# RDF Doctor: Holistic RDF Syntax Validation and Error Correction

Ahmad Hemid

Matriculation number: 2915574

July 27, 2018

Master Thesis

**Computer Science**

Supervisors:

Prof. Dr. Jens Lehmann
Lavdim Halilaj

Institut für Informatik III

Rheinische Friedrich-Wilhelms-Universität Bonn

# Declaration of Authorship

I, Ahmad Hemid, declare that this thesis, titled "RDF Doctor: Holistic RDF Syntax Validation and Error Correction", and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. Except for such quotations, this thesis is entirely my own work. I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: ________________________________________

Date: ________________________________________

# Acknowledgements

I would like to thank

- My parents
- My wife
- My children

# Contents

# List of Figures

## Abstract

# Chapter 1

# Introduction

## 1.1 Introduction

More and more the usage of RDF is increasing in many fields in computer science. RDF data representation helps in supporting the machine to perform the normally manual computation work in an automatic fashion. Moreover, the machines will be smarter to understand the data which is represented in RDF format.

The quality of RDF data needs to be ensure before proceeding of any further processing. Most of current parsers which focus on detection of the syntax error fail to detect more than one error, especially, of RDF data represented in Turtle or NTriples format.

This study was encouraged by the tremendous data representation of either Turtle and NTriples. Hence, the intention of the study to afford a user-friendly syntax checker or parser. Such parser or syntax checker should give all errors can be detect inside such data.

### 1.1.1 Motivation

This study was motivated by several scenarios which require syntax checking of RDF data as an input and ensuring of its quality. To mention one of such scenarios, let's discuss an example shown in *Figure 1.1. The example describes a collaboration system for processing an input data, say for example to perform machine learning analysis. Of course,in a such case, a valid input data to the system is must. The input data is verified for further data processing. Most of the current existing systems ensuring syntax-error-free RDF data, are stopped parsing at the first syntax error occurrence, as will be followed in Section* **??**.

*To stop parsing when the first syntax error is found will introduce much complications. Assuming, the input RDF data contains, for example, 10 syntax errors. Normally, what is happing when an error is found, the system will proceed with no further processing, instead it will report error's existence in the inserted data. The reported error then should be corrected by the user, then after correction data will be send back for re-checking of data's syntax. To make it more complex, imagine that the user will do such correction process for 10 times (remember that data includes 10 errors). Then, what if the data contains hundreds or thousands of errors.*

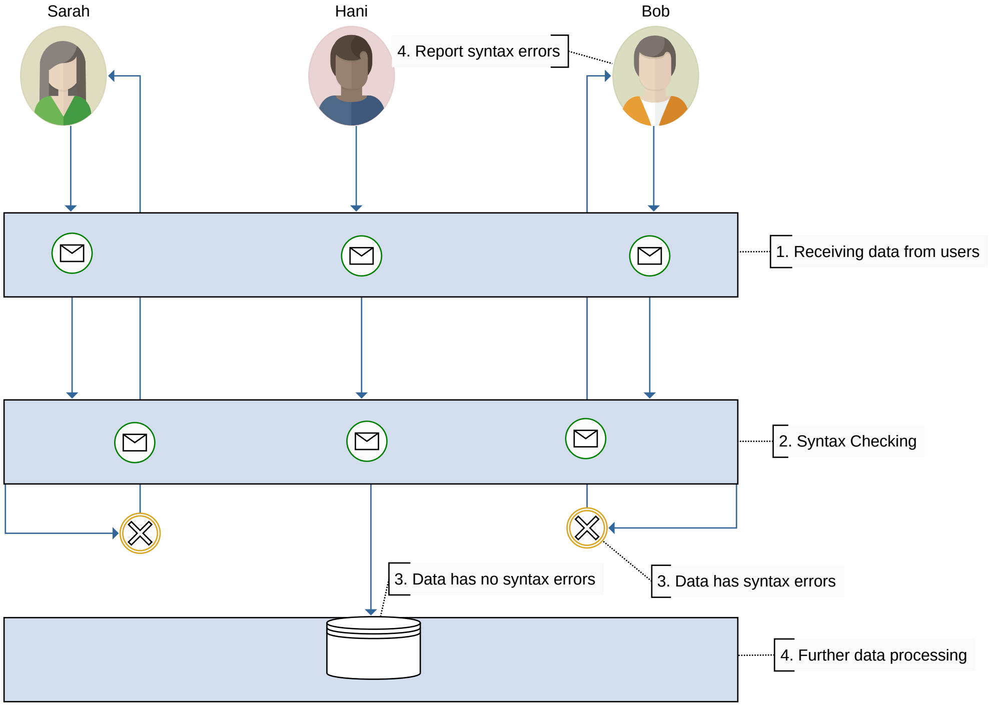
Sarah
Hani
Bob
4. Report syntax errors
1. Receiving data from users
2. Syntax Checking
3. Data has no syntax errors
3. Data has syntax errors
4. Further data processing

