Extracting Linguistic Resources from the Web for Concept-to-Text Generation

Gerasimos Lampouras lampouras06@aueb.gr
Department of Informatics, Athens University of Economics and Business, Greece

Ion Androutsopoulos ion@aueb.gr
Department of Informatics, Athens University of Economics and Business, Greece

Abstract

Many concept-to-text generation systems require domain-specific linguistic resources to produce high quality texts, but manually constructing these resources can be tedious and costly. Focusing on NaturalOWL, a publicly available state of the art natural language generator for OWL ontologies, we propose methods to extract from the Web sentence plans and natural language names, two of the most important types of domain-specific linguistic resources used by the generator. Experiments show that texts generated using linguistic resources extracted by our methods in a semi-automatic manner, with minimal human involvement, are perceived as being almost as good as texts generated using manually authored linguistic resources, and much better than texts produced by using linguistic resources extracted from the relation and entity identifiers of the ontology.

1. Introduction

The Semantic Web (Berners-Lee, Hendler, & Lassila, 2001; Shadbolt, Berners-Lee, & Hall, 2006) and the growing popularity of Linked Data (data published using Semantic Web technologies) have renewed interest in concept-to-text generation (Reiter & Dale, 2000), especially text generation from ontologies (Bontcheva, 2005; Mellish & Sun, 2006; Galanis & Androutsopoulos, 2007; Mellish & Pan, 2008; Schwitter, Kaljurand, Cregan, Dolbear, & Hart, 2008; Schwitter, 2010; Liang, Stevens, Scott, & Rector, 2011; Williams, Third, & Power, 2011; Androutsopoulos, Lampouras, & Galanis, 2013). An ontology provides a conceptualization of a knowledge domain (e.g., wines, consumer electronics) by defining the classes and subclasses of the individuals (entities) in the domain, the possible relations between them etc. The current standard to specify Semantic Web ontologies is OWL (Horrocks, Patel-Schneider, & van Harmelen, 2003), which is based on description logics (Baader, Calvanese, McGuinness, Nardi, & Patel-Schneider, 2002), RDF, and RDF SCHEMA (Antoniou & van Harmelen, 2008). OWL2 is the latest version of OWL (Grau, Horrocks, Motik, Parsia, Patel-Schneider, & Sattler, 2008).[1] Given an OWL ontology for a knowledge domain, one can publish on the Web machine-readable data pertaining to that domain (e.g., catalogues of products, their features etc.), with the data having formally defined semantics based on the ontology.

Several equivalent OWL syntaxes have been developed, but people unfamiliar with formal knowledge representation have difficulties understanding them (Rector, Drummond, Horridge, Rogers, Knublauch, Stevens, Wang, & Wroe, 2004). For example, the following statement defines the class of St. Emilion wines, using the functional-style syntax of OWL, one of the easiest to understand.[2]

```
SubClassOf(:StEmilion
  ObjectIntersectionOf(:Bordeaux
```

1. Most Linked Data currently use only RDF and RDF SCHEMA, but OWL is in effect a superset of RDF SCHEMA and, hence, methods to produce texts from OWL also apply to Linked Data. Consult also `http://linkeddata.org/`.
2. Consult `http://www.w3.org/TR/owl2-primer/` for an introduction to the functional-style syntax of OWL.

```
ObjectHasValue(:locatedIn :stEmilionRegion)
ObjectHasValue(:hasColor :red)
ObjectHasValue(:hasFlavor :strong)
ObjectHasValue(:madeFrom :cabernetSauvignonGrape)
ObjectMaxCardinality(1 :madeFrom)))
```

To make ontologies easier to understand, several *ontology verbalizers* have been developed (Cregan, Schwitter, & Meyer, 2007; Kaljurand & Fuchs, 2007; Schwitter et al., 2008; Halaschek-Wiener, Golbeck, Parsia, Kolovski, & Hendler, 2008; Schutte, 2009; Power & Third, 2010; Power, 2010; Schwitter, 2010; Stevens, Malone, Williams, Power, & Third, 2011; Liang et al., 2011). Although verbalizers can be viewed as performing a kind of light natural language generation (NLG), they usually translate the axioms (in our case, OWL statements) of the ontology one by one to controlled, often not entirely fluent English statements, typically without considering the coherence of the resulting texts. By contrast, more elaborate NLG systems (Bontcheva, 2005; Androutsopoulos, Oberlander, & Karkaletsis, 2007; Androutsopoulos et al., 2013) can produce more fluent and coherent multi-sentence texts, but they need domain-specific linguistic resources. For example, NaturalOWL (Androutsopoulos et al., 2013), a publicly available NLG system for OWL ontologies, produces the following description of St. Emilion wines from the OWL statement above. It needs, however: a *sentence plan* for each relation (e.g., `:locatedIn`) of the ontology, i.e., a linguistically annotated template showing how to express the relation; a *natural language name* for each class and individual, i.e., a linguistically annotated noun phrase to be used as the name of the class or individual; a text plan specifying the order in which relations should be expressed etc. Similar domain-specific linguistic resources are used in most concept-to-text systems (Reiter & Dale, 2000). Manually constructing resources of this kind, however, can be tedious and costly.

> St. Emilion is a kind of red, strong Bordeaux from the St. Emilion region. It is made from exactly one grape variety: Cabernet Sauvignon grapes.

Instead of requiring domain-specific linguistic resources, simpler verbalizers use the OWL identifiers of classes and individuals (e.g., `:cabernetSauvignonGrape`) typically split into tokens as their natural language names, they express relations using phrases obtained by tokenizing the OWL identifiers of the relations (e.g., `:hasColor`), they order the resulting sentences following the ordering of the corresponding OWL statements etc. Without domain-specific linguistic resources, NaturalOWL behaves like a simple verbalizer, producing the following lower quality text from the OWL statement above. A further limitation of using tokenized OWL identifiers is that non-English texts cannot be generated, since OWL identifiers are usually English-like.

> St Emilion is Bordeaux. St Emilion located in St Emilion Region. St Emilion has color Red. St Emilion has flavor Strong. St Emilion made from grape exactly 1: Cabernet Sauvignon Grape.

Previous experiments (Androutsopoulos et al., 2013) indicate that the texts that NaturalOWL generates with domain-specific linguistic resources are perceived as significantly better than (i) those of SWAT, one of the best available OWL verbalizers (Stevens et al., 2011; Williams et al., 2011), and (ii) those of NaturalOWL without domain-specific linguistic resources, with little or no difference between (i) and (ii). The largest difference in the perceived quality of the texts was reported to be due to the sentence plans, natural language names, and (to a lesser extent) text plans.

