SimE4KG: Distributed and Explainable Multi-Modal Semantic Similarity Estimation for Knowledge Graphs

Carsten Felix Draschner
Information Systems and Artificial Intelligence
University of Bonn
Bonn, Germany
0000-0002-1006-146X

Hajira Jabeen
Big Data Analytics, GESIS
Leibniz Institute for the Social Sciences
Cologne, Germany
0000-0003-1476-2121

Jens Lehmann
Amazon
(Done while at Uni. Bonn)
Dresden, Germany
0000-0001-9108-4278

***Abstract*—In recent years, more and more exciting sources of data have been modeled as Knowledge Graphs (KGs). This modeling represents both structural relationships and the entity specific multi-modal data in KGs. In various data analytic pipelines and Machine Learning (ML), the task of semantic similarity estimation plays a significant role. Assigning similarity values to entity pairs is needed in recommendation systems, clustering, classification, entity matching/disambiguation and many others. Efficient and scalable frameworks are needed to handle the quadratic complexity of all pair semantic similarity on Big Data KGs. Moreover, heterogeneous KGs demand multi-modal semantic similarity estimation to cover the versatile content like categorical relations between classes or their attribute literals like strings, timestamp or numeric data. In this paper we propose SimE4KG framework as a resource providing generic open source modules that compute semantic similarity estimation in multi-modal KGs. To justify the computational costs of similarity estimation, the SimE4KG generates reproducible, reusable and explainable results. The pipeline results are a native semantic RDF-KG, including the experiment results, hyper-parameter setup, and explanation of the results, like the most influential features. For fast and scalable execution in memory, we implemented the distributed approach using Apache Spark. The entire development of this framework is integrated into the holistic distributed semantic analytics stack SANSA.**

***Index Terms*—Semantic Similarity, Knowledge Graphs, Knowledge Engineering, Distributed Computing, Explainable Artificial Intelligence, Scalable Semantic Processing, RDF, Apache Spark, Machine Learning**

I. Motivation

In the field of data analytics and Machine Learning (ML), there are many applications based on the similarity estimation of entities. On the one hand, there are data optimizing algorithms like entity disambiguation, entity resolution, and unsupervised classification. On the other hand, some algorithms produce results to improve the accessibility of content references or recommendations like clustering or recommendation systems. One common data source for these analytic pipelines is heterogeneous Knowledge Graphs (KGs). These KGs represent information about entities in linked data structures[1] [1]–[3]. For a standardized data representation, the W3C standard RDF is used [4]. The integrated data are multimodal and require type-specific handling for proper similarity estimations. Moreover, the results of these algorithms should be reproducible and reusable so that the effort of the predictions can be validated, and the predictions can also be reused in building applications. In addition, it is desirable to have explainable results such that the use of results is supported in interpreting them and maybe further adjusting to the desired outcome instead of being confronted with an ML black box. Further, the user should be capable of influencing the ML result to the desired behavior. Due to the sheer size of RDF KGs, which in the era of big data no longer fit into the main memory of classical consumer devices, and due to the quadratic complexity of all pair similarity, the solution calls for generic, scalable distributed similarity estimation frameworks. The advantage of distributed computing is that individual computers cannot be scaled arbitrarily vertically since, at certain performance levels, the costs increase not proportionally when leaving consumer hardware or individual hardware is no longer feasible. Providing such a framework with good documentation and open-source is crucial since this ensures transparency and adaptability. An easy-to-use open-source framework is missing that offers explainable and adjustable semantic similarity estimation for multi-modal KGs, simultaneously capable of scaling horizontally to a multi-node cluster. The main contributions of this work are:

- Semantic similarity estimation SimE4KG algorithm for multi-modal RDF KGs.
- Novel generic distributed in memory downstream machine learning modules within the open-source SimE4KG framework, which is specially designed to scale with heterogeneous KGs. [5].
- Introduction of multi-dimensional weighting techniques of domain-aware similarity scores.
- Semantification of similarity estimation improves the explainability, reusability, and reproducibility, completing the original RDF Knowledge Graph.
- Documentation within ReadMe, Scala Docs, and sample

[1]https://lod-cloud.net

pipelines with accessible notebooks, integrated into the holistic Semantic Analytics Stack SANSA [6].

II. Preliminaries

a) Apache Jena: is an open-source Apache framework [7] programmed in Java to develop Linked Data, RDF Data, and Semantic Web programs.

b) Apache Spark: is a framework for cluster computing [8]. It is available under the Apache Open Source License. Apache Spark encapsulates software modules that optimize the execution of big data analytics pipelines [9].

c) Resource Description Framework (RDF): is the W3C standard to represent metadata and designed as a core technology for building a web of data [4] with the envisioned goal to realize the Semantic Web [10].

*d) SANSA - **S**emantic **An**alytics **S**t**a**ck:* is an open source framework to process large scale RDF data based on Apache Spark, Apache Flink, and Apache Jena, within various tasks like: semantic data representation, querying, inference, and analytics [11].

e) Scala: is an object-oriented and functional programming language. Scala uses static types to reduce bugs in complex applications. One can naively use Java libraries within Scala code, as Scala is being executed within the JVM (Java Virtual Machine) [12].

III. Related Work

A. Knowledge Graphs (KG)

In the last years, more and more KGs have been created, such as Wikidata [13], DBpedia [1], YAGO [3], FreeBase [2], Linked Movie Database [14] and many more. KGs are used as a representation of the data integration of linked open data [15] on the Semantic Web [10]. The W3C standard RDF [4] serves as a possibility to form uniform terms by URIs and IRIs and thus facilitate data integration.

