NIF Session at SD-LLOD-17

Release 0.1

Markus Ackermann

Jun 27, 2017

Contents

These pages provide instructions and advice for the practical tutorial session on the NLP Interchange Format (NIF) at the Second Summer Datathon on Linguistic Linked Open Data (SD-LLOD-17).

The NLP Interchange Format (NIF) is an RDF/OWL-based format that aims to achieve interoperability between Natural Language Processing (NLP) tools, language resources and annotations.

CHAPTER 1

Preparations

Suitable Text Editor

You should have a text editor set up which offers syntax-highlighting for Turtle (an RDF serialisation format) and (optionally) for SPARQL queries.

Several more powerful editor (e.g. Emacs, Vim, Notepad++) either support support for these syntaxes out of the box or there are appropriate community extensions. Thus, if you are experienced in their usage, setting up your favorite of these editors for Turtle and SPARQL might be your best option.

For all users with no prior experience with the aforementioned editors, we recommend installing and setting up Atom for this session. Installers for the recent version for all common platform are available at GitHub. After installation of Atom, you can install the packages language-rdf and language-sparql directly from within the Editor itself, using the Settings. If you prefer the command line way, just type: **`apm install language-rdf`** and **`apm install language-sparql`**

Java 8 JRE

Most tools that will be used in the session require the Java Runtime Environment (or Java Development Kit) Version 8. For recent Linux distributions, a Java 8 JRE should be available in the official repositories. If you are on Windows or Mac and do not have Java 8 set up already, please obtain it from Oracle.

Please also ensure that the directory with the Java executables has been added to your `$PATH`. To verify that `$PATH` is set correctly, enter `java -version` into a command line prompt. The output should contain `version "1.8.0` as a substring.

Notes on Command Prompt Usage (only relevant on Windows)

We will use the command prompt regularly in this session. For the sake of simplicity, the rest of the documentation will give the command examples generally with the command names for BASH, the default command prompt on

(almost) all Linux versions.

Current versions of Windows mostly ship with the PowerShell, which has aliases for all the common BASH commands occurring in this tutorial to their Windows counterparts, thus, the commands should be executable in a PowerShell session without alterations. If you use the traditional Windows Command Prompt, some command names and or options will have to be adjusted by you (e.g. `dir` instead of `ls`). Alternatively, Babun can be used.

CHAPTER 2

Excercises

Excercise A: Understanding and Modifying a NIF document

A1: Finding One Way Through a NIF Document

The file `data/wilde.ttl` contains a smalll NIF document representing a quote from Oscar Wilde. Open it in your text editor with syntax highlighting for Turtle and try to get an overall impression of the structure described by the triples for a minute.

This auto-generated documentation page for NIF Core vocabulary contains explanation on the semantice of the occurring classes and properties.

Next, ensure that the *Fuseki SPARQL Server* has been started and start browsing with the *LodLive Web of Data Browser* by either visiting the the URL `file://$MATERIALS_ROOT/tools/lodlive/app_en.html?http://sd-llod-nif-session.nlp2rdf.org/text/ow/offset_0_49`[1] or start the naviation as described in the tools page. Then you start browsing from the `nif:Context` of that doument in the . Expand all outgoing `nif:word` relations from the context and arrange the `nif:word` circles in a way that the linear chain of words in the sentence can be followed easily. (This might look like `includes/exA-lodlive-Wilde-initial.png`).

A2: Modifying a NIF Document

Now think of a little adjustment of the statement achived by adding a short (up to four words) sentence or replacing/inserting a few words in the additional sentence. For example:

> *One can survive everything nowadays except death. What about Two?*

(Please pardon the lowest form of humour[2] and feel invited to use a creative alternative.)

Start adjusting the document so that it represents the altered/extended version. For the above example, the following table provides the character offsets:

[1] *$MATERIALS_ROOT* here is an insertion mark for the folder where you placed the extracted contents of the materials archive.
[2] "A pun is the lowest form of humor—when you don't think of it first." - Oscar Levant

token	begin	end
What	50	54
about	55	60
Two	61	64
?	64	65

Note: In the original document the `nif:Context` and the single `nif:Sentence` were the same string individual. This combination of roles is not applicable any more if another sentence is added.

