Semantic Analytics in the Palm of your Browser

Carsten Felix Draschner[1], Farshad B. Moghaddam[1], Jens Lehmann[1], and Hajira Jabeen[2]

[1] Smart Data Analytics Research, University of Bonn, Germany
[2] University of Cologne, Germany

Abstract. Linked open data sources and the semantic web has become a precious data source for data analytics tasks and data integration. The growing data set sizes of RDF Knowledge Graph data need scalable processing and analytics techniques. The processing power of in-memory frameworks which can perform scalable distributed semantic analytics like SANSA, make use of Apache Spark and Apache Jena to provide start-to-end extensive scalable analytics on RDF knowledge graphs. The setup of a technical system with all dependencies and environments can be a tough challenge and might also require sufficient available processing power. To reduce the entry barriers for getting started in evaluating and testing all opportunities of the SANSA framework and even bring this technology to production only from the browser. We introduce within this paper how to get the SANSA stack running within Databricks, with no need for special Apache Spark skills or any installations. This simplified usage offers distributed large-scale processing of RDF data from mobile devices. In addition, the availability of Hands-On Sample Notebooks increases the reproducibility of complex framework evaluation experiments. This paper shows that the startup of a very complex scalable semantic data analytics stack framework does not need to be complicated.

Keywords: Semantic Analytics, Machine Learning, Semantic Similarity, Distributed Processing, Apache Spark, Resource Description Framework, RDF, SANSA, Databricks

1 Motivation

An increasing number of data sets based on the linked open data principle have appeared in recent years. These offer tremendous potential for data integration through the use of *IRIs* and *URIs*. Moreover, in the area of energy data, semantic data is emerging in many projects and is being used for various data analytics tasks. Due to the large amount of data, solutions in the area of big data processing are necessary.

These technologies, such as Apache Spark[3], enable fast memory and parallel processing of data to scale arbitrarily across parallel cores by optimizing them for distributed cluster computing. The Scalable Semantic Analytics Stack

[3] https://spark.apache.org

SANSA[1] leverages Apache Spark and Apache Jena[4] to provide an open-source framework for start-to-end data analytic pipelines for large-scale RDF Knowledge Graphs [10,11].

Distributed frameworks like the SANSA stack have a vast potential in the energy domain to process large-scale Knowledge Graphs. However, the technical requirements in the initial setup and the first experimentation for less mature Data Scientists and Machine learning engineers are challenging and overwhelming entry hurdles for the development of first experiments, proof of concepts, or Minimal Valuable Products.

To reduce this hurdle, we would like to show how users from the energy sector and all other fields which want to build large-scale RDF Knowledge graph data analytic pipelines can use SANSA with minimal technical requirements and entry hurdle.

1.1 Contributions

- Introduction of Scalable Semantic Analytics Stack in Browser through Databricks
- Sample Explanatory Notebooks for Hands-on Interaction with RDF data
- Guideline of How to Use Third-Party Apache Spark Frameworks within Platform as a Service Providers
- Showcasing Recent Modules and developments of the SANSA stack

2 Related Work

In recent years, it has been recognized that creating complex technical environments is a major challenge. Therefore, virtualization environments have been developed. On the one hand, there are virtualization environments that native allow an entire image of a complete operating system *VMbox*[5], *Parallels*[6], *VMware*[7], while on the other hand, there are containerization platforms like *Docker* [2] orchestrated by *Swarm*[8] or *Kubernetes*[9], which through their architecture creates a clean replication and scaling of complex technical dependencies. In data science, libraries like *pipenv*[10] or *poetry*[11] are popular, enabling a project level or repository level encapsulated environment.