B. Distributed KG Machine Learning (ML)

These KGs have become so large that they no longer fit into conventional computers' RAM. Therefore distributed ML frameworks are necessary. Technologies have been developed for latent embedding based pipelines [16]–[18] and also feature based pipelines [11], [19], [20]. These enable horizontally scalable ML pipelines on KGs. Due to a lack of explainability, especially in latent embedding [21], classical feature-based approaches can be used [22], [23] to explain the semantics behind ML.

C. Explainable and Reproducible KG ML

KG-based ML and Semantic Similarity Estimations for KG are multidimensional [24]. Several approaches have been developed to improve it. On the one hand, the results become better explainable by naively annotating the environment and hyper-parameter setup of the experiment pipeline through metadata models [25], [26]. On the other hand, latent embeddings are only partially explainable [21], [27] where latent ML approaches can be either transformed into feature-based models [23] or rely on classical features right away [22].

D. Semantic Similarity Estimation:

For various tasks (see Section I and Section IV-G0b), similarity scores are needed to calculate the similarity between entities. Various scores have been introduced in the last century based on a feature basis (Jaccard [28], Tversky [29], Batet [30]), distance in the graph (Shortest Path [31], Weighted Links [32], Wu&Palmer [33]), or information content (Resnik [34], Lin [35]). Frameworks that apply these similarity scores to actual data have been developed [36]–[38]. Based on these approaches, scalable similarity estimations were developed for distributed cluster computation using Apache Spark [8], [39], [40], based on the probabilistic approach Minhash locality sensitivity hashing [41] and Bucket Random Projection [8]. DistSim [37] was developed and integrated into the SANSA Stack [11]. It uses the first feature-based similarity scores for Apache Spark but does not separate the features by their types and considers all neighboring nodes in the graph as categorical features [37]. Due to the complexity of all pair similarities, a scalable solution is needed; the ML results should be reproducible and reusable. While many similarity assessment measures exist, they do not cover the multiple aspects in one approach: multi-modality, distributed scalability, and explainability in conjunction with KGs within a framework.

IV. Similarity Assessment Architecture

This section gives insight into the structure of the SimE4KG framework and the computation of new generic modules within the pipeline for scalable distributed domain-aware multi-modal semantic similarity estimation.

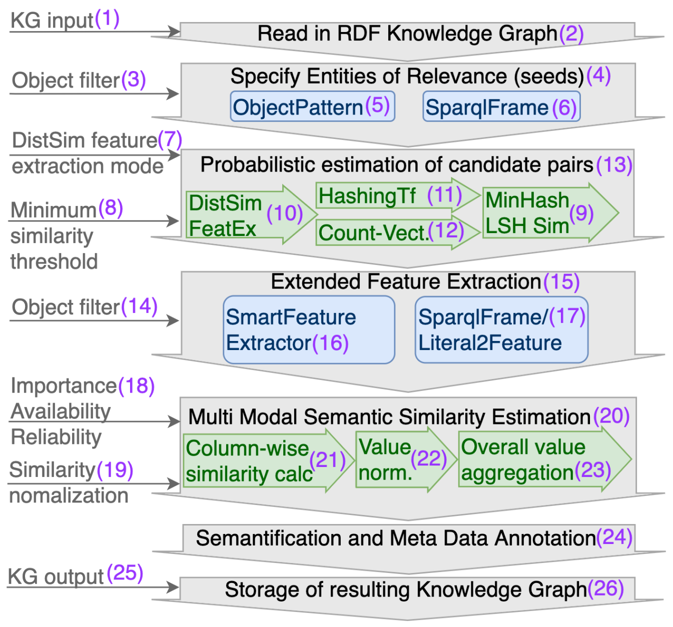
KG input(1)
Read in RDF Knowledge Graph(2)
Object filter(3)
Specify Entities of Relevance (seeds)(4)
ObjectPattern(5)
SparqlFrame(6)
DistSim feature(7) extraction mode
Probabilistic estimation of candidate pairs(13)
Minimum(8) similarity threshold
DistSim FeatEx (10)
HashingTf (11)
Count-Vect.(12)
MinHash LSH Sim(9)
Extended Feature Extraction(15)
Object filter(14)
SmartFeature Extractor(16)
SparqlFrame/(17) Literal2Feature
Importance(18) Availability Reliability
Multi Modal Semantic Similarity Estimation(20)
Column-wise similarity calc(21)
Value norm.(22)
Overall value aggregation(23)
Similarity(19) nomalization
Semantification and Meta Data Annotation(24)
KG output(25)
Storage of resulting Knowledge Graph(26)

Fig. 1: SimE4KG Pipeline Overview

A. Pipeline Architecture

The SimE4KG framework execution is performed by a pipeline of generic newly developed modules. Figure 1 shows the structure of the pipeline. In the following subsections, IV-B-IV-F, the approaches of the single steps are described, and technical insights into the implementation are given. The pipeline consists of seven steps (see Fig. 1). Initially, the data is Read-In. The relevant entities of the KG that should be within the similarity assessment can then be specified. Subsequently, a probabilistic approach is used to calculate promising candidates. Extensive and detailed features are extracted from the KG for these promising candidates. Similarity scores are calculated and accumulated for the promising candidate pairs based on the multi-modal features and weights. The results of the semantic similarity estimation are transformed into a semantic native form, including the pipeline metadata like the hyper-parameters. Finally, the RDF Knowledge Graph data is stored.

