Analysis And Evolution Of Comic Hero Networks

Julian Pascal Wittker*
University Osnabrück
jwittker@uni-osnabrueck.de

ABSTRACT

Knowledge graphs have received a lot of attention in several industries and was subject of a lot of research. Representing real-world networks, knowledge graphs offer a variety of possibilities like data and relationship visualization, analysis and evolution. In this paper concepts and methods for graph refinement, community detection and graph comparison are presented and applied to two datasets describing comic characters and their relationships. When running the concepts on the exemplary datasets different measurements are used and compared.

1. INTRODUCTION

1.1 Overview

In section 1 the concepts of knowledge graphs, graph refinement, community detection and answer set programming (ASP) are presented. Also the research questions for this paper are set up. Section 2 shows a summary of related work to this paper. In 3 the implementation of before mentioned concepts and algorithms is explained, together with the results compared to initial assumptions and expectations. At the end in section 4, the paper is concluded and further questions and ideas are raised.

1.2 Knowledge Graph

A graph $G = (V, E)$ is defined by a set of vertices *V* and a set of edges *E* with and edge $e := \{\{u, v\} : u, v \in V\}$ being defined by both its endpoints / nodes. The amount of vertices is denoted as $n = |V|$ and edges as $m = |E|$. A knowledge graph is a more specific graph type with attributes. Each vertex or node and each edge can have a set of attribute values describing a real-world entity. Types of attributed graphs are *node-attributed* and *edge-attributed*. For nodes, such real-world entities could be actual persons like scientific researchers. An edge could represent the fact that two researchers worked together for a paper or other publication. In this example the knowledge graph can be regarded as an collaboration network of scientific researchers. If such a knowledge graph is build it can not only be used for the representation of the network itself. It can offer the possibility to suggest e.g. a scientific researcher with which other person a collaboration would be interesting, resulting in an evolving network. Use cases for knowledge graphs can be retail, entertainment, finance, healthcare and more. Commonly known examples are DBPedia or Wikidata representing for example the linking structure of Wikipedia and the Google Knowledge Graph visualizing the search behavior of users [3].

1.3 Refinement and Completion

A knowledge graph can not only be seen as a static representation of knowledge. In [13] a variety of methods are presented of how a knowledge graph can be further evolved after its first creation. Obviously knowledge graphs can be edited, adjusted and completed by humans directly. But refining, evolving and completing a knowledge graph can also be done automatically with information within the graph, its original data or even from external data. In [13], several graph methods got categorized in *Completion vs. Error Detection*, *Refinement* and *Internal vs. External*. The first category includes the addition of missing or the identification of false data. *Refinement* can be seen as a sub category of *Completion*. Here methods like link prediction or linking of two or multiple knowledge graphs are located. The last category determines the source of information which is used to for example refine a graph.

In this paper I concentrate on *Refinement* methods, more specifically on link prediction, both internal and external. A knowledge graph can be evolved over the course of several steps. A graph G_s in step *s* can be moved into the next step G_{s+1} by applying one iteration of link prediction. The goal is to increase the number of edges using internal or external information based on several measures. Possible measures include several aspects of the knowledge graph. [1] lists *node features, neighborhood features* and *path features* as usable measure bases. The later used *common neighbors* measurement locates in the *neighborhood features*. For two not connected nodes the amount of common neighbors is counted. If the amount is over a certain threshold, a link is predicted. While being a fairly simple measurement, the *common neighborhood* can be a promising reference point when trying to predict for example a new interaction between two persons if they have common friends. Other measures are e.g. the *Jaccardi coefficient*, also taking the total number of neighbors into account, or the distance between two nodes. Obviously, the used measurement should depend on the dataset, the scenario and what should be accomplished or simulated. For more measurements and detailed explanations see [1].

