Teaching Semantic Web Technologies through Interactive Jupyter Notebooks

✉ Lars Pieschel[1], ✉ Sascha Welten[1], ✉ Lars Gleim[1], and Stefan Decker[1,2]

[1] Databases and Information Systems, RWTH Aachen University, Germany
[2] Fraunhofer FIT, Sankt Augustin, Germany

Abstract. Getting new generations of developers excited about the benefits of the Semantic Web and familiar with the underlying technologies is one of the biggest challenges for furthering its adoption. Inspired by the success of structured digital learning materials in other domains of computer science, we present *SemWebNotebooks*, a portfolio of interactive Jupyter Notebook tutorials for teaching Semantic Web technologies interactively. By introducing the *Jupyter-RDFify* plugin, we seamlessly integrate support for RDF and related standards into the Jupyter ecosystem. Motivated by the overwhelmingly positive feedback provided by the students using these materials at RWTH Aachen University and correspondingly high technology acceptance, we release both *Jupyter-RDFify* and *SemWebNotebooks* to the community as open source for open reuse and further collaborative improvement.

1 Introduction

In order to drive the adoption of Semantic Web technologies, it is essential to educate new generations of developers with the underlying standards, tools, and methodologies as efficiently as possible. While a large variety of open resources such as W3C's technology primers and massive open online courses (MOOCs) on the subject are already available today, in our own experience with teaching Semantic Web Technologies to Master-level computer science students, learners frequently struggle to apply these tools in practice. It is our understanding that part of the underlying problem is the lack of coherently structured interactive learning materials for a relevant intersection of Semantic Web technologies.

In order to support the learning experience of our students, we developed *SemWebNotebooks*, a portfolio of interactive Jupyter Notebook [3] tutorials covering RDF basics such as CURIEs, Blank Nodes, Literals and the serialization formats Turtle and JSON-LD, the query language SPARQL, the shape constraint language ShEx, as well as data modeling using OWL 2. Using Jupyter's modular architecture, we provide convenient features such as syntax checking and RDF graph plotting directly from within the Jupyter Notebook cells through *Jupyter-RDFify*, a custom-developed plugin providing a seamless learning environment without any boilerplate code that could divert the attention of the students. Based on the NBGrader framework [1], we enable automatic feedback and grading for students, which reduces the teacher's workload to a necessary minimum and increases the time for direct student mentoring.

The remainder of this paper is structured as follows: Section 2 summarizes relevant related work in both the Semantic Web and in other domains employing Jupyter Notebooks to support eLearning. Section 3 details the features provided by *Jupyter-RDFify*, while Section 4 summarizes its usage workflow. Section 5 presents a quantitative evaluation of our contribution conducted in the context of our Semantic Web lecture at RWTH Aachen University before we conclude with a short discussion of impact in Section 6.

Background. The constant rising trend of Semantic Web technologies in different domains has emerged the need for suitable courses and tools to teach the theory and practice of Semantic Web. Semantic Web courses such as the MOOC ”Knowledge Engineering with Semantic Web Technologies” offered by OpenHPI[3] deliver the theoretical knowledge needed to work with technologies such as RDF, Turtle, and SPARQL. As for the practical knowledge, these courses offer quizzes and exemplary SPARQL endpoints or encourage students to configure their own triple store. Especially, self-hosted triple stores (e.g. Apache Jena Fuseki[4]) allow for a maximum level of customization and adjustments according to the course contents. However, the effectiveness of these teaching strategies suffers from the lack of sufficient interactive feedback or initial barriers induced by the configuration of these tools.

Jupyter Notebooks[5] have been proposed to lower these barriers by providing a single exercise-sheet-style interface, which wraps a customizable and shippable backend. While read-only text fields can provide task descriptions or further explanations, the executable code cells enable an interactive possibility to enter and validate code according to the task description. The methodology of using Jupyter notebooks as an educational resource[6] has been employed numerous times, especially in the domain of data science [6] but also in the Semantic Web domain [2] as a solution for interactive SPARQL queries. Another advantage of using Jupyter notebooks as explained above is, that one can use frameworks like NBGrader to automatically grade notebooks and generate feedback for them [1].

