam wikibase:language "en" }}"""
```

2.1. Find a path between 2 entities to check whether one is a subclass of the other (but not a direct parent). For example: there is a path between entity1: **semiconductor device** and entity2: **electric device**: semiconductor device>>electronic component>>chemical element>>electronic device (it could takes time or make time out request when an entity have too many parents)

When crawling top down from a root node, we will only extract items in the list of paths between these entities

2.2. Check whether entity_id1 is direct parent of entity_id2

```
"""SELECT ?item ?itemLabel WHERE{wd:"""+ entity_id+""" wdt:"""+propertyID + """?item .
SERVICE wikibase:label { bd:serviceParam wikibase:language "en" }}"""
```

3. Crawling top down from a node which is not a child of any nodes

Disadvantage:

- Time performance O(n*2) with n > 5000 and easy to get time out request because of find intersections in step 2 because of P279+ (find all upper parents)

Conclusion:

This method could not be implemented any more because or timeout request or too long to wait (at least 5 days)

Method 2:

1. Step 1: Using RDF_GAS_API (this API has been learning about the constraints that allow for scalable, multi-machine, and (with MapGraph), massively parallel graph algorithms) to find parent in upper 3 levels and the link between all nodes in only 1 step instead of find direct parent for 3 times which leading to timeout request or malfunction request

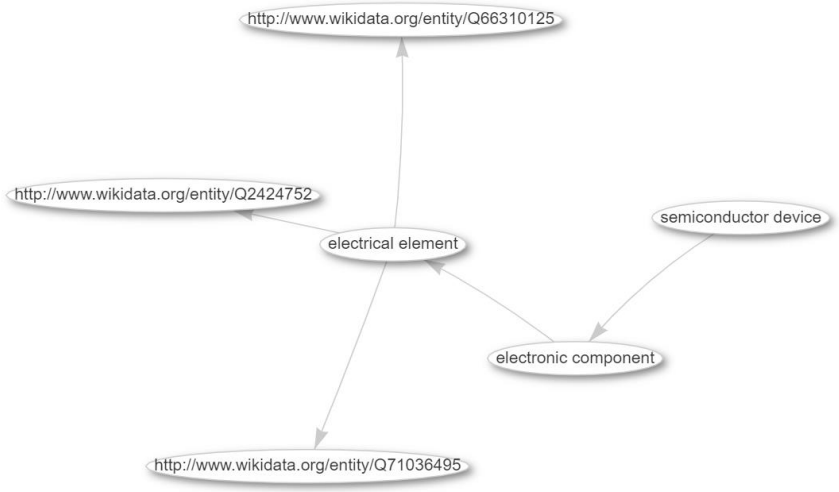
```
""PREFIX gas: <http://www.bigdata.com/rdf/gas#>

SELECT ?item ?itemLabel ?linkTo
WHERE
{
  SERVICE gas:service {
    gas:program gas:gasClass "com.bigdata.rdf.graph.analytics.SSSP" ;
                gas:in wd:"""+entity_id+""";
                gas:traversalDirection "Forward" ;
                gas:out ?item ;
                gas:out1 ?depth ;
                gas:maxIterations """+str(iteration)+""" ;
```

```
            gas:linkType  wdt:P279 .
  }
  OPTIONAL { ?item wdt:P279 ?linkTo }

  SERVICE wikibase:label {bd:serviceParam wikibase:language  "en" }
}"""
```

item	itemLabel	linkTo
wd:Q11653	electronic component	wd:Q210729
wd:Q175805	semiconductor device	wd:Q11653
wd:Q210729	electrical element	wd:Q2424752
wd:Q210729	electrical element	wd:Q66310125
wd:Q210729	electrical element	wd:Q71036495



2. Update label, short_name, description for all upper level parents

