Order Matters: Matching Multiple Knowledge Graphs

Sven Hertling
sven@informatik.uni-mannheim.de
Data and Web Science Group
University of Mannheim, Germany

Heiko Paulheim
heiko@informatik.uni-mannheim.de
Data and Web Science Group
University of Mannheim, Germany

ABSTRACT
Knowledge graphs (KGs) provide information in machine-interpretable form. In cases where multiple KGs are used in the same system, that information needs to be integrated. This is usually done by automated matching systems. Most of those systems consider only 1:1 (binary) matching tasks. Thus, matching a larger number of knowledge graphs with such systems would lead to quadratic efforts. In this paper, we empirically analyze different approaches to reduce the task of multi-source matching to a linear number of executions of binary matching systems. We show that the matching order of KGs and the multi-source strategy actually matter and that near-optimal results can be achieved with linear efforts.

CCS CONCEPTS
• Information systems → Data extraction and integration; Information extraction.

KEYWORDS
Knowledge Graph; Multi-source matching; Incremental merge

ACM Reference Format:
Sven Hertling and Heiko Paulheim. 2021. Order Matters: Matching Multiple Knowledge Graphs. In Proceedings of the 11th Knowledge Capture Conference (K-CAP ’21), December 2–3, 2021, Virtual Event, USA. ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3460210.3493556

1 INTRODUCTION
During the past years more and more knowledge graphs (KGs) and ontologies were generated. The Linked Open Data (LOD) Cloud\(^1\) counts more than 1,250 datasets [33], and the LOD-a-lot dataset [13] has more than 650k datasets. Querying and utilizing this data in a uniform way is only possible if the schema and instances are aligned because each dataset can use their own identifiers (URIs) for the same concept. One key idea of the Semantic Web is to connect representations of the same entities with the \texttt{\texttt{owl:}\text{sameAs}} relation. There are also dedicated properties for specifying that two URIs describe the same class (\texttt{\texttt{owl:}\text{equivalentClass}}) and property (\texttt{\texttt{owl:}\text{equivalentProperty}}).

Usually, these knowledge graphs contain too many resources (classes, properties, and instances) to be manually mapped, and thus many matching systems were developed in the past. Starting from 2004, the Ontology Alignment Evaluation Initiative [11] (OAEI) takes care of evaluating such matching systems. The matchers are uploaded by developers and executed by track organizers. The matching interface originates from [12] and consists of one function which expects two KGs and an optional input alignment which contains already aligned entities. The result of this function is an alignment which consists of correspondences. Since 2018, the OAEI has a dedicated track for matching knowledge graphs [19].

Similarly, many systems are developed in the field of entity resolution/alignment. The main difference to KG matching is the static schema which is required by most systems. In other words, those systems expect the schemas of the KGs to be already aligned.

Due to the interface of all matching systems, they can only align two KGs at the same time, but the usual problem at hand is to align multiple data sources together. Even the conference track [6] at OAEI is originally a multi-source matching problem which is broken down into 21 binary test cases. If the ultimate goal is to obtain an alignment between \(n\) sources, this would require \(O(n^2)\) executions of a matching system.

The goal of this paper is thus to reuse the well researched and developed 1:1 matching systems for multi-source matching. The canonical approach matches each source pair, but it has a quadratic runtime. As an alternative, we propose an incremental approach, which matches and merges KGs successively. We show that such incremental merge based approaches can reach near-optimal results in linear runtime. Moreover, we analyze different ordering schemes for the incremental approaches, and analyze their impact on the matching result.

The contributions of this paper are:
• An extension of the general matcher and evaluation interface in MELT \(^2\) (an open-source matching framework recommended by OAEI),
• an implementation of different multi-source approaches in the aforementioned framework,
• an evaluation of those approaches, and
• analysis of the impact of different orderings and multi-source approaches.

The paper is structured as follows: Section 2 discusses the related work about multi-source matching and cluster repair strategies. It is followed by the description of the different approaches and orderings (Section 3). In the subsequent sections, we outline our experimental setup, discuss results, and conclude with an outlook on future works.

Throughout the paper, we assume that the knowledge graphs to be matched are duplicate free in themselves (which is the case most of the time). In case they are not, the sources could be deduplicated.

\(^1\)https://lod-cloud.net

\(^2\)https://github.com/dwslab/melt
The presented core idea of the overview paper is to do holistic data integration which starts with one source as an initial KG. Then, further KGs are merged to it. For each entity, it has to be decided if it should be merged to an existing entity or if it is a new concept not appearing yet. The order of how this is done, plays an important role. However, to the best of our knowledge, there is no work so far which analyzes the impact of the ordering.

The holistic clustering approach mentioned before is implemented by [25] in a distributed manner for Linked Data. It uses Apache Flink for parallelization and clusters each entity by simple linguistic similarity on labels. Afterwards, the type of the entity is used to further refine the clusters. Another parallel implementation from the same department is called Dedoop [21] which does the entity resolution in a map reduce fashion.