Nevertheless, these Notebooks conventionally rely on convoluted boilerplate code (e.g. Python) and additional packages in order to process, e.g., the entered RDF or SPARQL statements. The arising consequence is that the students do not get familiar with the actual syntax yielding a distraction of the actual task or the implementation of additional features like keyword highlighting is complicated. In our work, we circumvent the incorporation of additional wrapper code and present a plugin for the processing of common Semantic Web languages to enable the advantages of Jupyter Notebooks for the Semantic Web community. The features of this plugin are part of the upcoming sections.

2 Jupyter-RDFify and SemWebNotebooks

Jupyter-RDFify (JRDF)[7] enables parsing, validating, visualizing, and querying RDF graphs directly within the Jupyter Notebook ecosystem. To ensure compatibility with the default IPython kernel backend of Jupyter and enable the seamless handling of the different Semantic Web technologies, we make use of Jupyter's modular plugin and its so-called magics system. Magics are special directives, which tell the IPython kernel that the following line or cell does not contain python code and should thus be treated differently. Jupyter-RDFify[7] implements support for Semantic Web technologies through the IPython extension mechanism. Once loaded, the RDF magic (`%%rdf` or `%rdf`) may be used to control the extension through the IPython kernel. Our extension itself is built modular as well, providing submodules for Turtle, JSON-LD, SPARQL, ShEx, etc. by interpreting the magic like a command-line interface. The submodules use the feature-rich RDFLib[7]

[3]https://open.hpi.de/courses/semanticweb2015

[4]https://jena.apache.org/documentation/fuseki2

[5]Project Jupyter: https://jupyter.org

[6]For example: https://github.com/BigDataAnalyticsGroup/bigdataengineering

[7]Code, Documentation, and Evaluation: https://github.com/SemWebNotebooks/Notebooks

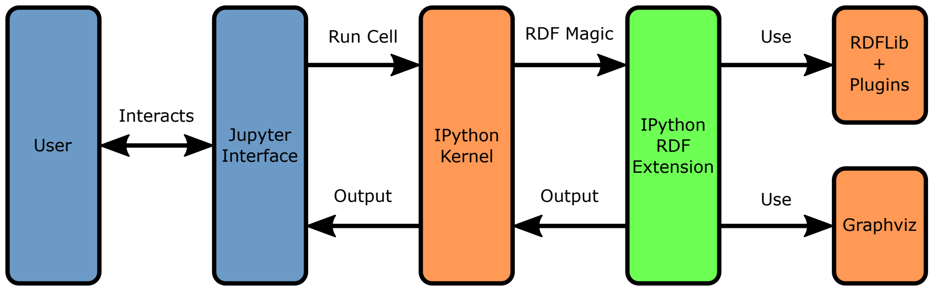


Fig. 1. Interaction between SemWebNotebooks components: Blue components are client-sided, orange and green components are server-sided. The contributed RDF extension is marked in green.

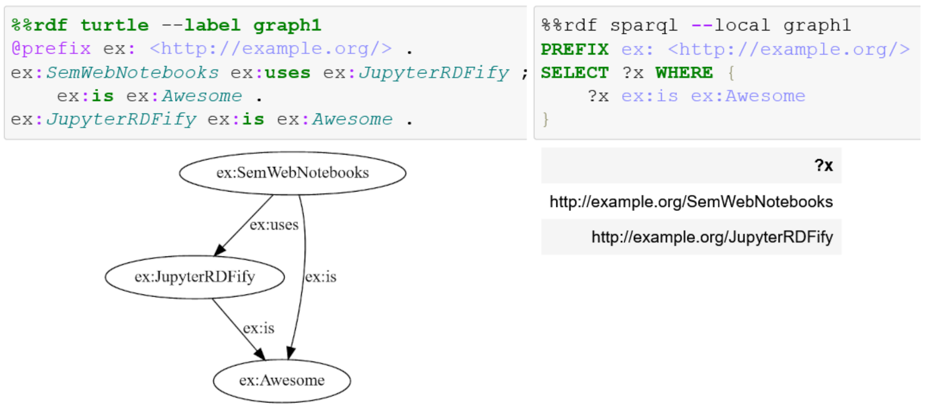
Python package in combination with plugins and extensions like RDFLib-jsonld[7] or OWL-RL[7] to handle the input. To visualize graphs, Graphviz[7] together with its Python interface is used. The output is then sent back and displayed to the user through the IPython kernel. Figure 1 visualizes the interaction between these components and the Jupyter framework.