Figure 1.1: A motivation example of syntax checking of data before further processing

*let's dig deep to explain what is there in Figure 1.1, It is showing a flow of data from clients or users seeking further data processing. 3 persons are shown in this Figure, their names are Bob, Hani, and Sarah. All of them start with the first phase by sending the data to be syntacticly checked. The parser starts checking if there syntax errors of the input data, then if such data passed with no syntax errors, it can be forwarded for further phases, i.e. for data processing, otherwise, the input data will send back to the user to correct the errors. Figure 1.1 clearly shows that Sarah and Bob have syntax errors in their input data,then they got an error report, including found errors. In the meanwhile, Hani has received his data processed without getting such an error report, since his input data has no syntax errors.*

*This study has been encouraged by the illustrated example to find a suitable solution for such cases. The proposed solution will focus on producing* *a software program that can detect all or almost syntax errors that can be detected in the input data.*

## 1.2 Proposed Problem

*The tackled problem in this study is to list all the detected syntax errors found in RDF files; in order to help ontology engineers and users to fix them in one shut. Such a list of errors provided in the output of the syntax checking phase will significantly assist to get rid of a loop of first error notification if multiple errors are found in the input data.*

## 1.3 Contributions

*The contributions of this study are:*

1. **Reporting list of detected syntax errors found in RDF data:**
2. **Showing expressive error messages:**
3. **Correcting some errors:**

## 1.4 Thesis Structure

*The remainder of this document is structured as follows. In the Related Work chapter*

*The thesis is organized as follows. The next section presents the problem description. Section* **??** *reviews the up-to-date research works. Section* **??** *describes the data set and Section* **??** *presents the results. Finally, Section* **??** *concludes*

# Chapter 2

# Related Work

*In order to validate RDF code, either by pasting URL where it exists or by uploading a file, almost the available tools and applications that we could find, will only give the first occurred error. Moreover, semantic developers and engineers will struggle in debugging their codes and they need alternative tools that could be more helpful. To the best of our knowledge, there is no comparable prior work regarding fault-tolerant tool to validate syntactically RDF serialization formats expect one that works only for RDF/XML format, the following text sheds light on this tool. The new proposed tool should feature prominently in listing of all errors included in the code.*

*This section reviews the related research works have been done and presents the current state of the art of RDF syntax validating. Despite the long record of RDF syntax validation research with many of theoretical models or practical tools, we can hardly find a research that describes the challenge of detection multiple syntax errors inside RDF code. During our journey of checking the existing tools that provide such validation service, The W3C RDF validation tool [**?** ]W3C:Validation:Online was firstly checked, it is available online for parsing and validating RDF/XML codes. It uses the ARP parser of Jena [5] as a backend. However, it fails in detection of multiple syntax errors, the first error in the order will be only released. K. Tolle developed a Validating RDF Parser (VRP) [9] in his thesis, VRP is a Java-based build tool, and it validates RDF/XML code semantically and syntactically. Nevertheless, the validation service provided by VRP is limited to RDF/XML and does not support other RDF serialization formats, especially those formats which are structured in triples such as N3, NTriple, and Turtle. In this work, extension of syntax validation to other formats is planned.*

