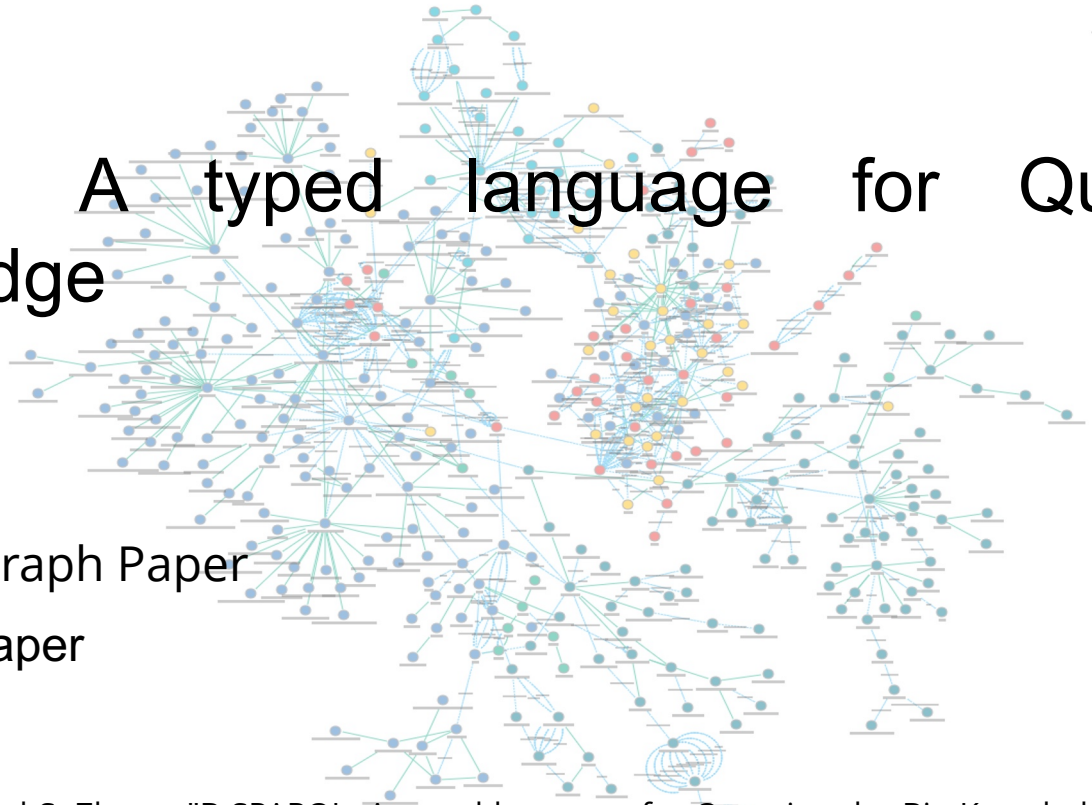
Individual Research Task by **Pallavit Aggarwal**
Course = MSc Intelligent Systems

B-SPARQL: A typed language for Querying the Big Knowledge



Tag: Knowledge Graph Paper

Tag: Application Paper

R. Lu, C. Wang, X. Huang and S. Zhang, "B-SPARQL: A typed language for Querying the Big Knowledge," 2020 IEEE International Conference on Knowledge Graph (ICKG), 2020, pp. 173-180, doi: 10.1109/ICBK50248.2020.00034