To simplify working with multiple RDF graphs, a graph can be associated with a label upon loading it. The graphs can then be referenced by their aforementioned label at a later stage to query or verify them. Figure 2 depicts a concrete example, in which the parsed graph is labeled ”graph1” and then queried using SPARQL. The supported magics and their usage are documented in the project repository[7].

Working with Jupyter-RDFify and SemWebNotebooks. A sample exercise notebook might be structured as follows: Instructors may provide an RDF graph in Turtle notation. Due to the IPython extension, the turtle serialization can be parsed and immediately visualized in the Notebook. Students are then asked to create queries, which are executed against the earlier defined and visualized graph, in succeeding cells. Instead of a pre-defined graph from the instructors, an experimental and interactive *playground*-Notebook might also be a promising method to motivate students to create and query their own graphs without much effort.

In our Semantic Web lecture at RWTH Aachen University, we used the above-mentioned methods in conjunction with the NBGrader framework [1] to create a series of automatically graded exercises, which we release as *SemWebNotebooks*. These SemWebNotebooks combine tutorials in the form of markup paragraphs together with hands-on exercises using our extension. A detailed overview of the covered topics can be found in the project repository[7]. The learners get immediate feedback either through the output graphs or, if their solution contains syntactic errors, through the parser errors which IPython passes back to the user. SemWebNotebooks additionally make use of NBGrader to run unit-tests and thus automatically grade the exercises and simultaneously generate feedback.

Jupyter-RDFify is available as open source software, can be installed through the python package manager (e.g. Pip), and loaded into any Jupyter Python Notebook on-demand using the predefined `%load_ext` magic of the IPython kernel.



```
%%rdf turtle --label graph1
@prefix ex: <http://example.org/> .
ex:SemWebNotebooks ex:uses ex:JupyterRDFify ;
    ex:is ex:Awesome .
ex:JupyterRDFify ex:is ex:Awesome .
```

```
%%rdf sparql --local graph1
PREFIX ex: <http://example.org/>
SELECT ?x WHERE {
    ?x ex:is ex:Awesome
}
```

```
                                  ?x
http://example.org/SemWebNotebooks
     http://example.org/JupyterRDFify
```

Fig. 2. Using Jupyter-RDFify: The left cell parses and visualizes an RDF graph given in Turtle format, the right cell queries the graph parsed from the left cell and outputs the variable bindings.

