From FOAF to English: Linguistic Contribution to Web Semantics

Max Silberztein

From FOAF to English: Linguistic Contribution to Web Semantics

Max Silberztein
Université de Franche-Comté
max.silberztein@univ-fcomte.fr

Abstract

This paper presents a linguistic module capable of generating a set of English sentences that correspond to a Resource Description Framework (RDF) statement; I discuss how a generator can control the linguistic module, as well as the various limitations of a pure linguistic framework.

1 Introduction

Automatic Natural Language Generation Software aims to express information stored in a knowledge database as a Natural Language text. The information at the source typically consists of a series of simple atomic statements, such as "John owns a house", "Mary lives in Manchester", "Peter is Ann's cousin". These statements are usually stored in a knowledge database or ontology in a formal notation such as Prolog (e.g. "Own(John,House)") or XML.

Translating each elementary statement into an isolated English sentence is straightforward; the difficulty arises when one tries to process a complex set of statements to generate a text that feels "natural": an entity that is mentioned several times might then have to be referred to by a pronoun, a possessive determiner (e.g. *He is her cousin*), or an anaphoric term (e.g. *The student is her cousin*); a complement might need to be brought into focus (e.g. *It is Peter who is Ann's cousin*); subsequent sentences might need to agree in tense and aspect, etc. For each original individual statement, there might be thousands of potential English sentences that can express it: the generator must then decide which sentence to produce. This article presents the linguistic component of such a system.

2 The linguistic framework

Based on the principles of linguistic approaches to generation laid out by Danlos (1987), I have used the NooJ[1] platform to construct a set of linguistic resources that parses a sequence of RDF statements and produces a corresponding set of English sentences. NooJ allows linguists to construct structured sets of linguistic resources ("modules") in the form of dictionaries and grammars[2] to formalize a large gamut of linguistic phenomena: orthography and spelling, inflectional, derivational and agglutinative morphology, local and structural syntax, transformational syntax, lexical and predicative semantics. All linguistic analyses are performed sequentially by adding and/or removing linguistic annotations to/from a Text Annotation Structure (TAS); at each level of the analysis, each parser uses the annotations that were added to the TAS by preceding parsers, and then adds new annotations to the TAS, or deletes annotations that have been proven to be incorrect. This architecture allows the system to perform complex linguistic operations that require information coming from all levels of analyses, even when total disambiguation was not possible at earlier stages of the analysis, thus avoiding the problems at the heart of criticisms against pure linguistic approaches.[3]

For instance, to generate the sentence "She is Joe's love" from the elementary statement "Joe loves Lea", a linguistic system needs to access the following information:

[1] See Silberztein (2016a). NooJ is a free, open-source linguistic development environment supported by the European Metashare program.

[2] Regular Grammars, Context-Free Grammars, Context-Sensitive Grammars and Unrestricted Grammars can be entered either in a textual or in a graphical form. The grammars shown in this article are graphical Context-Sensitive Grammars.

[3] See for instance the "generation gap" discussed by Gardent Perez-Beltrachini (2017).

Proceedings of the Linguistic Resources for Automatic Natural Language Generation Workshop, pages 1–9,
Santiago de Compostela, Spain, September 4, 2017.

- the word "loves" can be a conjugated form of lexical entry *to love*;[4]
- the verb *to love* can be nominalized into the Human Noun *a love*;[5]
- the structure N_0 V N_1 can be restructured as N_1 is N_0's V-n;
- the Noun *Lea* is feminine therefore it can be replaced with pronoun *she* when it is in a subject position.

One important characteristic of NooJ resources is that they are "application-neutral": they can be used both by parsers and by generators. This allows a single software application to both:

- parse sentences, e.g., from sentence "*It is not Lea that he loves*", produce the analysis "*Joe loves Lea* +Focus1 +Neg +Pron1",
- or, the other way around, given the elementary sentence "*Joe loves Lea*" and the series of operators "+Pro0 +Preterit +AspCont +Intens2", generate the complex sentence "*He continued to love Lea for a long time*".

