Distributed Holistic Clustering on Linked Data

Markus Nentwig(✉), Anika Groß, Maximilian Möller, and Erhard Rahm

Database Group, University of Leipzig, Leipzig, Germany
{nentwig,gross,rahm}@informatik.uni-leipzig.de,
m.moeller@studserv.uni-leipzig.de

Abstract. Link discovery is an active field of research to support data integration in the Web of Data. Due to the huge size and number of available data sources, efficient and effective link discovery is a very challenging task. Common pairwise link discovery approaches do not scale to many sources with very large entity sets. We propose a distributed holistic approach to link many data sources based on a clustering of entities that represent the same real-world object. Our approach provides a compact and fused representation of entities, and can identify errors in existing links as well as many new links. We support distributed execution, show scalability for large real-world data sets and evaluate our methods with respect to effectiveness and efficiency for two domains.

1 Introduction

Linking entities from various sources and domains is one of the crucial steps to support data integration in the Web of Data. A manual generation of links is very time-consuming and nearly infeasible for the large number of existing entities and data sources. As a consequence, there has been much research effort to develop link discovery (LD) frameworks [10] for automatic link generation. Platforms like `datahub.io` and `sameas.org` or repositories such as LinkLion [11] collect and provide large sets of links between numerous different knowledge sources. They can be reused to avoid an expensive re-determination of the links. It is particularly complex to ensure high link quality, i.e., the generation of correct and complete link sets. Existing link repositories cover only a small number of inter-source mappings and automatically generated links can be erroneous in many cases [2]. Despite the huge number of sources to be linked, most LD tools focus on a pairwise (binary) linking of sources. However, LD approaches need to scale for n-ary linking tasks as well as for an increasing number of entities and sources that are added to the Web of Data over time [13].

To address these shortcomings we recently proposed an approach to cluster linked data entities from multiple data sources into a holistic representation with unified properties [8]. The method combines entities that refer to the same real world object in one compact cluster instead of maintaining a high number of binary links for k sources. The approach is based on existing `owl:sameAs` links and can deal with entities of different semantic types as they occur in many sources (e. g., for geographical datasets, countries, cities, lakes). Input links are checked for consistency and new links (e. g., for previously unconnected sources) are identified.

Considering the huge size and number of sources to be linked, scalability becomes a major issue. Linking and clustering approaches usually comprise complex operations such as similarity computations to identify similar entities or clusters. These complex work steps can often be parallelized in a distributed environment in order to reduce execution time significantly. Big Data frameworks like Apache Spark or Apache Flink [1] provide execution engines to process very large datasets in a distributed environment. With regard to the ever increasing amount of data that needs to be linked and integrated in typical big data processing workflows, it is essential to develop scalable solutions for link discovery and holistic entity clustering.

Herein we study a distributed holistic clustering approach. In contrast to the previous work, we support blocking strategies to reduce unnecessary comparisons and present a comprehensive evaluation for quality and efficiency on real-world data for different domains. An extended version with more details on Flink implementation and creation of a reference dataset can be found in [9]. We make the following contributions:

- We present a distributed holistic clustering approach for linked data to enable an effective and efficient clustering of large entity sets from many data sources.
- We evaluate the efficiency and effectiveness of the distributed holistic clustering for very large datasets with millions of entities from two domains.

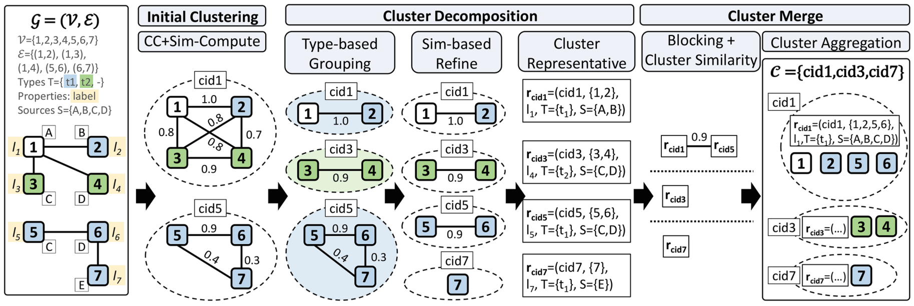
We present the implementation of the distributed holistic clustering in Sect. 2. Then, we show evaluation results in Sect. 3. Finally, we discuss related work in Sect. 4 and conclude in Sect. 5.