Reproducing and illustrating machine learning pipelines is increasingly enabled by formats such as markdown documents or, more preferably, notebooks. Both have the advantage of representing a variety of content in the same document. There can be text and graphical sections as in classical literature, in

[4] https://jena.apache.org
[5] https://www.virtualbox.org
[6] https://www.parallels.com/
[7] https://www.vmware.com/
[8] https://docs.docker.com/engine/swarm/
[9] https://kubernetes.io
[10] https://github.com/pypa/pipenv
[11] https://python-poetry.org

addition to code sections. In the same notebook, the results of the code cells can be mapped to show, for example, the mapping of a generated data frame, a figure, or an arbitrary plot. The popular examples of notebooks are *Jupyter Notebook*[3], *Jupiter Lab*[12] and Zeppelin *Notebooks*[4].

The idea of collaborative editing of resources within the browser were first introduced by office solutions from *Google Docs*[13], *Next-cloud Office*[14], *Overleaf*[15] and many more. This opportunity to share and edit at the same time in parallel documents with a group of users was adapted by notebook based programming platforms like *Google-Colab*[16] and *Databricks*[17].The processing power is provided by the platform providers. The focus of *Google Colab* is the default python data science stack, while *Databricks* focuses on *Spark* processing. These platforms offer a free plan with limited processing power and functionality, sufficient for most first hands-on notebooks to demonstrate functionalities and usage of libraries and frameworks.

3 SANSA through Databricks

A complex and heterogeneous framework like *SANSA*[18] requires several technical prerequisites to run initial experiments. On the one hand, the computation is done in memory and it is crucial to have enough memory to manage the data, On the other hand, the computation is done on the CPU side. *Apache Spark* is designed for multi-core and cluster computation. In order to use the framework, Apache Spark must be available in the required version (in our case (3.x) and also *Scala* in version 2.12. Setting up this hardware and software can be eased by using *Databricks* since even in the community edition (free plan), a two-core system with 15GB of memory is already available. Furthermore, there are predefined images like the combination of different Apache Spark versions and Scala versions. The following sections will guide through the setup, and explain working with SANSA on RDF data.

3.1 Get Access to Platform

Databricks[5] is one of several Platform as a Service (PaaS) providers. Several alternatives do not offer the simplicity of setting up an Apache Spark instance for use on notebooks and making it accessible through notebooks in a user-friendly way. For registering Data Bricks in the Free Plan, the Community Edition is suitable. More information can be found on the Databricks FAQ[19].

[12] https://jupyterlab.readthedocs.io
[13] https://www.google.com/docs/about/
[14] https://nextcloud.com/onlyoffice/
[15] https://www.overleaf.com/
[16] https://colab.research.google.com
[17] https://databricks.com
[18] https://github.com/SANSA-Stack/SANSA-Stack
[19] https://databricks.com/product/faq

3.2 Upload needed Data

Once logged in into the platform, an opportunity is given to *import libraries*. The SANSA stack needs to be uploaded as a *jar*. The *jar* can be fetched from the most recent Release in the SANSA stack GitHub page[20]. The name will be given automatically according to the filename of the jar. Due to jars size, the upload process will take a few minutes. After the upload is done, the process can be confirmed with "*Create*" button. Next, we need to make our desired data available. In the next step we add the Knowledge Graph data to our Databricks file system. We will introduce the usage of Sansa stack based on the *Linked Movie Database* dataset[6] which is a LOD RDF dataset containing 40 thousand movies like their title, runtime, list of actors, genres and publish date. This dataset represents a multimodal Knowledge Graph for several example pipelines. The import can be started from main pages *Import and Explore data*. In the overlay menu, one could drag and drop the file. Other files can be found on web pages like https://lod-cloud.net or https://www.w3.org/wiki/DataSetRDFDumps. Once the data is uploaded, the menu shows the path where it got stored. An example might be: *"FileStore/tables/linkedmdb-18-05-2009-dump.nt"*.