Given the elementary sentence *Joe loves Lea*, Silberztein (2016b) showed that by combining linguistic operations such as negation (e.g. *Joe does not love Lea*), focus (e.g. *It is Joe who loves Lea*), tense (e.g. *Joe loved Lea*), aspect (e.g. *Joe has stopped loving Lea*), modality (e.g. *Joe should love Lea*), intensity (e.g. *Joe loves Lea passionately*), pronominalization (e.g. *He loves her*), nominalization (e.g. *Joe is in love with Lea*), etc., a system can generate over a million declarative sentences (e.g. *It is not her that he stopped loving*), about 500,000 nominal phrases (e.g. *Joe's passionate love for her*) and over 3 million questions (e.g. *When did Joe start loving her?*). Each generated sentence is associated with the series of transformations (e.g. +Passive, +Focus1) used to produce it.

In this article, I show how this system can be adapted so that an NLG system can control what exact English sentence(s) need to be generated.

3 FOAF Predicates

The Semantic Web[6] constitutes a gigantic network of ontologies that contain elementary pieces of information, written in the RDF syntax. A typical RDF statement is a triple that contains one subject entity, one predicate and one object entity; the predicate states the type of relationship between the two entities. All three elements are identified by a URI. For instance, the following RDF triple states that the person "Mark_Twain" is the author of the book "Huckleberry_Fin":[7]

```
<http://example.org/Mark_Twain>
<http://example.org/author>
<http://example.org/Huckleberry_Fin>.
```

In this article, I focus on the *Friend Of A Friend* (FOAF) ontology (FOAF Vocabulary Specification 2010), which contains a set of classes for entities:

Agent, Document, Group, Image, Organization, Person, Project...

and a set of properties (i.e. predicates), e.g.:

account, age, based near, birthday, currentProject, familyName, gender, givenName, interest, knows, name, title…

4 From RDF to English

A linguistic module capable of parsing RDF statements and producing the corresponding potential English sentences needs the following resources:

- a set of lexical and morphological resources to link all words and expressions to their actual inflected and derived forms (I am using NooJ's default English module);
- a syntactic grammar to parse an RDF statement and extract from it the value of its entities and predicate;
- one syntactic grammar for each FOAF property, in order to describe the set of English sentences that can be used to express it.

The grammar for FOAF property *currentProject* shown in Figure 1 contains four parts:

- **Turtle**: this grammar describes RDF statements expressed in the simple Turtle notation;

[4] There are other possible analyses such as in *loves* = Plural of Noun *a love*.

[5] There is a second nominalization that is not in play here : *Joe's love for Lea*, where *love* is an abstract noun.

[6] See Berners-Lee et al. (2001).

[7] I am using the Terse RDF Triple Language (Turtle) syntax, see RDF1.1. Turtle (2014).

- **declarative**, this grammar describes declarative sentences, e.g. *Tim Berners-Lee is currently working on the World Wide Web project*
- **noun phrase** describes noun phrases, e.g. *Tim Berners-Lee's World Wide Web current project*
- **question** describes questions, e.g. *Is Tim Berners-Lee involved in the World Wide Web project?*

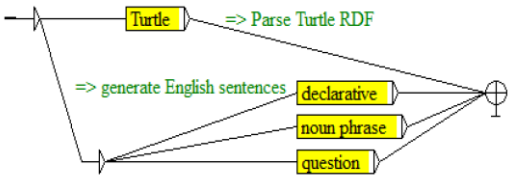

Figure 1: Parse RDF and generate sentences

Note that the same grammar can be used to parse an RDF statement and produce the corresponding English equivalent (sentences, phrases and questions), or reciprocally, to parse any English sentence, phrase or question and produce the corresponding RDF statement. In this article, I assume that the system receives an RDF statement as its input; it will then produce the corresponding English declarative sentences, phrases and questions.

4.1 Parsing an RDF statement

Parsing an RDF statement written in the simplified Turtle notation is straightforward: a statement is a sequence of three XML tags followed by a period; each tag contains an URI that represents an entity or a predicate. For instance, consider the following triple:

```
<http://dbpedia.org/Tim_Berners-Lee>
<http://xmlns.com/foaf/0.1/currentProject>
<http://dbpedia.org/World_Wide_Web>.
```

Grammar **XML**, shown in Figure 2, extracts the suffix of each tag's URI and stores it in variable **$Suf**. Note that the suffix may contain any number of letters, digits, periods, dashes and underscore characters.[8]

The main grammar **Turtle** shown in Figure 3 contains three references to the **XML** graph: it parses a sequence of three consecutive XML tags and computes the value of each variable **$Suf**. Each subsequent value of **$Suf** is then copied to the corresponding global variables **@Subject**, **@Predicate** and **@Object**.[9]