a) Pipeline Architecture - Technical Implementation Details: Many new modules were developed during the technical implementation based on existing widely used libraries and frameworks. Data management is implemented by the Hadoop File System (HDFS). All pipeline modules use Apache Spark [8] for distributed execution and Apache Jena [7] for native handling of Semantic RDF KGs. The reading and writing of RDF data are implemented by the SANSA RDF layer [11]. The initial optional filtering of the entities is implemented by the triple pattern matching options or the SparqlFrame Transformer [20]. The probabilistic gathering of candidate pairs reuses the DistSim [37] pipeline (see Subsection IV-B). For feature extraction, the significantly faster SmartFeatureExtractor was developed as an alternative to the existing SparqlFrame/Literal2Feature [20], [22] Transformer (see Subsection IV-C). The multi-modal semantic similarity estimation and the semantification of the results were implemented as new Transformers (see Subsection IV-D-IV-E and List. 1), which are aligned to the well-known libraries, e.g. Apache Spark MLlib and scikit-learn [9], [42].

```
val ds: Dataset[Triple] = spark.rdf([...]
val SimE4KG = new DaSimEstimator()
  .setObjectFilter([...]
  .setDistSimFeatureExtractionMethod("os")
  .setSimilarityValueStreching(true)
  .setImportance(Map("actor_sim" ->
      0.2[...]
val similarityDf: DataFrame = SimE4KG
  .transform(ds)
similarityDf.show()
```

Listing 1: Example code usage of SimE4KG module

B. Probabilistic Gathering Candidate Pairs

The RDF KG is read by using the existing SANSA RDF layer API [11] (see Fig. 1 (1,2)). Afterward, the KG is available as a Spark Dataset of Jena Triples. Since the complexity (see Fig.7 top left) of multi-modal and domain-aware similarity scores has, in principle, a quadratic running time, the probabilistic feature-based similarity score is performed based on the DistSim approach [37]. For this, the relevant entities are initially determined by a filter in the KG (see Fig. 1 (3-6)). A SPARQL query or a triple filter can be used based on the object, such as a concrete rdfType. The object filter method is faster (see Evaluation and Fig. 10) but allows less complexity for the specification. The SPARQL filter is executed by the SparqlFrame transformer, a generic SANSA stack module that internally uses SPARQLIFY [20], [43]. For the filtered entities, features are extracted using the DistSim feature extractor. These features are transformed into numerical feature vectors using two different options. The options are either the HashingTF or the CountVectorizer of Spark [8], [9]. For the selected entities, the upper triangular matrix of similarity scores is computed and filtered by a minimum threshold of entity pair similarity. The similarity scores are generated using the MinHashLSH Apache Spark implementation [40] (see Fig. 1 (7-13)). Since the DistSim approach treats all KG features as categorical features within the MinHashLSH, this step is only used as a first guess to reduce the complexity of the multi-modal estimations (see Subsec. IV-C & IV-D). The result is the basis for all entity pairs to be computed in the multi-modal semantic similarity score procedure.

```
val ds: Dataset[graph.Triple] = [...]
/** Smart Feature Extractor */
val sfe = new SmartFeatureExtractor()
  .setSp[...]
/** Feature Extracted DataFrame */
val featureExtractedDF: DataFrame = sfe
  .transform(ds)
featureExtractedDF.show()
```

Listing 2: Example code usage of SimE4KG Pipeline Transformer SmartFeatureExtractor

entity (URI)	runtime (Double)	actors (Array[URI])	release date (Timestamp)	title (String)	fN ...
https://.../film1	141.0	[htt//.../a4, ...a8]	2002-01-01	Catch Me If y...	...
https://.../film2	142.0	[htt//.../a1, ...a2]	1994-01-01	The Shawshan...	...
https://.../film3	189.0	[htt//.../a1, ...a4]	1999-01-01	Green Mile	...
https://.../filmN	...	...	...	...	...

TABLE I: Sample multi-modal result SmartFeatureExtractor

C. Feature Extraction

Based on the feature extractor implementations in the SANSA stack (*FeatureExtractorModel*, *SparqlFrame*), a feature extractor Transformer *SmartFeatureExtractor* was developed that combines the strengths of both. The speed is increased by using the parallelizable and efficient pivot function in Apache Spark [8], but also the resulting DataFrames are collapsed, so that for each sample, a row exists [20] (see Fig. 1 (14-17) and Fig.2 (1-3)). The columns are automatically annotated to the literal types[2] mounted when given so that native operations can be performed on the given values (see Table I and Fig. 2 (3,4)). This predicate-based and multi-modal feature extraction significantly improves the approach

[2]https://www.w3.org/TR/swbp-xsch-datatypes/

from DistSim. The similar functionality of the *SparqlFrame* requires a call to the underlying SPARQLIFY [22], [43] framework, which introduces significant time complexity, although this is essential due to the higher variability of a SPARQL query in certain use cases. So, for most use cases, the new *SmartFeatureExtractor* enables faster (see Sect. V) but also domain-aware and multi-modal feature extraction in contrast to the DistSim feature extractor model [37] and SparqlFrame Feature Extractor [20].

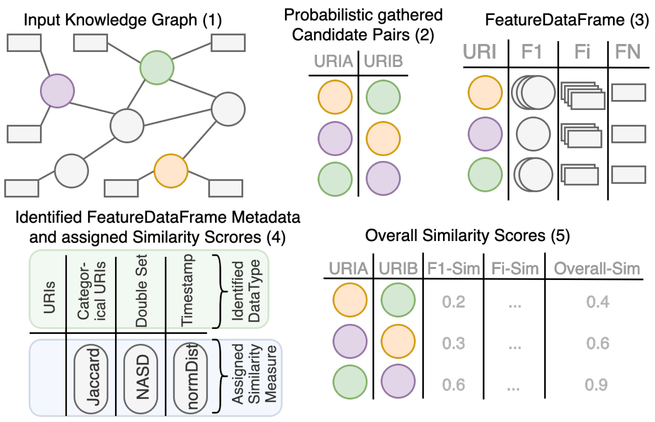
Input Knowledge Graph (1)
Probabilistic gathered Candidate Pairs (2)
URIA
URIB
FeatureDataFrame (3)
URI
F1
Fi
FN
Identified FeatureDataFrame Metadata and assigned Similarity Scores (4)
URIs
Categor-ical URIs
Double Set
Timestamp
Identified DataType
Jaccard
NASD
normDist
Assigned Similarity Measure
Overall Similarity Scores (5)
URIA
URIB
F1-Sim
Fi-Sim
Overall-Sim
0.2
...
0.4
0.3
...
0.6
0.6
...
0.9