It is important to note, that most link prediction methods only work with already existing graphs or networks. When looking at the *common neighborhood* it is easy to see, that a graph only consisting in nodes and no edges would not be refined or evolved by applying a link prediction with this measurement. The initial creation of connections is called *Cold Start Problem* [12]. This is an interesting problem because the non-existence of a network can be diverse. The probably simplest possibilities are that either no network structure exists in the first place or the structure is hidden for the user. Since no graph characteristics can be used to initially predict links, the nodes attributes have to be looked at. While [12] proposed the usage of a probabilistic graph, [10] proposed the usage of a bipartite graph based on the graphs attributes constructed in Answer Set Programming.

1.4 Community Detection

A community of a graph refers to a subset of nodes of that graph. More specifically a community describes a subgraph with a denser intrinsic structure and a less dense connection outside of the community. Each community should be distinguishable from other communities based on a certain characteristic. Community detection is the task of finding such communities which can represent certain interesting properties. Properties can be structural and then focus on dense graph structures or the quality of a community depends on the graphs attributes.

*This author is the one who did all the really hard work.

For communities only based on the graph structure, again, multiple measurements also called quality functions can be applied. The basis of these functions build the following numbers:

- $d(u)$ - the *degree* of a node *u* / number of neighbors of *u*
- $\bar{d}_c(u) := |\{\{u, v\} \in E : v \notin C\}|$ - the *out-degree* of a node *u* / number of edges from *u* that go outside of a community *C*
- $m_c := |\{\{u, v\} \in E : u, v \in C\}|$ - number of *intra-edges* / number of edges inside a community *C*
- $\bar{m}_c := |\{\{u, v\} \in E : |\{u, v\} \bigcap C| = 1\}|$ - the number of *inter-edges* / number of edges that go outside of a community *C*

Based on this, helpful local quality functions for a community *C* can be defined:

- $IAODF(C) := 1 - \frac{1}{n_c} \sum_{u \in C} \frac{\bar{d}_C(u)}{d(u)}$ - the *inverse average out-degree fraction*
- $SIDX(C) := 1 - \frac{\bar{m}_c n(n-1)}{2mn_c(n-n_c)}$ - the *segregation index*
- $MODL(C) := \frac{m_c}{m} - \sum_{u,v \in C} \frac{d(u)d(v)}{4m^2}$ - the *modularity contribution*

First community detection algorithms can be based on almost solely on those measures like the greedy modularity maximization approach. The problem of approaches which are only based on the graph structure is the interpretability. For a human it is difficult to pinpoint the meaning of a community and why a specific set of nodes were combined to a community.
On the other side are algorithms using the attributes of nodes to not only just detect communities but also directly describing them. Then a community is a collection of nodes which have a set of same attribute values and are densely connected, compared towards nodes outside of the community. Such community description can be very helpful in explaining the result of the algorithm. An example for such a descriptive community detection algorithm is COMODO presented in [5]. COMODO works with an exhaustive search using extended frequent pattern trees (FP-Trees [11], [7]). The result are the top *k* communities pruned by optimistic-estimate quality functions, which are based on the three mentioned local quality functions. In the beginning every attribute value is called a *basic pattern*. During the algorithm a *pattern p* for a community gets extended by possibly adding multiple *basic pattern*. The *pattern* then gets mapped to a value which indicates the interestingness of the *pattern*. The optimistic-estimate of a quality function is used for the branch&bound aspect of the algorithm as an upper bound of interestingness of a *pattern* to be able to only look at probably interesting patterns. For the evolution of communities, extended FP-trees in an adapted FP-growth manner are used. Adding graph information needed for the calculation of the quality functions to the frequency of a pattern in a FP-node allows for an efficient FP-tree mining.