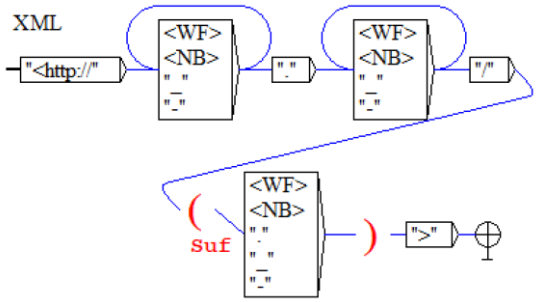

Figure 2: Parse XML tags

After parsing the previous RDF statement, variable **@Subject** is set to "Tim_Berners-Lee", variable **@Predicate** is set to "currentProject" and variable **@Object** is set to "World_Wide_Web".

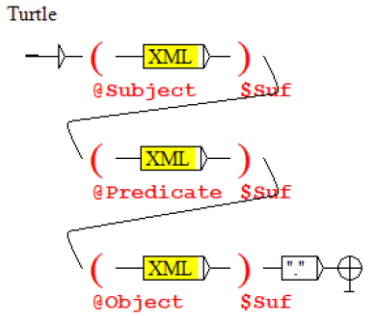

Figure 3: Parsing Turtle RDF Statements

4.2 Generating English Sentences

Each property from the FOAF ontology corresponds to a set of English sentences that can be used to express it. In this approach, one must construct one grammar to generate all the English sentences that correspond to each of the FOAF properties *name*, *firstName*, *givenName* and *familyName* (e.g. *His first name is Tim, Berners-Lee's given name is Tim*), a grammar that corresponds to the FOAF property *age* (e.g. *John is 18-month old; Mary is 12; Joe is still a teenager; Lea is a senior citizen*), a grammar that expresses the fact that a person knows another person (e.g. *John is acquainted with Mary, Lea has met Joe already*), a grammar that expresses the fact that a person is currently working on a project, another for

[8] <WF> matches any sequence of letters; <NB> matches any sequence of digits.

[9] Variables with prefix "@" have a global scope. This allows the system to link a given entity to all its references (e.g. "Lea" with Pronoun "her") across a grammar that may contain dozens of graphs. Here, we want to link **@Predicate** to all its English corresponding terms, whether they are Verbs (e.g. *works on*), Adjectives (*is involved in*) or even Nouns (e.g. *head of*).

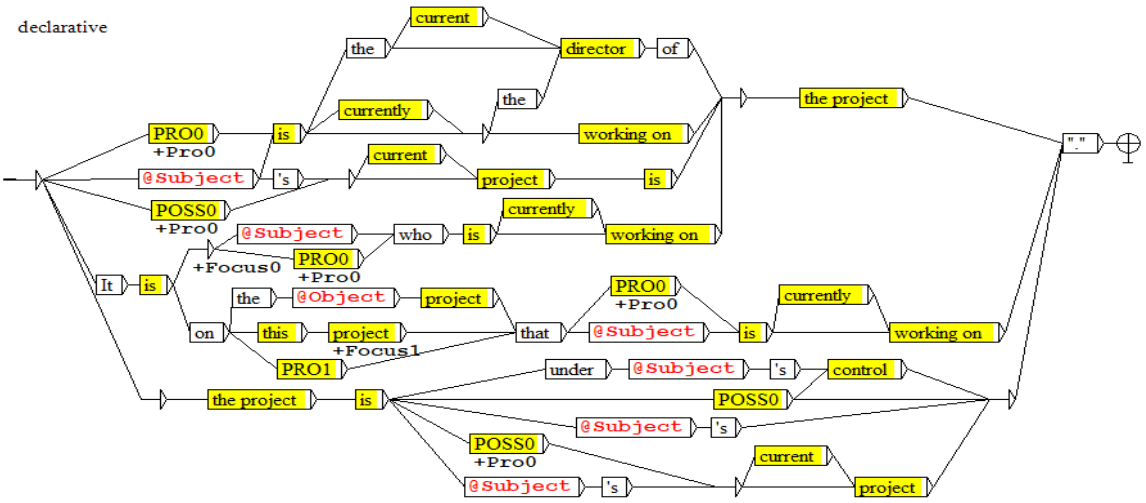

Figure 4: Declarative Sentences

property *pastProject*, another for property *enemyOf*, etc.