In this paper, we present methods to automatically or semi-automatically extract from the Web the natural language names and sentence plans required by NaturalOWL for a given ontology. We do not examine how other types of domain-specific linguistic resources (e.g., text plans) can be

generated, leaving them for future work. We base our work on NaturalOWL, because it appears to be the only open-source NLG system for OWL that implements all the processing stages of a typical NLG pipeline (Reiter & Dale, 2000), it supports OWL2, it is extensively documented, and has been tested with several ontologies. The processing stages and linguistic resources of NaturalOWL, however, are typical of NLG systems (Mellish, Scott, Cahill, Paiva, Evans, & Reape, 2006). Hence, we believe that our work is also applicable, at least in principle, to other NLG systems. Our methods may also be useful in simpler verbalizers, where the main concern seems to be to avoid manually authoring domain-specific linguistic resources. Experiments show that texts generated using linguistic resources extracted by our methods with minimal human involvement are perceived as being almost as good as texts generated using manually authored linguistic resources, and much better than texts produced by using tokenized OWL identifiers.

Section 2 below provides background information about NaturalOWL, especially its natural language names and sentence plans. Sections 3 and 4 then describe our methods to extract natural language names and sentence plans, respectively, from the Web. Section 5 presents our experimental results. Section 6 discusses related work. Section 7 concludes and suggests future work.

2. Background information about NaturalOWL

Given an OWL ontology and a particular target individual or class to describe, NaturalOWL first scans the ontology to select statements relevant to the target. It then converts each relevant statement into (possibly multiple) *message triples* of the form $\langle S, R, O \rangle$, where S is an individual or class, O is another individual, class, or datatype value, and R is a relation (property) that connects S to O. For example, the `ObjectHasValue(:madeFrom :cabernetSauvignonGrape)` part of the OWL statement above is converted to the message triple <`:StEmilion`, `:madeFrom`, `:cabernetSauvignonGrape`>. Message triples are similar to RDF triples, but they are easier to express as sentences. Unlike RDF triples, the relations (R) of the message triples may include *relation modifiers*. For example, the `ObjectMaxCardinality(1 :madeFrom)` part of the OWL statement above is turned into the message triple <`:StEmilion`, `maxCardinality(:madeFrom)`, 1>, where `maxCardinality` is a relation modifier. In this paper, we consider only sentence plans for message triples without relation modifiers, because NaturalOWL already automatically constructs sentence plans for triples with relation modifiers from sentence plans for triples without them.

Having produced the message triples, NaturalOWL consults a user model to select the most interesting ones that have not been expressed already, and orders the selected triples according to manually authored text plans. Later processing stages convert each message triple to an abstract sentence representation, aggregate sentences to produce longer ones, and produce appropriate referring expressions (e.g., pronouns). The latter three stages require a sentence plan for each relation (R), while the last stage also requires natural language names for each individual or class (S or O).

2.1 The natural language names of NaturalOWL

In NaturalOWL, a natural language (NL) name is a sequence of slots. The contents of the slots are concatenated to produce a noun phrase to be used as the name of a class or individual. Each slot is accompanied by annotations specifying how to fill it in; the annotations may also provide linguistic information about the contents of the slot. For example, we may specify that the English NL name of the class `:TraditionalWinePiemonte` is the following.[3]

3. The NL names and sentence plans of NaturalOWL are actually represented in OWL, as instances of an ontology that describes the domain-dependent linguistic resources of the system.

$$[\,]^{1}_{article,\,indef,\,agr=3}\ [\text{traditional}]^{2}_{adj}\ [\text{wine}]^{3}_{headnoun,\,sing,\,neut}\ [\text{from}]^{4}_{prep}\ [\,]^{5}_{article,\,def}\ [\text{Piemonte}]^{6}_{noun,\,sing,\,neut}\ [\text{region}]^{7}_{noun,\,sing,\,neut}$$

The first slot is to be filled in with an indefinite article, whose number should agree with the third slot. The second slot is to be filled in with the adjective 'traditional'. The third slot with the neuter noun 'wine', which will also be the head (central) noun of the noun phrase, in singular number, and similarly for the other slots. NaturalOWL makes no distinctions between common and proper nouns, but it can be instructed to capitalize particular nouns (e.g., 'Piemonte'). In the case of the message triple <:wine32, instanceOf, :TraditionalWinePiemonte>, the NL name above would allow a sentence like "This is *a traditional wine from the Piemonte region*" to be produced.

The slot annotations allow NaturalOWL to automatically adjust the NL names. For example, the system also generates comparisons to previously encountered individuals or classes, as in "Unlike the previous products that you have seen, which were all *traditional wines from the Piemonte region*, this is a French wine". In this particular example, the head noun ('wine') had to be turned into plural. Due to number agreement, its article also had to be turned into plural; in English, the plural indefinite article is void, hence the article of the head noun was omitted.

As a further example, we may specify that the NL name of the class FamousWine is the following.

$$[\,]^{1}_{article,\,indef,\,agr=3}\ [\text{famous}]^{2}_{adj}\ [\text{wine}]^{3}_{headnoun,\,sing,\,neut}$$

If <:wine32, instanceOf, :TraditionalWinePiemonte> and <:wine32, instanceOf, :FamousWine> were both to be expressed, NaturalOWL would then produce the single, aggregated sentence "This is a famous traditional wine from the Piemonte region", instead of two separate sentences "This is a traditional wine from the Piemonte region" and "This is a famous wine". The annotations of the slots, which indicate for example which words are adjectives and head nouns, are used by the sentence aggregation component of NaturalOWL to appropriately combine the two sentences. The referring expression generation component also uses the slot annotations to identify the gender of the head noun, when a pronoun has to be generated (e.g., "it" when the head noun is neuter).

We can now define more precisely NL names. A NL name is a sequence of one or more slots. Each slot is accompanied by annotations requiring it to be filled in with exactly one of the following:[4]

(i) *An article*, definite or indefinite, possibly to agree with another slot filled in by a noun.

(ii) *A noun flagged as the head.* The number of the head noun must also be specified.

(iii) *An adjective flagged as the head.* For example, the NL name name of the individual :red may consist of a single slot, to be filled in with the adjective 'red'; in this case, the adjective is the head of the NL name. The number and gender of the head adjective must also be specified.

(iv) *Any other noun or adjective*, (v) *a preposition*, or (vi) *any fixed string.*

Exactly one head (noun or adjective) must be specified per NL name. For nouns and adjectives, the NL name may require a particular inflectional form to be used (e.g., in a particular number, case, or gender), or it may require an inflectional form that agrees with another noun or adjective slot.[5]

When providing NL names, an individual or class can also be declared to be *anonymous*, indicating that NaturalOWL should avoid referring to it by name. For example, in a museum ontology, there may be a coin whose OWL identifier is :exhibit49. We may not wish to provide an NL name for this individual (it may not have an English name); and we may want NaturalOWL to avoid referring to the coin by tokenizing its identifier ("exhibit 49"). By declaring the coin as anonymous, NaturalOWL would use only the NL name of its class (e.g., "this coin"), simply "this", or a pronoun.