1.5 Answer Set Programming

ASP programs are build by defining a set of facts and rules. A rule consists of a head, positive and test body: $H : -B_1, ..., B_n, C_1, ..., C_m$. When modeling the program various aggregations like `count` help defining strong rules. During the solving of the program multiple new facts are derived from the input facts and rules. Besides being an evaluation of the initial problem, the answer to a program can be used to explain the result itself. The model used in a program contains all facts and rules with parameters / atoms for which they were fulfilled. With reconstructive explanation methodologies ([16], [16]), the model and therefore the result can be explained. Since providing highly needed strong grounder and solver toolkits such as Clingo ([9], [2]) can be used to model and solve ASP programs.

1.6 Research Questions

Similar to [4], in this paper I try to compare different knowledge graphs with the help of different measures and algorithms. In contrast to [4] I want to compare the *Marvel Universe Social Network* with the *Super Hero* dataset since they both come from the same subject area. I propose the following two research questions:

- To what extent can those different datasets result, especially regarding their structure, in similar or comparable networks?
- To what extent can link prediction help creating a more detailed network out of the two datasets?

2. RELATED WORK

Knowledge Graphs are used in different scenarios. Not only can they represent real-world applications or situations, they can also be used as a database. In the already mentioned papers different scopes, techniques and concepts of knowledge graphs are presented. In addition to those proposals, in papers like [5], the proposed concepts are directly tested on exemplary datasets. Mostly datasets representing some type of social-networks are used and therefore likely show real-world properties. Interestingly the first dataset (*The Marvel Social Network*) was already used in a comparison with collaboration networks in [4]. It was stated that real-world properties of collaboration networks e.g. for scientific researchers show the following two properties among others:

- every pair of nodes is connected by a shortest path
- two nodes are more likely to be connected if they have a common neighbor

These two network characteristics build the *small-world* features. The second aspect supports the idea of using common neighbors as a measure for link prediction. For analysis aspects not appearing in this paper, I refer to [4].

3. EXPERIMENTS

3.1 Datasets

For the experiments two datasets dealing with possible networks of comic characters are used. The first dataset, *The Marvel Universe Social Network* [14], contains information about comic appearances of marvel characters and interactions between heroes, hence the name *Social Network*. The second dataset, *Super Hero Dataset (Comic Super Hero)* [15], contains comic characters from different universes and for each characters a variety of attributes are given (e.g. power stats, group-affiliations).
For the implementation, I chose the Python library *networkx* as graph representation. Not only does *networkx* use a fast dictionary-based data structure as basis, but it also provides a lot of useful measures and algorithms on graphs. Using measures like centralities or degree histograms allowed for a further graph comparison besides link prediction and community detection, see the following sections.

3.2 Cold Start

Regarding the creation of a knowledge graph for each of those datasets the link prediction problem called *Cold Start* was looked at in this paper. Using the Answer Set Programming approach presented in [10] it was possible to create networks based on comic appearances and attributes. Both graph creations benefit from a bipartite graph assumption. The comic-hero connections create a bipartite graph with heroes on one and comics on the other side. Describing this graph type as ASP in Clingo ([9], [2]) allows a graph creation based on common comic appearance of two characters. To gain more control about the resulting graph or network, the implemented function was given a parameter *number_common_neighbors*. This allows the user to determine how many common comic appearances two characters have to have in common and therefore gives the user more control

Common Comic Appearances	$\|E\|$
2	31210
5	5355
10	1114
20	233

Table 1: Comic Appearance Cold Start

Initial Edges/Common Neighbors	31210	5355	1114	233
2	174054	6489	278	6
5	15128	196	2	0
10	1141	15	0	0
20	138	2	0	0

Table 2: *Marvel* link prediction

over the size of the resulting graph. Especially the number of edges and the degree of nodes can be influenced. This influence was crucial for the analysis. When using one common comic appearance as a quality for a possible link prediction, either the Clingo model or the resulting graph surpassed my 7GB RAM. So the parameters of the link prediction can help reducing the size of the graph and represent the actual meaning of a predicted connection or relationship.