https://ieeexplore-ieee-org.elib.tcd.ie/document/9194459

Area/Motivation

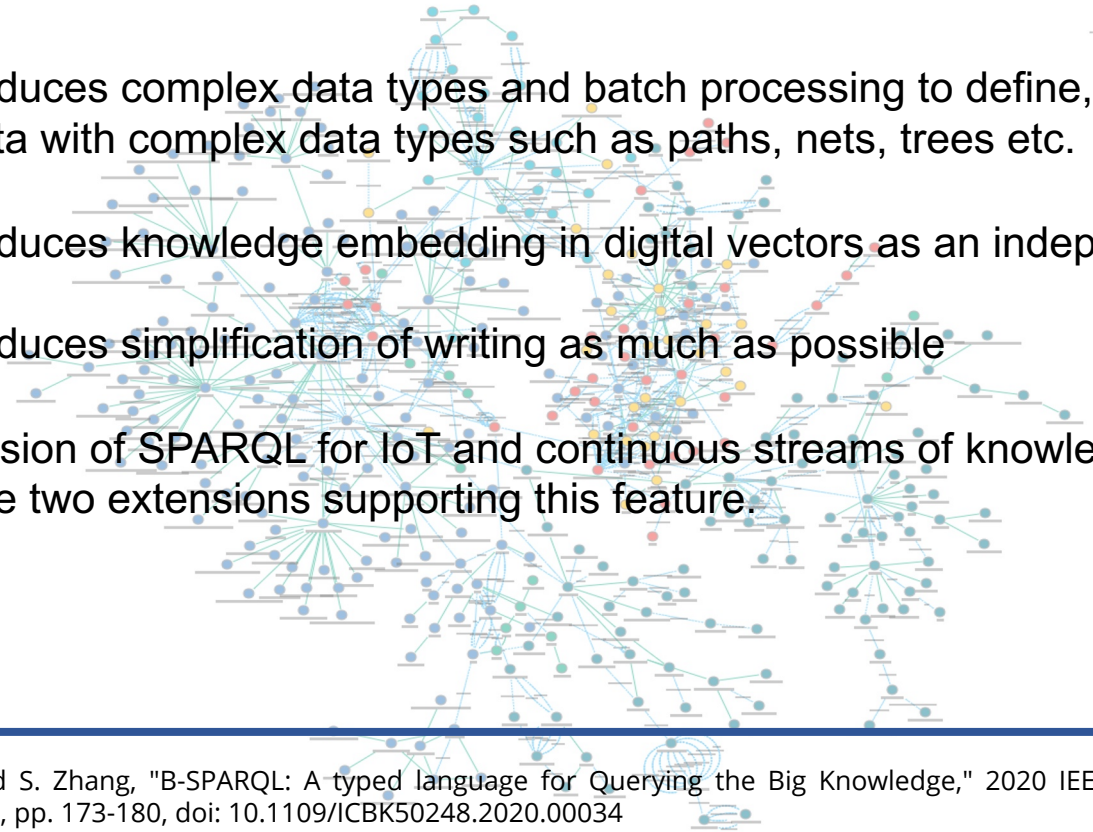
i. Under the background of BK and BD, SPAQRL is not powerful enough to meet growing requirements

ii. B-SPARQL introduces complex data types and batch processing to define, operate and query massive size data with complex data types such as paths, nets, trees etc.

iii. B-SPARQL introduces knowledge embedding in digital vectors as an independent data type

iv. B-SPARQL introduces simplification of writing as much as possible

v. Discusses extension of SPARQL for IoT and continuous streams of knowledge. C-SPARQL and DESERT are two extensions supporting this feature.



- R. Lu, C. Wang, X. Huang and S. Zhang, "B-SPARQL: A typed language for Querying the Big Knowledge," 2020 IEEE International Conference on Knowledge Graph (ICKG), 2020, pp. 173-180, doi: 10.1109/ICBK50248.2020.00034
- Ruqian, L.U., Chaoqun, F.E.I., Chuanqing, W.A.N.G., Shunfeng, G.A.O., Han, Q.I.U., Zhang, S. and Cungen, C.A.O., 2020. HAPE: A programmable big knowledge graph platform. *Information Sciences*, *509*, pp.87-103.
- https://www.oracle.com/in/big-data/what-is-big-data/

Area/Motivation

- *?ID(Xany)* means queries an entity which is an identifier.
- *?SEQ(anytype)* means querying a sequence of data separated by commas.
- *?STR(anytype)* means querying a string of symbols not including separators such as comma, blanc, point, colon, etc.
- *?B(anytype)* means querying a bag (multi-set) of data of any type.
- *?E(anytype)* means querying a vector embedded from some data of any type.
- *?N(anytype)* means querying a network of data of any type.
- *?P(anytype)* means querying a path in some network.
- *?C(anytype)* means querying a chain in some network.

?Xany means querying an entity *Xany*;

?Xpany means querying a predicate *Xpany*;

?UPR(Xany1, XAny2) means querying an unordered pair *Xany* of entities;

?OPR(Xany1,Xany2) means querying an ordered pair *Xany* of entities;

?T(Xany) means querying a triplet *Xany*.