2 Distributed Holistic Clustering Approach

In this section we outline the workflow and implementation for our distributed holistic clustering approach based on the big data stream and batch processing system Apache Flink [1]. Starting with a introduction to Apache Flink (Sect. 2.1), we present the transformation and adaptation of the holistic clustering approach towards a distributed processing workflow (Sect. 2.2).

2.1 Apache Flink and Gelly API

Apache Flink's batch processing provides the DataSet API and well-known dataset transformations like *filter*, *join*, *union*, *group-by* or *aggregations* (relational databases) and *map*, *flat-map* and *reduce* (MapReduce paradigm). Special in-memory, distributed data structures called DataSets store data within Flink programs. DataSets can be manipulated based on so called transformations that return a new DataSet. Some transformation operations make use of user-defined functions (UDFs) and allow for customized definitions how DataSet values need to be changed. We make use of the graph processing library (Gelly) in our



$\mathcal{G} = (\mathcal{V}, \mathcal{E})$
$\mathcal{V}$={1,2,3,4,5,6,7}
$\mathcal{E}$={(1,2), (1,3), (1,4), (5,6), (6,7)}
Types T={ t1, t2, -}
Properties: label
Sources S={A,B,C,D}
Initial Clustering
CC+Sim-Compute
Cluster Decomposition
Type-based Grouping
Sim-based Refine
Cluster Representative
Cluster Merge
Blocking + Cluster Similarity
Cluster Aggregation
$\mathcal{C}$ ={cid1,cid3,cid7}
r_{cid1}=(cid1, {1,2}, l_1, T={t_1}, S={A,B})
r_{cid3}=(cid3, {3,4}, l_4, T={t_2}, S={C,D})
r_{cid5}=(cid5, {5,6}, l_5, T={t_1}, S={C,D})
r_{cid7}=(cid7, {7}, l_7, T={t_1}, S={E})
r_{cid1}=(cid1, {1,2,5,6}, l_1,T={t_1}, S={A,B,C,D})

Fig. 1. Example clustering workflow.

holistic clustering workflow. In particular, we employ Gelly graphs containing a `DataSet<Vertex<K, VV>> vertices` and a `DataSet<Edge<K, EV>> edges`.

The complex data types `Vertex` and `Edge` are inherited from the Flink `Tuple` classes `Tuple2<K,VV>` (type `K` as vertex id, `VV` as vertex value), `Tuple3<K,K,EV>` (source vertex id, target vertex id (each type `K`) and `EV` as edge value), respectively. Operators like *join*, *filter* or *group-by* rely on tuple positions (starting from 0), e. g.,

```
vertices.join(edges).where(0).equalTo(1)
    .with((vertex, edge) -> new Tuple1<>(edge.getSimilarity))
    .filter(tuple -> tuple.f0 >= 0.9);
```

will join all edges with the vertices where the vertex id (position 0 in `vertices`) equals the target id of the edge (position 1 in `edges`) and returns the similarity value if the accompanied filter function is evaluated and returns true.

Besides the used graph data model we benefit from Flink's and Gelly's abstract graph processing operators like graph neighborhood aggregations or abstracted models for iterative computations. In particular, we will make use of the Flink delta iteration in different variations as discussed in the following sections.

2.2 Distributed Holistic Clustering

In this section, we will discuss the transformation and adaptation of the holistic clustering workflow towards a distributed processing workflow in Apache Flink. From a high-level perspective, we read input entities and links into a Gelly graph $\mathcal{G}$ with vertices $\mathcal{V}$ and edges $\mathcal{E}$ and apply a set of transformation operators to generate entity clusters $\mathcal{C}$. We illustrate the workflow steps using the running example in Fig. 1. There are six input edges $\mathcal{E}$ having optional similarity values and seven input vertices $\mathcal{V}$ further described by a label (l_1, l_2, ...), the originating data source S and colored dependent on their semantic type ($t1$, $t2$ or no type).