While the first dataset provided information of character connections either in an explicit manner or via the comic appearance, the second dataset only provides a list of individuals with a list of attributes. In [10] such attributes are used to create a bipartite graph in ASP connecting each individual node with a node for each attribute value. Then two characters are connected if they have a common neighbor in this bipartite graph, i.e. they have the same attribute value. The corresponding ASP program can be seen in 3.2

```
# a node for each character
node(character).
# a node for each attribute value
node_attr(attribute_value).

# one edge for each attribute value
# to a character
edge(character, attribute_value).

# the bipartite graph is undirected
edge(X,Y) :- edge(Y,X).

# X is a common neighbor of Y and Z
c_attr(X,Y,Z) :- edge(X,Y), edge(X,Z),
                 not edge(Y,Z), Y!=Z.

# predict link if no edge exists
# and the nodes have a common attribute value
cn_lp(Y,Z) :- node_attr(Y), node_attr(Z),
              not edge(Y,Z),
              1=#count{X:c_attr(X,Y,Z)}.
```

Only a selection of attributes was chosen to contribute in this procedure. Attributes like *powerstats_intelligence* or *appearance_gender* would also work with this program, but with the goal of creating a network representing an interaction network, such attributes would not provide a meaningful connection. The selection consists of *biography_publisher*, *work_base* and *connections_group-affiliation*. In theory these characteristics are most likely to result in a useful network and even may be comparable to the comic appearances from the first dataset. Note that *biography_publisher* is a very strong attribute in the sense that e.g. the value *Marvel Comics* results in a very dense connection for a lot of nodes. Unfortunately most attributes are not uniformly described. For example the *group-affiliation Avengers* and *formerly Avengers* does not yield in a connection since only exactly the same attribute values contribute to a link prediction. Keeping that in mind *biography_publisher* greatly helped in building a greater network. With this attribute 72094 edges were created while only 239 edges were created without it. Fortunately, [5] mentions that especially the *MODL* quality function works good on dense graphs. Besides link prediction, the addition of new nodes could also be seen as a refinement of a knowledge graph. The *Super Hero Dataset* also contains information about the *connection_relatives* of a character. I did not follow this idea for this paper, but one could create new nodes for each relative and connect them to each other. When using any community detection method such family clusters could be found in the final community structure.

3.3 Structured Link Prediction

Next to the cold start link prediction problem, I also used structured link prediction to refine and evolve the knowledge graphs. Using an adaptation of the earlier mentioned ASP programs, the structured link prediction counts the common neighbors of two not connected nodes. If the amount of common neighbors is over a certain user defined threshold, a link is predicted. Again the amount of predicted links depends on the previously created connections, see table 2.

3.4 External Link Prediction

Again as an adaptation of the original ASP program from [10], I tried to apply external refinement to one of the graphs. Since the *Super Hero* graph is significantly smaller both edge- and node-wise it made the most sense to try to predict links in that graph. More specifically, I tried to write a program, which adds links to the *Super Hero* graph based on connections in the *Marvel* graph. In Clingo it is possible to define a python function which can be called inside an ASP program. The function *is_similar_node(X,Y)* takes to symbols from the program which represent a node from the target and the source graph and very simply compares them. For that a new fact and a new rule are added to the program:

```
# fact used to compare result of is_similar_node
check(1).

# rule using is_similar_node
ext_lp(X,Y) :- target_node(X), target_node(Y),
    not target_edge(X,Y),
    1<=#count{W,V:source_node(W),
        source_node(V), source_edge(W,V),
        check(@is_similar_node(X,W)),
        check(@is_similar_node(Y,V))}.
```

This method could be improved with better node comparison or preprocessing which further aligns the datasets. Without such improvements with 2 common comic appearances in the *Marvel* graph and not using the *biography_publisher* attribute, to increase the probability of increasing links to test the method itself, 7277 edges got added by this procedure, proving that external refinement measures works even if the datasets are not uniformly named or no special comparing mechanisms are in place. Note that probably due to the high amount of nodes in both graphs (*Marvel*: 6439, *Super Heroes*: 712) the program did take a long time to finish (over an hour).

