Approximate Sub-graph Matching over Knowledge Graph

Jiyuan Ren(✉), Yangfu Liu, Yi Shen, Zhe Wang, and Zhen Luo

Northeast Branch of State Grid Corporation of China, Liaoning, China
582854926@qq.com

Abstract. With the rapid development of the mobile internet, the volume of data has grown exponentially, and the content of data become more complicated. It is hard for people to select useful information from such a large number of data. In this paper, we study the problem of approximate sub-graph matching over knowledge graph. We first propose two algorithms to reduce the scale of knowledge graph. Next, we use an efficient algorithm to find similarity sub-graphs. Thirdly, we use skyline technique to further select high quality sub-graphs from the matching results. Theoretical analysis and extensive experimental results demonstrate the effectiveness of the proposed algorithms.

Keywords: Knowledge graph · Sub-graph matching · Compression · Skyline

1 Introduction

With the rapid development of the mobile internet [1,2], the volume of data has grown exponentially, and the content of data become more complicated. It is hard for people to select useful information from such a large number of data. In addition, when the target of a user is an object- portfolio other than a signal object, the problem turns to more difficult since users should fully consider relationships among different data.

For example, in a event-based mobile social network [3,4], users often select object-portfolio, i.e., shop, sports, and cinemas based on the their location, their business hours and so on. For example, Peter usually attends sports activities organized on Meetup. In the evening of Friday, Peter faces a dilemma since Meetup recommends four three conflicting sport activities on Sunday: a hiking trip from 7:00 a.m. to 11:00 p.m. (at the place A), a badminton game from 10:00 a.m. to 14:00 a.m (at the place D), a basketball game from 11:30 a.m. to 2:30 p.m. (at the place F), and a football game from 1:30 a.m. to 3:30 p.m. (at the place G). Though Peter is interested in all these four sports, he can only attend few of them. Thus, Peter must make a decision to select an object- portfolio.

In a network attack defense system, the system should process data package generated from a large number of IP addresses, and it has to identify malicious network attack and attack mode. Usually, the attack mode is complex. As an example, a compound attack often involves multiple attacks step, we use could use a graph structure to express the relationship among attackers and hosts.

J. Liu et al. (Eds.): MobiCASE 2020, LNICST 341, pp. 198–208, 2020.
https://doi.org/10.1007/978-3-030-64214-3_14

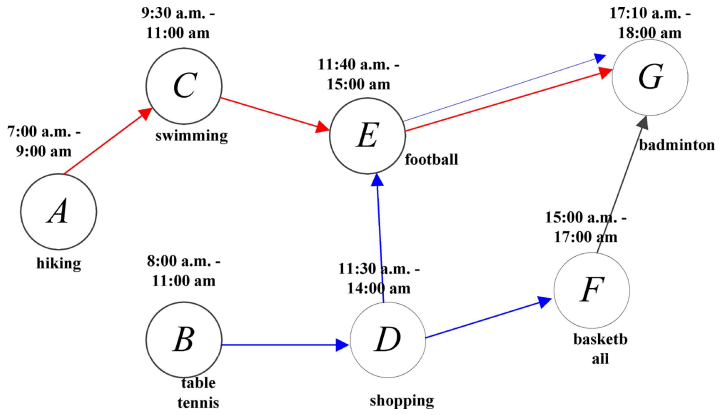


Fig. 1. Event-based mobile social network

However, compared with simple graph structure [5–7], examples discussed above is very complex. The main reason is we should associate each nodes with many information. Back to the example in Fig. 1. We should maintain the starting (also ending) moment for each node. In addition, we should maintain the keywords for each node. Thus, we introduce knowledge graph (short for KG). A KG is a heterogeneous graph [8–10], where nodes function as entities, and edges represent relations between entities. Items and their attributes can be mapped into the KG to understand the mutual relations between items. Moreover, users and user side information can also be integrated into KG. The benefit is users can accurately capture relationship among users and items.

In this paper, we study the problem of subgraph matching over large scale knowledge graph. Since it is a NP-hard problem, in order to solve this problem, we propose an approximate framework named CBSM (short for Compress-based Sub-graph Matching). Given a graph $G(V, E)$, we first compress it to a sub-graph G'. Compared with G', both the edge scale and vertex scale are all reduced a lot. In this way, we achieve the goal of using a small number of objects to support query processing. Next, we propose a skyline-based sub-graph matching algorithm. Compared with other algorithms, it considers both the similarity of graph structure and the similarity of key words. Above all, the contribution of this paper is as follows.