Since one can easily creates syntax mistakes when editing Turtle manually (unless one is routined doing so), use `turtle` / `turtle.bat` from the *Apache Jena Command Line Tools* by the following command line invocation (with the working directory at *$MATERIALS_ROOT*) `./tools/jena-cmds/bin/turtle --validate data/wilde.ttl` (Linux/Mac) or `.\\tools\\jena-cmds\\bat\\turtle.bat --validate data\\wilde.ttl` (Windows).

When there are no messages about syntax errors/warnings from the validation tool, you can stop and restart the Fuseki Server. This will allow to browser the NIF graph that will reflect the changes you applied.

Further, you should use RDFUnit to check whether your modified document still adheres to fundamental constraints of NIF documents. To do so, invoke `./tools/run-rdfunit-on-wilde.sh` or `.\\tools\\run-rdfunit-on-wilde.bat` at *$MATERIALS_ROOT* in a command prompt. A successful execution will terminate with lines similar to these:

```
[INFO SimpleTestExecutorMonitor] Tests run: N, Failed: F, Timeout:
0, Error: E. Individual Errors: I [INFO ValidateCLI] Results
stored in: rdfunit/results/sd-llod-nif-session.nlp2rdf.org_text_ow.
extendedTestCaseResult.*
```

(If the execution instead fails with a stack trace, you might have submitted a malformed Turtle document - please re-validate with the `turtle` command to exclude this error source. If the unsuccessful execution persists, let the tutor have a look.)

Opening the file `rdfunit/results/sd-llod-nif-session.nlp2rdf.org_text_ow.extendedTestCaseResult.html` in your browser will give a detailed listing of errors found in the RDFUnit run and the subject resource IRIs of the violating statements. Rectify all issues and repeat the validation & check procedure above.

A solution for the modification suggested earlier can be found in `./examples-solutions/wilde-extended.ttl`

Excercise B: Querying NIF data with SPARQL

For this excercise, use the Fuseki dataset `brown`. It contains a subset of the documents from the traditional and eminent BROWN corpus of English, converted to NIF, with Part-of-Speech-Tags (POS-Tags) represented as OWL instances from the Ontology of Linguistic Annotations project.

To get a first feel how information on the POS annotation is represented, start again using *LodLive Web of Data Browser* navigate the graph, for example from either

```
http://brown.nlp2rdf.org/linkeddata.php?t=url&f=xml&i=http://brown.
nlp2rdf.org/corpus/a01.xml#offset_0_155 or http://brown.nlp2rdf.org/
linkeddata.php?t=url&f=xml&i=http://brown.nlp2rdf.org/corpus/a02.
xml#offset_367_443.
```

First nagivate to one or two word resources, from them follow `nif:oliaLink` and then investigate the inheritance relations (`rdfs:subClassOf`) starting from classes that were assigned to the OLiA tag instances.

Tip: You can inspect the classes and individuals of the BROWN tag set representation of OLiA using an HTML page generated by LODE or by using Protege opening `ontologies/brown.owl`.

Now it's time to compose the first SPARQL query over this graph. Ensure that the *Fuseki SPARQL Server* is running and then visit `http://localhost:3030/dataset.html` in your browser. You can enter queried directly in the text field overed in the query view, but it is recommended to save earlier attempts and query snippets in an editor in parallel via copy & paste, as you might want to re-use and adapt earlier attempts. (The Fuseki query does not offer a query history feature.)

Tip: To look up specific details about the SPARQL syntax, definded functions etc., you can consulted in the official SPARQL 1.1 Spec from W3C[3].

B1: Finding Adjectives

As first query, retrive a sample of eight words tagged as adjectives (`olia_brown:JJ`). The binding in the result set should contain both the IRIs of the word resources and their sub-strings they point at.

B2: Basic Prevalence Statistics for the POS Tags

Now, find out which are the four POS tags that were assigned most frequently to the tokens occuring in the graph. You can compare your query with a version optimized for brevity in `examples-solutions/most-common-brown-tags.sparql`.