4. NaturalOWL also supports Greek. The possible annotations for Greek NL names (and sentence plans, see below) are slightly different, but in this paper we consider only English NL names (and sentence plans).

5. We use SIMPLENLG (Gatt & Reiter, 2009) to generate the inflectional forms of nouns, adjectives, and verbs.

2.2 The sentence plans of NaturalOWL

In NaturalOWL, a sentence plan for a relation R specifies how to construct a sentence to express any message triple of the form $\langle S, R, O \rangle$. Like NL names, sentence plans are sequences of slots with annotations specifying how to fill the slots in. The contents of the slots are concatenated to produce the sentence. For example, the following is a sentence plan for the relation `:madeFrom`.

$$[ref(S)]^1_{nom}\ [\text{make}]^2_{verb,\ passive,\ present,\ agr=1,\ polarity=+}\ [\text{from}]^3_{prep}\ [ref(O)]^4_{acc}$$

Given the message triple <`:StEmilion`, `:madeFrom`, `:cabernetSauvignonGrape`>, the sentence plan would lead to sentences like "St. Emilion is made from Cabernet Sauvignon grapes", or "It is made from Cabernet Sauvignon grapes", assuming that appropriate NL names have been provided for `:StEmilion` and `:cabernetSauvignonGrape`. Similarly, given <`:Wine`, `:madeFrom`, `:Grape`>, the sentence plan above would lead to sentences like "Wines are made from grapes" or "They are made from grapes", assuming again appropriate NL names. As another example, the following sentence plan can be used with the relations `:hasColor` and `:hasFlavor`.

$$[ref(S)]^1_{nom}\ [\text{be}]^2_{verb,\ active,\ present,\ agr=1,\ polarity=+}\ [ref(O)]^3_{nom}$$

Given the message triples <`:StEmilion`, `:hasColor`, `:red`> and <`:StEmilion`, `:hasFlavor`, `:strong`>, it would produce the sentences "St. Emilion is red" and "St. Emilion is strong", respectively.

The first sentence plan above, for `:madeFrom`, has four slots. The first slot is to be filled in with an automatically generated referring expression (e.g., pronoun or name) for S, in nominative case. The verb of the second slot is to be realized in passive voice, present tense, and positive polarity (as opposed to expressing negation) and should agree (in number and person) with the referring expression of the first slot ($agr = 1$). The third slot is filled in with the preposition 'from', and the fourth slot with an automatically generated referring expression for O, in accusative case.

NaturalOWL has built-in sentence plans for domain-independent relations (e.g., `isA`, `instanceOf`). For example, <`:StEmilion`, `isA`, `:Bordeaux`> is expressed as "St. Emilion is a kind of Bordeaux" using the following built-in sentence plan; the last slot requires the NL name of O without article.

$$[ref(S)]^1_{nom}\ [\text{be}]^2_{verb,\ active,\ present,\ agr=1,\ polarity=+}\ [\text{"a kind of"}]^3_{string}\ [name(O)]^4_{noarticle,nom}$$

Notice that the sentence plans of NaturalOWL are not simply slotted string templates (e.g., "X is made from Y"). Their linguistic annotations (e.g., POS tags, agreement, voice, tense, cases) along with the annotations of the NL names allow NaturalOWL to produce more natural sentences (e.g., turn the verb into plural when the subject is also plural), produce appropriate referring expressions (e.g., pronouns in the correct cases and genders), and aggregate shorter sentences into longer ones. For example, the linguistic annotations of the NL names and sentence plans allow NaturalOWL to produce the aggregated sentence "St. Emilion is a kind of red Bordeaux made from Cabernet Sauvignon grapes" from the triples <`:StEmilion`, `isA`, `:Bordeaux`>, <`:StEmilion`, `:hasColor`, `:red`>, <`:StEmilion`, `:madeFrom`, `:cabernetSauvignonGrape`>, instead of three separate sentences.

We can now define more precisely sentence plans. A sentence plan is a sequence of slots. Each slot is accompanied by annotations requiring it to be filled in with exactly one of the following:

(i) *A referring expression for the S* (a.k.a. the *owner*) of the message triple in a particular case.
(ii) *A verb* in a particular polarity and form (e.g., tense), possibly to agree with another slot.
(iii) *A noun or adjective* in a particular form, possibly to agree with another slot.
(iv) *A preposition*, or (v) *a fixed string.*
(vi) *A referring expression for the O* (a.k.a. the *filler*) of the message triple.

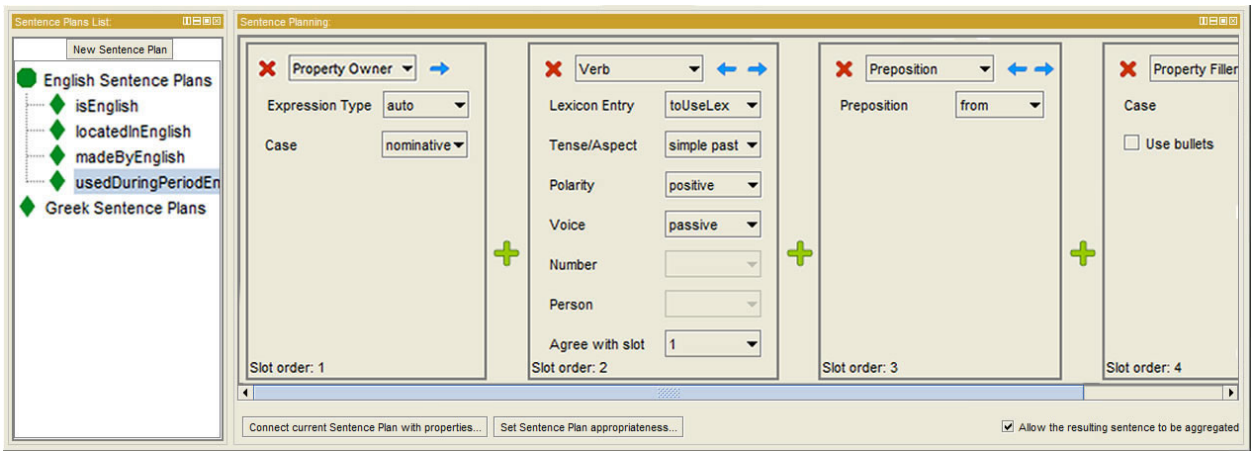


Figure 1: A manually authored sentence plan in the Protégé plug-in of NaturalOWL.

More details about the NL names and sentence plans of NaturalOWL and their roles in sentence aggregation, referring expressions etc. can be found elsewhere (Androutsopoulos et al., 2013). Both sentence plans and NL names were so far authored manually, using a Protégé plug-in (Fig. 1).[6]

3. Our method to extract natural language names from the Web

Given a target class or individual t that we want to produce an NL name for, we first extract from the Web noun phrases that are similar to the OWL identifier of t. The noun phrases are ranked by aligning their words to the tokens of the identifier. The top-ranked noun phrases are then enhanced with linguistic annotations (e.g., POS tags, agreement, number), missing articles etc., turning them into NL names. We aim to identify the best few (up to 5) candidate NL names for t. In a fully automatic scenario, the candidate NL name that the method considers best for t is then used. In a semi-automatic scenario, the few top (according to the method) NL names of t are shown to a human author, who picks the best one; this is much easier than manually authoring NL names.