- We propose a novel algorithm to compress knowledge graph. It first computes the weight of all edges. Based on the computing result, we propose a randomly algorithm to delete edges (also vertexes) that are not important. Since the scale of the knowledge graph is reduced, the cost of the subgraph matching algorithm also could be reduced.
- We propose a skyline-based algorithm to find similarity sub-graphs from knowledge graph G. It evaluates the similarity via the following two facets. Firstly, it evaluates the similarity based on the graph structure. Secondly, it evaluates the similarity based on key word set difference between query graph and sub-graphs in G.

The rest of this paper is organized as follows: Sect. 2 gives background, Sect. 3 presents our proposed solution. Section 4 evaluates the proposed methods with extensive experiments. Section 5 is the conclusion.

2 Background

In this section, we first review the algorithms about sub-graph matching. Thereafter, we explain the problem definition (Table 1).

Table 1. The summary of notations

Notation	Definition
KG	A knowledge graph
V	The vertex set
E	The edge set
K	The key word set
Q_S	The skyline sub-graph
KG_C	The compressed knowledge graph

2.1 Related Work

Matching over Static Graph. Many researchers studied the problem of static graph matching. Lee et al. compared five classic static graph matching algorithms under the same environment, i.e., vF2, quicksi, graphql, Gaddi, sPath and so on. They explain the advantages and disadvantages of these five algorithms. Based on the application types, Yu Jing et al. classify graph matching based on different application, such as structural matching, semantic matching, precise matching, approximate matching as well as optimal algorithm and approximate algorithm. It focuses on the algorithms about static graph matching over static graph. However, it is not suitable to solve the problem of pattern matching in dynamic graph data.

Matching over Dynamic Graph. Many researchers studied the problem of sub-graph matching over dynamic graph. They mainly focus on the incremental maintenance algorithm when a new vertex (or a new edge) is inserted into the graph. Wang et al. propose a neighbour-tree based method to support subgraph matching over dynamic graph. Their key idea is using neighbour-tree to filter sub-graphs that have no chance to become query results. The benefit is the number of sub-graphs that should be verified could be effectively reduced.

With the emergence of data center, many researches use network topology structure to express relationships among different centers. Compared with traditional dynamic graph, the topological structure of graph data will not change frequently. However, the attribute values of nodes may be changed frequently. In order to process this problem, zhong et al. proposed the gradin algorithm. The key idea behind it is it uses the

attribute value of each node in the frequent subgraph as a data dimension. In this way, the frequent subgraph can be represented as an n-dimensional coordinate vector in the n-dimensional grid index. In this way, all frequent subgraphs of the data graph can be indexed.

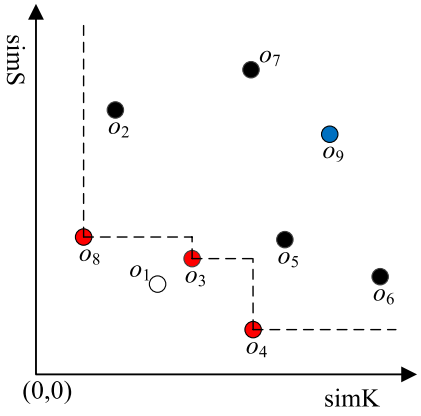
In many cases, the system has the requirement of real time. Thus, approximate algorithms with error guarantee is more preferred in many cases. Among all the algorithms, IncIsoMatch, SJ$®Tree and Gradin are representative ones. Their key idea is using relationship among vectors to approximately evaluate similarity between query graph and sub-graphs. Thus, the complex of these algorithms are reduced a lot.

2.2 Problem Definition

In this section, we first explain the concept of Dominance. Thereafter, we formally discuss the problem definition. For simplicity, let o be an object in $d-$dimensional space. Its coordinate is expressed the tuple $\langle o[1], o[2], \ldots, o[d] \rangle$. Without loss of generality, we assume that a smaller value indicates a better performance in all dimensions. Assuming $o_1[i]$ refers to the value of o_1 in the i-th dimension, we say $o_1 \prec o_2$ if $\forall i o_1[i] \leq o_2[i]$ and $\exists j o_1[j] \leq o_2[j]$.