Being able to automatically produce questions would be useful for a few specialized applications such as literature or language teaching (whereas a software automatically generates questions from a study text that students are expected to answer) or question answering, whereas sentences recognized by the **declarative** grammar are potential answers for any question recognized by the **question** grammar.[10] In this article, I present the **declarative** and **noun phrase** grammars.

4.3 Declarative Sentences

The entrance point for the grammar that represents (i.e. can parse and/or generate) the declarative sentences for property *currentProject* is shown in Figure 4.

The grammar uses the value of the variables **@Subject** and **@Object** (in red in the graph) that were set by the parsing of the *currentProject* RDF statement. This graph contains references to embedded graphs (in yellow in the graph) such as **current**, **project**, **the project**, etc. For instance, the embedded grammar **current** represents the following Adjectives:

- **current** = current | in progress | ongoing | present | present-day;

The grammar **project** contains the following nouns:

- **project** = activity | affair | adventure | assignment | business | creation | enterprise | job | project | scheme | task | venture;

The graph for **the project** is displayed in Figure 5.

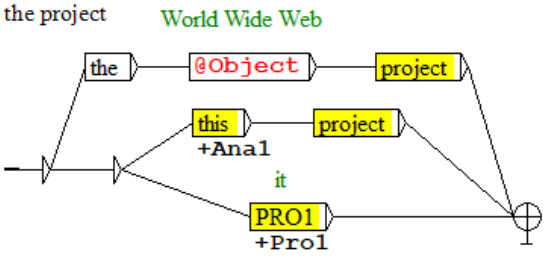

Figure 5: Graph for "the project"

Note that this graph can produce anaphoric terms as well as pronouns, e.g.:

> *Tim Berners-Lee is currently working on* **that enterprise**. **It** *is under Tim Berners-Lee's control.*

The **declarative** grammar for property *currentProject* contains over 30 graphs and represents (i.e. can both parse or generate) over 50,000 declarative sentences.

[10] The **question** grammar generates over one million questions, such as *Who is working on the World Wide Web project? What is Tim Berners-Lee's current project? Is Tim Berners-Lee involved in the World Wide Web project?*

4.4 Noun Phrases

The entrance point for the grammar that represents the noun phrases that might be used to express property *currentProject* can be seen in Figure 6.

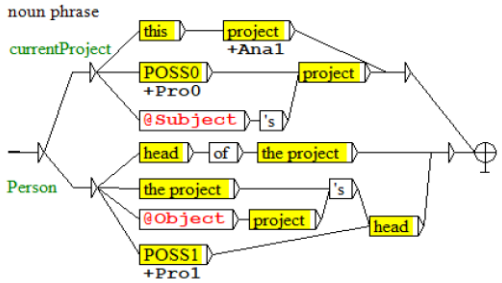

Figure 6: Noun Phrases

This grammar represents (i.e. parses or generates) two types of noun phrases: phrases that focus on the *currentProject* entity, e.g.:

> *Tim Berner-Lee's World Wide Web, his ongoing project, etc.*

and phrases that focus on the *person* entity, e.g.:

> *The World Wide Web project's current director, the present head of that enterprise, its director, etc.*

This grammar does not describe phrases that focus on the date (e.g. *the moment when Tim Berners-Lee's project is the World Wide Web*), even though the information that the project is "current" is a crucial part of the information represented by the RDF statement. Generating phrases from RDF statements that explicitly refer to a project's initial and/or ending dates will require other grammars for dates, such as the default one available in the NooJ's English module.

5 Pronouns and anaphora

The grammar *currentProject* produces certain sentences and phrases that should not be generated in isolation, e.g.:

> *He is currently involved in that project.*
> *His project.*

If the goal is to produce one isolated sentence or phrase, that sentence or phrase should not contain any pronoun, possessive determiner or anaphoric term, otherwise the original information would be lost. However, most NLG applications aim at producing texts that are sequences of related sentences and phrases: in order to keep the resulting text natural, it is then important to be able to use pronouns, possessive determiners, anaphoric terms, as well as every linguistic operator the language offers: aspect, derivation, focus, intensity, modality, tense, etc.[11]