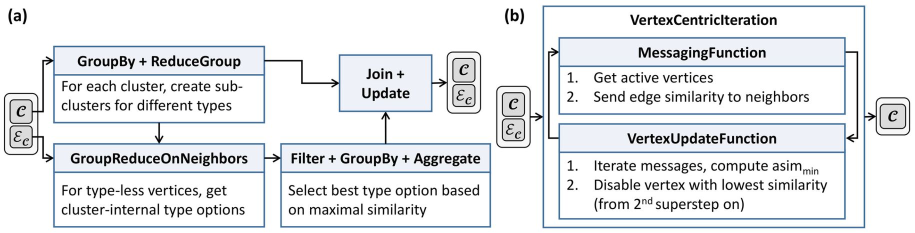

(a)
GroupBy + ReduceGroup
For each cluster, create sub-clusters for different types
GroupReduceOnNeighbors
For type-less vertices, get cluster-internal type options
Join + Update
Filter + GroupBy + Aggregate
Select best type option based on maximal similarity
(b)
VertexCentricIteration
MessagingFunction
1. Get active vertices
2. Send edge similarity to neighbors
VertexUpdateFunction
1. Iterate messages, compute asim_min
2. Disable vertex with lowest similarity (from 2nd superstep on)

Fig. 2. Sub-workflows with operators for type-based grouping (a) and similarity-based refinement (b).

Preprocessing. During preprocessing we apply several user-defined functions on the input graph, e. g., to harmonize semantic type information, remove inconsistent edges and vertices and normalize the label property value. First, we compute similarities only for given input edges based on vertex property values. For each vertex, we carry out a consistency validation using grouping on adjacent vertices and associated edges, and remove neighbors with equal data sources (details in [8]). We omit the preprocessing in the example (Fig. 1) and directly start with the preprocessed input graph $\mathcal{G}$.

Initial Clustering. To determine initial clusters, we determine the connected components (CC) within $\mathcal{G}$ and assign a cluster id to each vertex. In the example, vertices 1-4 obtain cluster ids *cid1* and vertices 5-7 *cid5*. Intra-cluster edges are then generated within each cluster accompanied by a similarity computation based on properties such as a linguistic similarity on labels or normalized geographical distance.

Cluster Decomposition. *Type-based grouping* is the first part of the decomposition to split clusters into sub-components dependent on the compatibility of semantic types. Figure 2a shows the sequence of applied transformations and short descriptions. Within clusters, a ReduceGroup function assigns new cluster ids based on semantic types, e. g., in the example vertex 3 and 4 are separated from vertex 2. Vertices without type (like vertex 1) require a special handling. We apply GroupReduceOnNeighbors (a Gelly CoGroup function to handle neighboring vertices and edges) to produce tuples for vertices with missing semantic type, e. g., vertex 1 creates a `(id,sim,type,cid)` for each outgoing edge $((1, 2), (1, 3), (1, 4))$, namely `(1, 1.0, t_1,cid1)` for edge $(1, 2)$ and `(1, 0.8, t_2, cid2)` for edge $(1, 3)$ and $(1, 4)$. Grouping on the vertex id executes an aggregation function for each group to return the tuple with the highest similarity per vertex, which is `(1, 1.0, t_1, cid1)` for vertex 1, processed vertices update their cluster id accordingly (e. g., vertex 1 $\rightarrow$ *cid*1). The result of the type-based grouping is a set of clusters with intra-cluster edges.