Fig. 2: SimE4KG Data Transformations

D. Domain Aware and Multi-Modal Similarity Estimation

For a final assignment of a similarity score, individual feature-specific similarity scores are calculated based on all extracted features (see Fig. 1 and Fig. 2 (3-5)). Matching similarity scores are automatically assigned and used for different multi-modal feature types. For example, for categorical feature lists such as an arbitrary number of genre annotations, similarity scores are computed using the Jaccard index [28], [37]. A normalized distance across the whole feature distribution is calculated for continuous values like runtime. For lists of timestamps, a normalized mean distance is calculated as the similarity score based on a transformed $unix\_timestamp$ representation of these features (see Fig. 2 (4) green and blue). Thus, all features are assigned a similarity score between 0.0 (no similarity) and 1.0 (exact similarity). Subsequently, all similarity scores can be normalized (see Listing 1) again to carry domain awareness correctly on the characteristics of the different KGs and similarity scores (see Fig.2 (5)). The supported and mapped similarity scores can handle the following multi-modal KG data types: Boolean, String, TimeStamp, Double, Int, URI/IRI [7], [8].

E. Weighting of Features and overall Similarity Assessment

The decided feature similarities are aggregated in a final overall similarity score based on a weighted sum. Likewise, the most significant feature similarities for explainability are collected. However, individual similarity scores can also be weighted. Several weights can be specified by the user as parameters or are automatically derived from a holistic data analysis of the KG. The weights are the following:

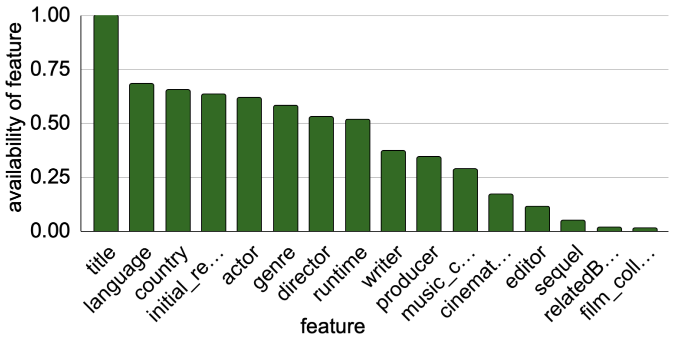

Fig. 3: Extract of feature availability within LMDB dataset of type movies

a) Importance: With the optional importance parameter, the user can give SimE4KG a personal preference for the relevance of features. For example, if this pipeline is an implementation of a recommendation system in the field of a movie database, the user can assign increased importance to concrete feature similarities such as a high similarity of genre and actors (see List. 1 line 6). The importance is optional; if it is not specified as a parameter, all features have equally distributed importance. If the importance is given, the respective values must sum up to one.

b) Availability: The availability weight is automatically calculated. The non-null, non-empty, or non-nan values accumulate to the availability ratio for each feature. It is used in the case of weighting similarity scores when a score is calculated from two entities concerning a concrete feature. In some cases, the feature value is not provided for one of the entities. Due to the open-world assumption, this can always happen (see Fig. 3). However, if, in principle, a feature is often null (low availability), a low similarity in this feature should not negatively impact the overall similarity score.

c) Reliability: It describes the likelihood of the information within a feature being correct; if a particular feature is less reliable, its respective similarity score can be reduced in the final influence of the overall result. The basic idea is that KGs are created by merging and integrating many data sources. The initial data sources building the data integrated KG are differently obtained, differently volatile, and differently ensured to be up to date. In contrast to availability, reliability must be handed over as a parameter because it needs expert knowledge about the data sources not stored within the actual data and not each user can provide.

F. Semantification of Results and Meta Data

The SimE4KG pipeline provides end-to-end RDF KG data handling (see Fig. 1). The results of the similarity estimations can be output as native RDF data. Not only the resulting similarity scores for pairs of entities are mapped (see Fig.4, blue) but also other metadata like the hyper-parameter setup of the pipeline (see Fig. 4, green and yellow). This data is provided for the results' reusability, reproducibility, and explainability. Here the explainability is the annotation of the most influential factors (one or multiple) e.g. for *film1* and *film2* the *most influential factor* is the overlapping *writer* (see Fig. 4 in yellow). Furthermore, the reproducibility is supported by including the hyperparameters of the whole pipeline. The reusability is made possible because the results coexist natively in the original KG (see Fig. 4 in grey) and can thus be included in queries or further analyses.