3 Evaluation

Before the evaluation, we have applied SemWebNotebooks as mandatory assignments for each participant of our Master-level Semantic Web lecture at RWTH Aachen University. Overall, we have released six exercises[7], which represent the foundation of our evaluation. We have conducted the evaluation of our work at the end of the semester. The evaluation was realized by an online survey distributed to each student and was structured as follows. The first twelve questions are aligned to the Technology Acceptance Model (TAM) questionnaire [4,5]. Firstly, these questions focus on the perceived usefulness and secondly, on the perceived ease-of-use to address the two main factors, which fuel the acceptance of our work [5]. In the second part of the survey, we asked six more specific questions about the auto-grading and the generated feedback. As the last item, we have provided a free text area such that each participant was able to add individual and more advanced feedback. The questions were answered using a Likert scale. At the end of the survey, we received 38 responses out of 104 students who submitted at least one exercise. The complete evaluation results and further dissection can be found in the project repository[7]. The quantitative evaluation and the qualitative discussion of the results are presented in the following based on three key questions of our survey. We further summarize qualitative textual feedback given by our students. Here, we concentrate on three questions: *I would find SemWebNotebooks in the SemWeb course useful.* **(A)**, *I would like to use Jupyter Notebooks in other lectures as well.* **(B)**, and *The automatically generated feedback was fair.* **(C)**. Both questions **(A)** and **(B)** show remarkable results with an average score of $4.95(\sigma=0.23)$ and $4.87(\sigma=0.34)$. Question **(C)** however, resulted in the overall lowest average score with only $3.56(\sigma=1.20)$. The high results of questions **(A)** and **(B)** show not only that most students found the Jupyter Notebooks useful in the SemWeb course but that they would also like to work with similar notebooks in other lectures as well.

The obtained quantitative feedback is also reflected in the participant's textual qualitative feedback. These results show that the SemWebNotebooks are very valuable for learners. The students especially benefit from the interactive design of each Notebook. Further, the immediate syntax check supports the students to produce proper and compilable code in a practical environment. Despite the mainly positive feedback, the survey has revealed some weak points of our automatic grading system and the corresponding provided automatically generated feedback. Although it was stated that the automatic grading was provided rapidly, the grading was perceived as unfair because small errors lead to bigger point losses (e.g. case sensitivity, changed prefixes) than through manual corrections. This perception is particularly presented by the low score of question **(C)**.We have tried to compensate and mitigate these very strict assessments by manual re-corrections for borderline cases. However, we strive to make our tool more flexible and robust against minor syntax and discrepancies as future work.

4 Conclusion

In this work, we have presented Jupyter-RDFify, which is an IPython kernel extension for Jupyter Notebooks. Based on the well-established *magics* functionality of the IPython kernel, our plugin poses a solution to process Semantic Web-related languages in Jupyter Notebooks. We further extend this plugin with additional packages such as *Graphviz* for visualization purposes or *RDFLib* for querying the input data. These submodules can be implemented according to the creator's needs and propose a flexible method to customize Jupyter Notebooks. We have published our plugin as open source with support for Turtle, JSON-LD, SPARQL, and ShEx. To evaluate our work, we have applied our SemWeb-Notebooks in combination with NBGrader to our Semantic Web course to facilitate the interaction with Semantic Web technologies for student learners and also the assessment of student's assignments. Our evaluation has shown that our SemWebNotebooks increased the learning outcome and obtained strong support from the participants. Therefore, our work poses a promising alternative to improve the teaching of Semantic Web technologies. In future work, we will improve auto-grading and implement support for SHACL tasks.

References

1. Blank, D.S., Bourgin, D., Brown, A., Bussonnier, M., Frederic, J., Granger, B., Griffiths, T.L., Hamrick, J., Kelley, K., Pacer, M., et al.: nbgrader: A tool for creating and grading assignments in the Jupyter Notebook. The Journal of Open Source Education **2**(11) (2019)
2. Gray, A.J.: Using a Jupyter Notebook to Perform a Reproducible Scientific Analysis Over Semantic Web Sources. In: SemSci@ ISWC. pp. 12–24 (2018)
3. Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S., Willing, C., development team, J.: Jupyter Notebooks – a publishing format for reproducible computational workflows. In: Loizides, F., Schmidt, B. (eds.) Positioning and Power in Academic Publishing: Players, Agents and Agendas. pp. 87–90. IOS Press (2016)
4. Lewis, J.R.: Comparison of Four TAM Item Formats: Effect of Response Option Labels and Order. Journal of Usability Studies **14**(4) (2019)
5. Park, S.Y.: An analysis of the technology acceptance model in understanding university students' behavioral intention to use e-learning. Journal of Educational Technology & Society **12**(3), 150–162 (2009)
6. Toomey, D.: Jupyter for Data Science. Packt Publishing (2017)