3.1 Extracting noun phrases from the Web

We first collect the OWL statements of the ontology that describe t, the individual or class we want to produce an NL name for, and turn them into message triples $\langle S = t, R, O \rangle$, as when generating texts. For example, for the class t = `:KalinCellarsSemillon` of the Wine Ontology, one of the ontologies of our experiments, three of the resulting message triples are:

<:KalinCellarsSemillon, isA, :Semillon>
<:KalinCellarsSemillon, :hasMaker, :KalinCellars>
<:KalinCellarsSemillon, :hasFlavor, :Strong>

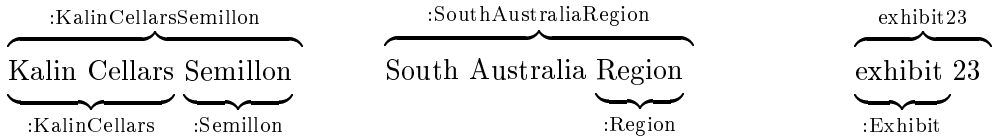
For each collected message triple $\langle S = t, R, O \rangle$, we then produce $tokName(S)$ and $tokName(O)$, where $tokName(X)$ is the tokenized identifier of X.[7] From the three triples above, we obtain:

$tokName$(`:KalinCellarsSemillon`) = "Kalin Cellars Semillon" , $tokName$(`:Semillon`) = "Semillon"
$tokName$(`:KalinCellars`) = "Kalin Cellars", $tokName$(`:Strong`) = "Strong"

6. Consult `http://protege.stanford.edu/` and `http://nlp.cs.aueb.gr/software.html`.
7. Most OWL ontologies use identifiers written in CamelCase, or identifiers that can be easily broken into tokens at underscores, hyphens etc. If the ontology provides an `rdfs:label` for X, we use its tokens as $tokName(X)$.

3.1.1 Shortening the tokenized identifiers

Subsequently, we attempt to shorten *tokName*(t), i.e., the tokenized identifier of the individual or class we wish to produce an NL name for, by removing any part (token sequence) of *tokName*(t) that is identical to the tokenized identifier of the O of any triple $\langle S = t, R, O \rangle$ that we collected for t. If the shortened tokenized identifier of t is the empty string or contains only numbers, t is marked as anonymous (Section 2.1). In our example, where t = `:KalinCellarsSemillon`, the tokenized identifier of t is initially *tokName*(t) = "Kalin Cellars Semillon". We remove the part "Semillon", because of the triple <`:KalinCellarsSemillon`, `:isA`, `:Semillon`> and the fact that *tokName*(`:Semillon`) = "Semillon", as illustrated below. We also remove the remaining part "Kalin Cellars", because of <`:KalinCellarsSemillon`, `:hasMaker`, `:KalinCellars`> and the fact that *tokName*(`:KalinCellars`) = "Kalin Cellars". Hence, `:KalinCellarsSemillon` is marked as anonymous.



Anonymizing `:KalinCellarsSemillon` causes NaturalOWL to produce texts like (a) below when asked to describe `:KalinCellarsSemillon`, rather than (b), which repeats "Semillon" and "Kalin Cellars":

(a) This is a strong, dry Semillon. It has a full body. It is made by Kalin Cellars.
(b) Kalin Cellars Semillon is a strong, dry Semillon. It has a full body. It is made by Kalin Cellars.

Similarly, if t = `:SouthAustraliaRegion` and we have collected the following message triple, the tokenized identifier of t would be shortened from "South Australia Region" to "South Australia". We use *altTokName* to denote the resulting shortened tokenized identifiers.[8]

<`:SouthAustraliaRegion`, `:isA`, `:Region`>
tokName(`:SouthAustraliaRegion`) = "South Australia Region", *tokName*(`:Region`) = "Region"
altTokName(`:SouthAustraliaRegion`) = "South Australia"

Also, if t = `:exhibit23` and we have collected the following triple, *altTokName*(`:exhibit23`) would end up containing only numbers ("23"). Hence, `:exhibit23` is marked as anonymous.

<`:exhibit23`, `:isA`, `:Exhibit`>
tokName(`:exhibit23`) = "exhibit 23", *tokName*(`:Exhibit`) = "exhibit"

3.1.2 Obtaining additional alternative tokenized identifiers

We then collect the tokenized identifiers of all the ancestor classes of t, also taking into account equivalent classes; for example, if t has an equivalent class t', we also collect the tokenized identifiers of the ancestor classes of t'. For t = `:KalinCellarsSemillon`, we collect the following tokenized identifiers, because `:Semillon`, `:SemillonOrSauvignonBlanc`, and `:Wine` are ancestors of t.[9]

tokName(`:Semillon`) = "Semillon", *tokName*(`:SemillonOrSauvignonBlanc`) = "Semillon Or Sauvignon Blanc", *tokName*(`:Wine`) = "Wine"

8. Strictly speaking, the values of *altTokName* should be shown as sets, since an individual or class can have multiple shortened tokenized identifiers (see below), but we show them as single values for simplicity.
9. We consider only classes with OWL identifiers when traversing the hierarchy, ignoring classes constructed using OWL operators (e.g., class intersection) that have not been assigned OWL identifiers. In ontologies with multiple inheritance, the same tokenized identifiers of ancestors of t may be obtained by following different paths in the class hierarchy, but we remove duplicate tokenized identifiers.

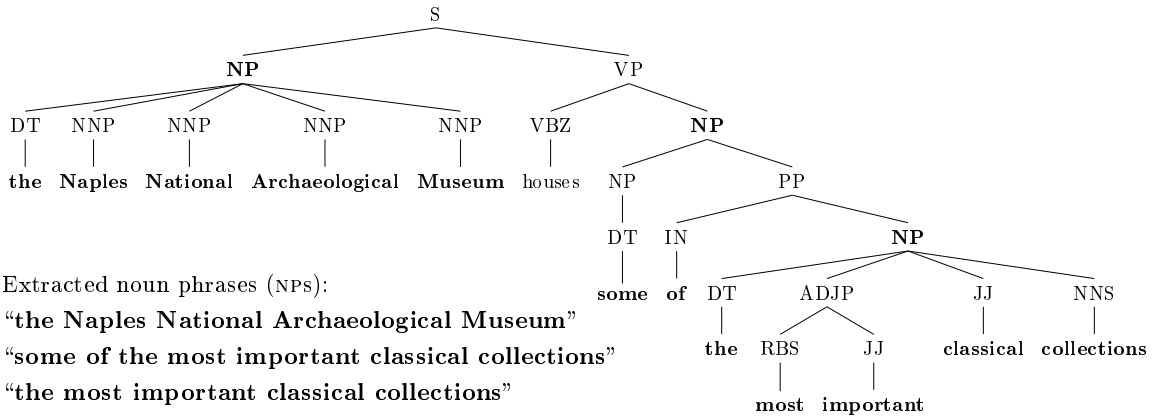


Figure 2: Parse tree of a retrieved sentence and its noun phrases.