Similarity-based Refinement. We further decompose clusters by removing non-similar entities from their cluster. We use a Gelly vertex-centric iteration using

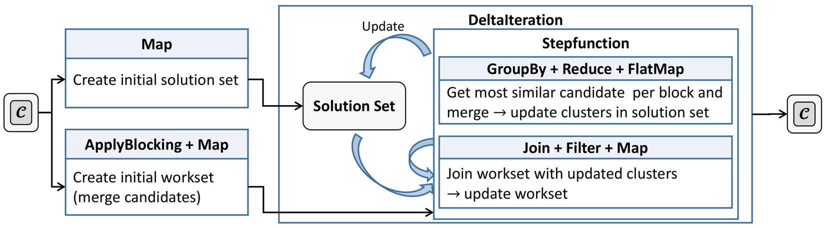

Map
Create initial solution set
ApplyBlocking + Map
Create initial workset (merge candidates)
Update
Solution Set
DeltaIteration
Stepfunction
GroupBy + Reduce + FlatMap
Get most similar candidate per block and merge → update clusters in solution set
Join + Filter + Map
Join workset with updated clusters → update workset

Fig. 3. Sequence of transformations for the cluster merge using Flink DeltaIteration

the core idea to iterate between a custom MessagingFunction and a VertexUpdateFunction (see Fig. 2b for details). In the first round, all vertices are active and send messages to all their neighbors. Messages are tuples containing the originating id, the edge similarity and an average edge similarity *asim* over all incoming messages (0 in the first iteration). Starting with the second iteration, we illustrate the sent messages for vertex 7 in cluster *cid*5 in our example: vertex 5 sends `(5, 0.4, 0.65)` to 7, vertex 6 sends `(6, 0.3, 0.6)` to 7 and vertex 7 sends messages to 5 and 6, resulting in $asim = (0.4 + 0.3)/2 = 0.35$ for vertex 7. Now in each cluster the vertex with the lowest *asim* will be deactivated (and is therefore excluded from the cluster) given that this *asim* is below a certain similarity threshold. In Fig. 1, vertex 7 will be deactivated and isolated into cluster *cid*7. Vertices send only messages if they are updated and deactivated vertices never send messages again, therefore, iteration termination is guaranteed.

Finally, we create a unified *Cluster Representative* for each cluster based on contained entities. Aggregation of property values is used for covered data sources and semantic types as well as selection of best label or geographic coordinates, see Fig. 1.

Cluster Merge. The main operators for the merge phase are sketched in Fig. 3. The implemented DeltaIterate function iteratively combines highly similar clusters into larger ones. To avoid the quadratic complexity comparing all clusters, we employ blocking strategies to avoid unnecessary comparisons, e. g., standard blocking on properties like label, not comparing representatives with incompatible semantic type and check for already covered data sources. In our example in Fig. 1 three blocks are created by applying blocking strategies. Only r_{cid1} and r_{cid5} need to be compared, such that a triplet $(r_{cid1}, 0.9, r_{cid5})$ is created as a merge candidate.

The delta iteration starts with an initial solution set containing the previously determined clusters and an initial workset (merge candidates) as seen in Fig. 3. Each iteration updates the workset applying a custom step function to generate changes for the solution set. In detail, for each block the merge candidate with the highest similarity is selected using a custom Reduce function. For our running example, $(r_{cid1}, 0.9, r_{cid5})$ is the best candidate and is merged using a custom FlatMap function. The new cluster r_{cid1} contains combined values for properties

S, T and l (see Fig. 1). This will directly affect the cluster representatives in the solution set, and the already merged cluster r_{cid5} will be deactivated in the solution set. Merge candidates in the workset are adapted based on changed clusters within the iteration step (see Fig. 3 solution set). Again, triplets are discarded if the data sources for the participating clusters overlap or exceed the maximum possible number of covered sources.

The delta iteration ends when the workset is empty (true for our running example after the first iteration). Note that parts of the dataset will converge faster to a solution, when clusters can not be merged anymore. These parts will not be recomputed in following iterations, such that only smaller parts of the data will be handled.

3 Evaluation

In the following we evaluate our distributed holistic clustering approach w.r.t. effectiveness and efficiency for datasets from the geographic and music domains. We first describe details of the used datasets (Sect. 3.1). We then evaluate the effectiveness and efficiency of our approach (Sect. 3.2).