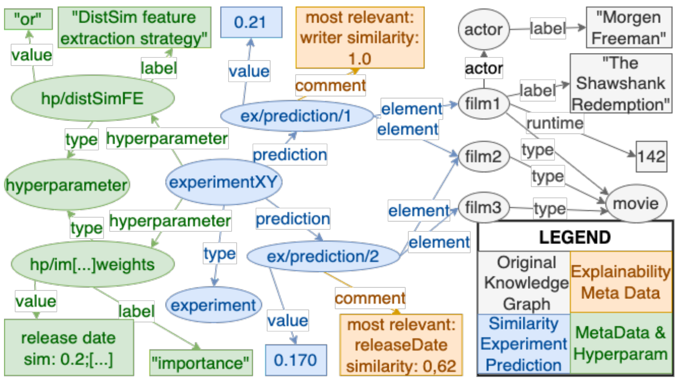

Fig. 4: SimE4KG Semantification Example

G. Impact & Use Cases

a) Potential Impact:: SimE4KG is an interesting resource for the AI/ML, Knowledge Engineering, and Semantic Web community because it offers scalable and multi-modal semantic similarity estimation as an easy-to-use framework integrated into SANSA. All generic modules are adjustable and usable outside the intended context to create arbitrary downstream ML pipelines for RDF KGs. Due to its usability, this source offers a solution to use KGs and ML results for tools like recommendation systems. Also, the metadata encourages the reproducibility of ML experiments. It makes it easier to use semantic data cause of its distributed scalable approach. Working with generic transformers, which are well established in libraries and frameworks, also offers many data scientists access to all the interesting semantic data sources. By semantification of the ML pipeline parameters and results, we introduce how downstream ML and similarity estimation pipelines create native semantic results. These results become explainable and reproducible. This representation also offers many data integration advantages to the source knowledge graph. The resource improves and widens the opportunities of already developed frameworks like the DistSim approach that could not produce explainable results and operate on multi-modal data, while the user can set parameters like the presented weighting.

b) Use Cases: Generic, scalable distributed semantic similarity estimation pipelines for RDF KGs are needed in several follow-up ML and data analytics tasks. The EU Horizon 2020 project PLATOON (Platform for Tools in Energy[3]) uses this framework to analyze large-scale energy RDF data. The database of the PLATOON project was made available for data integration from many data sources in RDF, which especially contain multi-modal features within the Literals. The data is of such a large-scale that individual systems can no longer process it. Another project, Simple-ML[4], develops on top of the generic data analytics options of SANSA a convenient (low-code, no-code) opportunity to stack ML approaches. The SANSA stack supports the data analysis in the Opertus Mundi[5] project.

[3]https://platoon-project.eu

H. Quality, Reusability, and Availability

a) Reusability: The resource is documented by ReadMes, where the framework's ideas, thoughts, and evaluations are presented[6]. All novel modules are documented by scala docs. We provide Hands-On tutorials for sample pipelines within the framework and supply sample Databricks notebooks. Due to the open-source and generic pipeline development approach, the framework is extensible. Within the multiple sources (ReadMe, GitHub Pages, Databricks sample notebooks), the framework is explained and how it can be used.

b) Design and Technical Quality: The resource modules are developed generically and are aligned with common downstream ML pipeline modules like scikit-learn [42] and Apache Spark MLlib [9] transformers where all (hyper-) parameters are set over setters (see List. 1). The maintenance is supported by the scala-docs, and unit tests are automatically called by GitHub actions. The ReadMes and Databricks Notebooks elaborate the downstream tasks to assist the use of specific modules to perform intended or adjusted similarity estimations. SimE4KG is integrated into SANSA framework and it reuses modules from the related approaches: DistSim [37], DistRDF2ML [20], SPARQLIFY [43] and Apache Spark MLlib MinHashLSH [40]. All the frameworks and data used within this tool, all the tools which are used, are openly available and mostly open source. The framework is evaluated in multiple domains, offering performance metrics in data size and processing power scalability, and presents the advantages and disadvantages of the novel introduced modules. The entire resource is available within the open source SANSA GitHub reprository[7]. All modules are merged over a pull request and available within the linked SimE4KG SANSA release[8] [5]. The framework is available under the Apache License 2.0[9]. The framework goes through release cycles twice a year, and novel modules are earlier available in pre-releases.

V. Experiment and Evaluation

In this section, we evaluate the performance and characteristics of the proposed SimE4KG framework. SimE4KG is evaluated in the dimensions of scalability over increasingly complex pipelines (dataset size and hyperparameters) and

[4]https://simple-ml.de
[5]https://www.opertusmundi.eu
[6]http://sansa-stack.github.io/SANSA-Stack/
[7]https://github.com/SANSA-Stack/SANSA-Stack
[8]https://github.com/SANSA-Stack/SANSA-Stack/releases/
[9]https://www.apache.org/licenses/LICENSE-2.0

the scalability of the approach over distributed cluster architectures. The newly developed approaches, such as feature extractor and similarity estimation, are compared with the distributed approaches from related work. All evaluations are presented here, and some further details are documented on our release page. The evaluations were performed on a three-node cluster, with each node having 64 cores and 256 GB RAM, configured with Spark 3.x, Scala 2.12, and Java 11. In addition, experiments were run on a Macbook Pro 16 (2019) with dataset sizes and hyperparameter configurations that did not exceed the available memory. The presented processing times can be reproduced by the available benchmarking and evaluation scripts. The experiments are evaluated on the multi-modal Linked Movie Data Base (LMDB) KG [14] resulting in 10^9 potential candidate pairs (see Fig. 5 and Fig. 7).