```
""" SELECT
  ?id?label?desc
  (GROUP_CONCAT(DISTINCT(?aka);  separator="|") AS ?akas)
WHERE{
  VALUES ?id { wd:"""+entity_id+"""  }
  OPTIONAL{ ?id rdfs:label ?label. FILTER(LANG(?label)="en")}
  OPTIONAL{ ?id skos:altLabel ?aka . FILTER(LANG(?aka)  = "en")}
  OPTIONAL{ ?id schema:description  ?desc . FILTER(LANG(?desc)  = "en")}
}
GROUP BY ?id?label?desc"""
```

3. Make tree builder from all links between entities with node level 0 = root

Advantage:

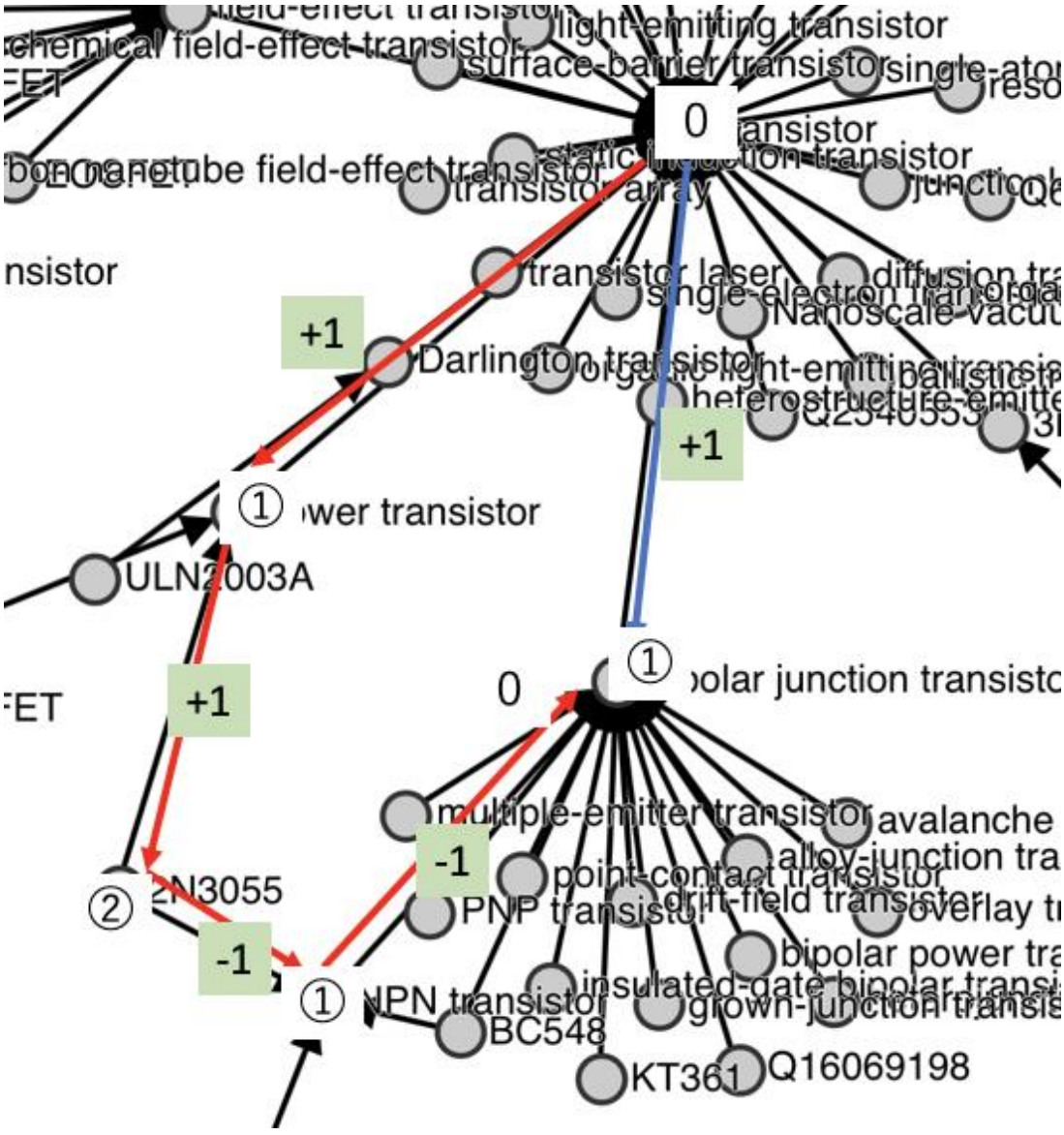
- Fast and prevent “timeout request”

- **Q35120 (entity)** can cover all main trees in patents (equipment, semiconductor device of electronic component tree, base material, physical property, physical state, physical phenomenon, structure, process, chemical element, chemical substances)
- Now one tree with root node entity can represent all those important trees instead of separated trees as last week

B – Knowledge Graph Level Labelling

Problem 1: Create a graph structure from crawled wiki data by building 3 classes Graph, Node, Level

Problem 2: How to decide level of an entity



Current solution: Level of an entity is defined by the shortest from that entity to root node in a tree

For example: there are 2 paths from Q35120 to Q107 -> Level_of(Q107)= min(2 paths, **key**=len)

```
"Q107":
["Q35120", "Q488383", "Q7184903", "Q930933", "Q4393498", "Q5469988", "Q36161",
"Q177646", "Q3054889"],
["Q35120", "Q16686448", "Q386724", "Q7184903", "Q930933", "Q4393498",
"Q5469988", "Q36161", "Q177646", "Q3054889"],
```

C - Graph Visualize Using Library of d3.js lib

Link https://github.com/AngryLoki/wikidata-graph-builder

NOTE: Code updated on repository
https://github.com/Cinnamon/sony_patent_evaluation