Definition 1 DOMINANCE. *Let $\mathcal{D}$ be a set of $d-$dimensional objects with size N. $\forall o_1, o_2 \in \mathcal{D}$, we say object o_1 dominates o_2 (denoted as $o_1 \prec o_2$) if o_1 is as good as or better than o_2 in all dimensions, and better in at least one dimension.*

Definition 2 SKYLINE. *For each object $o \in \mathcal{D}$, if o is not dominated by any objects in $\mathcal{D}$, we regard o as a skyline object. All skyline objects in $\mathcal{D}$ make up of the skyline set, i.e., denoted as $\mathcal{S}$.*



(b) Skyline

Fig. 2. An example of skyline

As is shown in Fig. 2, there are 9 2-dimensional objects. Among them, o_3, o_4 and o_8 are skyline sub-graphs, we regard them as skyline objects. After discussing the concept of dominate and skyline, we now explain the concept of RDF graph and sub-graph matching over RDF graph.

Definition 3 KNOWLEDGE GRAPH. *Let E, R and S be three sets. They refer to the URI, blank nodes and literal. We call a tuple $\langle s, p, o\rangle \in (U \cup B) \times (U \cup B \cup L)$. Here, s is the subject, p refers to the predicate, and o refers to the object.*

Definition 4 SUB-GRAPH MATCHING OVER KNOWLEDGE GRAPH. *Let KG be a RDF Graph, $q\langle K, G\rangle$ be a query graph. q search on KG, find a sub-graph G' satisfying that the key word set of G' equals to $q.K$. In addition, G' should equals to $q.G$.*

Definition 5 APPROXIMATE SUB-GRAPH MATCHING OVER KNOWLEDGE GRAPH. *Let KG be a RDF Graph, $q\langle K, G\rangle$ be a query graph. q search on KG, find a sub-graph G' satisfying the following two cases. Here, $\mathsf{simK}(q.K, G'.K)$ refers to the similarity between keyword set of q and G'. $\mathsf{simS}(q.S, G'.S)$ refers to the similarity between structure of q and G'.*

- $\mathsf{simK}(q.K, G'.K) \leq \varepsilon$
- $\mathsf{simS}(q.S, G'.S) \leq \tau$

We find that, in real applications, it is difficult to find such a perfect sub-graph. In addition, the computational cost is also high. In most cases, we can find the solution in real time. Thus, in this paper, we propose the approximate RDF Sub-graph Matching over RDF Graph. We now formally discuss the problem definition.

Definition 6 SKYLINE BASED SUB-GRAPH MATCHING. *Let KG be a RDF Graph, $q\langle K, G\rangle$ be a query graph. q search on KG, find a sub-graph KG' satisfying that $\mathsf{sim}(q.K, KG'.K)$ is smaller than ε. In addition, $\mathsf{sim}(q.G, KG'.G)$ is smaller than η. Let $\mathcal{G}$ be a set of sub-graphs satisfying the above conditions, the find result set $\mathcal{G}$ is the skyline sub-graphs among all the sub-graphs in $\mathcal{G}$.*

3 Skyline Based Sub-RDF Matching

In this section, we propose a novel framework to support skyline based Sub-RDF matching. First of this section, we explain how to compress a given RDF. Next, we discuss the approximate matching algorithm. Last of this section, we discuss the skyline algorithm.

3.1 The Compression Algorithm

Recalling Sect. 2.2, a given KG could be expressed by the tuple $\langle E, R, S\rangle$. Since both the set E and R contains multitudes of key words, the space cost of a KG is high. Lucky, we find that some key words are similar with each other, we could use one key word to express them. In addition, some entries are all similar with each other, we also could use one entry to express it.