B3: Searching for Usage of Passive Voice

For combined usage of NIF structure and OLiA, try now to compose a query of two adjacent words that are part of a verb phrase in passive voice, i.e. we look for two adjacent words where the first one is a form of *to be* (the tag should carry be an instance of any sub-class of `olia_brown:Be`) and the second is a past participle (the tag schould be an instance of `olia_brown:LexicalVerbPastParticiple`).

Tip: To check for whether `?classA` is a (transitive) subclass of `?classB`, the Property Path feature of SPARQL is helpful.

After completing a query with the constraints described above you will notice that the result bindings also contain phrases like *'been charged'* that are rather part of the past perfect constructions. These cases can be excluded by disallowing that the first word described in the query is preceded by a form of have. Extend the query accordingly .

Tip: The `FILTER NOT EXISTS { [...] }` construct of SPARQL, will be useful for the exclusions.

You can compare your (extneded) query with `examples-solutions/passive-voice.sparql`.

[3] Disregarding the section 18 completely, which is only relevant for creating implementations of the lanugage. Parts of the language described in sections 13, 14 and 16.2 to 16.4 will not be needed for these excercises as well.

B4: Searching for Zero-Conversions

In Englisch derivations involving word class change between adjective and verb without any change in the written representation is quite common. Formulate a query that looks for candidates of evidence of such conversions in the BROWN graph, i.e. word-forms that occur both as adjective and verb.

Tip: This means we are looking for a pair of word instances that share their `nif:anchorOf` value.

You can compare your query with *examples-solutions/adj-verb-zeroconv.sparql*

After having the query that collects individual points of evidence, modify the query so that for each these candiate word forms, the number of occurrences as adjective and as verb is calculated.

Tip: Achieving this will require joined sub-queries.

You can compare your query with *examples-solutions/adj-verb-zeroconv-with-stats.sparql*

CHAPTER 3

Tools

Apache Jena Command Line Tools

Jenas command line tools are a collection of Shell scripts/Batch scripts that allow to access functionalities of that RDF/OWL programming library without composing Java programs. For the excercises proposed, usage will probably be limited to the tool that reads Turtle RDF documents to either validate them syntactically or to convert them to another RDF serialisation format. (`turtle`/ `turtle.bat`).

Fuseki SPARQL Server

Fuseki allows to query and update configured RDF datasets using SPARQL queries. It also offers a browser-based application to formulate and execute SPARQL queries with syntax highlighting and hints for syntactic error in entered queries.

You find a copy of Fuseki pre-configured for the excercises in `tools/fuseki`. To start it, oben a new terminal window, navigate to that directory and run the start script `./fuseki-server` (Linux/Mac) or `.\fuseki-server.bat` (Windows). The command prompt will remain occupied by the server process during execution and log messages about the initlization process, received queries etc. will be printed. To terminate the server process, send `SIGINT` by pressing **CTRL +C** in the terminal with the server process running.

Once Fuseki has been started, you can visit `http://localhost:3030/dataset.html` to access the graphical SPARQL user interface. Ensure that the appropriate dataset for the excercise you intend to tackle is selected in the *'Dataset'* drow-down menu at the top. (`/wilde` for Excercise A, `/brown` for Exercise B).

LodLive Web of Data Browser

LodLive is a Javascript-based browser application that allows stepwise exploration of RDF knowledge graphs. A copy of LodLive slightly modified and pre-configured for this session is located in the `tools/lodlive` directory.

To lauch, open the `app_en.html` start page in that directory in your browser. When invoking this HTML document without additional query parameters in the URL, you will be presented with a start screen. The boxes in the lower row in light green represent the configured local datasets. With the contained dropdown menus you can select example resources to start a browsing session. If you already know a specific resource IRI you would like to use as starting point, you can either enter it in the blue box on the start screen or add it to the URL presented in the navigation bar of your browser:

`file://`*`$MATERIALS_ROOT`*`/tools/lodlive/app_en.html?`*`$RESOURCE_IRI`*

When after selecting your entry resource, you will see the actual graph exploration view with the representation of the selected resource as first circle in the center of the viewport. Although usage for further exploration is quite intutive, please consider opening and skimming the in-app explanation offered by buttons on the left side of the viewport.

Note: LodLive retrieves the statements to describe and visualise resources from the local *Fuseki SPARQL Server*, so please ensure that it is up and running before using LodLive.

Index

Symbols

E