3.5 Community Detection

For the choice of community detection method the datasets have to be taken into account. The *Marvel Universe Social Network* does not provide any attributes for the nodes and therefore no method can be used which relays and results in a description of a community and its nodes. Since the second knowledge graph was already created using the characters attributes, a descriptive community detection method like COMODO [5] could be used. As described earlier, COMODO is based on a FP-tree and therefore on a FP-growth algorithm. For the implementation, code from [7] and [8] was used and slightly adapted. In [5], the adaptation not only included the adaption of the tree data structure but also the analysis of a community during the algorithm. Also important to note is the selection of attribute values which should contribute to a community choice and description. All

three attributes from the cold start problem were used again and I added the *biography_alignment* to the selected attributes. The distinction between *good* and *bad* could result in more specific communities especially against the background of using *biography_publisher* as a cold start characteristic. Further, the paper [5] states, as a result from experiments on different exemplary data, that the *MODL* quality function works best in most cases, especially for dense graphs. Due to the introduction of the *biography_publisher* attribute, the *Super Hero* graph became dense enough to use the *MODL* as quality function as well, see Section 3.6. Unfortunately, even with using the FP-tree implementation from [8] the adaptation of said data structure into a *community pattern tree* posed a to big task for me. Therefore the COMODO algorithm could not be evaluated with the *Super Hero* dataset.
Because the modularity can be used in the COMODO algorithm, regarding the first dataset, I used greedy modularity maximization as community detection, which is also provided by the *networkx* library. The results can be seen in figures 2. The graphs are drawn with *networkx's spring_layout* which pulls nodes that are connected together and pushes not-connected nodes away from each other (*Fruchterman-Reingold force-directed algorithm*). Since the nodes in both figures are close together it can be assumed that they are strongly connected, even if the modularity maximization found several communities. Also note that the increase in both the common comic appearances and the common neighbors helped reduce the size of the graph. Since the COMODO community detection could not be finished, I also applied this community detection method to the *Super Hero* dataset, see figure 2. The result suggests that the communities are highly influenced by the *publisher* attribute which suggests that COMODO would result in a comparable community structure. Note that I removed all not-connected nodes for both graphs in the community visualization.

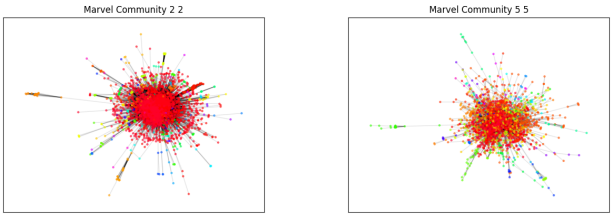

Figure 1: Marvel Communities with 2 (5) common comic appearances per hero and one step of structured link prediction with 2 (5) common neighbors as measurement. Each color refers to a community

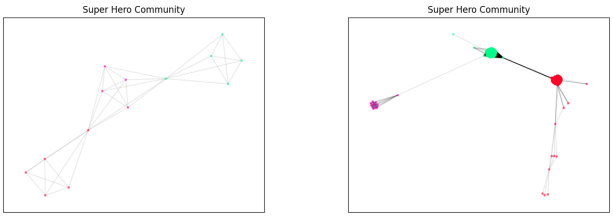

Figure 2: Marvel Communities with one step of structured link prediction with 2 common neighbors as measurement. Left is without the *publisher* attribute, the right is with this attribute. Each color refers to a community

3.6 Graph Comparison

After every step of graph refinement, I compared the two knowledge graphs with a variety of measures.
Simple measurements of a graph are the *degree* of a node and the *clustering coefficients* of a node. The later is the fraction of *number of links between neighbors of a node* over *the number of potential links between neighbors of a node*. In figures 3, similar to the figures 2, the high clustering coefficients show a high density of the marvel graph. The *Super Hero* graph on the other hand shows a big difference depending on whether the *biography_publisher* attribute is used in the attributed cold start or not, see figures **??** and **??**.

