DistSim - Scalable Distributed in-Memory Semantic Similarity Estimation for RDF Knowledge Graphs

Carsten Felix Draschner
carsten.draschner@uni-bonn.de
University of Bonn
Bonn, Germany

Jens Lehmann
jens.lehmannn@cs.uni-bonn.de
University of Bonn
Bonn, Germany

Hajira Jabeen
hajira.jabeen@uni-koeln.de
University of Cologne
Cologne, Germany

***Abstract*— In this paper, we present DistSim, a Scalable Distributed in-Memory Semantic Similarity Estimation framework for Knowledge Graphs. DistSim provides a multitude of state-of-the-art similarity estimators. We have developed the Similarity Estimation Pipeline by combining generic software modules. For large scale RDF data, DistSim proposes MinHash with locality sensitivity hashing to achieve better scalability over all-pair similarity estimations. The modules of DistSim can be set up using a multitude of (hyper)-parameters allowing to adjust the trade-off between information taken into account, and processing time. Furthermore, the output of the Similarity Estimation Pipeline is native RDF. DistSim is integrated into the SANSA stack, documented in scala-docs, and covered by unit tests. Additionally, the variables and provided methods follow the Apache Spark MLlib name-space conventions. The performance of DistSim was tested over a distributed cluster, for the dimensions of data set size and processing power versus processing time, which shows the scalability of DistSim w.r.t. increasing data set sizes and processing power. DistSim is already in use for solving several RDF data analytics related use cases. Additionally, DistSim is available and integrated into the open-source GitHub project SANSA.**

I. Introduction

Various information domains are modeled using knowledge graphs. A popular standardized format to encode knowledge graphs as linked data is RDF. For optimizing available RDF data, we perform algorithms like Entity Resolution, Entity Linking, and Classification. Additionally, methods are needed to gain insights into the data and to find relevant entities for advanced analytics like Recommendation Systems, Clustering, and Anomaly Detection. The commonality in all of these methods is that they rely on computing similarities or distances between entities. RDF data can have arbitrary sizes up to billions of triples and gigabytes of volume[1]. This large scale data indicates the need for distributed computing because it is cheaper and sometimes more convenient to scale horizontally rather than vertically.Processing large scale data can lead to extensive processing times, especially when the algorithms have non-linear complexity. For some of the non-linear complexity algorithms, there are probabilistic alternatives that reduce the complexity [1]. This reduction in complexity offers a trade-off between run time and result quality. However, whether a reduction in quality is acceptable depends on the use case. Therefore, it is desired to develop algorithms by providing an appropriate trade-off between processing time and quality of results. The contributions of this work are:

- A scalable **Dist**ributed in-Memory Semantic **Sim**ilarity Estimation Framework for RDF Knowledge Graphs
- Integration of DistSim into the holistic SANSA stack over a set of generic modules
- Representation of Semantic Similarity Estimation Experiments and their results in native RDF format
- Evaluation of scalability of different distributed similarity estimation approaches

[1]https://www.w3.org/wiki/DataSetRDFDumps

II. Preliminaries

A. Resource Description Framework

RDF is the W3C standard to represent semantic linked data. RDF data is, in principle build-up by triples. Each Triple has a *Subject*, a *Predicate*, and an *Object*. The elements can be *IRI*, *Blank Node*, or *Literals*. *Subjects* can be *IRI* or *Blank Node*, *Predicates* are *IRI* and *Objects* can be *IRI* or *Literal*. *IRIs* are *"Internationalized Resource Identifier"*. *Blank nodes* are nodes in the Knowledge Graph without explicit given *IRI*. *Literals* are leaves in an RDF graph and represent explicit values like Strings, Integers or date-time information.

B. Apache Spark

Apache Spark is a framework for cluster computing, available in scala, python, and java. The building block components of Apache Spark are Spark SQL, Spark Streaming, MLlib, and GraphX. Large scale data sets can be processed in memory when a sufficient cluster hardware setup is available. Apache Spark MLlib is Spark's scalable machine learning library consisting of standard learning algorithms and utilities, including feature transformation, clustering, hashing, and classification.