If *tokName*(t) does not contain any of the collected tokenized identifiers of the ancestor classes of t, we create additional alternative tokenized identifiers for t, also denoted *altTokName*(t), by appending to *tokName*(t) the collected tokenized identifiers of the ancestor classes of t. For example, if t = `:red` and `:Color` is the parent class of t (<`:red`, `isA`, `:Color`>), we also obtain "red color":

tokName(`:red`) = "red", *tokName*(`:Color`) = "color", *altTokName*(`:red`) = "red color"

By contrast, if t = `:KalinCellarsSemillon`, no *altTokName*(t) is produced from the ancestors of t, because *tokName*(t) = "Kalin Cellars Semillon" contains *tokName*(`:Semillon`) = "Semillon", and `:Semillon` is an ancestor of `:KalinCellarsSemillon`.

Furthermore, we create an additional *altTokName*(t) by removing all numbers from *tokName*(t); for example, from *tokName*(t) = "Semillon 2006" we obtain *altTokName*(t) ="Semillon". Lastly, if *tokName*(t) contains brackets, we create an *altTokName*(t) for each part outside and inside the brackets; for example, from "gerbil (dessert rat)" we get "gerbil" and "dessert rat".

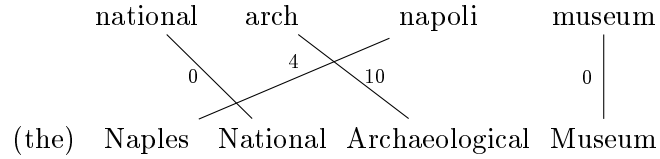
3.1.3 Retrieving web pages, extracting and ranking noun phrases

Subsequently, we formulate a Boolean Web search query for *tokName*(t) (e.g., "South" `AND` "Australia" `AND` "Region") and each *altTokName*(t) (e.g., "South" `AND` "Australia"); recall that t is the individual or class we wish to produce an NL name for.[10] We convert the retrieved pages of all the queries to plain text documents and parse every sentence of the text, if any stemmed word of the sentence is the same as any stemmed word of any *tokName*(t) or *altTokName*(t).[11] We then extract the noun phrases (NPs) from every parsed sentence. For example, from the sentence "the Naples National Archaeological Museum houses some of the most important classical collections" we extract the NPs "the Naples National Archaeological Museum", "some of the most important classical collections", and "the most important classical collections" (Fig. 2).

For each extracted NP, we compute its similarity to *tokName*(t) and each *altTokName*(t). Let *np* be an extracted NP and let *name* be *tokName*(t) or an *altTokName*(t). To compute the similarity between *np* and *name*, we first compute the character-based Levenshtein distance between each token of *np* and each token of *name*; we ignore upper/lower case differences, articles, and connectives (e.g. "or"), which are often omitted from OWL identifiers. In the following example, *np* = "the Naples National Archaeological Museum" (but "the" is ignored) and *name* = "national arch napoli

10. If the search engine suggests corrections to any query, we use the corrected form of the query as well.
11. We use JSOUP (`http://jsoup.org/`) to obtain the plain texts from the Web pages, Porter's stemmer (`http://tartarus.org/martin/PorterStemmer/`), and the Stanford parser (`http://nlp.stanford.edu/`).

museum"; this *name* is an *altTokName*(t) produced by appending to *tokName*(t) the tokenized identifier of the parent class (`:Museum`) of t (Section 3.1.2). The Levenshtein distance between "national" and "National" is 0 (upper/lower case differences are ignored). The distance between "napoli" and "Naples" is 4; a character deletion or insertion costs 1, a replacement costs 2.



We then form pairs of aligned tokens $\langle t_{name}, t_{np} \rangle$, where t_{name}, t_{np} are tokens from *name*, *np*, respectively, such that each token of *name* is aligned to at most one token of *np* and vice versa, and any other, not formed pair $\langle t'_{name}, t'_{np} \rangle$ would have a Levenshtein distance (between t'_{name}, t'_{np}) larger or equal to the minimum Levensthein distance of the formed pairs.[12] In our example, the pairs of alinged tokens are <"national", "National">, <"arch", "Archaeological">, <"napoli", "Naples">, <"museum", "Museum">.

The similarity between *np* and *name* is then computed as follows, where A is the set of aligned token pairs, *Levenshtein*(a) is the Levenshtein distance (normalized to $[0, 1]$) between the t_{name} and t_{np} of pair a, and $|np|$, $|name|$ are the lengths (in tokens) of *np* and *name*, respectively.

$$similarity(np, name) = \frac{\sum_{a \in A}(1 - Levenshtein(a))}{max\{|np|, |name|\}} \quad (1)$$

For each extracted NP of t, we compute its similarity to every possible *name*, i.e., *tokName*(t) or *altTokName*(t), as discussed above, and we assign to the NP a score equal to the largest of these similarities. Finally, we rank the extracted NPs of t by decreasing score. If two NPs have the same score, we rank higher the NP with the fewest crossed edges in its best alignment with a *name*. If two NPs still cannot be distinguished, we rank them by decreasing frequency in the parsed sentences of t; and if their frequencies are equal, we rank them randomly.

3.2 Turning the extracted noun phrases into natural language names

The extracted NPs are not yet NL names, because they lack the linguistic annotations that NaturalOWL requires (e.g., POS tags, agreement, number); they may also lack appropriate articles. To convert an NP to an NL name, we first obtain the POS tags of its words from the parse tree of the sentence the NP was extracted from.[13] For example, the NP "the Red Wine" becomes:

$$\text{the}_{\text{POS}=DT}\ \text{Red}_{\text{POS}=JJ}\ \text{Wine}_{\text{POS}=NN}$$

For every noun, adjective, article, preposition, we create a corresponding slot in the NL name; all the other words of the NP become slots containing the words as fixed strings (Section 2.1). For nouns and adjectives, the base form is used in the slot (e.g., "wine" istead of "wines"), but slot annotations indicate the particular inflectional form that was used in the NP; e.g., the NN POS tag shows that "wine" is singular. A named-entity recognizer (NER) and an on-line dictionary are

12. In our experiments, we actually formed the pairs greedily, by computing the distances of all the possible pairs and iteratively selecting the pair with the smallest distance whose elements did not occur in any other already selected pair. A non-greedy search (e.g., using Integer Linear Programming) makes little difference in practice.