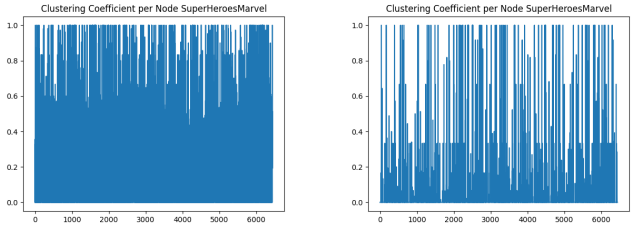

Figure 3: Marvel clustering coefficients with 2 (5) common comic appearances per hero and one step of structured link prediction with 2 (5) common neighbors as measurement.

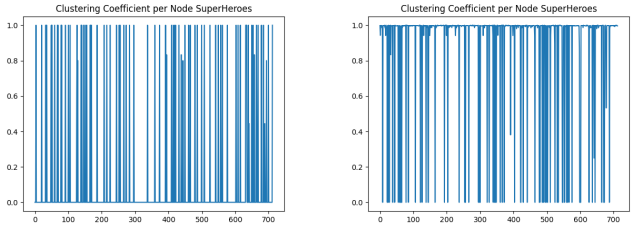

Figure 4: Super Hero degree histogram with (left) and without (right) the attribute *biography_publisher* and one step of structured link prediction with 2 common neighbors as measurement.

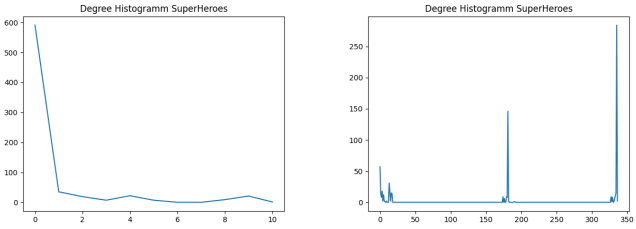

Figure 5: Super Hero degree histogram with (left) and without (right) the attribute *biography_publisher* and one step of structured link prediction with 2 common neighbors as measurement.

By comparing the centralities of nodes it is possible to figure out which node is the most interesting in a corresponding network. In the scope of the used datasets, an interesting node regarding the node centrality can also be seen as an important node to the real-world network. The importance of a hero could be both, having a lot of companions and/or a lot of enemies, which both are listed in the datasets. In figures 7 and 6, the *betweenness* and *closeness* centralities are shown. *Betweenness* $bet(v) := \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}$ - with σ_{st} being the number of shortest paths between s and t and $\sigma_{st}(v)$ being the number of shortest paths between s and t that go through v - denotes the number of shortest paths of all node pairs that go through a specific node. *Closeness* is the length of these shortest paths and the shorter the path is, the higher the node ranks.

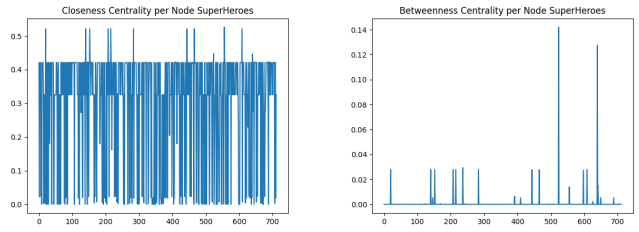

Figure 6: Super Hero Centralities with 2 common neighbors for link prediction and using the *biography_publisher* attribute.