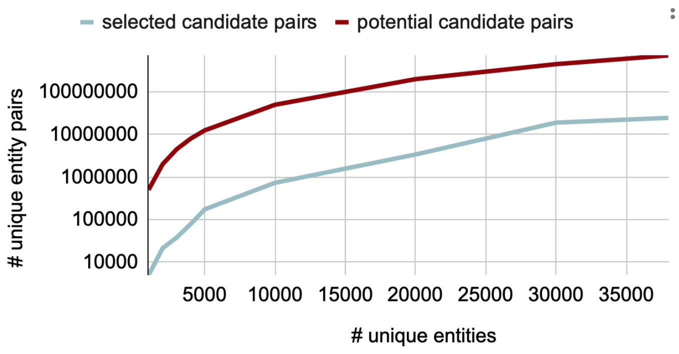

Fig. 5: Evaluation dataset size scalability LSH effect

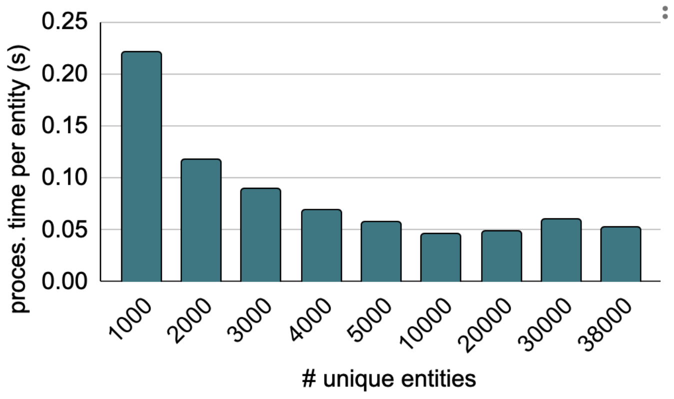

Fig. 6: Evaluation of Spark Overhead by Dataset Size

A. Dataset Size vs. Processing Time

The use of distributed technologies results from the fact that semantic data and KGs can exist in data sizes that exceed the available memory on local computers. However, since memory provides a fast way to process data, the capabilities of distributed in-memory cluster computation are a significant advancement. In this evaluation (see Figs. 5,6,7), we show the scalability of SimE4KG over increasing data set sizes. In Fig. 5, we present that the quadratic complexity of all pair similarities is reduced by two orders of magnitude using the MinHash LSH approach (from 10^9 to 10^7 similarity scores for entity pairs). Fig. 7 shows that the framework scales linearly among, in principle, quadratic increasing candidate pairs computed on the cluster (the 30k+ unique entities already correspond to 10^7 selected candidate pairs and 10^9 potential candidate pairs, see Fig. 5). The spark overhead also is proportionally reduced in more extensive data set size scenarios (see Fig. 6). We also evaluated the scalability of local execution between a linearly increasing number of movie entities and increasing quadratic candidate pairs (see Fig. 5). Additionally, the almost linear scaling is observed among quadratically increasing problems due to the MinHashLSH approach and the usage of Spark Cluster computation.

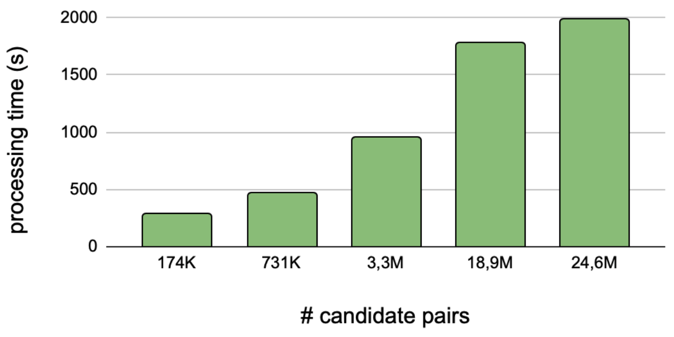

Fig. 7: Evaluation SimE4KG dataset size scalability on LMDB subsets

B. Processing Power vs. Processing Time

The additional performance on cluster computation with the increased number of computation units (cores) can only be used by parallelizable and efficient algorithms. To evaluate the scalability of performance over increasing computational power, we configure our cluster with different numbers of executor cores so that a reduction of the processing time becomes transparent with constant LMDB KG data. Fig. 8 shows how the processing time can be reduced with more processing power. In addition, we would recommend a well-balanced spark cluster setup to even bring down the processing time with the same overall processing power (see Fig. 9). The more executors a cluster has, the more the implementation can be executed in parallel. Still, all those parallel jobs need to be managed and synced by a driver, creating an overhead. Our observation in Fig. 9 shows that having multiple executors (more executors than nodes) but each having still quite a lot of processing power results in faster performance.

C. Comparison of SimE4KG to Distsim and DistRDF2ML

In contrast to existing DistSim frameworks, SimE4KG can deal with the multi-modal nature of RDF KG and produces explainable results that the user can better understand. Moreover, the ideas of the SPARQLIFY-based SparqlFrame are extended for the generic feature extractor. These and other modules have been tested through (hyper-)parameter evaluations. The left table within Fig. 10 presents that the novel SmartFeatureExtractor outperforms the SparqlFrame feature

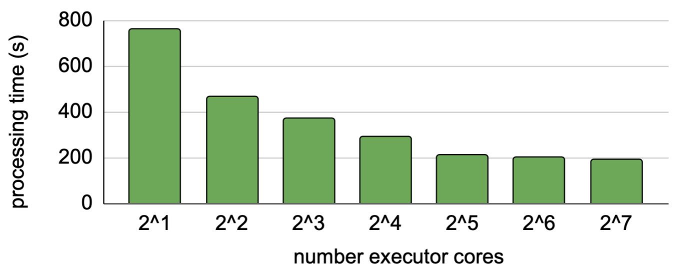

Fig. 8: Evaluation processing power scalability

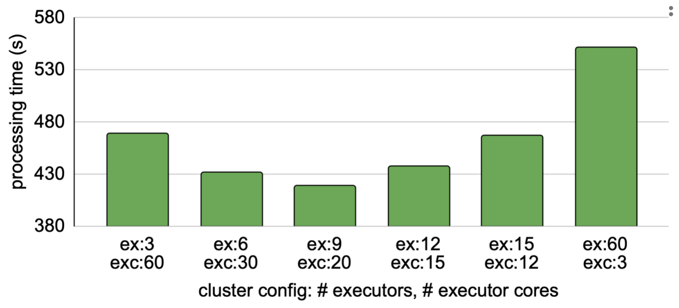

Fig. 9: Evaluation spark cluster configuration effect

extractor in the full 26 features (projection variable = PV) with one layer depth. So if the extended feature extraction (see Fig. 1(15,16,17)) is only interested in the features of distance one in the KG, the SmartFeatureExtractor from SimE4KG should be selected. If a precise and deep traversal of the KG structure is needed, the existing SparqlFrame can be used. In addition, the initial seed gathering (see Fig. 1(4,5,6)) has been extended and evaluated. For the 3,5M triples containing information about roughly 40K entities of type movie, the initial time for collection of seed entities could be reduced from 15 seconds using the SparqlFrame, to 2 seconds with the object pattern matching based on the Apache Spark/Jena Triple matching (The processing time measures shown in Fig. 10 result from local execution on MP16-2019).