13. If an NP has been extracted from multiple sentences and their parse trees provide different POS tag assignments to the words of the NP, we create a separate NL name for each POS tag assignment.

employed to detect nouns that refer to persons and locations.[14] The genders of these nouns are determined using the on-line dictionary, when possible, or defaults otherwise (e.g., the default for person nouns is a 'person' pseudo-gender, which leads to "he/she" or "they" when generating a pronoun). Nouns not referring to persons and locations are marked as neuter. Since the NPs are extracted from Web pages, there is a risk of wrong capitalization (e.g., "the RED wine"). For each word of the NL name, we pick the capitalization that is most frequent in the retrieved texts of the individual or class we generate the NL name for. Hence, the NP "the Red Wines" becomes:

$$[]^1_{article,\,def}\ [\text{red}]^2_{adj}\ [\text{wine}]^3_{noun,\,sing,\,neut}$$

which requires a definite article, followed by the adjective "red", and the neuter "wine" in singular.

A dependency parser is then used to identify the head of each NL name (Section 2.1) and to obtain agreement information.[15] Adjectives are required to agree with the nouns they modify, and the same applies to articles and nouns. At this stage, the NP "the Red Wines" will have become:

$$[]^1_{article,\,def,\,agr=3}\ [\text{red}]^2_{adj}\ [\text{wine}]^3_{headnoun,\,sing,\,neut}$$

We then consider the main article (or, more generally, determiner) of the NL name, i.e., the article that agrees with the head (e.g., "a" in "a traditional wine from the Piemonte Region"). Although the NL name may already include a main article, it is not necessarily an appropriate one. For example, it would be inappropriate to use a definite article in "*The* red wine is a kind of wine with red color", when describing the class of red wines. We modify the NL name to use an indefinite article if the NL name refers to a class, and a definite article if it refers to an individual (e.g., "the South Australia region").[16] The article is omitted if the head is an adjective (e.g., "strong"), or in plural (e.g., "Semillon grapes"), or if the entire NL name (excluding the article, if present) is a proper name (e.g., "South Australia") or a mass noun phrase without article (e.g., "gold"). Before inserting or modifying the main article, we also remove any demonstratives (e.g., "*this* statue") or other non-article determiners (e.g., "some", "all") from the beginning of the NL name . In our example, the NL name is to be used to refer to the class `:RedWine`, so the final NL name is the following, which would lead to sentences like "*A* red wine is a kind of wine with red color".

$$[]^1_{article,\,indef,\,agr=3}\ [\text{red}]^2_{adj}\ [\text{wine}]^3_{headnoun,\,sing,\,neut}$$

Recall that NaturalOWL can automatically adjust NL names when generating texts (Section 2.1). For example, in a comparison like "Unlike *the previous red wines* that you have seen, this one is from France", it would use a definite article and it would turn the head noun of the NL name to plural, also adding the adjective "previous". The resulting NL names are finally ranked by the scores of the NPs they were obtained from (Section 3.1.3).

3.3 Inferring interest scores from natural language names

The reader may have already noticed that the sentence "A red wine is a kind of wine with red color" that we used above sounds redundant. Some message triples lead to sentences that sound redundant, because they report relations that are obvious (to humans) from the NL names of the individuals or classes. In our example, the sentence reports the following two message triples.

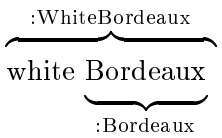
14. We use the Stanford NER (`http://nlp.stanford.edu/`), and Wiktionary (`http://en.wiktionary.org/`).
15. The Stanford parser also produces dependency trees. The parser is applied to the sentences the NPs were extracted from. If multiple parses are obtained for the NP an NL name is based on, we keep the most frequent parse.
16. This arrangement works well in the ontologies we have experimented with, but it may have to be modified for other ontologies, for example if kinds of entities are modelled as individuals, rather than classes.

<:RedWine, isA, :Wine>, <:RedWine, :hasColor, :Red>

Expressed separately, the two triples would lead to the sentences "A red wine is a kind of wine" and "A red wine has red color", but NaturalOWL aggregates them into a single sentence. The "red color" derives from an *altTokName* of :Red obtained by considering the parent class :Color of :Red (Section 3.1.2). It is obvious that a red wine is a wine with red color and, hence, the two triples above should not be expressed. Similarly, the following triple leads to the sentence "A white Bordeaux wine is a kind of Bordeaux", which again seems redundant.

<:WhiteBordeaux, isA, :Bordeaux>

NaturalOWL provides mechanisms to manually assign *interest scores* to message triples (Androutsopoulos et al., 2013). Assigning a zero interest score to a triple instructs NaturalOWL to avoid expressing it. Manually assigning interest scores, however, can be tedious. Hence, we aimed to automatically assign zero scores to triples like the ones above, which report relations that are obvious from the NL names. To identify triples of this kind, we follow a procedure similar to the one we use to identify individuals or classes that should be anonymous (Section 3.1.1). For each $\langle S, R, O\rangle$ triple that involves the individual or class S being described, we examine the NL names of S and O. If all the (lemmatized) words of the phrase produced by the NL name of O (e.g., "a Bordeaux"), excluding articles, appear in the phrase of the NL name of S (e.g., "a white Bordeaux"), we assign a zero interest score to $\langle S, R, O\rangle$.



4. Our method to automatically extract sentence plans from the Web

To produce a sentence plan for a relation, we first extract slotted string templates (e.g., "X is made from Y") from the Web using seeds (values of X, Y) from the ontology. We then enhance the templates by adding linguistic annotations (e.g., POS tags, agreement, voice, tense) and missing components (e.g., auxiliary verbs) turning the templates into candidate sentence plans. The candidate sentence plans are then scored by a Maximum Entropy classifier to identify the best few (again up to 5) candidate sentence plans for each relation.[17] In a fully automatic scenario, the sentence plan that the classifier considers best for each relation is used. In a semi-automatic scenario, the few top sentence plans of each relation are shown to a human author, who picks the best one.

4.1 Extracting templates from the Web

For each relation R that we want to generate a sentence plan for, our method first obtains the OWL statements of the ontology that involve the relation and turns them into message triples $\langle S, R, O\rangle$, as when generating texts. For example, if the relation is :madeFrom, two of the triples may be:

<:StEmilion, :madeFrom, :cabernetSauvignonGrape>, <:Semillon, :madeFrom, :SemillonGrape>

To these triples, we add more by replacing the S, O, or both of each originally obtained triple by their classes (if S or O are individuals), their parent classes, or their equivalent classes. For example, from <:StEmilion, :madeFrom, :cabernetSauvignonGrape> we also obtain the following three triples, because Wine is a parent class of StEmilion, and Grape is a parent class of :cabernetSauvignonGrape.

17. We use the Stanford classifier; consult http://nlp.stanford.edu/software/classifier.shtml.

<:Wine, :madeFrom, :cabernetSauvignonGrape>
<:StEmilion, :madeFrom, :Grape>, <:Wine, :madeFrom, :Grape>

We obtain the same additional triples from <:Semillon, :madeFrom, :SemillonGrape>, because Wine and Grape are also parent classes of Semillon and SemillonGrape, but we remove duplicates.