3.3 Setup Cluster

One must set up a cluster in the platform used for executing the notebooks. First, create the cluster as a new cluster and give it a unique name like *SANSA-tryout-cluster*. Next, select the fitting image named Spark Runtime Version to the pair *Scala 2.12* and *Scala 3.x*. Then specify the spark config by pasting the following three key-value pairs shown in figure 1 and figure 2. They correspond to the default Databricks and SANSA Spark setup. This Cluster configuration has to be confirmed with "*Create Cluster*". Confirmation over "*create cluster*" opens up the overview of the respective created cluster. In the Libraries tab, it is needed to "*install new*" the previously uploaded *SANSA jar*. The uploaded jar is within the user's workspace. This process has to be confirmed with "*install*". After some seconds the SANSA library will change status from *installing* to *installed*.

3.4 Setup Notebook

Now, we have to open up or create a desired notebook. Either one can start with a blank notebook, but it is easier to use the provided sample notebooks[21]. These sample notebooks can be imported by using the import option from users' workspace. In the pop-up window, one can import the notebook over the notebook URL. The import will directly add the notebook to the workspace and open it up.

[20] https://github.com/SANSA-Stack/SANSA-Stack/releases
[21] https://github.com/SANSA-Stack/SANSA-Databricks

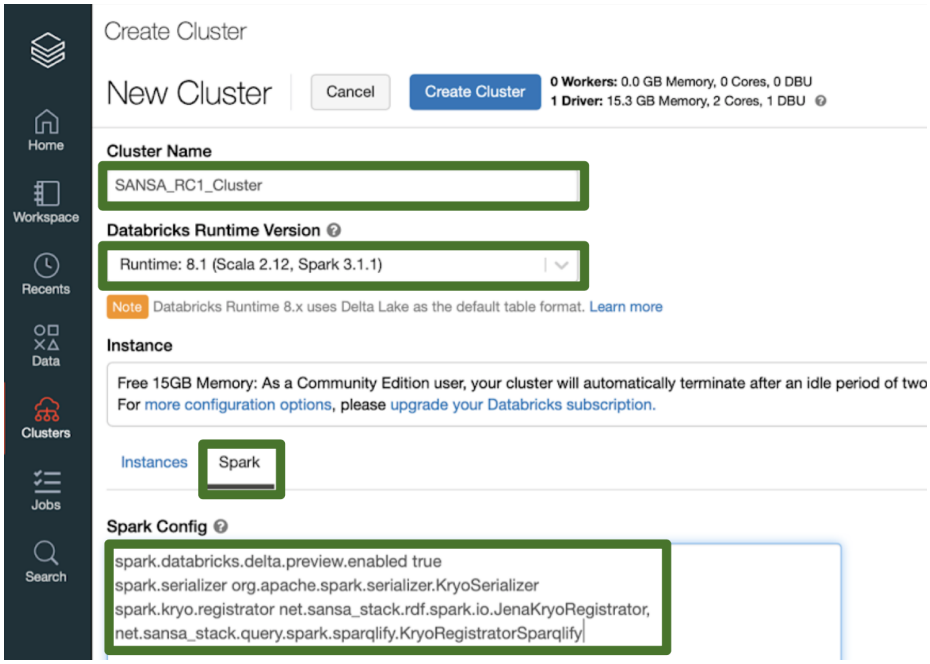

Fig. 1. Configuration of Cluster

```
1 spark.databricks.delta.preview.enabled true
2 spark.serializer org.apache.spark.serializer.KryoSerializer
3 spark.kryo.registrator net.sansa_stack.rdf.spark.io.
      JenaKryoRegistrator, net.sansa_stack.query.spark.
      sparqlify.KryoRegistratorSparqlify
```

Fig. 2. Spark Configuration modules

3.5 Execution of Sample Notebooks

The notebook needs to get assigned a cluster. The cluster should be present as previously configured (see figure 5) and contains the SANSA framework as a library. After selecting the cluster, it gets attached and will be ready after some seconds. This enables the execution of notebook cells with SANSA module functionalities.