#PV	#FeatureEx	prcstime(s)
3	SparqlFrame	41
4	SparqlFrame	49
26	SmartFeatureExtractor	50
5	SparqlFrame	55
6	SparqlFrame	66
10	SparqlFrame	123
26	SparqlFrame	1638

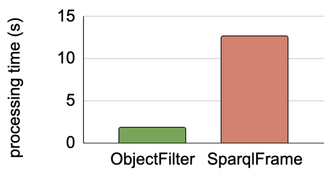

Fig. 10: Evaluation and Comparison to available SANSA modules

VI. Discussion

The evaluation has shown that the developments of SimE4KG are both scalable for increasingly complex data and scale well horizontally over increasing resources. This scalability enables more complex similarity estimations using the framework SANSA. The integration and documentation of SimE4KG and its scalable performance provide a reasonable basis for follow-up ML or data-analytic pipelines in SANSA. The pipelines can be used for implementing recommendation systems, entity matching, or clustering. In the multi-dimensional evaluation, it becomes clear that the various feature extractors of DistSim, DistRDF2ML, and SimE4KG differ in complexity, both in the set of possible features and their runtime. In the present use cases, the newly developed feature extractor compromises the fast but non-differentiated collection of features and the various configurable but computationally intensive SPARQL-based feature extractors. The design of the similarity estimator allows far more influence by the pipeline's user. Also, SimE4KG can cover many more possible multi-modal knowledge graph data types. There are fast probabilistic methods as well as more accurate non-probabilistic algorithms. Also, the improved annotation of the similarity estimations allows an increased explainability of the created results.

VII. Conclusion & Future Work

This work presents the SimE4KG framework integrated into the semantic analytics stack SANSA. SimE4KG is an explainable, scalable, distributed multi-modal in-memory semantic similarity estimation approach for RDF KGs. The similarity assessment is performed by a pipeline of generic, easily configurable software modules. All of them are aligned with the common usage of ML pipeline transformers, offering standard usability. In addition, the entire similarity estimation is optimized to fit the multi-modal nature, and the results are presented in an explainable, reproducible, and reusable native RDF format. The extensive evaluation of framework performance shows the significant scaling of distributed computing. The entire framework is open-source and available through the corresponding GitHub repository, while the documentation in the form of a tutorial like ReadMe, scala-docs, hands-on example classes, and Databricks notebooks is presented. Due to the generic nature and the integration into the SANSA stack, SimE4KG can be extended, and the individual software modules can be used outside the initially intended usage.

A. Future Work

The approach presented here can be further explored in several possible directions. Further similarity measurements can be added to address use-case-specific requirements better or handle other possible multi-modal literals of a KG, such as images or audio data. Embeddings or visual bag-of-words transformers can be possible options to achieve this for images. Moreover, with the number of open-source datasets available, many exciting hands-on pipeline creations are possible in domains like recommendation systems for streaming content.

Acknowledgement

This work was partly supported by the EU Horizon 2020 project PLATOON (Grant agreement ID: 872592). We would also like to thank the SANSA development team for their helpful support.

References

[1] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann, "Dbpedia - a crystallization point for the web of data," *Journal of web semantics*, vol. 7, no. 3, pp. 154–165, 2009.

[2] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: A collaboratively created graph database for structuring human knowledge," in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, 2008, pp. 1247–1250.

[3] F. M. Suchanek, "Yago: A core of semantic knowledge," in *Proceedings of the 16th international conference on World Wide Web*, 2007, pp. 697–706.

[4] E. Miller, "An introduction to the resource description framework," *Bulletin of the American Society for Information Science and Technology*, vol. 25, no. 1, pp. 15–19, 1998.

[5] C. Draschner, "Sime4kg github release," https://github.com/SANSA-Stack/SANSA-Stack/releases/tag/v0.8.2.3_SimE4KG.

[6] SANSA-Dev-Team, "Sansa github repository," https://github.com/SANSA-Stack.

[7] B. McBride, "Jena: A semantic web toolkit," *IEEE Internet computing*, vol. 6, no. 6, pp. 55–59, 2002.

[8] M. Zaharia, R. S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. J. Franklin *et al.*, "Apache spark: A unified engine for big data processing," *Communications of the ACM*, vol. 59, no. 11, pp. 56–65, 2016.

[9] X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. Tsai, M. Amde, S. Owen *et al.*, "Mllib: Machine learning in apache spark," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1235–1241, 2016.

[10] T. Berners-Lee and J. Hendler, "Publishing on the semantic web," *Nature*, vol. 410, no. 6832, pp. 1023–1024, 2001.

[11] J. Lehmann, G. Sejdiu, L. Bühmann, P. Westphal, C. Stadler, I. Ermilov, S. Bin, N. Chakraborty, M. Saleem, A.-C. N. Ngomo *et al.*, "Distributed semantic analytics using the sansa stack," in *International Semantic Web Conference*. Springer, 2017, pp. 147–155.

[12] M. Odersky, L. Spoon, and B. Venners, *Programming in Scala*. Artima Inc, 2008.

[13] D. Vrandečić and M. Krötzsch, "Wikidata: a free collaborative knowledgebase," *Communications of the ACM*, vol. 57, no. 10, pp. 78–85, 2014.

[14] O. Hassanzadeh and M. P. Consens, "Linked movie data base," in *LDOW*, 2009.

[15] L. Yu, "Linked open data," in *A Developer's Guide to the Semantic Web*. Springer, 2011, pp. 409–466.