C. SANSA

The open-source SANSA stack[2] uses Apache Spark and Apache Flink, which offer fault-tolerant, highly available, and scalable approaches to efficiently process massive sized data sets. SANSA provides various layers containing modules for semantic data representation, querying, inference, and analytics. The SANSA stack is available over GitHub.

III. Related Work

A. Developments of Semantic Similarity Estimations

In recent years, many different semantic similarity estimators have been developed for various use cases[3], [4]. Three types of Semantic Similarity Estimation have been proposed[5], [6]. Structure/Path-based semantic similarity estimations assign the similarity over two entities' distance in a knowledge graph. The smaller the distance is, the higher is the similarity (Shortest Path[7], Weighted Links[8], Wu and Palmer[9]). These and further approaches differ in how the knowledge graph structure path is deduced and weighted for the resulting similarity value. The second class of semantic similarity estimations is based on Information Content (Resnik et al.[10], Lin et al.[11]). These approaches assign the similarity by the highest information content of shared features. The information content is calculated based on inverse feature frequency. The rarer a feature is, the higher is its Information Content. The third group is feature-based semantic similarity measures. The feature-based methods differ in the normalization of the calculated overlap of features. The initial approach is Jaccard Index[12]. Jaccard defines the similarity as the cardinality of the intersection of features divided by the cardinality of the union of features. Based on these feature-based semantic similarity measures, probabilistic approaches like minHash[13] were developed. MinHash reduces the processing time in computing all pair similarity by representing the sparse hot encoded feature vector in a dense minHashed vector. Based on these min-hashed vectors, elements are grouped in buckets over Locality Sensitivity Hashing. This bucketing results in a much smaller search space for each entity to calculate similarity values.

B. Distributed Semantic Similarity Estimation Frameworks

In recent years frameworks were developed for semantic analytics like similarity estimation [14], [4], [3], but these developments are not optimized for large data, and scalability over distributed computing. Apache Spark is the state of the art, open Source Framework for distributed data analytics. Spark MLlib provides two similarity estimation modules (Bucketed Random Projection for Euclidean Distance and MinHash[15] for Jaccard Distance. Apache Spark does not incorporate RDF data out of the box. On the other hand, Knowledge Graph and RDF do not have native feature set vectors, such that they can be used out of the box.

IV. DistSim

A. Pipeline Architecture

The Scalable **Dist**ributed in-Memory Semantic **Sim**ilarity Estimation is implemented as a stacked pipeline aligned with the standards of Apache Spark MLlib. Figure 1 shows that the approach consists of six modules: ReadIn, Feature Extraction, Count Vectorization, Similarity Estimation, Metagraph Creation, and WriteOut. DistSim makes use of ReadIn and WriteOut software modules from the SANSA stack RDF Layer to read and write RDF data. For scalable similarity estimation, it

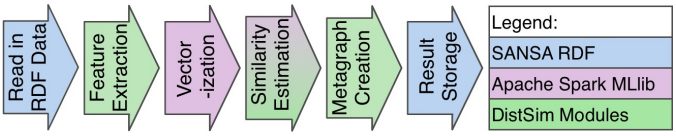

Fig. 1. DistSim Pipeline Architecture)

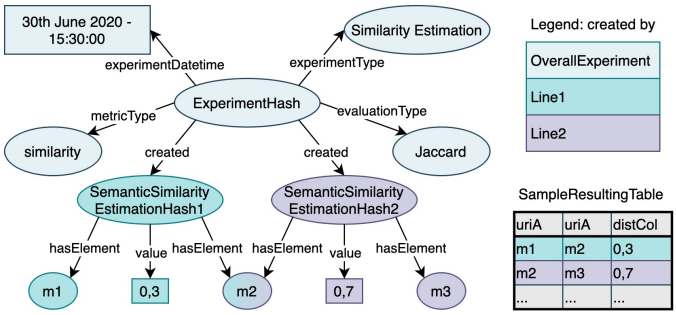

Fig. 2. Sample resulting meta graph from DistSim

uses Count Vectorizer and MinHash with Locality Sensitivity Hashing from Apache Spark MLlib. In addition, DistSim provides feature extraction, a set of similarity estimation models, and the meta graph creation as novel contributions. The pipeline for semantic similarity estimation reads-in and writes-out RDF data.