Algorithm 1: The Compression Algorithm

Input: Node Set $\mathcal{E}$, Knowledge Graph KG
Output: Compressed Knowledge Graph KG'

```
while E is not empty do
    Node e ← E[0];
    for i from 2 to |E| do
        if sim(e, E[i]) ≥ ζ then
            M ← E[i];
            E ← E − E[i];
    E' ←construction(M);
for i from 1 to |E| do
    Node e ← E[i];
    for j from 1 to |e| do
        sum ← sum + e[j].w;
    for j from 1 to |e| do
        I[j] ← e[j].w / sum;
        if I[j] ≤ τ then
            delete(e, e[j]);
            if |e[j]| = 0 then
                delete(e[j]);
return;
```

Take an example in Fig. 2. The node A, B, and C are three basketball venues. The service provided by them are expressed a group of key words. Since the key word set of them are similar with each other, we could construct a super node U. It contains the union of A, B, C's key words. Accordingly, we could use the nodes $U - A$, $U - B$, and $U - C$ to express nodes A, B, and C. Compared with the orient ones, the space cost could be reduced a lot (Fig. 3).

In order to compress key word information of each node, for each node $e \in E$, we compute the similarity of e and all the other nodes. After computing, we select all the ones whose similarity to e is smaller than a threshold ζ, construct a super node based on e and these nodes, and remove these nodes from E.

From then on, we repeat the above operations to process other nodes. The above operations is terminated until E turns to empty. Besides, we also should delete edges (vertexes) which importance is low. For example, given a vertex v and a group of other vertexes $\{v_1, v_2, \ldots, v_m\}$, for each i, it satisfied that there exists an edge $e(v, v_i)$. We evaluate the importance of an edge based on its weights. It can be computed based on Eq. 1.

$$\mathbf{I}_i = \frac{|w_i|}{\sum_{i=j}^{i=m} |W_j|} \tag{1}$$

It implies if the importance of an edge is high, the weight of this edge is high. Otherwise, the weight is low. Thus, if the importance of edge is smaller than a threshold τ, we could delete it directly. Based on the about observations, we could access every edge to compute its weight, and delete the ones whose importance is smaller than a threshold τ. In particularly, if a vertex has no edge, this vertex could be deleted directly.

As is shown in Algorithm 2, it contains two steps. In the first step, we access every node $e \in E$, find other nodes which are similar with e. After finding, we construct a super node based on these nodes. From then on, we repeat the above operations to process other ones. In the second step, we access every node $e \in E$, compute the weight sum of all vertexes that are connected with e. After computing, we delete all unimportant ones. In particularly, if we find another node e' which has no edge, we delete it. This algorithm is terminated until processing all nodes.

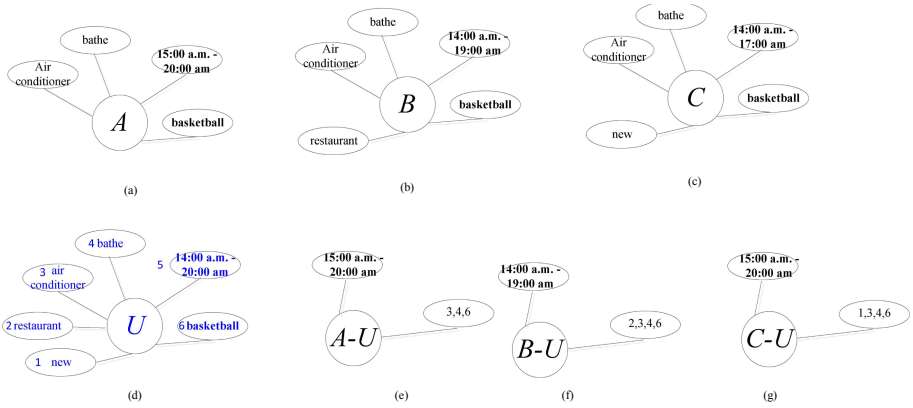


Fig. 3. Key word based compression

3.2 The Sub-graph Matching Algorithm

Given a query graph $G\langle S, K\rangle$, we first find the enumerate sequence. Since the enumeration algorithm have been discussed in many algorithms, we skip the details for saving space. Let CG be a set of enumerated result. These sub-graphs contain two types, that are, connected-graph and unconnected-graph. If a sub-graph is a unconnected-graph, it must not be a query result. Otherwise, we should further process it.

