Knowledge Graphs Alignment Based on Learning to Rank Methods

Victor Yamamoto[a] and Julio Cesar dos Reis[b]
University of Campinas, Campinas, SP, Brazil

Keywords: Knowledge Graphs, Entity Alignment, Supervised Ranking Aggregation.

Abstract: Knowledge graphs (KGs) define facts expressed as triples in representing knowledge. Usually, several knowledge graphs are published in a given domain. It is relevant to create alignments both for classes that model concepts and between instances of those classes defined in different knowledge graphs. In this article, we study techniques for aligning entities expressed in KGs. Our solution explores the supervised ranking aggregation method in the alignment based on similarity values. Our experiments rely on the dataset from the *Ontology Alignment Evaluation Initiative* to evaluate the proposed method in experimental analyzes. Obtained results indicate the effectiveness in our alignment technique in the investigated datasets.

1 INTRODUCTION

The term knowledge graphs was coined by Google when it introduced this technology as the basis for a new web search strategy in 2012 (Ringler and Paulheim, 2017a). A traditional search method uses keywords to search for the expected results, but the terms can be ambiguous and limit the retrieved information. The use of knowledge graphs allows the search to be carried out for objects that represent real entities such as places, people and movies. These entities and their relationships allow performing information retrieval by using a context in which the term is searched for. This helps in reducing the ambiguity of terms, and improving the quality of information returned when using the entities' relationships (Singhal, 2014).

Large-scale knowledge graphs (KGs) like *DBpedia* [1], *YAGO*[2] and *Wikidata* [3] play a central role as a source of general knowledge. These KGs present a good coverage regarding the entities represented and expressed in several domains. However, they lack covering specific topics or they usually are addressed with little detail (Hertling and Paulheim, 2018). The aforementioned KGs present similar information, as they all use *Wikipedia* as the basis for creating entities and their relationships. Their use in a combined way requires creating links between entities of different KGs (Ringler and Paulheim, 2017b). For example, "*Black Panther*" [4] is an entity from Marvel Cinematic Universe Wiki, which is mapped to the Marvel Database entity "*Black Panther*" [5].

Hofman *et al.* (Hertling and Paulheim, 2018) used a two-step method to create mappings between KGs generated from Wikis. First, mappings were generated between each Wiki and DBpedia. Using these mappings, the KGs were grouped in blocks and the mappings between Wikis were created only between graphs of the same group. To obtain the related entities, a string distance algorithm was used based on the labels of the entities.

Learning to Rank is a machine learning technique for training ordering models. This technique can be used in several areas such as information retrieval, natural language processing and data mining. An example of application of such technique is the retrieval of documents. A system manages a set of documents and when a query is executed, the system searches for documents containing the queried terms, order the documents based on different processed ranking lists and returns the best results (LI, 2011).

In the creation of an alignment (process of generating mappings between KG entities), the simplest approach might look for all possible pairs in a set. However, this approach becomes prohibitive for larger sets. Locality-sensitive Hashing (LSH) (Leskovec et al., 2014) allows the comparison between similar

[a] https://orcid.org/0000-0002-3825-6461
[b] https://orcid.org/0000-0002-9545-2098
[1] http://dbpedia.org
[2] https://yago-knowledge.org
[3] https://www.wikidata.org/wiki/Wikidata:Main_Page
[4] https://marvelcinematicuniverse.fandom.com/wiki/Black_Panther
[5] https://marvel.fandom.com/wiki/Black_Panther

Yamamoto, V. and Reis, J.
Knowledge Graphs Alignment Based on Learning to Rank Methods.
DOI: 10.5220/0012258100003598
In *Proceedings of the 15th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2023) - Volume 2: KEOD*, pages 315-322
ISBN: 978-989-758-671-2; ISSN: 2184-3228

pairs. The items are provided in a hash, in which the probability of two items having the same hash is based on the similarity value between them. In this sense, the candidate pairs are those that are in the same hash bucket .

In this article, we propose and evaluate a method for the alignment of KGs based on Learning to Rank techniques. In our approach, we investigate the candidate reduction methods based on Locality-sensitive Hashing. We experimentally evaluate our proposal based on the test dataset offered by the OAEI (*Ontology Alignment Evaluation Initiative*) competition [6]. The quality of the mappings created by our solution are evaluated by comparing them with the alignment provided by the competition (as a gold standard). In our results, the proposed technique obtained high recall in the created mappings, but affected precision in several cases. Compared to existing baseline systems, our approach presents lower f-measure for class and instance, but higher f-measure for property.