B. DistSim as Resource

DistSim is developed as open source and is fully integrated into the SANSA-Stack and documented with scala-docs. The estimators provide methods for *nearestNeighbors* which estimates for one URI the k most similar elements represented by their URI. Alternatively, *allPairSimilarity* calculates the similarity of all pairs of URIs from the two DataFrames (of length n and m). DistSim provides the output RDF data enriched with similarity annotations and meta-information (see figure 2). The annotated meta information of the similarity estimation not only makes the results reproducible, but it also allows the possibility to comprehend the conditions and parameters used for the estimation. For every novel developed module, we provide unit tests.

C. DistSim Feature Extraction

The Semantic Similarity Estimations of DistSim operate on feature sets. These feature sets are derived from the reading the RDF data set using the Feature-Extractor Module, which is implemented as a Transformer. Developers can set the Feature-Extractor methodology using modes. The mode specifies how the information stored in triples for a specific URI is transformed into the assigned feature set. In this paper, we present two out of twelve available feature extraction modes, supplied as a parameter in Feature Extractor Transformer initialization. The corresponding figures 3, 4 show on the left-hand side the sample KG. with the entities in blue, the used triple information for features in green, and the ignored information in red. On the right-hand side, the corresponding schematic feature

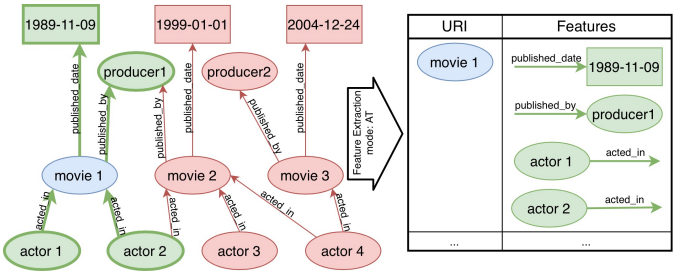

Fig. 3. Feature Extraction using predicate and node in same feature

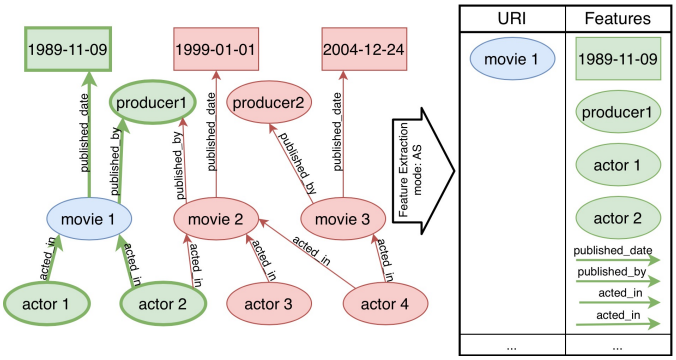

Fig. 4. Feature Extraction using predicate and node in separate feature

vector representation of the entity is shown. In the feature extraction approach shown in figure 3 the information of the property and connected node are kept together for feature generation compared to the stacking approach presented in figure 4

D. Semantic Similarity Estimation Models

DistSim provides feature set based semantic similarity estimations (see table I). The scalable alternative MinHashLSH[15], [13] can be used for the probabilistic approach in calculating Jaccard[12] similarity. This scalable but approximate method is optimal for large scale data scenarios. In addition, we can stack MinHash with different DistSim models, such that we calculate a set of first estimates in scalable processing time and call in a second step more accurate functions only on promising candidates.