The difference is still that the schema is considered to be the same or at least matched. However, for popular KGs, such as DBpedia [2], YAGO [36], or Wikidata [37], this assumption cannot be made. [15] argued that not only the data, but also the schema needs to be integrated. They also propose a holistic approach based on rules, but only examples are given and no system was implemented nor evaluated.

When multiple sources are integrated, a candidate pre-selection is usually the first step. Afterwards, the similarities within each cluster are computed. Based on this similarity graph, various cleaning and improvement strategies can be applied. The main idea behind such algorithms is to use the graph structure and confidence values to find strongly connected components which usually represent one concept. If multiple resources are weakly connected, this can be a signal for wrongly stated correspondences. One approach was implemented by Raad et al. [30] which computes an error score for each owl:sameAs link in the LOD cloud. It is based on the community detection algorithm Louvain [5]. The FAMER system [32] also has an incremental clustering and repairing step after the linking phase. Multiple clustering approaches are compared such as Center, Star, and CLIP.

Summarizing, while systems for knowledge graph matching exist, they do not consider the case of multi-source matching. On the other hand, multi-source entity matchers assume a pre-aligned schema, which is not given for most real-world knowledge graphs, or only look at entity matching. Hence, to the best of our knowledge, this paper is the first to address both of these challenges simultaneously. Moreover, for multi-source matching, the order in which the sources are processed is assumed to have an impact on the results, but that impact has not been analyzed so far.

### 2 RELATED WORK

Starting in 2018, the OAEI campaign [1] has a new knowledge graph track which requires instance and schema matches. Over the years, developers adapted their matching components such that they return useful results. Therefore, systems like AgreementMakerLight (AML) [24], LogMap [20], ATBox [18], and ALOD2Vec [28] are able to match the schema as well as the instances. Another related approach is PARIS [35] which computes probabilities for each correspondence at the relation, instance, and schema level and uses this information to improve and validate other correspondences. Therefore, the instance alignment can help to improve the class alignment and also the other way around. However, all those systems are designed in a fashion that they match two knowledge graphs at a time and do not provide means for multi-source matching.

In the field of entity resolution and record linkage, there are some approaches which support the case of multiple sources [7, 32]. Thus, the closest related work to the presented one here, is multi-source entity resolution which covers a similar matching task, but does neither align the schema nor reuse it to improve performance. One reason is that in a typical entity resolution scenario, all sources have to have the same schema (all instances are from the same class and have the same – or at least manually matched – properties).

Also in that field, blocking [23, 40] is usually used as a strategy for restricting the search space. Based on a rather simple condition like sharing the same label, all resources are clustered. Each resource in a cluster will thus have the same label and possibly many of them represent the same real world concept. This matching strategy is most common in the field of entity matching [38]. As shown in [26] the blocking key for instances can be the corresponding classes. Here, the same observation holds: blocking keys can only be defined if the schemas of the data sources are already matched. Moreover, blocking keys usually need to be defined per class. In large, cross-domain knowledge graphs, there are often thousands of classes [16], all of which would require the definition of different blocking keys.

An overview of integrating multiple sources is given by Rahm [31]. In this work, the integration of KGs is also discussed as one use case among others, such as integrating open data. It is stated that the main challenges are data quality and semantic heterogeneity [8]. The presented core idea of the overview paper is to do holistic data integration which starts with one source as an initial KG. Then, further KGs are merged to it. For each entity, it has to be decided if it should be merged to an existing entity or if it is a new concept not appearing yet. The order of how this is done, plays an important role. However, to the best of our knowledge, there is no work so far which analyzes the impact of the ordering.

### 3 APPROACH

In the following section, different approaches are described which solve the task of multi-source matching reusing 1:1 matching systems. In comparison to the usual matching interface [12], the new interface consists of a function which expects a list of KGs, an input alignment, and additional parameters. The return value is again an alignment which is represented as a set of correspondences of the form \( \text{entityOne, entityTwo, } \ldots, 1, 0 \). Throughout the paper, this result is also called system alignment. Even though multiple sets of same entities can be used to represent the alignment, we stick to the usual format to be able to capture similarity graphs (a graph where the edges represents correspondences).

Figure 1 shows an example of four KGs A, B, C, and D which are matched with different multi-source approaches. All of them will be explained in more detail in the following sections.
Approaches with this strategy use a linear set of $n - 1$ matching problems. Matches between other KGs are then found by creating the transitive closure of the resulting alignments. All of those approaches require an ordering of KGs. In this paper, multiple orderings are tried out. The orders are defined by extracting different KG measures such as the number of classes, instances, or the total number of statements (model size). It might sometimes be the case that multiple of these measures produce the same order. All measures can be used in ascending and descending order which doubles the number of possible orderings.

(ii) Transitive Pairs: Windowing. The goal of this approach is to reduce the number of matching executions while keeping at least a (transitive) connection between all sources. It should be clear that these approaches reduce the number of executions with the drawback that not all matches might be found. E.g. if there is a concept in source A named $A_i$ and corresponding match in source C named $C_i$, then it is possible to find the correspondence $<A_i,C_i,=,1.0>$ only in the case where source B also contains a resource $B_j$ which should be matched both $A_i$ and $C_i$. Creating the execution plan for this approach requires an order of the KGs. Here the windowing approach is followed, which means that the first KG (in the example it is A) is matched to the second KG (B) and then the second KG to the third one and so on (window size of two).