*The journey to check the existing tools that validate RDF serialization formats other than RDF/XML is continued. As previously stated, Jena RDF toolkit [5] offers validation service based on ARP parser. It can be used as a command-line program (standalone) or as an API within another application. Despite its ability to validate numerous RDF serialization formats, including RDF/XML, again, the first error is only reported. Some of the tools validating RDF formats use the following core techniques as a significant part of their implementations:*

- ***ARP-parser-dependable approach :*** *both W3C RDF validation tool [3] and RDF Validator and Converter [1] use ARP parser of Jena [5]. Moreover, the latter focuses more on triple-based serialization formats, validating them and converting from one format to another, where the former validates only RDF/XML format.*

- ***N3-parser-dependable approach :*** *N3 parser can also be used for syntax validation. In the online IDLab Turtle Validator [8], N3 parser powered by N3 NodeJS library is used. As well, same approach was used to build a turtle editor with syntax validation in [6].*
- ***Shape expressions approach :*** *in [7] a turtle parser was developed based on shape expressions. Shape expressions validates RDF through declaring of constraints on the RDF model, if the declared constraints are violated, then RDF is invalid. Furthermore, Shape expressions describes the RDF graph on regular expressions base.*

*When it comes to the application side, N3-parser-dependable approach can be fitted perfectly in the new tool, since it is built using Nodejs library. This can improve the performance of the tool, especially when it is validating a large RDF code. Moreover, the first two approaches are more expressive in explaining the syntax error and its location, where as, the tool used the third approach is less expressive.*

*In this research, our intention goes toward inventing a fancy tool that lists all syntax errors with an improved performance. The proposed tool can have a solution for the explained issue in either two ways:*

- ***Patching the output errors of parsers :*** *while reviewing the source codes of others' tools, an error event by an error handler will be emitted to show the first occurred error. An idea of looping inside the RDF code and fixing eachtime the first error can be suggested. Fixing the error can be by either deleting the triple made the error, removing or adding a punctuation, inserting a dummy IRI for an incorrect or missing one, etc, then reprase the RDF code again and again till the end of the code .*
- ***Parser Optimization :*** *this needs to review deeply the whole code of the parser and improve its method. The improvement should list all syntax errors that the parser can detect. Both parsers built with N3-parser-dependable approach or Shape expressions approach can be optimized to reach our goals while the optimization of the latter inherits more complexity.*

*To end this section, after describing the actual issue, reviewing the state of art of research works related to it, and finally presenting the possible solutions, we can say that both two solutions can solve the issue, but it seems to us that the second solution more efficient than the first, since this is the normal way* *how acually most of editors of programming languages work, to alert on-the-fly syntax errors to the programmer, even before compilation .*

# Chapter 3

# Preliminaries

## 3.1 RDF Model

*The "Semantic Web" [2] term has appeared during the transformation process of Web Development from "Web of documents" to "Web of data", similar to those data are found in databases. W3C defines it as "the Web of linked data" . Figure 3.1 describes the Semantic Web Stack, proposed by W3C. It can be seen, that it contains several technologies to enable users of creating their own data stores on web, building vocabularies, and enforcing processing rules on such data.*

*In order to make data more and machine-readable, RDF Model has been proposed. RDF Model is considered as a model for data interchange in the new generation of web stack (called Semantic Web).*