6 Aspect and Tense

The *currentProject* property limits the possible aspect and tense of the generated English sentences to present or present progressive: the linguistic module generates sentences such as the following ones:

> *Tim Berners-Lee (works | is working | is currently working) on the World Wide Web project; Tim Berners-Lee (is involved | is currently involved) in the World Wide Web project; Tim Berners-Lee's (current | in progress | on going | present | present-day) project is the World Wide Web project,* etc.

but it may also generate sentences such as the following ones:

> [+Tense]: *Tim Berners-Lee (was working | has worked | will be working) on the World Wide Web project.*
>
> [+Aspect]: *Tim Berners-Lee started working on the World Wide Web project (in 1989); Tim Berners-Lee has been working on the World Wide Web (for 20 years); Tim Berners-Lee will stop working on the World Wide Web (next year).*
>
> [+Modality]: *Tim Berners-Lee should work on the World Wide Web project; Tim Berners-Lee can work on the World Wide Web; Tim Berners-Lee might work on the World Wide Web.*

as well as sentences that contain any combination of Tense / Aspect / Modality variants:

> *Tim Berners-Lee could have started to work on the WWW project (one year earlier). Steve Jobs has worked on the iPhone project (for a long time). Jürgen E. Schrempp initiated the Chrysler-Daimler merger task (last year). Has Larry Page been involved in the Alphabet Inc.*

[11] See Lloret Pastor (2011) on how the COMPENDIUM automatic summary system manages redundancy and information fusion. The WebNLG Challenge also aims at producing a sequence of sentences that might contain elisions, anaphora or pronouns.

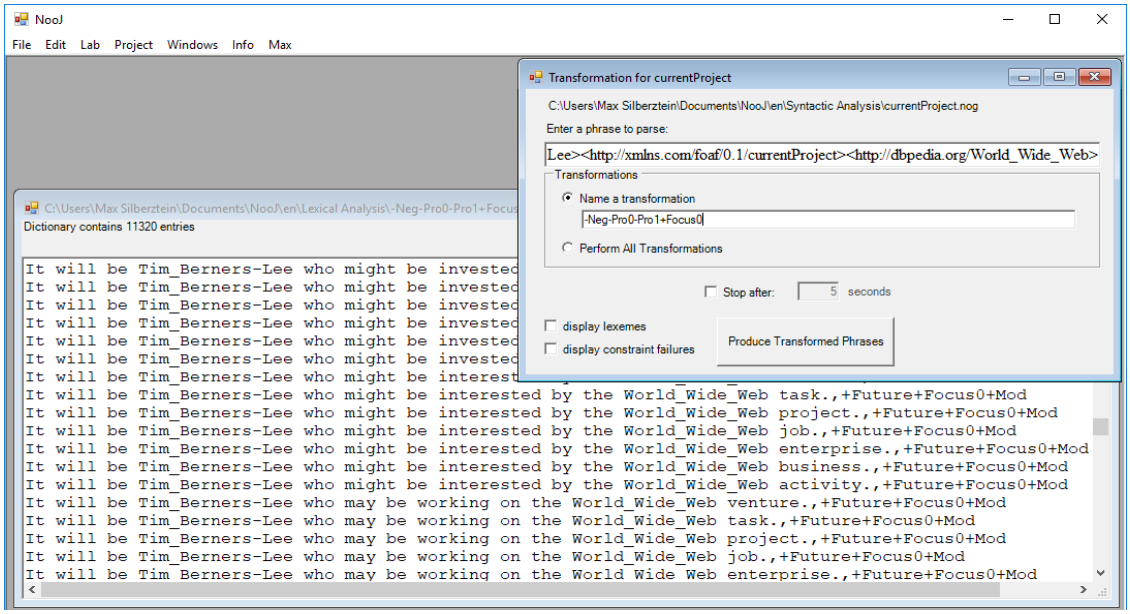

Figure 7: Operator-controlled generation

Company adventure (from the very beginning)?

Based on the sole *currentProject* property, it might not be appropriate to generate these sentences; however, text generators are always used to express more than one piece of information; these sentences will be useful if the generator needs to produce sentences that express properties such as *pastProject*, or if the generator has access to date information such as: *when did Tim Berners-Lee start to work on the WWW*, *when does Jürgen E. Schremp plan to stop working on the merger*, *how long has Larry Page been working on the creation of the Alphabet Inc. Company?* etc.