The remaining of this article is organized as follows: Section 2 presents a literature review. Section 3 introduces the necessary formalization; Section 4 describes our proposal; Section 5 presents the experimental results conducted; Section 6 discusses the achieved results; Section 7 summarizes the conclusions.

2 BACKGROUND

Several systems have proposed creating mappings between entities of KGs. AgreementMakerLight Ontology Matching System (AML) (Faria et al., 2013) is a framework that maps ontologies using four types of matchers and implements a ranked selector. The matchers are lexicon, lexicon mediated by a third ontology; words using the Jaccard index (Jaccard, 1912); and a set of parameters exploring string similarity methods. To obtain the best mappings, the results of the matchers are ordered in a unified list from the best to the worst. The mappings are created based on such an ordered list.

FCAMap-KG system (Chang et al., 2019) uses the analysis of formal concepts to create mappings. This system organizes the process into three stages: lexical match, structural match, and match filter. The first step creates three formal contexts for classes, properties, and instances based on keys. In OAEI KG context, those types were created to fuse different KGs into one coherent KG. Schema type (classes and properties) derives from wikis' constructs, and instance type derives from pages about real-world entities (Hertling and Paulheim, 2020). A mapping is created when a formal concept contains objects from the two KGs under alignment. In the second stage, the previously obtained mappings are used to create a structured formal context. This step focuses on creating mappings between the instances using RDF triples whose properties and subsequent instances were already mapped. In the last step, the mappings are selected so that each entity has only one mapping. This operation can be carried out because the OAEI 2019 KG competition uses only 1:1 matches. If an entity has more than one mapping, the mapping with more structural attributes and lexical keys in common is selected in this technique. DOME system (Deep Ontology MatchEr) (Hertling and Paulheim, 2019) uses doc2vec method to obtain the mappings. Doc2vec is an algorithm that learns fixed-length feature representations from variable-length pieces of text. Each document is represented as a vector trained to predict words in the documents

Learning to Rank (LTR) is a machine learning technique for training the model to rank (LI, 2011). The system manages a system of documents. When queried, the system retrieves documents related to the query, ranks the documents, and returns the top-ranked documents. The differences between learning to rank and other models are that LTR does not need to predict the absolute value of the items (regression); it does not need to predict the class of items (classification); the important thing is to obtain the relative ranking of items. Our method explored *LambdaMart*, a pairwise learning-to-rank technique. Destro *et al.* (Juliana M. Destro and da S. Torres, 2019) explored several rank aggregation techniques for aligning cross-lingual ontologies and found that *LambdaMart* have the best results.

3 FORMAL PRELIMINARIES

Knowledge Graph. A knowledge graph $\mathcal{K}$ accumulates and conveys knowledge in term of entity and relations (Hogan et al., 2020). Formally, a knowledge graph $\mathcal{K} = (\mathcal{E}_{\mathcal{K}}, \mathcal{R}_{\mathcal{K}})$ consists of a set of entities $\mathcal{E}_{\mathcal{K}}$ represented as nodes; and entities are interrelated by directed relationships $\mathcal{R}_{\mathcal{K}}$. Each entity $e \in \mathcal{E}_{\mathcal{K}}$ has a unique identifier and type. The entity identifier uses Universal Resource Identifier (URI) that is a string used to identify resources and provide a mean of locating the resource Entity type can be of schema level, formed by properties and classes, and instance level. Each relationship $r(e_1, e_2, e_3) \in \mathcal{R}_{\mathcal{K}}$ is a triple consisting of a subject, a predicate and an object.

[6]http://oaei.ontologymatching.org

Mapping. Given two entities e_s and e_t from two different KGs, a mapping m_{st} is defined as:

$$m_{st} = (e_s, e_t, conf) \quad (1)$$

The $conf$ is the similarity value between e_s and e_t indicating the confidence of their relation. We define $\mathcal{M}_{ST}^{j}$ as a set of mappings m_{st} between two KGs $\mathcal{K}_S$ and $\mathcal{K}_T$.