(iii) Transitive Pairs: First vs Rest. This approach is similar to the previous one but it will not match the first one (e.g. the one with the highest or lowest number of classes) with the second one in the list. Instead the first KG is chosen as a hub and all other KGs are matched to it. The complexity and runtime is thus the same as in the previous approach.

(iv) Transitive Pairs: Similarity. Approaches (ii) and (iii) need an ordering to obtain the final execution plan. It is also clear that if a concept is not appearing in all KGs, that ordering can lead to missed correspondences. E.g. if source C and source D in approach (iii) contain a similar concept which is not contained in source A, that correspondence cannot be found. In the general case of KG matching, all of these sources can have different main topics or cover a variety of domains. This can also be observed in the Linked Open Data Cloud which lists multiple datasets clustered to domains like geography, government, life sciences, media etc.

The main idea of similarity based approaches is to find KGs with a similar topic and to match them first. Therefore, a minimum spanning tree of these sources are computed with Kruskal’s algorithm [22]. Computing such a tree always requires weighted edges which should represent in this case the topical closeness of KGs.

In this paper, we use textual signals to compute the topical similarity of two KGs. The weighting is computed by the cosine similarity of tf-idf vectors to mitigate the impact of terms which often appears (such as upper level ontology terms). Those vectors are generated by iterating over all literals in the KG which contains text and all URI fragments (part after the last slash ”/” or hashtag “#”). Literal texts are processed by sentence splitting, tokenization, lowercasing, stopword removal, and stemming. The fragments are tokenized additionally with camel case and some common separators such as minus (-), underscore (_), and tilde (~). The sentence splitting is left out for the fragments because it usually does not form a whole sentence.

Computing the final execution plan works as follows: (1) For each source KG, the tf-idf vector based on the texts in the KG is computed. (2) The cosine similarity for each possible matching pair is calculated. (3) The distances are sorted in ascending order. (4) Each pair is added as long as there is no cycle / the source and target KG does not belong to the same transitive closure. For an efficient implementation of the last step, we implemented a disjoint-set data structure to create the transitive closure of KGs and to check if a cycle would be generated. It is based on the algorithm described in [3] (Section 3.3).

In this approach, it is quite likely that hubs emerge. In Figure 1, assuming that A and C are from one domain and B and D are from another, a new KG E from the first domain would be matched to B and another KG F from the second domain would be matched to A such that KGs A and B both become hubs.

1Literals which either have a language tag or the datatype xsd:string, or rdfs:langString
2https://en.wikipedia.org/wiki/Disjoint-set_data_structure
3https://en.wikipedia.org/wiki/Camel_case
3.2 Incremental Merge

Incremental Merge based approaches, in contrast to the transitive pairs ones, do not leave the input knowledge graphs untouched, but merge two KGs after matching them. This might add some computational complexity, but the assumption is that the merged KGs then contain more information and can be matched better to other KGs. Moreover, the problem of missing single correspondences, as discussed above, can be mitigated in those cases.

(v) Incremental Merge: Order Based. The order based approach matches the first two KGs together, creates the union of them, and continues with the next KG in the list. All available orderings from (ii) can be reused.

One important step is to create the union of the two KGs given the alignment between them. This merging is performed as follows: First, we determine which of the two KGs acts as a source and which as target. A merged KG has always precedence for being the target. In case both KGs are leaf nodes, then the one with more triples is selected as the target. Because no assumption about the storage of KGs is made, leaf nodes which first acts as a target during the merge, are copied. This prevents modifications of KGs which are stored on disk (e.g. Jena TDB or HDT file [14]). With such a storage option, also huge KGs can be processed.

Now that the source and target are fixed for the merge, all triples of the source are merged into the target KG. All subjects, predicates, and objects of each triple are checked and resources (represented by a URI) are replaced with the URI of the target concept. The potential disadvantage of such a merge strategy is that all information/triples of a resource are added to the union. This means every wrong match of two concepts, also adds e.g. the label to the merged concept. It will only happen when the system is able to cope with multiple labels or other multi-valued properties. In the end, this can lead to topic/concept drifts which are most often analyzed in the time dimension [34, 39].

Preventing such a concept drift is done by adding only information/triples of non-matched entities to the union. Such a merging strategy has to be defined on a level of RDF triples. If the subject of a source triple has a mapping to a concept in the target KG, this triple does not need to be added to the union because it will only provide more information to an already existent concept. If only the predicate is mapped (due to the fact that also the schema is matched), the information is new and can be added to the union. If a resource in the object position has a correspondent concept, this is again an information about an already existent concept in the union. Therefore, it is not added. To sum up this kind of strategy, only triples where the subject or object is not matched, will be added to the union. This prevents a topic drift because the first seen concept is fixed and not further modifications are made even when similar concepts appear in other sources.