3.1 Datasets

We use five datasets of different sources from the music and geographic domains (Table 1). Datasets DS1 and DS3 are used to evaluate the quality of entity clusters generated by the distributed holistic clustering while DS2, DS4 and DS5 are used to analyze the efficiency and scalability (see Sect. 3.2).

We use two datasets (DS1, DS2) from the *Geographic Domain*, covering entities from the data sources DBpedia, GeoNames, NY Times, Freebase for DS1 and additionally LinkedGeoData for DS2. Entities for both datasets have been enriched with properties like entity label, semantic type and geographic coordinates by using SPARQL endpoints or REST APIs. DS1 is based on a subset of existing links provided by the OAEI 2011 Instance Matching Benchmark[1]. For DS1, clusters and links have been manually checked and create a novel reference dataset for multi-source clustering [9]. We provide this dataset covering

Table 1. Overview of evaluation datasets. Number of resulting clusters and deduced correct links are given for reference datasets.

Domain	Entity properties	Dataset	#entities	#sources	#correct links	#clusters
Geography	label, semantic type	DS1	3,054	4	4,391	820
	longitude, latitude	DS2	1,537,243	5	-	-
Music	artist, title, album	DS3	19,375	5	16,250	10,000
	year, length, language	DS4	1,937,500	5	1,624,503	1,000,000
	number	DS5	19,375,000	5	16,242,849	10,000,000

[1] http://oaei.ontologymatching.org/2011/instance/.

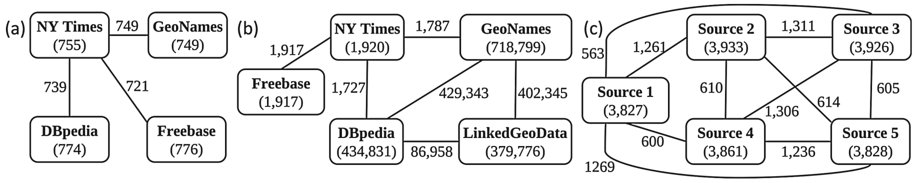


Fig. 4. Dataset structure for DS1 (a), DS2 (b) and DS3 (c) with number of entities and links.

the input dataset and the perfect cluster result as JSON files[2]. Dataset DS2 (Fig. 4b) originates from the link repository LinkLion [11]. We reuse about 1 Mio existing `owl:sameAs` links from LinkLion as input for the holistic clustering. However, there is no reference dataset available to evaluate the quality of created clusters for dataset DS2. We use DS2 to evaluate the scalability of our approach for very large entity sets.

For the *Music Domain*, we use the publicly available Musicbrainz dataset covering artificially adapted entities to represent entities from five different data sources [4]. Every entry in the input dataset represents an audio recording and has properties like title, artist, album, year, language and length. The property values have been partially modified and omitted to generate a certain degree of unclean data and duplicate entities that need to be identified. Beside a set of artificially created duplicates, each dataset covers cluster ids from which links between entities, that refer to the same object, can be easily derived. DS3 will be used for quality evaluation, while DS4 and DS5 are used to analyze the scalability of the distributed holistic clustering.

3.2 Experimental Results

We now present evaluation results w.r.t. the quality of the determined clusters as well as the scalability of the distributed holistic clustering for the five datasets DS1-DS5.

Setup and Configurations. The experiments are carried out on a cluster with 16 workers (Intel Xeon E5-2430 6x 2.5 GHz, 48 GB RAM) operating on OpenSUSE 13.2 using Hadoop 2.6.0 and Flink 1.1.2. All experiments are carried out three times to determine the average execution time.

We created input links for DS1 using three different configurations (confs) - computing similarities based on JaroWinkler on the entity label; confs 2 and 3 additionally compute a normalized geographic distance similarity below a maximum distance of 1358 km. Conf 1 applies a minimal similarity threshold of 0.9 for labels while confs 2 and 3 apply threshold 0.85 and 0.9 for the average label and geographic similarity, respectively.

[2] https://dbs.uni-leipzig.de/research/projects/linkdiscovery.

Table 2. Evaluation of cluster quality for geography dataset DS1 w.r.t. precision (P), recall (R) and F-measure (F1).