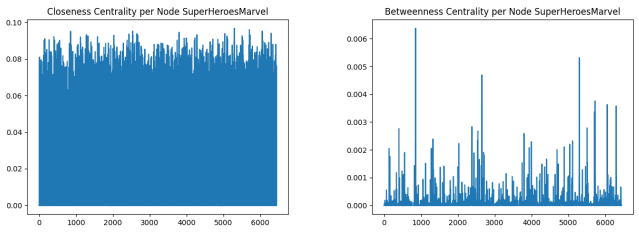

Figure 7: Marvel Centralities with 5 common comic appearances and 5 common neighbors for link prediction.

4. CONCLUSION

A comparison of the two datasets with help of their corresponding knowledge graph has proven to be difficult. The *Marvel* dataset only included characters from the *Marvel Comics* and provided no attributes for the characters. That is why this graph could only be compared with measures concentrating on the graph structure alone. On the other hand, the *Super Hero* dataset contained attributes not named uniformly enough to properly apply attributed link prediction resulting in a poor graph structure.

Interestingly, the structured link prediction approach showed great potential not only in evolving the knowledge graph. The memory problem showed that it could be possible to use a combination of graph reduction techniques and link prediction to save a small version of knowledge graphs for memory efficiency. Link prediction then would generate the complete structure of the graph. I did not find any paper regarding this, probably because knowledge graphs do not have to be send over the internet for example.

Unfortunately the implementation of the COMODO algorithm posed a too difficult task because of the specific data structure adaptation. The algorithm itself is pretty straightforward. A comparison with other descriptive community detection algorithms like MinerLSD [6] comes to mind.

5. REFERENCES

[1]

[2] Clingo api. https://potassco.org/clingo/python-api/5.4/, 2021.

[3] Use cases of knowledge graphs. https://www.ibm.com/cloud/learn/knowledge-graphtoc-use-cases–$_iXkAhCl$, 2022.

[4] ALBERICH, R., MIRO-JULIA, J., AND ROSSELLÓ, F. Marvel universe looks almost like a real social network. *arXiv preprint cond-mat/0202174* (2002).

[5] ATZMUELLER, M., DOERFEL, S., AND MITZLAFF, F. Description-oriented community detection using exhaustive subgroup discovery. *Information Sciences 329* (2016), 965–984.

[6] ATZMUELLER, M., SOLDANO, H., SANTINI, G., AND BOUTHINON, D. Minerlsd: efficient mining of local patterns on attributed networks. *Applied Network Science 4*, 1 (2019), 1–33.

[7] CHONYY. Fp growth: Frequent pattern generation in data mining with python implementation. https://towardsdatascience.com/fp-growth-frequent-pattern-generation-in-data-mining-with-python-implementation-244e561ab1c3, 2022.

[8] CHONYY. Fp-growth github. https://github.com/chonyy/fpgrowth$_py$, 2022.

[9] GEBSER, M., KAMINSKI, R., KAUFMANN, B., AND SCHAUB, T. Clingo= asp+ control: Preliminary report. *arXiv preprint arXiv:1405.3694* (2014).

[10] GÜVEN, Ç., AND ATZMUELLER, M. Applying answer set programming for knowledge-based link prediction on social interaction networks. *Frontiers in big Data 2* (2019), 15.

[11] HAN, J., PEI, J., AND YIN, Y. Mining frequent patterns without candidate generation. *ACM sigmod record 29*, 2 (2000), 1–12.

[12] LEROY, V., CAMBAZOGLU, B. B., AND BONCHI, F. Cold start link prediction. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining* (2010), pp. 393–402.

[13] PAULHEIM, H. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic web 8*, 3 (2017), 489–508.

[14] SANHUEZA, C. The marvel universe social network. https://www.kaggle.com/csanhueza/the-marvel-universe-social-network, 2021.

[15] VERMA, A. Super hero dataset (comic super hero). https://www.kaggle.com/aakashverma8900/superhero-api-dataset, 2021.

[16] WICK, M. R., AND THOMPSON, W. B. Reconstructive expert system explanation. *Artificial Intelligence 54*, 1-2 (1992), 33–70.