?Tr(Xany) means querying a tree *Xany*.

?A(Xany) means querying an algorithm name *Xany*.

?S(anytype) means querying a set of data of any type. This set may be a knowledge graph, part of a knowledge graph, a SPARQL endpoint or just several triplets.

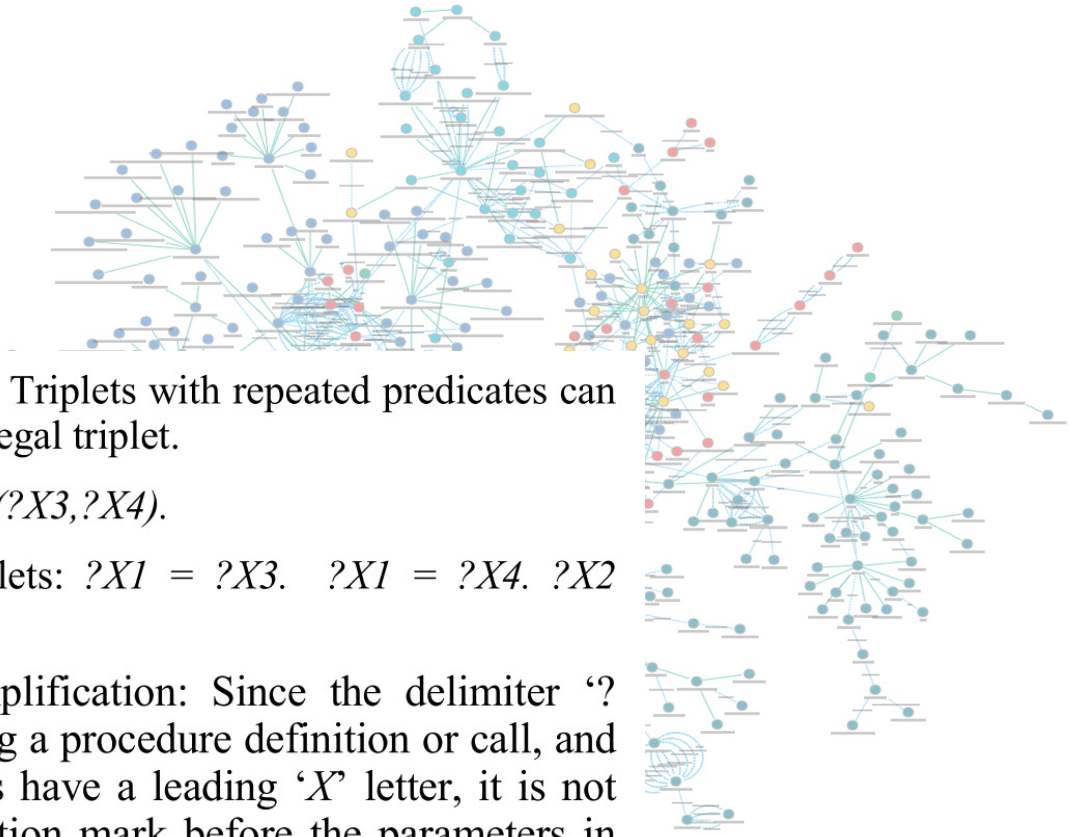
?L(anytype) means query a list of data separated by blancs.

Eg: ?S(S(T(X)))

- R. Lu, C. Wang, X. Huang and S. Zhang, "B-SPARQL: A typed language for Querying the Big Knowledge," 2020 IEEE International Conference on Knowledge Graph (ICKG), 2020, pp. 173-180, doi: 10.1109/ICBK50248.2020.00034

Area/Motivation

B-SPARQL introduces simplification of writing as much as possible



Triplet simplification: Triplets with repeated predicates can be saved. Following is a legal triplet.

all (?X1, ?X2) = all (?X3,?X4).

It equals to four triplets: *?X1 = ?X3. ?X1 = ?X4. ?X2 = ?X3. ?X2 =* ?X4.

Procedure head simplification: Since the delimiter '? Select' is always heading a procedure definition or call, and since all variable names have a leading '*X*' letter, it is not necessary to put a question mark before the parameters in procedure head.