	Config 1			Config 2			Config 3		
	P	R	F1	P	R	F1	P	R	F1
Input links	0.933	0.806	**0.865**	0.964	0.938	0.951	0.981	0.799	0.881
Best (star1, star2)	0.863	0.844	0.853	0.963	0.941	**0.952**	0.951	0.838	0.891
Holistic	0.903	0.824	0.862	0.913	0.919	0.916	0.968	0.836	**0.897**

For the music dataset *DS3* we created input links using a soft TF/IDF implementation weighted on title (0.6), artist (0.3) and album (0.1) with a threshold of 0.35. *DS4* and *DS5* are used to show scalability, we simply create edges based on the cluster id from the perfect result by linking the first entity of each cluster with all its neighbors.

Quality. We analyze the achieved cluster quality for all datasets based on precision, recall and F-measure. The input links DS1 (see Fig. 4 a) are manually curated, therefore, they achieve a precision of 100%. However, missing links between lead to a recall of only 50%, resp. F-measure of 66.7%. With the holistic clustering approach, we achieve very good results w.r.t. recall (97.1%) while preserving a good precision (99.8%) resulting in the F-measure of 98.5%. This shows that we produce high-quality clusters based on existing input links thereby finding many new links.

However, as input mappings are not perfect in real-world situations, we used automatically generated input links ations (config 1-3) as described above. To evaluate the cluster quality, we further compare our results with the best configurations recently published results in [14]. Star clustering creates overlapping clusters, thus clusters may contain duplicates. Besides, star clustering does not create a compact cluster representation. Table 2 shows results w.r.t. the cluster quality for the computed input links, the best result of [14] and our approach. The distributed holistic clustering nearly retains the input link quality for config 1, while best(star1, star2) achieves slightly worst results. For config 2, the star2 implementation achieves a slightly better F-measure compared to the input mapping. For config 3 the holistic clustering improves the quality of the input mapping by 1.6% w.r.t. F-measure.

For the music domain, we evaluate the cluster quality for DS3 using a set of computed input links (see setup in Sect. 3.2). Overall, the quality of the input links is lower than for DS1. Due to strongly corrupted entities and more properties, DS3 is more difficult to handle. Applying the holistic clustering, we identify a quality improvement for both precision (89.0%) and recall (86.1%), resulting in a significant increase of F-measure by approx. 7% to 87.6% showing that our approach is able to handle such unclean data.

	#workers	pre	dec	merge	total
DS2	1	312	668	351	1331
	2	164	367	268	799
	4	79	231	207	518
	8	45	130	186	361
	16	23	42	162	227
DS4	1	423	419	608	1450
	2	224	236	417	876
	4	121	123	301	545
	8	62	73	238	372
	16	40	35	237	312

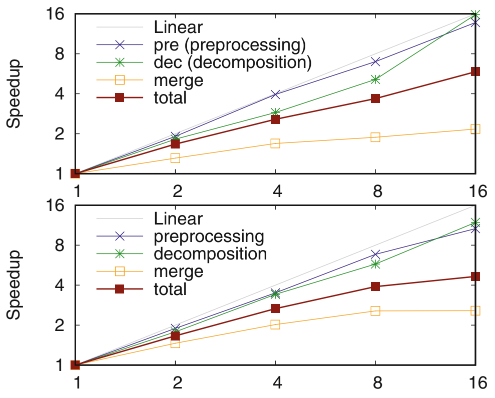


Fig. 5. DS2 and DS4 execution times (left) in seconds and speedup (DS2 top right, DS4 bottom right) for the single workflow phases and total workflow.

Overall, the holistic approach achieves competitive results although the DS1 dataset facilitates achieving relatively good input mappings making it difficult for any clustering approach to find additional or incorrect links.

Scalability. To evaluate the distributed holistic clustering w.r.t. efficiency and scalability, we determine the absolute execution times as well as the speedup for the very large geographic (DS2) and music datasets (DS4, DS5). To show scalability, we vary the number of Flink workers and use a parallelism equal to the number of workers.