3.6 Usage of SANSA Sample Notebooks

The provided notebooks show on the one hand how to read in RDF Knowledge graphs, how to query data over SPARQL, and execute elements from the ML layer. One can find many more generic modules for designing the desired start-to-end Apache Spark/SANSA pipeline for RDF Knowledge Graph Analytics and

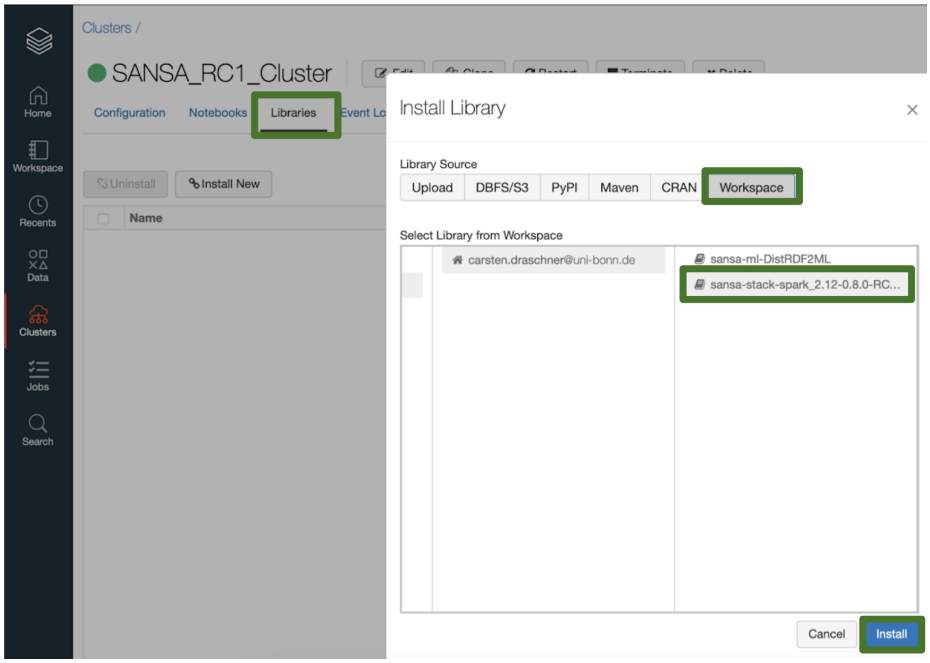

Fig. 3. Installation of SANSA library

processing from the SANSA documentation[10,**?**]. One of the recent examples is the DistSim[7] approach, which calculates Similarity Scores for RDF entities, which can then be used for various follow-up approaches like Clustering, Entity Linking, Classification, or Recommendation Systems [8]. A complete tutorial including links to sample notebooks can be found in an uploaded presentation and within the corresponding GitHub repository[9].

4 Conclusion and Future Work

This paper demonstrates that a complex and holistic framework for scalable semantic analytics can be made easily accessible, by showcasing sample notebooks hosted and running within the service as a platform provider, Databricks. This guideline offers the opportunity to take the first steps when exploring and porting their semantic data analytical pipeline ideas. On the one hand, the need to have a hardware setup with appropriate computational power and ram is not needed for the first step because notebooks are running on Databricks infrastructure. On the other hand, the installation and handling of installing the appropriate Scala and Spark Versions is automatically provided. All of the provided code within the sample notebooks can also run and scale among Distributed Spark Clusters. Within multiple collaborations we identified the need of high level RDF data analytic APIs. The partners can solve their Use cases of large scale Knowledge