- R. Lu, C. Wang, X. Huang and S. Zhang, "B-SPARQL: A typed language for Querying the Big Knowledge," 2020 IEEE International Conference on Knowledge Graph (ICKG), 2020, pp. 173-180, doi: 10.1109/ICBK50248.2020.00034

Overview of the Solution Proposed

B-Spa1.0 is the first implementation of B-SPARQL

Written in Python, extends open-source library RDFLib

Syntax -> B-SPARQL's query clause identical to SPARQL, B-SPARQL's conscise representation, New Semantic primitives

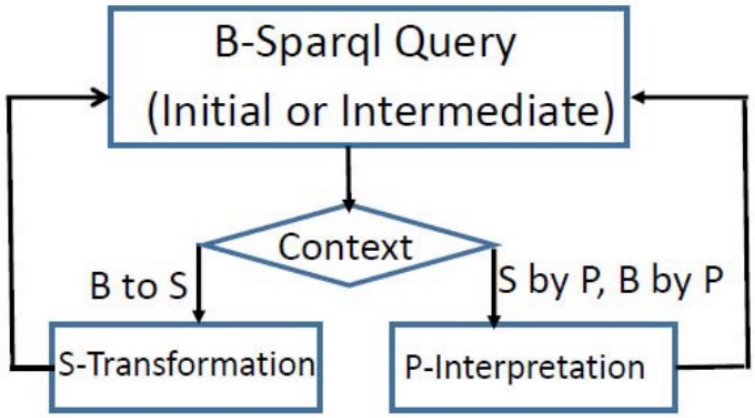


Fig. 1 Transformational Semantics of B-Sparql

- T.E. Oliphant, Python for scientific computing, Comput. Sci. Eng. 9 (3) (2007) 10–20.
- D. Krech, Rdflib: a python library https://github.com/RDFLib/rdflib (2006).

Example 3.1.1

Example 3.1.1: Find K most frequent predicates from M largest knowledge areas:

Select ?KMmost(S(OPR(XPre, Ynum)); Yk, Ym, S(S(T(Ygl))))

Where{Call select ?Mmost(S(S(T(Ypart))); Ym, S(S(T(Ygl)))).

Call select ?Kmost(S(OPR(XPre, Ynum)); Yk, S(T(Yloc)))

where {?S(T(Yloc)) in ?S(S(T(Ypart)))}.}

Group by ?XPre.

Order by desc(?Ynum).

Limit ?Yk.

Select ?Kmost(S(OPR(XPre, Ynum)); Yk, S(T(Yloc)))

Where {{?Xa ?XPre ?Xb.} in ?S(T(Yloc)).}

Group by ?XPre.

Order by desc (count(?XPre)).

Limit ?Yk.

?Ynum := count(?XPre).

Select ?Mmost(S(S(T(Ypart))); Ym, S(S(T(Yglob))))

Where { ?S(S(T(Ypart))) subsetOf ?S(S(T(Yglob))).}

Order by desc(count(anyof(?S(S(T(Ypart)))))).

Limit ?Ym.

The call statement is:

Call select ?KMmost(S(OPR(XPre, Ynum)); 10, 3, {DBpedia, YAGO, Wikidata}).

Call **select ?KMmost(S(OPR(XPre, Ynum)); 10, 3, {DBpedia, YAGO, Wikidata}).**

Select ?KMmost(S(OPR(XPre, Ynum)); Yk, Ym, S(S(T(Ygl))))
Where{
Call select **?Mmost(S(S(T(Ypart))); Ym, S(S(T(Ygl)))).**
Call select **?Kmost(S(OPR(XPre, Ynum)); Yk, S(T(Yloc)))**
where { S(T(Yloc)) in ?S(S(T(Ypart))) }. }
Group by ?XPre.
Order by desc(?Ynum).
Limit ?Yk.

Select ?Mmost(S(S(T(Ypart))); Ym, S(S(T(Yglob))))
Where { ?S(S(T(Ypart))) subsetOf ?S(S(T(Yglob))).}
Order by desc(count(anyof(?S(S(T(Ypart)))))).
Limit ?Ym.