The linguistic module cannot perform extra-linguistic computations, such as producing complements such as *for 28 years* by subtracting the initial project's date from the current date, by itself. It can, however, perform simple equality tests by using constraints such as <$gender="Male"> (to pronominalize *Tim Berners-Lee* as *he*), and <$pastProject=$currentProject> (to produce sentences such as *Tim Berners-Lee is still working on the World Wide Web*).

7 Controlling the linguistic module

To control what sentence is to be generated, the generator that pilots the linguistic system must send a set of operators that act as parameters. Following are examples of sentences generated, given a set of operators:

- [+AspTilNow+Pro0+Focus1]:
  It is on the World Wide Web venture that he has been working until now.
- [+When+Preterit+Pro0+Pro1]:
  When did he work on that enterprise?
- [+Neg+Future+AspStop]:
  Tim Berners-Lee will no longer work on the World Wide Web adventure.

Operators can be sent to the linguistic module with a "+" or a "-" prefix, to control whether the generator wants to activate, or filter out, the corresponding sentences and phrases. For instance, the generator may filter out sentences that contain a negation or a pronoun with the following sequence of operators: [-Neg-Pro0-Pro1]. Figure 7 shows that this exact sequence of operators makes the linguistic system produce over 11,000 declarative sentences, none of which include a negation or a pronoun.

7.1 Incorrect information

One problem with the pure linguistic approach is that, if not properly controlled, the linguistic module will also generate sentences that misrepresent the initial FOAF information, e.g.:

- [+Neg]:
  Tim Berners-Lee is not currently working on the World Wide Web project
- [+Future+AspCont+Intens2]:
  Tim Berners-Lee will keep on working on the World Wide Web project forever

However, even though the previous sentences are not appropriate, some combinations of these operators may produce correct statements, e.g.:

- [+Neg+Future+AspCont+Intens2]:
  Tim Berners-Lee will not keep on working on the World Wide Web project forever

In other words, linguistic operators such as +Neg or +Future are not "bad" intrinsically: they must be controlled by the generator, just like any other linguistic operator: it is the responsibility of the calling application (here, the generator) to control the linguistic module by setting the correct parameters in order to enable or disable the production of each sentence and phrase.

8 Limitations

There are a few problems with the prototype as it is now.

8.1 Missing information

The single FOAF statement that constitutes the input of the linguistic prototype presented in this article does not mention the entities' names. Therefore, the sentences generated by the prototype actually resemble the following:

> *Tim_Berners-Lee is currently working on the World_Wide_Web project.*

In a finalized software application, the generator should retrieve the value of the person's name property, available as an FOAF property:

```
<foaf:name xml:lang="en"> Tim Berners-Lee </foaf:name>
```

Using the value of the FOAF *givenName*, *firstName* and *familyName* properties for *person* entities would allow the linguistic component to generate abbreviated variants such as "Berners-Lee", or even "Tim" (in a casual context, for instance). In the same manner, the linguistic module would need to access a list of variants and abbreviations for each *project* entity, such as "the Web" or "WWW" for entity World_Wide_Web.

Another important piece of information is the gender of each *person* entity: for Tim_Berners-Lee, the generator needs to combine operator +Pro0 with operator +Mas to stop the linguistic module from generating incorrect feminine or neutral pronouns or possessive determiners such as in: *The World Wide Web is (her | its) current project*.[12] As this information is available in FOAF:

```
<foaf:gender
xml:lang="en">Male</foaf:gender>
```

Another possibility is to add this FOAF statement to the linguistic module to its input, store the value of the gender in a variable (e.g. $gender), and add a constraint on the variable in the grammar everywhere we need to produce a pronoun, such as in Figure 8.

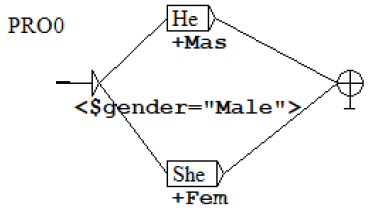

Figure 8: Adding gender information

8.2 What is a *project*?

Because the Web Semantics' entities are meant to represent elements of meanings independent from the languages, they tend to be more generic than actual English terms, which makes it difficult to compute back the sets of English terms they represent.[13]

For instance, the FOAF *project* class regroups entities that are not always easily referred to by the English term "project": it makes sense to qualify the World Wide Web as a project, an enterprise or even a program, but it is much more difficult to use the following terms:

> *Tim Berners-Lee is currently working on the World Wide Web (activity | affair | assignment | business | creation | job | management | scheme | task | venture)*

[12] *gender* is an FOAF property attached to class Agent rather than its subclass *Person*. The generator will therefore need to make entity Tim_Berners-Lee inherit its gender property to make it explicit to the linguistic module.