Similarity. Given two entities e_i and e_j, the similarity between them is defined as a function that calculates the similarity score between them returning a real value in the interval [0,1]. The function can explore different techniques for similarity computation, like string-based processing and semantic-based techniques which explore background knowledge for similarity computation. Formally:

$$sim(e_i, e_j) = f(e_i, e_j) \quad (2)$$

Rank Aggregation. Consider a set of rankings R_1, $R_2, R_3, ..., R_n$ formed by set of similarity proposals S_1, S_2, S_3, ... S_m. Each similarity list S_k requires a different function $sim(e_i, e_j)$ with a source e_i and target e_j. Each $s \in S$ is defined as a triple $s = (e_i, e_j, sim(e_i, e_j)$ and the entity e_i can be used to retrieve a set of similarity. Rank Aggregation produces one single rank that aggregates all given ranking. Equation 3 is used to include new set of ranking to aggregate. Each column represents a ranking and in each rank, the yellow box represents an element that is ranked in different positions for each rank. An aggregated rank is created and used to map entities in our solution.

$$rankAggregation.aggregate(R) = R_{agg} \bigcup R \quad (3)$$

Given an aggregated rank *rankAggregation* and an entity e, it is possible to query the aggregated rank presenting e. Equation 4 represents a function that returns a ranking formed by a set of similarities that have input e as a query filter.

$$rankAggregation.query(e) = R, s \in R_{agg}, s.id = e \quad (4)$$

Locality-Sensitive Hashing. An entity e is converted to a set of n-grams and added to a bucket of the hash. The equation 5 describes a function that insert an entity e to the hash H. Locality-sensitive Hash is an algorithm that uses random projections to construct hash codes in which pairwise distance is preserved when the length of codes is sufficiently large (Tsai and Yang, 2014). Given an entity, e and a locality-sensitive hash LSH, entity e can be used to query a similar entity inside such hash based on a similarity method. The equation 6 describes a function that queries for similar entities compared to entity e_i and returns a set of entity E where $e_j \in E$ have similarity $sim(e_i, e_j)$ greater than a given threshold τ (Leskovec et al., 2014).

$$LSH.insert(e_i) = H \bigcup Hash(e_i) \quad (5)$$

$$LSH.query(e_i) = E, e_j \in E, sim(e_i, e_j) > \tau \quad (6)$$

Problem Statement. Given two KGs $\mathcal{K}_S$ and $\mathcal{K}_T$, the problem addressed in this work is to obtain all mappings between entities from these KGs by using rank aggregation techniques and reduce explored candidates with the use of Locality-sensitive Hashing.

4 OUR PROPOSAL TO LINK KNOWLEDGE GRAPHS

Our goal is to create an appropriate mapping for each entity of a source KG $\mathcal{K}_S$ to a target KG $\mathcal{K}_T$. Figure 1 shows our defined workflow for this purpose. It is organized in four main steps: input process, candidate pairing, similarity calculation, and map creation.

In the first step, both KGs are processed and their entities are divided by their type (class; property; instance). Every triple formed by subject, predicate and object is extracted if the predicate is an RDFS label. The Resource Description Framework (RDF) is a framework for representing information in the Web, and RDFS is the schema used to model RDF data. RDFS:label is an instance that provides a human-readable resource's name. If the subject of the triple is an URI, a new entity is created and the extracted URI identifies it. The entity type is defined by analysing if the type name is contained in the URI. Class, property, and instance is identified by class, property, and resource, respectively.

In the second step (candidate pairing), the method creates the list of candidate entities for each entity from the source KG $\mathcal{K}_S$. For schema types (class and property), all target candidates are considered as candidates. For instance type, locality-sensitive hashing is used to create the list of candidate entities All entities from target KG $\mathcal{K}_T$ are inserted to the hash $\mathcal{H}$. After this process, each entity e from source KG $\mathcal{K}_S$ query hash $\mathcal{H}$ to retrieve the set of candidate entities $\mathcal{E}$.

In the third step (similarity calculation), each pair of source entity and candidate entity similarity is calculated. The similarity is calculated using entity name from URI. URI is formed by three main components: scheme, authority and path. Only the path after the type identifier is used to calculate the similarity. Our solution explored four different methods: Levenshtein, Jaro, Babelnet and Wordnet (cf. Subsection 4.2 for the explored similarity methods).

In the fourth step (mapping creation), similarity values are aggregated by using the *LambdaMart* method. Our solution generates the final classification (pair of entities expressing the mapping). Each source KG entity receives an alignment with the candidate entity with the best classification. Alignments are filtered to remove multiple alignments to the same entity from the target KG and below than a threshold.