After one merge is finished, the result is used directly in the next run of the binary matcher. Thus, systems which are aware of being used in such a scenario, can reuse a potentially build index for the next run. For such a warm start, all triples which are added to the union, are also fed into the matching system for updating necessary indices (i.e., it does not reload the knowledge graphs from scratch in the next run). If the matcher has no warm start feature, in each run a larger KG needs to be read.

The complexity of approach (v) is \( n - 1 \) but the runtime is calculated by the sum of time used by the matcher plus the merge time after each run.

(vi) Incremental Merge: Similarity. Similar to the approach (iv), the incremental merge strategy cannot only be used with the order based approach but also with the textual similarity of KGs. The idea is again to first match sources which share the same topic and then all others. The actual execution plan is calculated by a hierarchical agglomerative clustering [10]. The result is a hierarchy of clusters which can directly be used as an execution plan. In the beginning, each KG is its own cluster and the algorithm searches for the next best KG to form a new cluster. The similarity between clusters can be determined with single, average or complete link strategies [10]. The clustering input is the tf-idf vector of each KG which is described in (iv). In comparison to the order based incremental search, it is now possible that two already merged KGs should be matched together. In such a case the larger union in terms of statements is selected as the target except that one of them is the result of a previous matching operation. The reason is that the matcher with the warm start feature needs to only update its index with the given subset of triples.

3.3 Other approaches

There are more approaches on how to reduce the multi-source KG matching task to multiple binary matching problems which are not evaluated in this paper. An example of such an approach would be to create the union of all KGs and match the union to itself. Such approaches are not included because they will not respect the goal of a 1:1 matching system to provide at most one corresponding concept for a given resource. Thus, most matching systems will only return the correspondences which match the element to itself because it will be the best match. Another approach of this kind would be to match source A to the union of all KGs except A and repeat this for each source. This would solve the before mentioned issue but again it is not clear to which concept of the union it is matched because the union can contain multiple relevant concepts (from different sources).

4 EVALUATION

During the evaluation the datasets and matching systems from OAEI are used. Many advantages result from this setting: (1) the matching systems are available and tested on the given datasets, (2) the gold standards evolved over time and are freely accessible, and (3) the datasets cover ontology matching as well as instance and schema matching datasets. The evaluation was executed on a server with 768GB of RAM and 40 CPUs (2.4 GHz) with Debian 10 operating system and Openjdk version 1.8.0_162.

4.1 Datasets

The conference dataset [41] is selected as a representative of a multi-source ontology matching problem. It consists of 16 ontologies which cover the domain of conference organization. Although the initial problem is to combine many ontologies, the actual test cases are reduced to a 1:1 matching problem with a corresponding gold standard. Out of these 16 ontologies, only seven are involved in the reference alignment which results in 21 test cases with a gold
standard. More information on the dataset can be found at the official web page\(^6\) and in [41].

The next track is called Knowledge Graph\(^7\) and requires schema as well as instance matches. It consists of eight KGs which cover different topics from the entertainment and gaming domain. They are generated by running a modified version of the DBpedia extraction framework [4] on Wikis from the Fandom\(^8\) Wiki hosting platform [17]. All in all, three main topics exist: Star Wars, Marvel, and Star Trek. For five pairs of sources, a reference alignment exist. Together, they contain 49 class, 181 property, and 15,129 instance correspondences [19]. The latter ones are extracted directly from the Wiki pages by inspecting links in sections with a header containing the word *link*. Nearly 25% of these matches are non-trivial (meaning they cannot be found by simple string comparison). The schema alignments are generated by experts. The whole gold standard in this track is partial because not all resources in the KG are matched.

### 4.2 Systems

In our experiments, we select the top performing systems from OAEI 2020 [29] in the respective tracks. This results in the following matching systems: AML, LogMap, ALOD2Vec, and ATBox. Due to the fact that LogMap runs into deadlocks in the KG track, it is replaced by the PARIS matching system [35]. The latter one is specifically developed for matching the schema as well as the instances and is therefore a perfect replacement. The equivalence result files of the last iteration of PARIS are used. Since the matcher only returns subsumption relations for classes in the conference framework [4] (Section 3.3). Our implementation does not use a database and keeps all data structures in memory to speed up the computation and also allows to add multiple elements which already belong to the same cluster.

After the transitive closure of the alignment is calculated, each cluster consists of similar entities (this corresponds to all resources shown in Figure 2b). Those have to be mapped to the corresponding test cases. In our example, where the result for the pair A-B is to be computed, all combinations of elements from A and B in the same cluster are considered, i.e., \(\langle A_5, B_5, =, 1.0 \rangle\) and \(\langle A_5, B_9, =, 1.0 \rangle\). Due to the fact that the confidence is not taken into account for this evaluation, it is set to 1.0.

Furthermore, for all tracks, a simple and an extended version exist. For the former, only the sources which participate in the gold standard are used. In the example shown in Figure 2a this will include only the sources A to E. For the extended version, additional data sources are added for which no reference alignment exists (these are KGs F, G, and H). In the conference track, all 16 ontologies are used, whereas for the KG track, we doubled the number of sources to be also 16. The added KGs are randomly drawn from the set of extracted Wikis available in Fandom which are not too small.