[13] The vagueness and inconsistency of the Semantic Web are its two most common criticisms.

The World Wide Web has existed too long to be qualified as an *affair*; it is too big to be qualified as a *task*; it is not an *assignment*, nor a *business*, Tim Berners-Lee does not "manage" it, etc.

However, these terms would be more appropriate for other *currentProject* entities, e.g.:

> *Larry Page is responsible for the Alphabet Inc. Company (adventure | affair | business | creation | task | venture)*

Other FOAF classes such as *Group* and *Organization* might be relevant for describing what the World Wide Web or the Alphabet Inc. Company are: having the information that the World Wide Web is both a *project* and an *organization* will allow the linguistic module to produce much better sentences.

8.3 What does the *person* do exactly?

A similar problem concerns the *person* entity: when a project is described in FOAF as someone's current project, it is not clear what this person does, exactly: Is Tim Berners-Lee the *originator*, or the *creator*, or the *inventor* of the Web? Is Steve Jobs the *designer*, or the *mastermind*, or the *leader* of the iPhone project? Is Larry Page the *founder*, or the *originator*, or the *father* of the Alphabet Inc. Company? Is *Jürgen E. Schrempp* the *artisan*, or the *architect*, or the *facilitator* of the Mercedes-Chrysler merger? Even though both the *person* and the *project* entities are well defined, at this point we do not have the capability to select which exact terms can be used naturally: therefore, at this point, the linguistic prototype produces a large number of not-so-natural phrases such as "the World Wide Web task" or "the iPhone affair".

8.4 How current is a currentProject?

When a project is described in FOAF as a *currentProject*, it is not clear whether it is possible or not to replace the prototypic adverb *currently* with expressions such as: *for the moment, right now, these days*, etc., and if tenses other than present or present progressive (such as present perfect or future) are adequate or not:

- [+PresentPerfect]:
  Tim Berners-Lee has worked on the World Wide Web project (OK)
- [+PresentPerfect+AspCont+Intens1]:
  Tim Berners-Lee has been working on the World Wide Web project for a long time (OK)
- [+Future]:
  Tim Berners-Lee will work on the World Wide Web project (not OK)
- [+Future+AspCont+Intens1]:
  Tim Berners-Lee will continue to work on the World Wide Web project for a long time (not OK)

It will be necessary to explore the FOAF ontology to check if the *currentProject* is also listed as a *pastProject*; if so, the generator can send the operators +AspCont and +PresentPerfect to the linguistic module. The more information the generator has access to, the more it will be able to generate sentences produced by the linguistic module. As of now, unfortunately, we need to restrict the generation capability of the linguistic module drastically: the system is far from producing most of the English sentences that occur on the Web, such as the following ones:

> *Under Jobs' exacting leadership, Apple pioneered many things with the iPhone. Tim Berners-Lee is the director of the World Wide Web Consortium. Larry Page: I am really excited to be running Alphabet as CEO with help from my capable partner, Sergey, as President.*

However, grammars developed with NooJ are meant to be used not only for the generation, but also for the parsing of any text, including the previous sentences: the fact that a large number of sentences generated by the linguistic module are not yet useable by the generator is a consequence of the extreme simplicity of the FOAF ontology, rather than of a shortcoming of linguistics.

9 Conclusion

It is possible to construct a system capable of translating RDF statements into a rich set of English sentences. As a generator taps into the power of expression of the English language, it needs to control it: this can be performed via the use of linguistic operators.

Some operators, such as +Focus0 or +Pro1, are "information neutral", in the sense that they do not produce English sentences that might betray the information of the original RDF statement: they are

typically used for rhetorical purposes, to make the resulting text more natural.

Other linguistic operators, such as +Neg, +Future or +AspCont, are more "dangerous" to use, but should be easy to control, for instance by exploring the FOAF ontology to obtain missing information, such as the person's gender or the project's initial date. Exploring the Semantic Web to get more and more relevant information will be crucial in any case, as a system needs to access enough information about projects (dates, duration, organization involved, type of business, etc.) to state something "interesting".