When a graph $CG[i]$ is a connected-graph, the matching contains two types. The first one is we could not find any prefix for q. The second one is we could find a few prefixes for q. Under case one, since q has at least one spanning tree, we access these spanning trees to find similar sub-graph, and insert all the query result into a set M. In other words, if we can find a sub-graph SG satisfying that $\mathsf{sim}(SG, q) \leq \delta$, we insert it into query result set. Under case two, if we can find the matching for the prefix of q, assuming the query graph q is the longest prefix, we check whether e_1, e_2, ... e_m are

Algorithm 2: The Sub-Graph Matching Algorithm

Input: Knowledge Graph $\mathcal{KG}$, query graph q
Output: Similarity Knowledge Graph Set SG

```
T ← gRandomSpanningTree(KG);
T ← nextsorting(T);
T ← jump-sorting(T);
£ CS ← Index-Matching(T);
for j from 1 to |CS| do
    θ ←simK(CS[i], q);
    if θ ≥ ε then
        CS ← CS − CS[i];
£ return;
```

contained in the graph G. If the answer is yes, it must have a matching. From then on, we repeat the above operations to find other sub-graphs.

After finding all these similarity sub-graphs, we then access their key word set. Our goal is to a group of sub-graphs whose corresponding key word set is also similar with that of the query graph. Here, we evaluate the relevance between the query q and the key word set of a graph via using Eq. 1. Here, $q(d)$ refers to a key words set of q. In order to achieve this goal, we scan all these sub-graphs, compute the key words similarity among these sub-graphs and the query graph. Last of all, we select sub-graphs satisfying that $\mathsf{simK}(g, q) \leq \varepsilon$.

$$simK(q.K, G.K) = \frac{|V(q.K) \cap q(q.K)|}{|q(G.K)|} \tag{2}$$

3.3 The Skyline Algorithm

Let CG be a set of sub-graphs. For each element $g \in CG$, it should satisfies that $\mathsf{simK}(q, g) \leq \varepsilon$ and $\mathsf{simS}(q, g) \leq \tau$. However, if we can find multitudes of sub-graphs from G, users still cannot select a property sub-graph. The reason behind it is it is difficult to find a suitable parameters δ and ε. In this section, we propose a skyline-based algorithm to solve this problem. Since the skyline query processing algorithms have been well studied, we skip the details for saving space.

4 Experimental Study

4.1 Experiment Setup

In this section, we use a real data set named Meetup as real data set. In this dataset, each node is associated with a set of tags and a location. Each event in the dataset is also associated with a location. They are used as vertexes. In addition, we use the reachability among different vertexes to construct edges. For example, given two event v_1 and v_2, there starting moment/end moment are [9.00pm, 11.pm], [9.30pm, 12.pm]

respectively. Since the ending time of v_1 is later than the starting time of v_2, we cannot construct an edge between v_1 and v_2. By contrast, let v_3 be another event. Its starting moment/ending moment is [12.00pm, 2.am]. Thus, we construct an edge between v_1 and v_3. We also use synthetic dataset for evaluation. We generate the utility values following Uniform, Normal and Power distributions respectively. The algorithms are implemented in C++, and the experiments were performed on a Windows 10 machine with Intel i7-7600 3.40 GHz 8-core CPU and 32 GB memory (Table 2).

Table 2. Summary of datasets

Dataset	Description	Vertex amount	Edge amount
MEETUP	Vancouver	225	2012
SYN-U	Synthetic data set	364	3367

For the parameter setting, we first evaluate the parameters τ and ε to the algorithm. In addition, we should evaluate the impaction of query graph scale to the algorithm performance. The parameter setting is shown in Table 3.

Table 3. Summary of parameters

Parameter	Description	Vertex amount
τ	Key word similarity	0.1, 0.15, 0.2, 0.25, 0.3
ε	Graph structure similarity	0.1, 0.15, 0.2, 0.25, 0.3
$\|q\|$	The number of vertex	2, 5, 10, 20, 30

4.2 Algorithm Performance

In this section, we compare the framework CBSM with a baseline algorithm. Compared with CBSM, it does not compress the graph. In addition, it does not terminate the searching the number of query results achieves to 100. In this paper, we call it as OSM. In the following, we first compare their performance under different parameter τ.

As is depicted in Fig. 4, CBSM performs best of all. The reason behind is, for one thing, we use two compression algorithms to reduce the space cost of knowledge graph. Thus, we only need to spend lower space cost in I/O. For another, since the graph scale turns to much smaller than the orient graph, the computational cost also could turn to small. For the stability, we compare CBSM with CBSM under synthetic dataset.