4.1 Learning to Rank for the Alignment of KGs

Figure 2 presents how learning to rank is applied to our study context. First, the rank aggregation model is created using a training set (cf. A in Figure 2). Second, the pair of source KG entity e_s and the target KG entity e_t have their similarity calculated for each similarity method *sim* (cf. Subsection 4.2) and the triple (e_s, e_t, sim) is recorded as an entry for the system (cf. B in Figure 2). Third, each entity of the source KG is used as a query to retrieve all similarity entries that form different ranks for each similarity method (cf. C in Figure 2). In the last step, retrieved ranks are aggregated using the trained model and return one rank as result (cf. D in Figure 2).

Learning to Rank techniques like *lambdaMart* (Burges, 2010) are supervised learning tasks. It needs training and testing phases. In the training phase, a model is created to be used later to aggregate ranks (cf. A in figure 2). The training data consists of queries set Q as a set of documents to be retrieved D and a set of possible labels Y. The training set S is formed by triples (q_i, d_j, y_{ij}), where $q \in Q$, $d \in D$ and $y \in Y$. The set $Y = 1, 2, ..., n; n \in N$ is the relevance of the document d_i for the query q_j. The process to put relevance to the pair query and document is analogous to the labeling in other techniques.

In our approach, a set of queries Q is formed by URI from source KG's entities; the set of retrieved documents D is formed by candidate entities from target KG's entities; and the set of labels Y receives a value according to the presence of the pair in the gold standard. If the pair (e_i, e_j) is in the gold standard, the triple (q_i, d_j, y_{ij}) receives values $(e_i, e_j, 1)$; and the triple receives value $(e_i, e_j, 0)$ if not present.

The training model creates a ranking model $f(q,d) = f(x)$ that assigns a score to a given pair query and document. For a set, the training model creates a ranking model $F(q,D) = F(X)$ that returns a list of scores that can be converted to a ranking of documents using the score from $f(q,d)$ to sort the documents D (LI, 2011). In our approach, the model creates a ranking model $F(e_i, C) = F(X)$, where e_i is an entity from source KG and C is the set of candidate entities from target KG. This model is used to sort all candidate entities for a certain entity from source KG.

4.2 Similarity Techniques

Our proposed technique explores four method for similarity computation to generate rankings to be aggregated.

- Levenshtein similarity, also known as edit distance, between two strings is the minimal number of insertions, deletions, and replacements to make two strings equal (Navarro, 2001).
- Jaro similarity between two string is shown in equation 7, where m is the number of matching characters, t is the number of transposition, $|s_1|$ and $|s_2|$ are string length.

$$Jaro(s_1, s_2) = \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right) \quad (7)$$

- Path-similarity is based on WordNet synset, as groups of synonymous words. The similarity between the two terms is the shortest path that connects the senses in the "is-a" taxonomy.
- Weighted Overlap is based on NASARI vector constructed using WordNet synsets and Wikipedia pages (Camacho-Collados et al., 2015). Given a pair of words w_i and w_j the algorithm checks if they are synonyms, returning a maximum similarity score if true. If they are not synonyms, it gets their respective NASARI vector and calculates weighted overlap synonyms. Weighted Overlap sorts the elements of each vector and harmonically weights the overlap between two vectors. Equation 8 defines weighted overlap for two vectors v_i and v_j, where O is the set of overlapping dimensions between the vectors and r_q^j is the rank of dimension q in the vector v_j.

$$WO(v_i, v_j) = \frac{\sum_{q \in O}(r_q^i + r_q^j)^{-1}}{\sum_{k=1}^{|O|}(2k)^{-1}} \quad (8)$$