[16] Z. Zhu, S. Xu, J. Tang, and M. Qu, "Graphvite: A high-performance cpu-gpu hybrid system for node embedding," in *The World Wide Web Conference*, 2019, pp. 2494–2504.

[17] A. Lerer, L. Wu, J. Shen, T. Lacroix, L. Wehrstedt, A. Bose, and A. Peysakhovich, "Pytorch-biggraph: A large scale graph embedding system," *Proceedings of Machine Learning and Systems*, vol. 1, pp. 120–131, 2019.

[18] D. Zheng, X. Song, C. Ma, Z. Tan, Z. Ye, J. Dong, H. Xiong, Z. Zhang, and G. Karypis, "Dgl-ke: Training knowledge graph embeddings at scale," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020, pp. 739–748.

[19] P. Ristoski and H. Paulheim, "Rdf2vec: Rdf graph embeddings for data mining," in *International Semantic Web Conference*. Springer, 2016, pp. 498–514.

[20] C. F. Draschner, C. Stadler, F. Bakhshandegan Moghaddam, J. Lehmann, and H. Jabeen, "Distrdf2ml - scalable distributed in-memory machine learning pipelines for rdf knowledge graphs," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, ser. CIKM '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 4465–4474.

[21] M. Palmonari and P. Minervini, "Knowledge graph embeddings and explainable ai," *Knowledge Graphs for Explainable Artificial Intelligence: Foundations, Applications and Challenges, IOS Press,, Amsterdam*, pp. 49–72, 2020.

[22] F. Bakhshandegan Moghaddam, C. Draschner, J. Lehmann, and H. Jabeen, "Literal2feature: An automatic scalable rdf graph feature extractor," in *Further with Knowledge Graphs*. IOS Press, 2021, pp. 74–88.

[23] N. Frosst and G. Hinton, "Distilling a neural network into a soft decision tree," *arXiv preprint arXiv:1711.09784*, 2018.

[24] I. Tiddi and S. Schlobach, "Knowledge graphs as tools for explainable machine learning: A survey," *Artificial Intelligence*, vol. 302, p. 103627, 2022.

[25] D. Esteves, D. Moussallem, C. B. Neto, T. Soru, R. Usbeck, M. Ackermann, and J. Lehmann, "Mex vocabulary: A lightweight interchange format for machine learning experiments," in *Proceedings of the 11th International Conference on Semantic Systems*, 2015, pp. 169–176.

[26] G. C. Publio, D. Esteves, A. Ławrynowicz, P. Panov, L. Soldatova, T. Soru, J. Vanschoren, and H. Zafar, "Ml schema: Exposing the semantics of machine learning with schemas and ontologies," in *ICML 2018 Workshop on Reproducibility in Machine Learning*, 2018.

[27] F. Lecue, "On the role of knowledge graphs in explainable ai," *Semantic Web*, vol. 11, no. 1, pp. 41–51, 2020.

[28] P. Jaccard, "Distribution de la flore alpine dans le bassin des dranses et dans quelques régions voisines," *Bull Soc Vaudoise Sci Nat*, vol. 37, pp. 241–272, 1901.

[29] A. Tversky, "Features of similarity." *Psychological review*, vol. 84, no. 4, p. 327, 1977.

[30] D. Sánchez, M. Batet, D. Isern, and A. Valls, "Ontology-based semantic similarity: A new feature-based approach," *Expert systems with applications*, vol. 39, no. 9, pp. 7718–7728, 2012.

[31] R. Rada, H. Mili, E. Bicknell, and M. Blettner, "Development and application of a metric on semantic nets," *IEEE transactions on systems, man, and cybernetics*, vol. 19, no. 1, pp. 17–30, 1989.

[32] R. Richardson, A. Smeaton, and J. Murphy, "Using wordnet as a knowledge base for measuring semantic similarity between words," 1994.

[33] Z. Wu and M. Palmer, "Verb semantics and lexical selection," *arXiv preprint cmp-lg/9406033*, 1994.

[34] P. Resnik, "Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language," *Journal of artificial intelligence research*, vol. 11, pp. 95–130, 1999.

[35] D. Lin, "Principle-based parsing without overgeneration," in *31st annual meeting of the association for computational linguistics*, 1993, pp. 112–120.

[36] S. Harispe, D. Sánchez, S. Ranwez, S. Janaqi, and J. Montmain, "A framework for unifying ontology-based semantic similarity measures: A study in the biomedical domain," *Journal of biomedical informatics*, vol. 48, pp. 38–53, 2014.

[37] C. F. Draschner, J. Lehmann, and H. Jabeen, "Distsim-scalable distributed in-memory semantic similarity estimation for rdf knowledge graphs," in *2021 IEEE 15th International Conference on Semantic Computing (ICSC)*. IEEE, 2021, pp. 333–336.

[38] G. Zhu and C. A. Iglesias, "Sematch: Semantic similarity framework for knowledge graphs," *Knowledge-Based Systems*, vol. 130, pp. 30–32, 2017.

[39] J. Wang, H. T. Shen, J. Song, and J. Ji, "Hashing for similarity search: A survey," *arXiv preprint arXiv:1408.2927*, 2014.

[40] R. B. Zadeh and A. Goel, "Dimension independent similarity computation." *Journal of Machine Learning Research*, vol. 14, no. 6, 2013.

[41] A. Gakhov, *Probabilistic Data Structures and Algorithms for Big Data Applications*. BoD–Books on Demand, 2019.

[42] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[43] C. Stadler, G. Sejdiu, D. Graux, and J. Lehmann, "Sparklify: A scalable software component for efficient evaluation of sparql queries over distributed rdf datasets," in *International Semantic Web Conference*. Springer, 2019, pp. 293–308.