Each $\langle S, R, O \rangle$ triple is then replaced by a pair $\langle seedName(S), seedName(O) \rangle$, where $seedName(S)$ is a word sequence generated by the NL name of S, and similarly for O. We assume that the NL names are manually authored, or that they are generated by our method of Section 3. In the latter case, we keep only one NL name per individual or class, the one selected by the human author (in a semi-automatic setting of NL name generation) or the top ranked NL name (in a fully automatic setting). The five triples above become the following pairs. We call pairs of this kind *seed name pairs*, and their elements *seed names*. If a seed name results from a class, parent-class, or an equivalent class of the original S or O, we consider it a *secondary seed name*.

<"St. Emilion", "Cabernet Sauvignon grape">, <"Semillon", "Semillon grape">
<"wine", "Cabernet Sauvignon grape">, <"St. Emilion", "grape">, < "wine", "grape">

We then retrieve Web pages using the seed name pairs (of the relation that we want to generate a sentence plan for) as queries. For each seed name pair, we use the conjunction of its seed names (e.g., "St. Emilion" AND "Cabernet Sauvignon grape") as a Boolean query.[18] We convert all the retrieved pages (of all the seed name pairs) to plain text documents and parse every sentence of the retrieved documents, if at least one stemmed word from each seed name of a particular pair is the same as a stemmed word of the sentence. We then keep every parsed sentence that contains at least two NPs matching a seed name pair. For example, the sentence "obviously Semillon is made from Semillon grapes in California" contains the NPs "Semillon" and "Semillon grapes" that match the seed name pair <"Semillon", "Semillon grape"> (Fig. 3). Two NPs of a sentence match a seed name pair if the similarity between any of the two NPs and any of the two seed names (e.g., the first NP and the second seed name) is above a threshold T and the similarity between the other NP and the other seed name is also above T. The similarity between an NP and a seed name is computed as their weighted cosine similarity, with $tf \cdot idf$ weights, applied to stemmed NPs and seed names, ignoring stop-words.[19] The tf of a word of the NP or seed name is the frequency (usually 0 or 1) of the word in the NP or seed name, respectively; the idf is the inverse document frequency of the word in all the retrieved documents of the relation. We call NP *anchor pair* any two NPs of a parsed sentence that match a seed name pair, and NP *anchors* the elements of an NP anchor pair.

From every parsed sentence that contains an NP anchor pair, we produce a slotted string template by replacing the first NP anchor by S, the second NP anchor by O, including between S and O in the template the words of the sentence that were between the two NP anchors, and discarding the other words of the sentence. In the example of Fig. 3, we would obtain the template "S is made from O". Multiple templates may be extracted from the same sentence, if a sentence contains more than one NP anchor pairs; and the same template may be extracted from multiple sentences, possibly retrieved by different seed name pairs. We retain only the templates that were extracted from at least two different sentences. We then produce additional templates by increasingly extending the retained ones up to the boundaries of the sentences they were extracted from. In Fig. 3, if the template "S is made from O" has been retained, we would also produce the following templates.

obviously S is made from O	obviously S is made from O in	obviously S is made from O in California
S is made from O in	S is made from O in California	

Again, we discard extended templates that did not result from at least two sentences.

18. We do not consider corrections suggested by the search engine, since the NL names are assumed to be correct.
19. We also remove accents, and replace all numeric tokens by a particular pseudo-token.

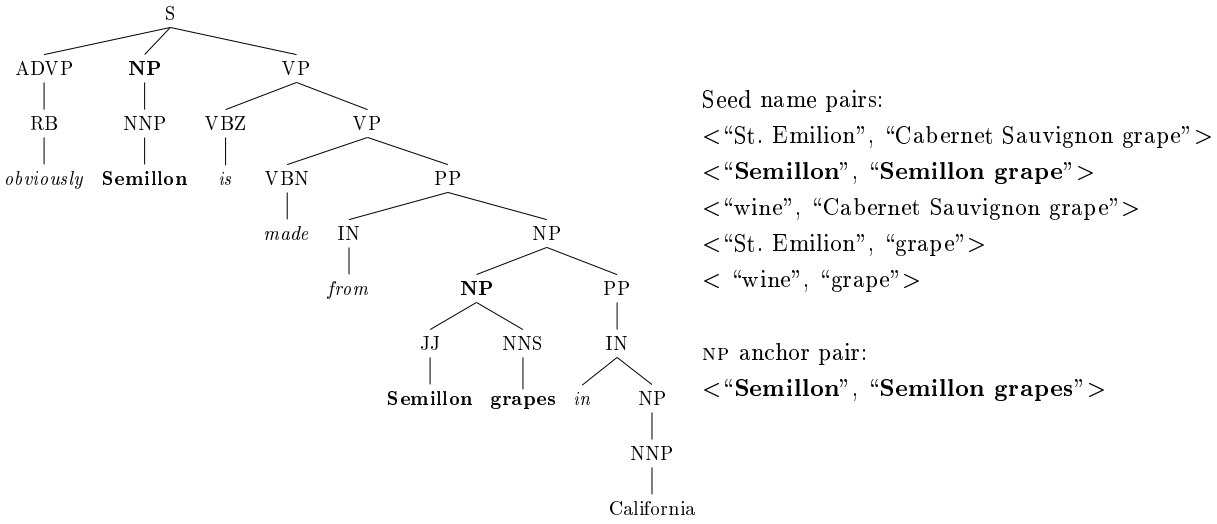


Figure 3: Parse tree of a retrieved sentence and its single NP anchor pair.

4.2 Turning the templates into candidate sentence plans

The templates (e.g., "S is made from O") are not yet sentence plans, because they lack the linguistic annotations that NaturalOWL requires (e.g., POS tags, agreement, voice, tense, cases), and they may also not correspond to well-formed sentences (e.g., they may lack verbs). The conversion of a template to a (candidate) sentence plan is similar to the conversion of Section 3.2. We start by obtaining POS tags from the parse trees of the sentences the template was obtained from. Recall that a template may have been extracted from multiple sentences. We obtain a POS tag sequence for the words of the template from each one of the sentences the template was extracted from, and we keep the most frequent POS tag sequence. We ignore the POS tags of the anchor NPs, which become S and O in the template. For example, the template "S is made from O" becomes:

$$S \text{ is}_{\text{POS}=VBZ} \text{ made}_{\text{POS}=VBN} \text{ from}_{\text{POS}=IN} \ O$$