However, the information stored in ontologies such as FOAF will never be as rich as necessary for an automatic generator to be able to produce all the English sentences that might express it. One solution for fixing the "vagueness" of the Semantic Web would be to enrich ontologies so that they contain information as precise as what the English language can express; in practice, this would require us to add to generic properties such as *currentProject* properties such as projectType, involvementType, projectOrganizationType, durationScale, involvementType, etc. to pinpoint what exact term is relevant for the project, what exact type of function and involvement the person has in the project (author a book, build a company, merge two companies, head an organization, design a product, oversight a business deal, chair a conference, etc.).

Reciprocally, producing RDF statements by parsing even complex English sentences has been proven to be feasible.[14] It seems to me that it would be therefore more sensible to develop linguistic resources to formalize more and more detailed information from the texts that already exist on the Web, rather than to store a simplified and redundant version of the information already available in English form on the Web in ontologies, and then try afterwards to compute back its equivalent English sentences.

References

Tim Berners-Lee, James Hendler and Ora Lassila. 2001. The Semantic Web. In *Scientific American*, pp 29-37.

Annibale Elia, Simonetta Vietri, Alberto Postiglione, Mario Monteleone and Federica Marano. 2010. Data Mining Modular Software System. In: SWWS2010 - *Proceedings of the 2010 International Conference on Semantic Web & Web Services*. Las Vegas, Nevada, USA 12-15 luglio 2010 CSREA Press, pp 127-133.

FOAF Vocabulary Specification 0.98. 2010. Namespace Document 9 August 2010. Marco Polo Eds. Available at:`http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.476.8247&rep=rep1&type=pdf`

RDF 1.1 Turtle – Terse RDF Triple LanguageTurtle. World Wide Web Consortium (W3C). 2014. Available at: `https://www.w3.org/TR/turtle`

Laurence Danlos. 1987. *The linguistic basis of Text Generation*. Studies in Natural Language Processing. Cambridge University Press Eds.

Maria Pia di Buono. 2017. Endpoint for Semantic Knowledge. In *Automatic Processing of Natural-Language Electronic Texts with NooJ*. Selected Papers from the 10th International NooJ Conference. (eds. Barone, Monteleone, Silberztein) CCIS Springer: Communication in Computer and Information Science #667, pp 223-233.

Héla Fehri, Kais Haddar and Abdelmajid Ben Hamadou. 2010. Automatic Recognition and Semantic Analysis of Arabic Named Entities. In *Applications of Finite-State Language Processing: Selected Papers from the NooJ 2008 International Conference* (Budapest, Hungaria) (eds. K. Judit, M. Silberztein, T. Varadi). Cambridge Scholars Publishing, Newcastle., UK, pp 101-113.

Claire Gardent and Perez-Beltrachini. 2017. A Statistical, Grammar-Based Approach to Mico-Planning. In *Computational Linguistics* 43:1. The MIT Press Eds, pp 1-30.

Kristina Kocijan and Marko Požega. 2015. Building Family Trees With NooJ. In *Formalising Natural Languages with NooJ 2014: Selected papers from the NooJ 2014 International Conference*, (eds. J. Monti, M. Silberztein, M. Monteleone, M. P. di Buono), Newcastle upon Tyne: Cambridge Scholars Publishing, pp 198-210.

Elena Lloret Pastor. 2011. *Text Summarization based on Human Language Technologies and its Application*. Tesis Doctorales. Universitad de Alicante Eds.

Max Silberztein. 2016a. *Formalizing Natural Languages: the NooJ approach*. ISTE-Wiley E

Max Silberztein. 2016b. *Joe loves Lea:* Transformational Analysis of Direct Transitive Sentences in *Automatic Processing of Natural-Language Electronic Texts with NooJ. NooJ 2015* (eds. T. Okrut, Y. Hetsevich, M. Silberztein, H. Stanislavenka). Communications in Computer and Information Science, vol 607. Springer, Cham, pp 55-65.

[14] NooJ has been used since 2002 to parse texts in a dozen languages and produce Semantic annotations in XML or Prolog notation, cf. for instance Fehri et al. (2010), Elia et al. (2010), Kocijan and Požega (2015), di Buono (2017).