TABLE I
FEATURE SET BASED SEMANTIC SIMILARITY ESTIMATION FORMULAS

Similarity Coefficient	Formula
Batet[16] Distance	$\log_2\left(1+\frac{\lvert X-Y\rvert+\lvert Y-X\rvert}{\lvert X\cap Y\rvert+\lvert X-Y\rvert+\lvert Y-X\rvert}\right)$
Braun-Blanquet[17] Similarity	$\frac{\lvert X\cap Y\rvert}{\max\{\lvert X\rvert,\lvert Y\rvert\}}$
Dice[18] Similarity	$\frac{2\lvert X\cap Y\rvert}{\lvert X\rvert+\lvert Y\rvert}$
Jaccard[12] Similarity	$\frac{\lvert X\cap Y\rvert}{\lvert X\cup Y\rvert}$
Ochiai[19] Similarity	$\frac{\lvert X\cap Y\rvert}{\sqrt{\lvert X\rvert*\lvert Y\rvert}}$
Simpson[20] Similarity	$\frac{\lvert X\cap Y\rvert}{\min\{\lvert X\rvert,\lvert Y\rvert\}}$
Tversky[21] Similarity	$\frac{\lvert X\cap Y\rvert}{\lvert X\cap Y\rvert+\alpha\lvert X-Y\rvert+\beta\lvert Y-X\rvert}$

E. DistSim (Hyper-)Parameter Setup

The modular DistSim Pipeline allows a multitude of adjustments over (hyper-)parameters that can reduce the memory usage and the processing time. The trade-off comes with a loss of information. The *CountVectorizer* transforms a set of features into a vector with a fixed length. The length is adjustable by minimal document frequency (*minDf*) and upper bound vector size (*maxVocabSize*). Small feature vectors need less memory and can be processed faster but store less information. In Semantic Similarity Estimation over *MinHashLSH*, a higher number of hash tables (*numHashTables*) reduce the false-negative rate in detecting similar elements but increase the processing time and memory usage. The *threshold* on minimal similarity, respectively, minimal distance can minimize memory usage and processing time. If this threshold is more strict, fewer pairs of similar values have to be processed over a distributed system in *allPairSimilarity*.

F. DistSim Use Cases

Scalable distributed semantic similarity estimations are needed in several Use Cases. The extended SANSA stack is in use as a generic Big Data Analytics Toolbox of the Horizon 2020 Project PLATOON. SANSA, as an underlying toolbox for semantic analytics in Opertus Mundi[2], provides with the novel developed semantic Similarity Pipeline needed software modules. The project Simple-ML[3] provides an easy to use generic stackable machine learning framework which uses SANSA and DistSim as underlying Semantic Similarity Framework for RDF data.

V. EXPERIMENT AND EVALUATION

DistSim implements well-established similarity estimation functions for RDF data. The evaluation is presented for the performance assessment of DistSim on different data sizes and varying cluster processing setups. Here, the processing time is an indicator of DistSim's distributed processing and scalability. The cluster processing power is adjusted over the spark-submit command, where the number of executor cores can be limited.

A. Data Sets

The evaluation of the scalability of DistSim is performed over multiple data sets of different sizes. The data set sizes are adjusted by creating synthetic data sets We use synthetic data sets to ensure equally distributed graph density. In real-world graphs, cutting off fractions could lead to an unnatural graph appearance. Figure 4 shows on the left hand side the principle structure of the generated data set.

B. Scalability over increasing horizontal Cluster Computation

The processing power is regulated over the number of available cores (from $2^2 = 4$ up to $2^7 = 128$). Table II shows the scalability over cluster setups. We see a clear decrease in processing time over increasing computation power.

[2]https://www.opertusmundi.eu
[3]https://simple-ml.de

TABLE II
DISTSIM PROCESSING POWER SCALABILITY

Processing Time in seconds	Number Executer Cores					
Estimation Approach	4	8	16	32	64	128
Jaccard	274	196	133	87	44	31
MinHash	33	31	22	15	10	10