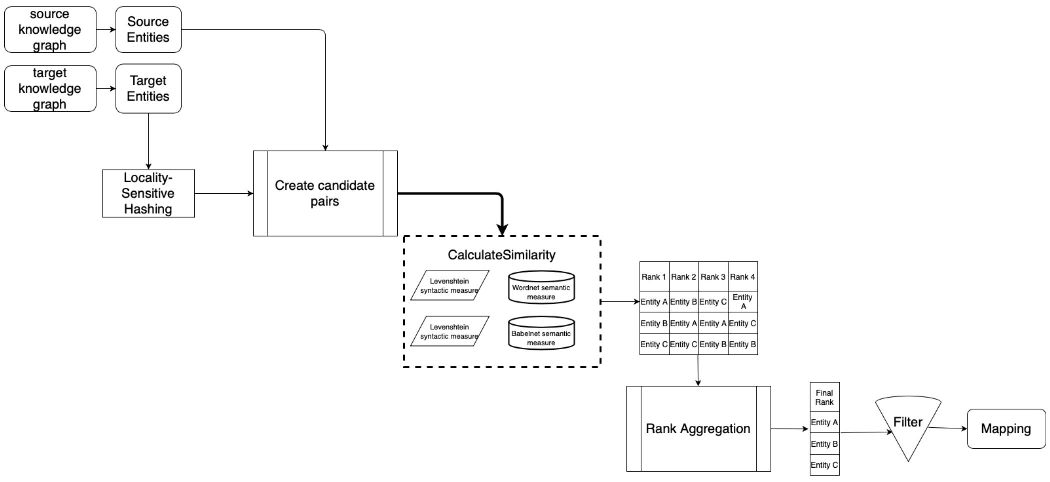

Figure 1: Our mapping technique workflow.

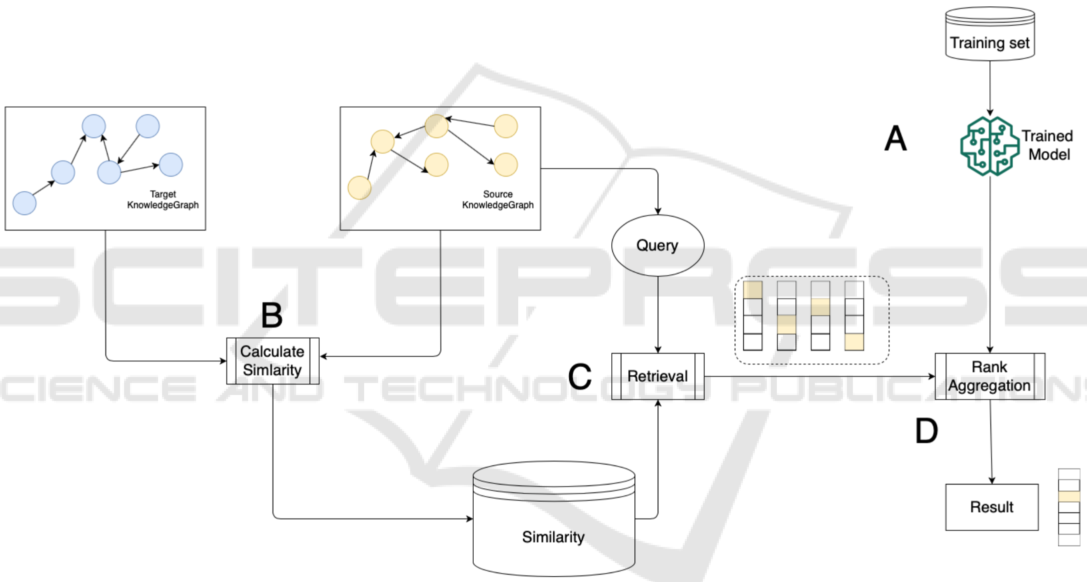

Figure 2: Learning to Rank technique applied to our approach (based on (Cummins and Briscoe, 2015)).

5 EXPERIMENTAL EVALUATION

Our goal is to analyze the quality of mappings generated in our approach for KG alignment. In the developed experiments, we used datasets from the OAEI (Ontology Alignment Evaluation Initiative) – Knowledge Graph Track released on 2019[7]. Datasets were created running DBpedia extraction framework on Wikis from the Fandom Wiki hosting platform (Hertling and Paulheim,). Each instance entity is created from a wiki page, and one triple is created for each entry in an infobox (Hertling and Paulheim, 2018). Table 1 describes statistics from the datasets used in our experimental evaluation. The source column has the name of the KG and the acronym used in this work.

The mappings created by our proposed approach were compared with the gold standard offered by OAEI – Knowledge Graph Track (2019 edition). Experts created the schema-level maps. Instance level maps were extracted using links present in sections with headers containing "link" to the corresponding page of another wiki (e.g. "External links"), removing all links where the source page linked to more than one page in another wiki and multiple links to the same concepts to ensure injectivity. The gold standard

[7]http : / / oaei.ontologymatching.org / 2020 / knowledgegraph/index.html

Table 1: Statistics of Knowledge Graphs (Hertling and Paulheim,).