As is depicted in Fig. 5, CBSM still performs best of all. Similar with the reason discussed before, we use two compression algorithms to reduce the space cost of knowledge graph. Another reason is, with the increasing of ε, it become easy to find subgraphs that satisfy the query conditions. Since CBSM stop searching when the number of query results achieve to 100, the running time of CBSM become lower when ε is high enough.

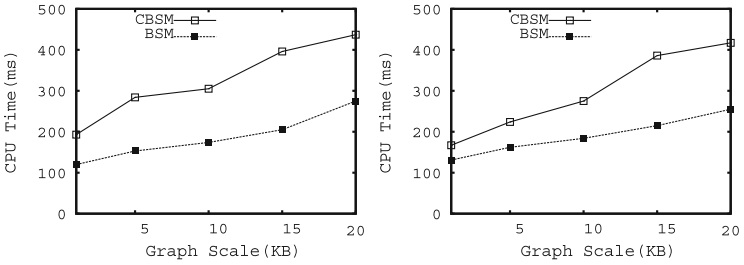


Fig. 4. Running time comparison under different graph scale

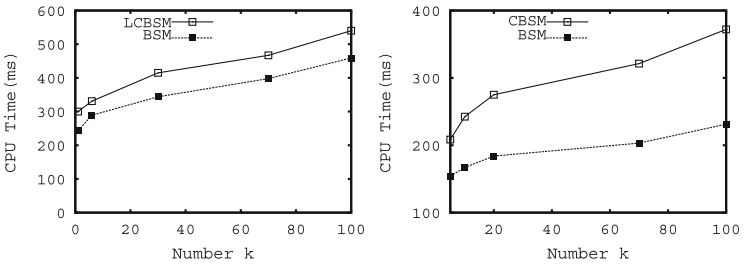


Fig. 5. Running time comparison under different ε

5 Conclusions

In this paper, we propose a novel framework named CBSM to support approximate sub-graph matching over knowledge graph. We first propose a compression algorithm to reduce the cost spent by keyword set. Next, we propose a weight based algorithm to delete unimportant edges (also vertexes). Based on these two algorithms, the scale of knowledge graph could be effectively reduced. Next, we use an efficient algorithm to find similarity sub-graphs and a skyline based method to select high quality sub-graphs. Theoretical analysis and extensive experimental results demonstrate the effectiveness of the proposed algorithms.

References

1. Amer-Yahia, S., Roy, S.B., Chawlat, A., et al.: Group recommendation: semantics and efficiency. Proceedings of the Vldb Endowment **2**(1), 754–765 (2010)
2. Bar-Yehuda, R., Bendel, K., Freund, A., Rawitz, D.: Local ratio: a unified framework for approximation algorithms. ACM Comput. Surv. (CSUR) **36**(4), 422–463 (2004). 1935–2004
3. Chen, C., Zhang, D., Guo, B., Ma, X., Pan, G., Wu, Z.: TripPlanner: personalized trip planning leveraging heterogeneous crowdsourced digital footprints. IEEE Trans. Intell. Transp. Syst. (T-ITS) **16**, 1259–1273 (2014)
4. Du, R., Yu, Z., Mei, T., Wang, Z., Wang, Z., Guo, B.: Predicting activity attendance in event-based social networks: content, context and social influence. In: UbiComp 2014: Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing, pp. 425–434 (2014)

5. Feng, K., Cong, G., Bhowmick, S.S., Ma, S.: In search of influential event organizers in online social networks. In: SIGMOD 2014: Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data, pp. 63–74 (2014). https://doi.org/10.1145/2588555.2612173
6. Wang, H., et al.: RippleNet: propagating user preferences on the knowledge graph for recommender systems. In: SIGMOD 2018, pp. 417–426
7. Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge. In: SIGMOD 2008, pp. 1247–1250
8. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: a core of semantic knowledge. In: World Wide Web 2007, pp. 697–706
9. Huang, X., Zhang, J., Li, D., Li, P.: Knowledge graph embedding based question answering. In: Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, pp. 105–113. ACM (2019)
10. Belleau, F., Nolin, M.-A., Tourigny, N., Rigault, P., Morissette, J.: Bio2RDF: towards a mashup to build bioinformatics knowledge systems. knowledge systems. J. Biomed. Inform. **41**(5), 706–716 (2008)