### 4.4 Results

All detailed results can be analyzed in a CSV file uploaded to Github\(^9\). It further distinguishes the class, property, and instance alignments and also shows the precision, recall, and \(F_1\) measure computed with micro and macro averages. All runtimes are comparable with each other because the time used to compute the

\(^{(a)}\) https://github.com/dwslab/melt/tree/master/examples/multisourceExperiment/multisourceResults

\(^{(b)}\) https://github.com/dwslab/melt/tree/master/examples/multisourceExperiment/multisourceResults/multisourceResults

\(^{(c)}\) https://github.com/dwslab/melt/tree/master/examples/multisourceExperiment/multisourceResults

\(^{(d)}\) https://github.com/dwslab/melt/tree/master/examples/multisourceExperiment/multisourceResults

\(^{(e)}\) http://oaei.ontologymatching.org/2020/conference/index.html

\(^{(f)}\) http://oaei.ontologymatching.org/2020/knowledgegraph/index.html

\(^{(g)}\) https://www.fandom.com
Table 1: Micro F-measure results for each approach and base matcher in the KG track. The extended version has eight more sources to be matched. Runtime is given in HH:MM:SS. TP means transitive pairs and IM represents the incremental merge approach.

| Approach                     | Knowledge Graph | Knowledge Graph Extended |
|------------------------------|----------------|--------------------------|
|                              | ALOD2Vec | AML | ATBox | Paris | ALOD2Vec | AML | ATBox | Paris |
|------------------------------|----------|-----|-------|-------|----------|-----|-------|-------|
| (t) All Pairs                | .89      | .87 | .87   | .90   | .89      | .87 | .90   | .90   |
| (ii) TP: Windowing           | .13      | .12 | .12   | .15   | .13      | .12 | .15   | .15   |
| ClassesDescending            | .69      | .67 | .71   | .70   | .69      | .67 | .70   | .70   |
| ModelSizeDescending          | .70      | .68 | .72   | .70   | .70      | .68 | .70   | .70   |
| ClassesAscending             | .13      | .12 | .12   | .15   | .13      | .12 | .15   | .15   |
| InstancesAscending           | .69      | .67 | .71   | .70   | .69      | .67 | .70   | .70   |
| ModelSizeAscending           | .70      | .68 | .72   | .70   | .70      | .68 | .70   | .70   |
| (iii) TP: First vs Rest      | .37      | .34 | .37   | .48   | .37      | .34 | .48   | .48   |
| ClassesDescending            | .26      | .23 | .26   | .31   | .26      | .23 | .31   | .31   |
| ModelSizeDescending          | .37      | .36 | .37   | .36   | .37      | .36 | .36   | .36   |
| ClassesAscending             | .18      | .16 | .15   | .21   | .18      | .16 | .21   | .21   |
| ModelSizeAscending           | .18      | .17 | .18   | .18   | .18      | .17 | .18   | .18   |
| (iv) TP: Similarity          | .88      | .87 | .89   | .90   | .87      | .86 | .88   | .89   |
| ClassesDescending            | .83      | .85 | .87   | .85   | .83      | .85 | .85   | .85   |
| ModelSizeDescending          | .85      | .85 | .86   | .86   | .85      | .85 | .86   | .86   |
| ClassesAscending             | .86      | .85 | .86   | .84   | .86      | .85 | .84   | .85   |
| ModelSizeAscending           | .87      | .85 | .84   | .81   | .87      | .84 | .81   | .84   |
| (v) IM: Order Based          | .87      | .86 | .87   | .87   | .87      | .86 | .87   | .86   |
| Single                       | .87      | .86 | .87   | .87   | .87      | .86 | .87   | .86   |
| Average                      | .88      | .86 | .87   | .87   | .88      | .86 | .87   | .86   |
| Complete                     | .88      | .86 | .87   | .87   | .88      | .86 | .87   | .86   |

merged knowledge graph for incremental merge based approaches is included as well.

4.4.1 Knowledge Graph Track. Table 1 shows the results for each approach in the KG track. The AML matcher is the slowest one and needs over 27 hours for matching all pairs in the extended version.

Thus, other approaches need to be applied. In general, approach (ii) performed the worst. The main reason is that it fixes one source and matches all others against it. Thus, the initially selected source places an upper limit on the set of concepts are able to be matched. Furthermore, the approach is very unstable because it relies on a good choice of a matching hub. It turns out that neither the classes nor the instances are a good measure but the total number of statements in the KG (ModelSize) returns better results. When using the latter measure, approach (iii) can reach 0.37 in terms of F1.

A better approach is to match different sources together like in (ii) where the order is used in combination with a windowing approach. Again, it is better to use an ordering based on the model size. Note here that descending and ascending results in the same execution plan with the only difference that source and target are interchanged (the same results support the assumption that the matchers do not care about switching source and target).