Source	Hub	Topic	# Instances	# Properties	# Classes
Star Wars Wiki (SWW)	Movies	Entertainment	145,033	700	269
The Old Republic Wiki (TOR)	Games	Gaming	4,180	368	101
Star Wars Galaxies Wiki (SWG)	Games	Gaming	9,634	148	67
Marvel Database (MDB)	Comics	Comics	210,996	139	186
Marvel Cinematic Universe Wiki (MCU)	Movies	Entertainment	45,828	325	181
Memory Alpha (MAL)	TV	Entertainment	45,828	325	181
Star Trek Expanded Universe (STX)	TV	Entertainment	13,426	202	283
Memory Beta (MBT)	Books	Entertainment	51,323	423	240

is a partial gold standard, because it does not contain all correct matches. A trivial match is an exact string match of the label, and a non-trivial match is when a string match is not exact (Hertling and Paulheim, 2020). Table 2 describes statistics of the gold standard used in our experiments.

The learning process used 100% of the mappings to train and validate between the datasets Star Wars Wiki and Star Wars Galaxies Wiki; Star Wars Wiki and The Old Republic Wiki; Memory Alpha and Memory Beta; and Memory Alpha and Star Trek Expanded Universe. Mappings between Marvel Cinematic Universe and Marvel Database were isolated to be used as an evaluation set. To this end, we applied our solution to them and analyzed the results based on objective metrics. MinHash Locality-sensitive hashing used 256 permutation, threshold of 0.75 and each entity were converted to a set of trigrams of entity name to hash. Trigram is a sequence of three consecutive character. We used three metrics to evaluate the results: Precision, Recall and F-Measure (F1-score).

Table 3 presents the precision, recall, and f1-score results for schema type. In the class type, we achieved precision higher than 0.5 in the datasets "Star Wars Wiki" and "Star Wars The Old Republic Wiki" (SWW-TOR). In the datasets, "Memory Alpha - Memory Beta" (MAL-MBT) and "Memory Alpha and Star Trek Expanded Universe" (MAL-STX), our solution presented recall lower than 0.9. The class matching for "Memory Alpha" had more Non-trivial mapping than other data sets, leading to lower recall. In analyzing the mappings regarding the properties, all datasets presented precision near 0.5 and recall higher than 0.9.

Table 4 presents the results for the mappings connecting the instances of the KGs. All datasets presented low precision, but achieved a recall higher than 0.7 except "Marvel Cinematic Universe - Marvel Database" (MCU-MDB) and "Star Wars Wiki and Star Wars Galaxies" (SWW-SWG).

Figures 3 and 4 compare our approach to the baseline matchers offered by OAEI Knowledge Graph Track organization. The baseline matcher used the label for each entity to create a mapping. The baseline matcher matches all resources which share the same *rdfs:label*. The baseline Alt Label additionally uses *skos:altLabel* as a predicate. Both baseline matchers used cross-product for all resources that have a common label.

Our approach and both baselines found all class mappings. Our approach found some false positive classes, so it lowered precision and f-measure. Our approach found all properties mapped and baseline found 36%, but all mapping found by baseline is correct. Overall, our approach exceeded the baseline for properties. Figure 3 shows the results obtained by our approach and the baseline matchers for schema-type entities mapping in "Marvel Cinematic Universe" (MCU) and "Marvel Database" (MDB).

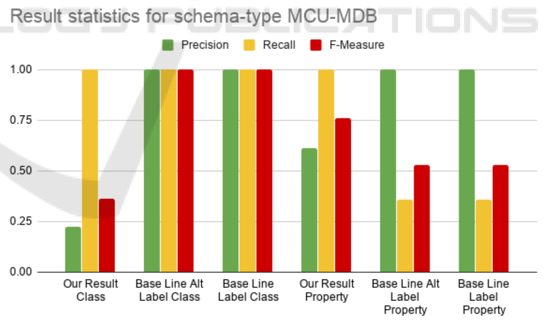

Figure 3: Comparison between our results and competition base line for schema-type entities in "Marvel Cinematic Universe" (MCU) and "Marvel Database" (MDB) mappings.

For instances, type of entities baseline presents better results than our approach for precision and recall. Figure 4 shows results for instance-type entities in "Marvel Cinematic Universe" (MCU) and "Marvel Database" (MDB). Baseline matcher presents better precision for all cases, but for recall the result is mixed. Our approach had better recall for property, same recall for class and worst for instance.

Table 2: Gold Standard statistics (Hertling and Paulheim, 2020).