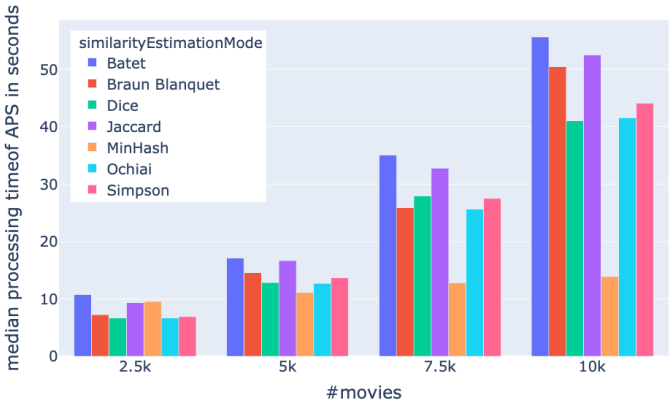

Fig. 5. Data Size Scalability of All-Pair-Similarity Estimation

C. Scalability over Data Set Size

The use of probabilistic similarity estimator MinHashLSH allows scalable processing of large scale RDF data. Figure 5 shows for the all pair similarity estimation that MinHash (orange) scales better than the other approaches. The approaches Batet, Braun Blanquet, Dice, Jaccard, Simpson, and Tversky scale similar. For Nearest Neighbor Estimation, all approaches are on a similar scalable level, including MinHash, because Nearest Neighbor is a linear operation and not quadratic in complexity like All-Pair Similarity.

VI. CONCLUSION AND FUTURE WORK

DistSim integrated into the SANSA stack provides a scalable distributed open-source framework for semantic similarity estimation on RDF Knowledge Graphs. Multiple projects are already using DistSim modules. The community is actively using the SANSA stack for scalable distributed semantic analytics on large-scale RDF data. The availability of an easy to use evaluation pipeline shows clear infer-able effects of (hyper-)parameters to the corresponding processing times. The storage in a tabular format and semantic data representation (see figure 2) allows high reproducibility and understanding of the needed pipeline setup. The results are human and machine-readable. Using DistSim and the proposed analytic pipeline modules for RDF processing, additional RDF data analytic algorithms can be easily ported to distributed processing. The evaluation shows scalability of DistSim over different data set sizes and processing power. We are currently developing more approaches for feature extraction and semantic similarity estimations to cover additional semantic information.

ACKNOWLEDGEMENT

This work was partly supported by the EU Horizon 2020 project PLATOON (Grant agreement ID: 872592).

REFERENCES

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

[2] J. Lehmann, G. Sejdiu, L. Bühmann, P. Westphal, C. Stadler, I. Ermilov, S. Bin, N. Chakraborty, M. Saleem, A. C. Ngonga Ngomo, and H. Jabeen, "Distributed semantic analytics using the SANSA stack," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10588 LNCS, no. iii, pp. 147–155, 2017.

[3] S. Harispe, S. Ranwez, S. Janaqi, and J. Montmain, "Semantic similarity from natural language and ontology analysis," *Synthesis Lectures on Human Language Technologies*, vol. 8, no. 1, pp. 1–254, 2015.

[4] G. Zhu and C. A. Iglesias, "Computing semantic similarity of concepts in knowledge graphs," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 1, pp. 72–85, 2016.

[5] 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.

[6] T. Slimani, "Description and evaluation of semantic similarity measures approaches," *arXiv preprint arXiv:1310.8059*, 2013.

[7] 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.

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

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

[10] 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.

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

[12] P. Jaccard, "Distribution de la flore alpine dans le bassin des Dranses et dans quelques régions voisines," *Bulletin de la Société Vaudoise des Sciences Naturelles*, vol. 37, no. May, pp. 241–272, 1901.

[13] A. Z. Broder, "On the resemblance and containment of documents," in *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No. 97TB100171)*. IEEE, 1997, pp. 21–29.

[14] 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.

[15] R. B. Zadeh and A. Goel, "Dimension independent similarity computation," *The Journal of Machine Learning Research*, vol. 14, no. 1, pp. 1605–1626, 2013.

[16] 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.

[17] J. Braun-Blanquet, *Pflanzensoziologie: grundzüge der vegetationskunde*. Springer-Verlag, 2013.

[18] T. A. Sorensen, "A method of establishing groups of equal amplitude in plant sociology based on similarity of species content and its application to analyses of the vegetation on danish commons," *Biol. Skar.*, vol. 5, pp. 1–34, 1948.

[19] A. Ochiai, "Zoogeographical studies on the soleoid fishes found in japan and its neighbouring regions-i," *Bull. Jpn. Soc. scient. Fish.*, vol. 22, pp. 522–525, 1957.

[20] G. G. Simpson, "Mammals and the nature of continents," *American Journal of Science*, vol. 241, no. 1, pp. 1–31, 1943.

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