When using the textual similarity for generating the order, results are getting even better. This holds for the transitive pairs (iii) as well as incremental merge (vi). The former one has very good results, but one has to be careful because the reference alignments exist for KGs which are also in the same topic. Thus, the incremental approach is better suited because all concepts are kept in the union and can be matched in later steps. The runtime increases in contrast to the transitive pair approaches because the matchers used in the experiments do not have a warm start feature and need to read the inputs over and over again (the merge itself also takes some time but everything is included in the final runtime). The base matchers behave similarly with the exception of ALOD2Vec which has better results in (v) when using the ascending order.

Overall, we can observe that the decomposition approaches can achieve results which are only one percentage point away from the pairwise approaches, but at significantly lower runtimes.

4.4.2 Conference Track. For the conference track, Table 2 lists the results for the extended version which contains 16 ontologies.

The results are comparable to the one from the KG track, despite the drop in F1 being higher compared to the all pairs approach. There are also some smaller differences. For example, the transitive pairs approach where one source is fixed (iii) is consistently better than the windowing approach (ii). This is the case because all sources have a similar topic and fixing one of it as a matching hub is not so severe. But even if the sources all have a similar topic, a clustering of sources based on text makes sense, and LogMap
Table 2: Micro F-measure results for each approach and base matcher in the extended conference track. The runtime is given in HH:MM:SS. TP means transitive pairs and IM represents the incremental merge approach.

| Approach | ALOD2Vec | AML | ATBox | LogMap |
|----------|----------|-----|-------|--------|
| (ii) TP: Windowing | .54 | .00:16:20 | .74 | .00:18:25 | .46 | .00:21:08 | .68 | .00:16:45 |
| ModelSizeAscending | .24 | .00:02:07 | .26 | .00:02:19 | .28 | .00:02:45 | .25 | .00:01:54 |
| ModelSizeDescending | .31 | .00:02:15 | .36 | .00:02:14 | .31 | .00:02:41 | .34 | .00:02:03 |
| ClassesAscending | .22 | .00:02:13 | .22 | .00:02:14 | .28 | .00:02:46 | .24 | .00:02:06 |
| ClassesDescending | .30 | .00:02:13 | .36 | .00:02:14 | .31 | .00:02:46 | .34 | .00:01:53 |
| (iii) TP: First vs Rest | .41 | .00:02:20 | .46 | .00:02:31 | .44 | .00:02:55 | .47 | .00:02:02 |
| ModelSizeAscending | .45 | .00:02:19 | .57 | .00:02:28 | .52 | .00:03:03 | .57 | .00:02:07 |
| ModelSizeDescending | .56 | .00:02:51 | .65 | .00:04:26 | .58 | .00:06:12 | .63 | .00:02:13 |
| ClassesAscending | .51 | .00:02:43 | .64 | .00:03:21 | .56 | .00:04:48 | .63 | .00:02:13 |
| ClassesDescending | .51 | .00:02:40 | .66 | .00:03:22 | .54 | .00:04:43 | .63 | .00:02:12 |
| (iv) TP: Similarity | .48 | .00:02:35 | .60 | .00:04:52 | .52 | .00:05:01 | .64 | .00:02:11 |
| Single | .50 | .00:02:36 | .59 | .00:04:37 | .52 | .00:04:52 | .63 | .00:02:11 |
| Complete | .42 | .00:02:31 | .xx | .xx:xx:xx | .50 | .00:03:52 | .64 | .00:02:08 |

Table 3: Results for all tracks when applying different clustering repair approaches on the approach all pairs (i). The values represent micro F1 measure.

| Approach | ALOD2Vec | AML | ATBox | LogMap |
|----------|----------|-----|-------|--------|
| Conference | All Pairs (Baseline) | .563 | .739 | .595 | .683 |
| CLIP | .580 | .729 | .598 | .663 |
| Error .90 | .593 | .739 | .595 | .683 |
| Connected Components | .563 | .739 | .595 | .688 |
| Conference extended | All Pairs (Baseline) | .563 | .739 | .595 | .683 |
| Error .90 | .577 | .739 | .609 | .676 |
| Knowledge graph | All Pairs (Baseline) | .890 | .870 | .891 | .896 |
| CLIP | .889 | .872 | .891 | .896 |

The gains highly depend on the base matcher and track. The performance of ALOD2Vec can be improved by three percentage points in term of F1 for the conference track when removing all correspondence with an error greater than 0.9. For ATBox, the gain is 0.3 when using the CLIP approach and 0.5 for LogMap when using Connected Components. For the KG track the CLIP approach works better but yields only an improvement of 0.2 F1 points for AML. All in all, it can be shown that the results cannot be further improved by a large margin except for ALOD2Vec in both conference tracks.

5 CONCLUSION AND OUTLOOK

In this paper we showed that the order and the multi-source strategy play an important role when matching multiple KGs. Two approaches turned out to work in most scenarios: 1) incremental merge with model size descending and 2) incremental merge with textual similarity (average link). They have the best trade-off between quality (overall F-measure) and efficiency (reduced runtime). Thus, we propose to first try out these methods to reuse an existing matching system in a multi-source setting. It reduces the quadratic time from the all pairs approach to a linear one with a near-optimal result. All methods and evaluation capabilities are implemented and documented in the open-source matching framework MELT to allow other researchers to reproduce and reuse the proposed approaches.