Mapping	Class Matches		Property Matches		Instance Matches	
	Total	Non-trivial	Total	Non-trivial	Total	Non-trivial
SWW-SWG	5	2	20	0	1,096	528
SWW-TOR	15	3	56	6	1,358	220
MCU-MDB	2	0	11	0	1,654	726
MAL-MBT	14	10	53	4	9,296	2,140
MAL-STX	13	6	41	3	1,725	274

Table 3: Mapping results for schema-type entities applying our solution.

Mapping	**Type**	**Precision**	**Recall**	**F1**
SWW-SWG	class	0.385	1.000	0.556
	property	0.435	1.000	0.606
SWW-TOR	class	0.515	0.944	0.667
	property	0.514	0.966	0.671
MCU-MDB	class	0.222	1.000	0.364
	propety	0.611	1.000	0.759
MAL-MBT	class	0.217	0.333	0.263
	property	0.559	0.963	0.707
MAL-STX	class	0.307	0.571	0.400
	property	0.549	0.975	0.703

Table 4: Mapping results for instance-type entities applying our solution.

Mapping	**Precision**	**Recall**	**F1**
SWW-SWG	0.337	0.494	0.400
SWW-TOR	0.273	0.746	0.400
MCU-MDB	0.357	0.460	0.403
MAL-MBT	0.298	0.739	0.425
MAL-STX	0.228	0.794	0.354

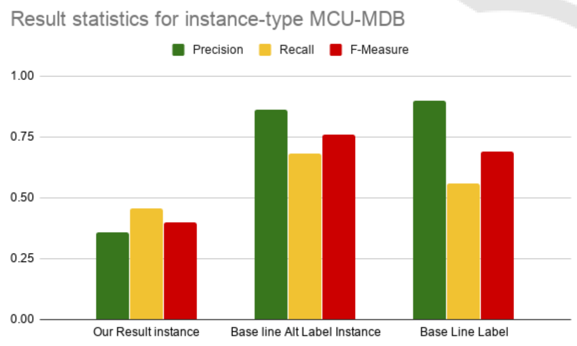

Figure 4: Comparison between our results and competition base line for instance-type entities in “Marvel Cinematic Universe” (MCU) and “Marvel Database” (MDB) mappings.

6 DISCUSSION

This investigation aimed to create mappings between KGs based on rank aggregation methods. Our results by experimenting our approach showed high recall for schema-type, but penalized the precision. For instance level type of entities, our approach presented low precision and acceptable recall.

The main difference between schema-type and instance-type mapping generations is in how candidate entities were created. Our approach used all entities as candidate entities for schema-type, but candidate entities are filtered using locality-sensitive hashing for instance type. For instance-type, it is not possible to use a cross-product approach to compare entities, because the number of instance-type entities is very large, so it needs to reduce the number of comparisons. For this reason, we were unable to calculate the similarity for all possible pairs. This difference is more clear for SWW-SWG and MCU-MDB, because both datasets have proportionally more non-trivial mappings than others.

Our approach found false positive mappings which affected the precision for almost all datasets studied. The explored gold standard is based on links created by the Wiki community on the page where entities were extracted. It means that the presence of mapping between entities depends on the interest of the community to enrich those pages.

Another case of false positive was caused by different specificities of the entities. In this case, the matcher creates new mappings between an entity and the more general entity, but the correct mapping was for the more specific entity. For example, our technique created a mapping between “Michael Duffy” from “Marvel Cinematic Universe” and “Michael Duffy” from “Marvel Database”. However, the correct answer was between “Michael Duffy” from “Marvel Cinematic Universe” and “Michael Duffy (Earth-616)” from “Marvel Database”.

Our approach found mapping with good recall for most cases in schema-type entities. With property, it exceeded baseline recall. Our approach uses a learning-to-rank technique that can be improved with more similarity techniques to aggregate ranking and more datasets without changing code structure.

7 CONCLUSION

The alignment of KGs remains an open research challenge. In this work, we proposed an approach based on rank aggregation and locality-sensitive hashing to create mappings between distinct KGs. Our approach used the entity URI to extract the set used to explore locality-sensitive hashing and similarities. We explored the hashing and four similarity techniques to create independent rankings that were aggregated using learning-to-rank techniques (in particular, we explored *lambdaMart*). We implemented the proposal and carried out experiments using OAEI competition datasets. Our solution was able to find most of the mappings between schema-level entities (good recall) although improvements are needed in terms of precision. Future work involves exploring more information from entities to get better results for hashing. We plan to explore other similarity techniques that do not use string as the main component. We also plan to experiment with our solution with additional datasets.