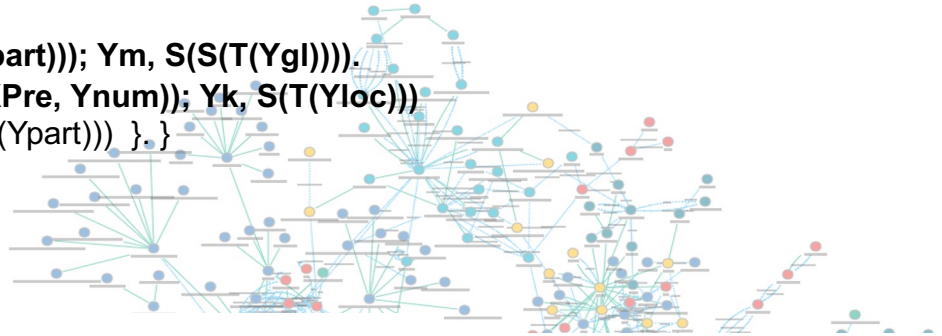
Select ?Kmost(S(OPR(XPre, Ynum)); Yk, S(T(Yloc)))
Where{ { ?Xa ?XPre ?Xb. } in ?S(T(Yloc)). }
Group by ?XPre.
Order by desc (count(?XPre)).
Limit ?Yk.
?Ynum := count(?XPre).

- R. Lu, C. Wang, X. Huang and S. Zhang, "B-SPARQL: A typed language for Querying the Big Knowledge," 2020 IEEE International Conference on Knowledge Graph (ICKG), 2020, pp. 173-180, doi: 10.1109/ICBK50248.2020.00034

Example 3.1.1 contd.

Call **select ?KMmost(S(OPR(XPre, Ynum)); 10 ,3, {DBpedia, YAGO, Wikidata}).**

Select ?KMmost(S(OPR(XPre, Ynum)); Yk, Ym, S(S(T(Ygl))))
Where{
  Call select **?Mmost(S(S(T(Ypart))); Ym, S(S(T(Ygl)))).**
  Call select **?Kmost(S(OPR(XPre, Ynum)); Yk, S(T(Yloc)))**
  where { S(T(Yloc)) in ?S(S(T(Ypart))) }. }
Group by ?XPre.
Order by desc(?Ynum).
Limit ?Yk.



- *?ID(Xany)* means queries an entity which is an identifier.
- *?SEQ(anytype)* means querying a sequence of data separated by commas.
- *?STR(anytype)* means querying a string of symbols not including separators such as comma, blanc, point, colon, etc.
- *?B(anytype)* means querying a bag (multi-set) of data of any type.
- *?E(anytype)* means querying a vector embedded from some data of any type.
- *?N(anytype)* means querying a network of data of any type.
- *?P(anytype)* means querying a path in some network.
- *?C(anytype)* means querying a chain in some network.

?Xany means querying an entity *Xany*;

?Xpany means querying a predicate *Xpany*;

?UPR(Xany1, XAny2) means querying an unordered pair *Xany* of entities;

?OPR(Xany1,Xany2) means querying an ordered pair *Xany* of entities;

?T(Xany) means querying a triplet *Xany*.

?Tr(Xany) means querying a tree *Xany*.

?A(Xany) means querying an algorithm name *Xany*.

?S(anytype) means querying a set of data of any type. This set may be a knowledge graph, part of a knowledge graph, a SPARQL endpoint or just several triplets.

?L(anytype) means query a list of data separated by blancs.

Example 3.1.1 contd.

Experiment VII.1 (Example 3.1.1)

In Table II we only list three most frequent predicates of each KG. The global result is (W1, W2, W3, Y1, Y2, D1, Y3, D2, D3). Roughly Wikidata > YAGO > DBpedia, except that D1 > Y3.