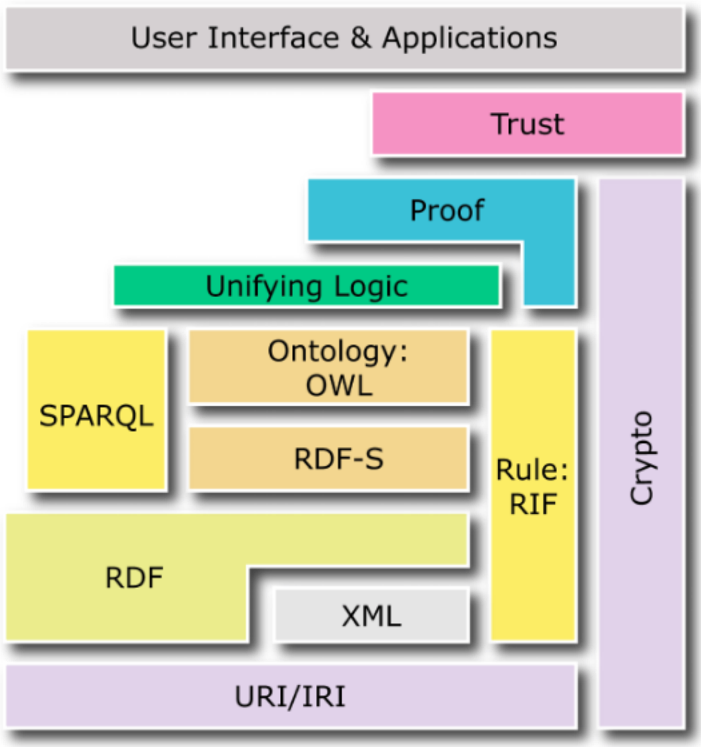

Figure 3.1: The Semantic Web Stack [4]

## 3.2 Parsing Methodologies

*If you have a chance to look to the red book of the compiler, then you absolutely know that there are several methods for building parsers.*

## 3.3 ANTLR Parser Generator

*ANTLR is an handy tool and easy way to have a parser. It is an parser generator where an normal user can build his own language's parser without much works. The basic principle used by ANTLR is define language's rules which draws the syntax and the semantic of the language the parser build for.*

*As has been previously discussed, the compiler has two main subsystems: lexer and parser. Both lexer and parser are needed to have their rules defined in ANTLR grammar file.*

# Chapter 4

# Approach

## 4.1 Proposed Solution

# Chapter 5

# Implementation

*In this chapter, we discuss in detail of the implementation*

## 5.1 Architecture

## 5.2 Modules

# Chapter 6

# Evaluation

### 6.0.1 Experimental Setup

### 6.0.2 Preliminary Results

#### Accuracy

#### Scalability

# Chapter 7

# Conclusions and Future Work

# Bibliography

[1] *RDF Validator and Converter @ONLINE . URL* `http://rdfvalidator.mybluemix.net/`.

[2] *W3C Semantic Web @ONLINE , . URL* `https://www.w3.org/standards/semanticweb/`.

[3] *W3C RDF validation Service @ONLINE , . URL* `http://www.w3.org/RDF/Validator/`.

[4] *Steve Bratt. Semantic Web, and Other Technologies to Watch. World Wide Web Consortium, 2007. URL* `http://www.w3.org/2007/Talks/0130-sb-W3CTechSemWeb`.

[5] *Brian McBride. Jena: A semantic web toolkit.* IEEE Internet Computing, *6(6):55–59, November 2002. ISSN 1089-7801. doi: 10.1109/MIC.2002.1067737. URL* `http://dx.doi.org/10.1109/MIC.2002.1067737`.

[6] *Niklas Petersen, Gökhan Coskun, and Christoph Lange. TurtleEditor: An ontology-aware web-editor for collaborative ontology development.*

[7] *Eric Prud'hommeaux, Jose Emilio Labra Gayo, and Harold Solbrig. Shape expressions: an rdf validation and transformation language. In* Proceedings of the 10th International Conference on Semantic Systems, *pages 32–40. ACM, 2014.*

[8] *Miel Vander Sande. IDLab Turtle Validator @ONLINE . URL* `http://ttl.summerofcode.be/`.

[9] *Karsten Tolle. Analyzing and Parsing RDF. Master's thesis, 2000.*

# Appendix A

# Parser Rules

*The contents...*

# Appendix B

# ANTLR Compiler Tutorial

*The contents...*