Figure 5 show the achieved execution times for DS2 and DS4 for different phases of the clustering workflow and the overall workflow execution time. For each phase, an increased number of workers leads to improved execution times. The best improvement can be achieved for the preprocessing (pre) and decomposition (dec). The merge phase is by far more complex. While preprocessing and decomposition operate within connected components and clusters, the merge phase attempts to combine similar clusters based on the assignment in the blocking step and therefore can suffer from data skew problems for some blocks. These effects become also clear in Fig. 5 showing the speedup results compared to the linear optimum. Preprocessing and decomposition achieve nearly linear speedup, while the merge phase shows decreased speedup values. In total, we achieve a good speedup of 5.86 for the large geographic dataset DS2 and 4.65 for the large music dataset DS4. For the largest dataset DS5, we could determine results for two configurations: 8 workers could finish the complex task in 43,589 seconds, 16 workers finished after 24,722 seconds (reduced by factor ≈1.8).

Overall, the distributed holistic clustering achieves good execution times and moderate scalability results for very large entity sets. The approach is scalable for different data sources and employs a multi-source clustering instead of basic binary linking of two sources. The distributed implementation further allows to scale for a growing number of entities and data sources and is very useful for complex data integration scenarios in big data processing workflows.

4 Related Work

Link discovery (LD) has been widely investigated and there are many approaches and prototypes available as surveyed in [10]. Typical LD approaches apply binary linking methods for matching two data sources but lack efficient and effective methods for integrating entities from k different data sources to provide a holistic view for linked data. Some approaches enable distributed link discovery or for matching two data sources, e. g., Silk [6] and Limes [5] realized LD approaches based on MapReduce before distributed data processing frameworks like Spark or Flink became state of the art, therefore they are suffering from limitations of MapReduce like repeated data materialization and lack of iterations. They further focus on pairwise matching and do not support reuse of existing links sets.

While LD is driven by pairwise linking of data sources, support for multiple data sources can be found in related research areas. In [3] ontology concepts from multiple data sources are clustered based on topic forests for extracted keywords from concepts and their descriptions to determine matching concepts within groups of similar topics. In [7] a maximum-weighted graph matching and structural similarity computations are applied to concepts of multiple ontologies to find high quality alignments. However, these holistic ontology matching approaches do not focus on clustering and have limitations w.r.t. scalability for LD.

There are few LD approaches for linked data on multiple sources. Thalhammer et al. [15] present a pipeline for web data fusion using multiple data sources applying hierarchical clustering. The unsupervised LD approach Colibri [12] considers error detection for LD in multiple knowledge bases based on the transitivity, while clustering of entities is not the main focus. Both approaches do not realize distributed execution and have not been evaluated w.r.t. scalability. The work in [14] considers the implementation of existing clustering algorithms on top of Apache Flink for entity resolution of several data sources. The approach does not handle incorrect links or semantic type information and does not create a compact cluster representation.

5 Conclusion

We presented a distributed holistic clustering workflow for linked data using the distributed data processing framework Apache Flink using dataset transformations and user-specific Flink operators. Our approach is based on the reuse of existing links and is able to handle entities from various data sources. We presented comprehensive evaluation results for datasets from two domains with up to 20 million entities showing that the proposed approach can achieve a very high cluster quality. In particular, we were able to find many new correct links and could remove wrong links. The distributed execution in a parallel cluster environment resulted in very good execution times for the considered dataset sizes and good overall scalability results.

For future work, we plan to further improve the scalability of our approach, e. g., by realizing sophisticated blocking and load balancing methods for the complex cluster merge phase. We further plan the development and combination with an incremental clustering to support the addition of new entities and data sources, particularly to address the ongoing growth of the Web of Data.

Acknowledgments. This research was supported by the Deutsche Forschungsgemeinschaft (DFG) grant number RA 497/19-2.