TABLE II. THREE MOST FREQUENT PREDICATES AND THEIR FREQUENCIES OF 3 KGS

DBpedia	YAGO	Wikidata
http://www.w3.org/1999/02/22-rdf-syntax-ns#type 113715836	http://www.w3.org/2000/01/rdf-schema#comment 139236795	http://schema.org/description 2205858413
http://www.w3.org/2002/07/owl#sameAs 33623686	http://www.w3.org/2000/01/rdf-schema#label 137296236	http://www.w3.org/1999/02/22-rdf-syntax-ns#type 1155990655
http://purl.org/dc/terms/subject 23990492	http://www.w3.org/1999/02/22-rdf-syntax-ns#type 55748756	http://wikiba.se/ontology#rank 1049614942

- R. Lu, C. Wang, X. Huang and S. Zhang, "B-SPARQL: A typed language for Querying the Big Knowledge," 2020 IEEE International Conference on Knowledge Graph (ICKG), 2020, pp. 173-180, doi: 10.1109/ICBK50248.2020.00034

Example 3.3.1

Select ?count(distinct(?XP1)) as ?Xcomplete

?count(distinct(?XP2)) as ?Xtotal ?Xcomprate
Where{ ?X2 ?XP1 ?Y2.

Where {?X1 ?XP1 ?Y1. ?X2 ?XP2 ?Y3. sameClass (?X1, ?X2) = true }}

Filter(STR(?X1) != STR(?X2))

?Xcomprate = ?Xcomplete/?Xtotal.

Select ?count(distinct(?XP1)) as ?Xcomplete
?count(distinct(?XP2)) as ?Xtotal ?Xcomprate

Where{
?X2 ?XP1 ?Y2.
Where {?X1 ?XP1 ?Y1. ?X2 ?XP2 ?Y3. sameClass (?X1, ?X2) = true }
}

Filter(STR(?X1) != STR(?X2))
?Xcomprate = ?Xcomplete/?Xtotal.

A. Experiment VII.2 (Example 3.3.1)

The data in Table III is the result of querying the KG NELL [54] for evaluating the degree of its predicate completeness. Table III shows its number of predicates and complete predicates and the completeness rate.

TABLE III. NELL'S COMPLETENESS OF PREDICATES

# predicates	# complete predicates	Completeness rate
828	355	355/828=0.429

- *T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, B. Yang, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. 2018. Never-ending learning. Commun. ACM 61, 5 (May 2018), 103–115. https://doi.org/10.1145/3191513*
- S. Issa, O. Adekunle, F. Hamdi, S. S. -S. Cherfi, M. Dumontier and A. Zaveri, "Knowledge Graph Completeness: A Systematic Literature Review," in IEEE Access, vol. 9, pp. 31322-31339, 2021, doi: 10.1109/ACCESS.2021.3056622.

Examples 4.1.2 , 5.1

?N (T(X)) ; ?P(T(X)) ; ?C (T(X))

Select ?C(X)

Where {?C(X) terminal e1. ?C(X) terminal e2.}

Following is the same query in detailed form which makes use of query procedure definition and a function 'undirect' producing a reversed form of a triplet as its alternative. More exactly, undirect *(?X1, ?X2', ?X3) = {?X1, ?X2', ?X3 || ?X3, ?X2', ?X1}*.

Select ? minechain (C(X);Y1,Y2)

Where {T1|| T2 T3 || T4 T5 call Select ? minechain (C(X);X2,X3) }

Above we use simplified notation to avoid very tedious writing, where

T1 = {?Y1 ?X1' ?Y2.} T2 = {?Y1 ?X1' ?X2.} T3 = {?X2 ?X4' ?Y2.}

T4 = {?Y1 ?X1' ?X2.} T5 = {?X3 ?X5' ?Y2.}

Call Select ? Ave-distance (Y4; 100, 100, 0).

Select ? c-distance (X ;ID(Y1),ID(Y2), X1).

Where { ?ID(Y1) = ?ID(Y2). ?X := ?X1.

|| all (?ID(Y1),?ID(Y2)) in ?T(Y). ?X := ?X1 +1.

|| all (?ID(Y1),?ID(Y3)) in ?T(Y). all (?ID(Y2),?ID(Y4)) in ?T(Y).

Call Select ? c-distance (X ;ID(Y3),ID(Y4), X1+2).

Select ? Ave-distance (Y4; Y5, Y6, Y7)

Where { ?Y5 > 0.