Current matching systems usually do not output only matches, but also confidence scores, which are then used, e.g., for downstream filtering. So far, especially in the transitive pairs approaches, we have not yet analyzed different alternatives for computing such scores. An example is depicted in Figure 4. It is a priori not clear what the confidence between A1 and D1 should be. In the current correspondence between C1 and D1 can be removed because all similar resource A1, B1, and C1 are not matched to D1 even though all data sources A, B, C, and D are matched together.

In this section, it is analyzed if the all pairs approach (i) followed by such a conflict resolution approach is worth the additional runtime it requires. Therefore, we reimplemented the approach from [30]. It computes an error degree for each correspondence and we filter them based on the following thresholds: 0.99, 0.95, 0.90, 0.80, 0.60. Additionally, all clustering approaches from FAMER [32] are tried out as well to further increase the performance. They need multiple parameters such as a priority selection. Therefore, we tried out all possible values and report the best ones in Table 3. CLIP, and Connected Components refers to FAMER based clustering approaches and Error 0.90 refers to [30] with the corresponding error threshold.
evaluation we just set it to the constant value of 1.0. It is important to note that in our work the correspondences are evaluated regardless of their confidence value and can thus be set to an arbitrary value. Nonetheless it is an interesting direction for researchers which require such values.

When matching a larger scale of input knowledge graphs, parallelization becomes an interesting option. While transitive pairs can be easily parallelized, this is not so straightforward for the incremental merge based approaches. Here, finding a value. Nonetheless it is an interesting direction for researchers which require such values.

As a next step, all KGs from the DBkWik project [17] should be matched together to create one consolidated KG. But matching is only one part during data integration. Afterwards the sources may provide different conflicting data values. A first step would be to find out which properties are functional and should only provide one value. For these cases conflict resolution methods should be applied.

REFERENCES

[1] Alsayed Algergawy, Michelle Cheatham, Daniel Faria, Alfio Ferrara, Irini Fundulaki, Ian Harrow, Sven Hertling, Ernesto Jiménez-Ruiz, Naouel Karam, Abderrahmane Khiat, et al. 2018. Results of the ontology alignment evaluation initiative 2018. In Ontology Matching Workshop @ ISWC. Vol. 2018. 76–116.

[2] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. DBpedia: A nucleus for a web of open data. In The semantic web 722–735.

[3] Wouter Beek, Joe Raad, Jan Wielender, and Frank Van Harmelen. 2018. sameas. cc: The closure of 500m owl: sameas statements. In ESWC. 65–80.

[4] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. 2009. DBpedia-A crystallization point for the Web of Data. Journal of web semantics 7, 3 (2009), 154–165.

[5] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. Journal of statistical mechanics: theory and experiment 2008, 10 (2008), P10008.

[6] Michelle Cheatham and Pascal Hitzler. 2014. Conference v2.0: An uncertain knowledgebase. In ISWC. 576–592.

[7] Vasilis Christophides, Vasilo Ethiblynious, Themis Palpanas, George Papadakis, and Kostas Stefanidis. 2020. An Overview of End-to-End Entity Resolution for Big Data. Comput. Surveys 53, 6 (2020).

[8] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horst, Ni Lao, Kevin Murphy, Thomas Strohmaier, Shouhao Sun, and Wei Zhang. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In SIGKDD. 601–610.

[9] Ahmed K Elmargamid, Panagiotis G Ipeirotis, and Vassilios S Verykios. 2006. Duplicate record detection: A survey. IEEE Transactions on knowledge and data engineering 19, 1 (2006), 1–16.

[10] David Eppstein. 2000. Fast hierarchical clustering and other applications of dynamic closest pairs. Journal of Experimental Algorithmics (JEA) 5 (2000), 1–es.

[11] Jérôme Euzenat, Christian Medlücke, Heiner Stuckenschmidt, Pavel Shvaiko, and Cäsar Trajano. 2011. Ontology alignment evaluation initiative: six years of experience. In Journal on data semantics XV. 158–192.

[12] Jérôme Euzenat, Pavel Shvaiko, et al. 2007. Ontology matching. Vol. 18.

[13] Javier D. Fernández, Wouter Beek, Miguel A Martinez-Prieto, and Mario Arias. 2017. LOD-a-lot. In ISWC. 75–83.

[14] Javier D. Fernández, Miguel A. Martinez-Prieto, Claudio Gutiérrez, Axel Polleres, and Mario Arias. 2013. Binary RDF Representation for Publication and Exchange (HDT). Web Semantics: Science, Services and Agents on the World Wide Web 19 (2013), 22–41.

[15] Laura M Haas, Martin Hentschel, Donald Kossmann, and Renée J Miller. 2009. Schema and data: A holistic approach to mapping, resolution and fusion in information integration. In International Conference on Conceptual Modeling. 27–48.

[16] Nicolas Heist, Sven Hertling, Daniel Ringler, and Heiko Paulheim. 2020. Knowledge Graphs on the Web-An Overview.

[17] Sven Hertling and Heiko Paulheim. 2018. DBwik: A consolidated knowledge graph from thousands of wikis. In 2018 IEEE International Conference on Big Knowledge (ICBK). 17–24.