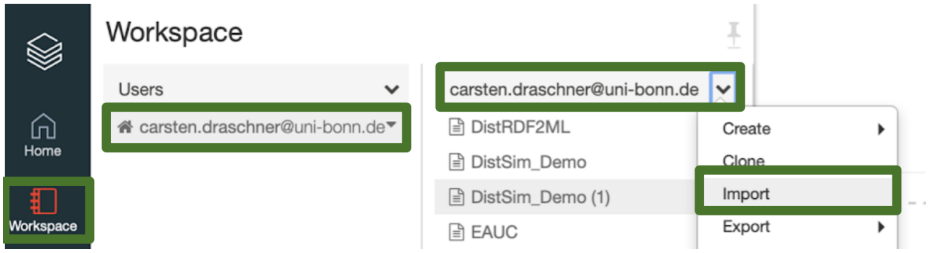

Fig. 4. Import Notebook

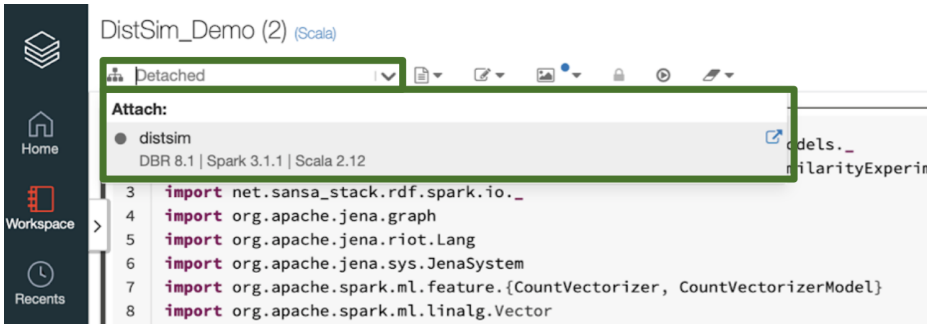

Fig. 5. Attach Cluster

Graph analytics through the generic modules of DistRDF2ML. The opportunity to postpone technical requirements set up post the first exploratory work can increase the tryout rate of complex frameworks.

Acknowledgement

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

References

1. 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.
2. C. Boettiger, "An introduction to docker for reproducible research," *ACM SIGOPS Operating Systems Review*, vol. 49, no. 1, pp. 71–79, 2015.
3. T. Kluyver, B. Ragan-Kelley, F. Pérez, B. E. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. B. Hamrick, J. Grout, S. Corlay *et al.*, *Jupyter Notebooks-a publishing*

format for reproducible computational workflows. Conference: 20th International Conference on Electronic Publishing, 2016, vol. 2016.
4. I. Ermilov, J. Lehmann, G. Sejdiu, L. Bühmann, P. Westphal, C. Stadler, S. Bin, N. Chakraborty, H. Petzka, M. Saleem *et al.*, "The tale of sansa spark." in *International Semantic Web Conference (Posters, Demos & Industry Tracks)*, 2017.
5. Databricks-Inc., "Databricks platform," https://databricks.com/product/data-lakehouse, 2021.
6. O. Hassanzadeh and M. P. Consens, "Linked movie data base." in *LDOW*, 2009.
7. 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.
8. SANSA-Team, "Sansa-stack - distsim github release and documentation," https://github.com/SANSA-Stack/SANSA-Stack/releases/tag/v0.7.1_DistSim_paper, 2020.
9. T. SANSA, "Semantic analytics in the palm of your browser slides," https://github.com/SANSA-Stack/SANSA-Databricks.
10. C. F. Draschner, C. Stadler, F. B. Moghaddam, J. Lehmann, and H. Jabeen, "DistRDF2ML-Scalable distributed in-memory machine learning pipelines for rdf knowledge graphs" in *2021 ACM International Conference on Information and Knowledge Management (CIKM)*. ACM, 2021.
11. F. B. Moghaddam, C. F. Draschner, J. Lehmann, and H. Jabeen, "Literal2Feature: an automatic scalable rdf graph feature extractor" in *Proceedings of the 17th International Conference on Semantic Systems, SEMANTICS 2021, Amsterdam, The Netherlands, September 6-9, 2021*. SEMANTICS, 2021.