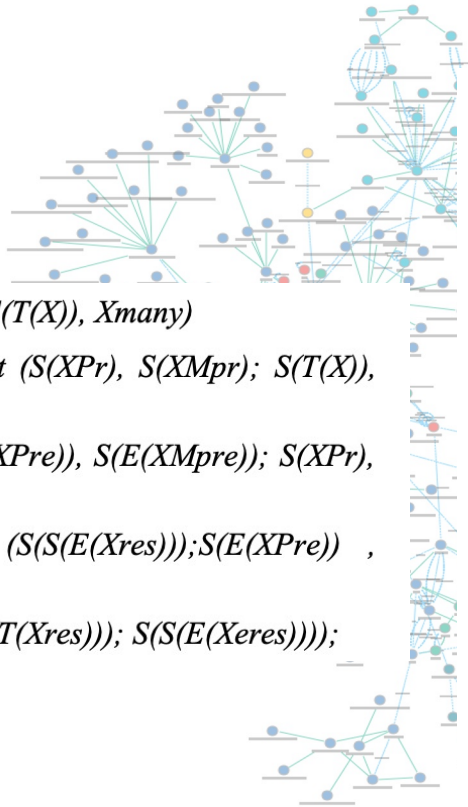
Call Select ? distance (X; Random (ID(Y), Y1) , Random (ID(Y), Y2), 0, Y),

Call Select ? Ave-distance (Y4; Y5-1, Y6, X+Y7, Y)}

TABLE IV. SMALL WORLD RATE OF THREE KGS

	DBpedia	**Orpha**	**Rhea**
Size	438336346	2296871	3291266
Average distance	3.32	2.24	1.95
S.W. rate	1.807	2.679	3.077

Example 6.1



Select ? Classify (S(S(T(Y)));S(T(X)), Xmany)

Where {Call Select ?Collect (S(XPr), S(XMpr); S(T(X)), Xmany)

Call Select ?Embed (S(E(XPre)), S(E(XMpre)); S(XPr), S(XMpr))

Call Select ? Cluster (S(S(E(Xres)));S(E(XPre)) , S(E(XMpre)),Xmany)

Call Select ? Reverse (S(S(T(Xres))); S(S(E(Xeres))));

TABLE V. CLASSIFY DBPEDIA IN THREE CLUSTERS

DBpedia	type	sameAs	subject
# predicates	686	1191	453
# triplets	130107097	59400920	36419711

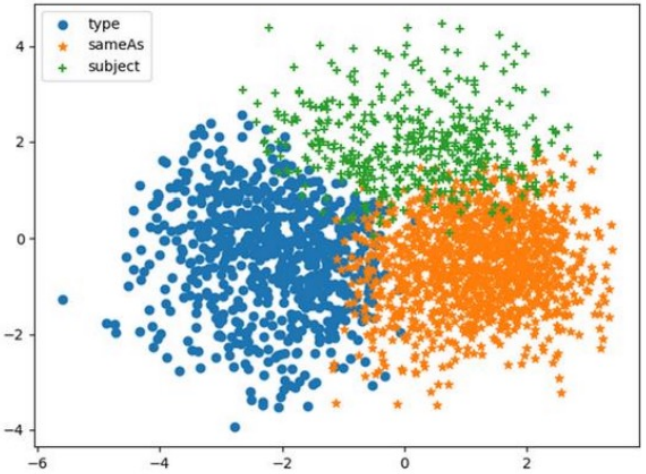


Fig. 2 Distribution of 3 clusters

Reflection

What the authors considered novel or the major contribution?

a) A new typed query language is introduced and its syntax is discussed
b) Authors extends RDF Lib and use Python as the programming language
c) B-SPARQL provides proper embedding and vector translation technique
d) Different enrichments in SPARQL like PSPARQL, SPARQLeR, GLEEN, TrQuery, Vec2SPARQL , FunSPARQL, CSPARQL and DESERT.
e) Different semantic examples using B-SPARQL

What I considered interesting?

a) SPARQL paves the way to improving our understanding to deal with Big Knowledge
b) There are different aspects of data, that stretch the current needs to explore the querying language in other dimensions as well.

What it tells me about the state of knowledge graphs?

a) Knowledge graphs are growing in size and data diversity
b) There are newer ways to compute and mine knowledge from Knowledge Graphs
c) There should be an effort to unify the current processes and languages to tareget the knowledge graphs more wholistically.