[18] Sven Hertling and Heiko Paulheim. 2020. ATBox results for OAEI 2020. In Ontology Matching Workshop @ ISWC. Vol. 2018. 168–175.

[19] Sven Hertling and Heiko Paulheim. 2020. The knowledge graph track at OAEI: Gold standards, baselines, and the golden hammer bias. In ESWC. Vol. 12123. 343–359.

[20] Ernesto Jiménez-Ruiz. 2020. LogMap family participation in the OAEI 2020. In Ontology Matching Workshop @ ISWC. Vol. 2018. 201–203.

[21] Lars Kolb, Andreas Thor, and Erhard Rahm. 2012. Dedoop: Efficient deduplication with hadoop. VLDBJ 5, 12 (2012), 1878–1881.

[22] Joseph B Kruskal. 1956. On the shortest spanning subtree of a graph and the traveling salesman problem. Proceedings of the American Mathematical society 7, 1 (1956), 48–50.

[23] Bo-Han Li, Yi Liu, An-Man Zhang, Wen-Huan Wang, and Shuo Wan. 2020. A survey on blocking technology of entity resolution. Journal of Computer Science and Technology 35, 4 (2020), 769–793.

[24] Beatriz Lima, Daniel Faria, Francisco M Couto, Isabel F Cruz, and Catia Pesquita. 2018. OAEI 2020 results for AML and AMLC. In Ontology Matching Workshop @ ISWC. Vol. 2018. 154–160.

[25] Markus Nentwig, Anika Groß, Maximilian Möller, and Erhard Rahm. 2017. Distributed holistic clustering on linked data. In OTM Confederated International Conferences. 371–382.

[26] Markus Nentwig, Michael Hartung, Axel-Cyrille Nongo Ngomo, and Erhard Rahm. 2017. A survey of current link discovery frameworks. Semantic Web 8, 3 (2017), 419–436.

[27] Heiko Paulheim. 2017. Knowledge graph refinement: A survey of approaches and evaluation methods. Semantic web 8, 3 (2017), 489–508.

[28] Jan Portisch, Michael Hladik, and Heiko Paulheim. 2020. ALOD2Vec matcher results for OAEI 2020. In Ontology Matching Workshop @ ISWC. Vol. 2018. 147–153.

[29] Nikosie Pour, Alsayed Algergawy, Reihaneh Amini, Daniel Faria, Irini Fundulaki, Ian Harrow, Sven Hertling, Ernesto Jiménez-Ruiz, Clement Jonquet, Naouel Karam, et al. 2020. Results of the ontology alignment evaluation initiative 2020. In Ontology Matching Workshop @ ISWC. Vol. 2018. 92–138.

[30] Joe Raad, Wouter Beek, Frank Van Harmelen, Nathalie Fernelle, and Fatihia Sais. 2018. Detecting erroneous identity links on the web using network metrics. In ESWC. 391–407.

[31] Erhard Rahm. 2016. The case for holistic data integration. In East European Conference on Advances in Databases and Information Systems. 11–27.

[32] Alihe Saiedi, Eric Peukert, and Erhard Rahm. 2018. Using link features for entity clustering in knowledge graphs. In ESWC. 576–592.

[33] Max Schmidtberg, Christian Bizer, and Heiko Paulheim. 2014. Adoption of the linked data best practices in different topical domains. In ISWC. 245–260.

[34] Thanos G Stavropoulos, Stelios Andreas, Efstratios Kontopoulos, and Ioannis Kompatsiaris. 2019. SemaDrift: A hybrid method and visual tools to measure semantic drift in ontologies. Journal of Web Semantics 54 (2019), 87–108.

[35] Fabian M. Suchanek, Serge Alehtehou, and Pierre Senellart. 2011. PARS: Probabilistic Alignment of Relations, Instances, and Schema. VLDBJ 5, 3 (2011), 157–168.

[36] Thomas Pellissier Tanon, Gerhard Weikum, and Fabian Suchanek. 2020. Yago 4: A reason-able knowledge base. In ESWC. 583–596.

[37] Denny Venetich und Markus Krötsch. 2014. Wikidata: a free collaborative knowledgebase. Commun. ACM 57, 10 (2014), 78–85.

[38] Juanna Wang, Guoliang Li, Jeffrey Xu Yu, and Jianhua Feng. 2011. Entity matching: How similar is similar. VLDBJ 4, 10 (2011), 622–633.

[39] Shenghui Wang, Stefan Scholobach, Janet Takens, and Wouter Van Atteveldt. 2009. Mapping-chains for studying concept shift in political ontologies. Ontology Matching 13 (2009).

[40] Steven Euijong Whang, David Menestrina, Georgia Koutrika, Martin Theobald, and Mario Arias. 2013. Binary RDF Representation for Publication and Exchange (HDT). Semantic Web: Research and Applications. 191–208.

[41] Ondrej Zamazal and Vojtech Svatek. 2017. The ten-year ontofarm and its fertilization within the onto-sphere. Journal of Web Semantics 43 (2017), 46–53.