References

1. Carbone, P., Katsifodimos, A., Ewen, S., Markl, V., Haridi, S., Tzoumas, K.: Apache flink^{TM}: stream and batch processing in a single engine. IEEE Data Eng. Bull. **38**(4), 28–38 (2015)
2. Faria, D., Jiménez-Ruiz, E., Pesquita, C., Santos, E., Couto, F.M.: Towards annotating potential incoherences in bioportal mappings. In: ISWC, pp. 17–32 (2014). doi:10.1007/978-3-319-11915-1_2
3. Grütze, T., Böhm, C., Naumann, F.: Holistic and scalable ontology alignment for linked open data. In: WWW2012 Workshop on Linked Data on the Web (2012)
4. Hildebrandt, K., Panse, F., Wilcke, N., Ritter, N.: Large-Scale data pollution with apache spark. IEEE Trans. Big Data **PP**(99), 1–1 (2017). doi:10.1109/TBDATA.2016.2637378
5. Hillner, S., Ngonga Ngomo, A.C.: Parallelizing LIMES for large-scale link discovery. In: I-Semantics 2011, pp. 9–16. ACM, New York (2011). doi:10.1145/2063518.2063520
6. Isele, R., Jentzsch, A., Bizer, C.: Silk Server - Adding missing Links while consuming Linked Data. In: Proceedings of the First International Workshop on Consuming Linked Data, CEUR Workshop Proceedings, vol. 665 (2010). CEUR-WS.org
7. Megdiche, I., Teste, O., Trojahn, C.: An extensible linear approach for holistic ontology matching. In: Groth, P., Simperl, E., Gray, A., Sabou, M., Krötzsch, M., Lecue, F., Flöck, F., Gil, Y. (eds.) ISWC 2016. LNCS, vol. 9981, pp. 393–410. Springer, Cham (2016). doi:10.1007/978-3-319-46523-4_24
8. Nentwig, M., Groß, A., Rahm, E.: Holistic entity clustering for linked data. In: Proceedings ICDM Workshops, pp. 194–201. IEEE (2016). doi:10.1109/ICDMW.2016.0035
9. Nentwig, M., Groß, A., Möller, M., Rahm, E.: Distributed holistic clustering on linked data. CoRR abs/1708.09299 (2017)
10. Nentwig, M., Hartung, M., Ngomo, A.N., Rahm, E.: A survey of current link discovery frameworks. Semant Web **8**(3), 419–436 (2017). doi:10.3233/SW-150210
11. Nentwig, M., Soru, T., Ngonga Ngomo, A.-C., Rahm, E.: LinkLion: a link repository for the web of data. In: Presutti, V., Blomqvist, E., Troncy, R., Sack, H., Papadakis, I., Tordai, A. (eds.) ESWC 2014. LNCS, vol. 8798, pp. 439–443. Springer, Cham (2014). doi:10.1007/978-3-319-11955-7_63

12. Ngonga Ngomo, A.-C., Sherif, M.A., Lyko, K.: Unsupervised link discovery through knowledge base repair. In: Presutti, V., d'Amato, C., Gandon, F., d'Aquin, M., Staab, S., Tordai, A. (eds.) ESWC 2014. LNCS, vol. 8465, pp. 380–394. Springer, Cham (2014). doi:10.1007/978-3-319-07443-6_26
13. Rahm, E.: The case for holistic data integration. In: Pokorný, J., Ivanović, M., Thalheim, B., Šaloun, P. (eds.) ADBIS 2016. LNCS, vol. 9809, pp. 11–27. Springer, Cham (2016). doi:10.1007/978-3-319-44039-2_2
14. Saeedi, A., Peukert, E., Rahm, E.: Comparative evaluation of distributed clustering schemes for multi-source entity resolution. In: Kirikova, M., Nørvåg, K., Papadopoulos, G.A. (eds.) ADBIS 2017. LNCS, vol. 10509, pp. 278–293. Springer, Cham (2017). doi:10.1007/978-3-319-66917-5_19
15. Thalhammer, A., Thoma, S., Harth, A., Studer, R.: Entity-centric data fusion on the web. In: Proceedings of the 28th ACM Conference on Hypertext and Social Media. ACM (2017). doi:10.1145/3078714.3078717