ACKNOWLEDGEMENTS

This work was supported by the São Paulo Research Foundation (FAPESP) (Grant #2022/15816-5)[8].

REFERENCES

Burges, C. J. (2010). From ranknet to lambdarank to lambdamart: An overview. Technical Report MSR-TR-2010-82, Microsoft Research.

Camacho-Collados, J., Pilehvar, M. T., and Navigli, R. (2015). NASARI: a novel approach to a semantically-aware representation of items. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 567–577, Denver, Colorado. Association for Computational Linguistics.

Chang, F., Chen, G., and Zhang, S. (2019). Fcamap-kg results for oaei 2019. In *Ontology Matching at International Semantic Web Conference, OM@ISWC*.

Cummins, R. and Briscoe, T. (2015). Learning to rank. *Advanced Topics in Natural Language Processing*.

Faria, D., Pesquita, C., Santos, E., Palmonari, M., Cruz, I., and Couto, F. (2013). The agreementmakerlight ontology matching system. In *On the Move to Meaningful Internet Systems, pp 527-541*, volume 8185.

Hertling, S. and Paulheim, H. Knowledge graph track. http://oaei.ontologymatching.org/2020/knowledgegraph/index.html.

Hertling, S. and Paulheim, H. (2018). Dbkwik: A consolidated knowledge graph from thousands of wikis. In *2018 IEEE International Conference on Big Knowledge (ICBK)*, pages 17–24.

Hertling, S. and Paulheim, H. (2019). Dome results for oaei 2019. In *OM 2019 : Proceedings of the 14th International Workshop on Ontology Matching co-located with the 18th International Semantic Web Conference (ISWC 2019) Auckland, New Zealand, October 26, 2019*, volume 2536, pages 123–130, Aachen. RWTH.

Hertling, S. and Paulheim, H. (2020). *The Knowledge Graph Track at OAEI: Gold Standards, Baselines, and the Golden Hammer Bias*, pages 343–359. Springer; 1st ed. 2020 edition.

Hogan, A., Blomqvist, E., Cochez, M., d'Amato, C., de Melo, G., Gutierrez, C., Gayo, J. E. L., Kirrane, S., Neumaier, S., Polleres, A., Navigli, R., Ngomo, A.-C. N., Rashid, S. M., Rula, A., Schmelzeisen, L., Sequeda, J., Staab, S., and Zimmermann, A. (2020). Knowledge graphs.

Jaccard (1912). The distribution of the flora of the alpine zone. In *New Phytologist*, volume 11, pages 37–50.

Juliana M. Destro, Javier A. Vargas, J. C. d. R. and da S. Torres, R. (2019). Exploring rank aggregation for cross-lingual ontology alignments. *14th International Workshop on Ontology Matching co-located with the 18th ISWC*.

Leskovec, J., Rajaraman, A., and Ullman, J. D. (2014). *Mining of Massive Datasets*. Cambridge University Press, USA, 2nd edition.

LI, H. (2011). A short introduction to learning to rank. *IEICE Transactions on Information and Systems*, E94.D(10):1854–1862.

Navarro, G. (2001). A guided tour to approximate string matching. *ACM Comput. Surv.*, 33(1):31–88.

Ringler, D. and Paulheim, H. (2017a). One knowledge graph to rule them all? analyzing the differences between dbpedia, yago, wikidata & co. In Kern-Isberner, G., Fürnkranz, J., and Thimm, M., editors, *KI 2017: Advances in AI*, pages 366–372, Cham. Springer.

Ringler, D. and Paulheim, H. (2017b). One knowledge graph to rule them all? analyzing the differences between dbpedia, yago, wikidata & co. In *978-3-319-67189-5*, pages 366–372.

Singhal, A. (2014). Introducing the knowledge graph: things, not strings. https://googleblog.blogspot.com/2012/05/introducing-knowledge-graph-things-not.html.

Tsai, Y. and Yang, M. (2014). Locality preserving hashing. In *2014 IEEE International Conference on Image Processing, ICIP 2014*, IEEE International Conference on Image Processing (ICIP 2014), pages 2988–2992, United States. IEEE.

[8]The opinions expressed in this work do not necessarily reflect those of the funding agencies.