For every verb form (e.g., "is made"), noun, adjective, and preposition, we create a corresponding slot in the sentence plan. For verbs, nouns, and adjectives, the base form is used in the slot; each verb slot is also annotated with the voice and tense of the corresponding verb form in the template. If a negation expression (e.g., "not", "aren't") is used with the verb form in the template, the negation expression is not included as a separate slot in the sentence plan, but the polarity of the verb slot is marked as negative; otherwise the polarity is positive. We determine the genders and capitalizations of nouns (and proper names) as in Section 3.2. The S and O are also replaced by slots requiring referring expressions. For example, the template "S is made from O" becomes:

$$[ref(S)]^1 \ [\text{make}]^2_{verb,\, passive,\, present,\, polarity=+} \ [\text{from}]^3_{prep} \ [ref(O)]^4$$

Agreement and case information is obtained using a dependency parser. The parser is applied to the sentences the templates were extracted from, keeping the most frequent parse per template. Referring expressions obtained from NP anchors that were verb subjects are marked with nominative case, and they are required to agree with their verbs. Referring expressions corresponding to verb objects or preposition complements are marked with accusative case (e.g., "from *him*"). Referring expressions corresponding to NP anchors with head nouns in possessive form (e.g., "Piemonte's") are marked with possessive case. In our example, we obtain:

$$[ref\,(S)]^1_{case=nom}\ [\text{make}]^2_{verb,\,passive,\,present,\,agr=1,\,polarity=+}\ [\text{from}]^3_{prep}\ [ref\,(O)]^4_{case=acc}$$

Any remaining words of the template that have not been replaced by slots (e.g., "obviously" in "obviously S is made from O") are turned into fixed string slots. Subsequently, any sentence plan that has only two slots, starts with a verb, or contains no verb, is discarded, because sentence plans of these kinds tend to be poor. Also, if a sentence plan contains a single verb in the past participle, in agreement with either S or O, followed by a preposition (e.g. "S made in O"), we insert an auxiliary verb to turn the verb form into present passive (e.g., "S is made in O"); in domains other than those of our experiments, a past passive may be more appropriate ("S was made in O). Similarly, if a single verb appears in the present participle (e.g. "S making O"), we insert an auxiliary verb to obtain a present continuous form. Both cases are illustrated below.

S made in $O \Rightarrow S$ <u>is made</u> in O $\qquad\qquad$ S making $O \Rightarrow S$ <u>is</u> making O

Lastly, we filter the remaining sentence plans through a Web search engine. For this step, we replace referring expression slots by wildcards, we generate the rest of the sentence (e.g., "* is made from *"), and we do a phrase search. If no results are returned, the sentence plan is discarded.

4.3 Applying a Maximum Entropy classifier to the candidate sentence plans

The retained candidate sentence plans are then scored using a Maximum Entropy (MaxEnt) classifier. The classifier views each candidate sentence plan p for a relation R as a vector of 251 features, and attempts to estimate the probability that p is a good sentence plan (positive class) for R or not (negative class). The 251 features provide information about p itself, but also about the templates, seed name pairs, and NP anchor pairs of p, meaning the templates that p was obtained from, and the seed name pairs and NP anchor pairs (Fig. 3) that matched to produce the templates of p.

4.3.1 Productivity features

The *productivity of the i-th seed name pair* $\langle n_{i,1}, n_{i,2}\rangle$ (e.g., $<n_{i,1}$ ="Semillon", $n_{i,2}$ = "Semillon grape">) of a relation R (e.g., R = `:madeFrom`) is defined as follows:

$$productivity(\langle n_{i,1}, n_{i,2}\rangle\,|R) = \frac{hits(\langle n_{i,1}, n_{i,2}\rangle\,|R)}{\sum_{j=1}^{N} hits(\langle n_{j,1}, n_{j,2}\rangle\,|R)} \tag{2}$$

where: $hits(\langle n_{i,1}, n_{i,2}\rangle\,|R)$ is the number of times $\langle n_{i,1}, n_{i,2}\rangle$ matched any NP anchor pair of the parsed sentences of R, counting only matches that contributed to the extraction (Section 4.1) of *any* template of R; N is the total number of seed name pairs of R; and $\langle n_{j,1}, n_{j,2}\rangle$ is the j-th seed name pair of R.[20] The intuition behind $productivity(\langle n_{i,1}, n_{i,2}\rangle\,|R)$ is that seed name pairs that match NP anchor pairs of many sentences of R are more likely to be indicative of R. When using the MaxEnt classifier to score a sentence plan p for a relation R, we compute the $productivity(\langle n_{i,1}, n_{i,2}\rangle\,|R)$ of *all* the seed name pairs $\langle n_{i,1}, n_{i,2}\rangle$ of p, and we use the *maximum*, *minimum*, *average*, *total*, and *standard deviation* of these productivity scores as five features of p.

The *productivity of a seed name* n_1 (considered on its own) that occurs as the first element of at least one seed name pair $\langle n_{i,1}, n_{i,2}\rangle = \langle n_1, n_{i,2}\rangle$ of a relation R is defined as follows:

$$productivity(n_1|R) = \frac{\sum_{i=1}^{N_2} hits(\langle n_1, n_{i,2}\rangle\,|R)}{\sum_{j=1}^{N} hits(\langle n_{j,1}, n_{j,2}\rangle\,|R)} \tag{3}$$

20. In the function $hits(\langle n_1, n_2\rangle\,|R)$, we actually multiply by $\frac{1}{2}$ the value of the function if exactly one of n_1, n_2 is a secondary seed name (Section 4.1), and we multiply by $\frac{1}{4}$ if both n_1, n_2 are secondary seed names. The same applies to the function $hits(\langle n_1, n_2\rangle, t|R)$ of Eq. 8 and the function $hits(\langle n_1, n_2\rangle, \langle a_1, a_2\rangle, t|R)$ of Eq. 9 below.

where: N_2 is the number of seed name pairs $\langle n_1, n_{i,2}\rangle$ of R that have n_1 as their first element; $hits(\langle n_1, n_{i,2}\rangle|R)$ is the number of times n_1 (as part of a seed name pair $\langle n_1, n_{i,2}\rangle$ of R) matched any element of any NP anchor pair of the parsed sentences of R and contributed to the extraction of *any* template of R; N and $hits(\langle n_{j,1}, n_{j,2}\rangle|R)$ are as in Eq. 2. Again, when using the classifier to score a sentence plan p for a relation R, we calculate the $productivity(n_1|R)$ values of *all* the (distinct) seed names n_1 that occur as first elements in the seed name pairs of p, and we use the *maximum*, *minimum*, *average*, *total*, and *standard deviation* of these productivity scores as five more features of p. We define similarly $productivity(n_2|R)$ for a seed name n_2 that occurs as the *second* element in any seed name pair $\langle n_{i,1}, n_2\rangle$ of R, obtaining five more features for p.

Similarly to Eq. 2, we define the *productivity of the i-th NP anchor pair* $\langle a_{i,1}, a_{i,2}\rangle$ (e.g., $<a_{1,1}$ ="Semillon", $a_{1,2}$ = "Semillon grapes"> in Fig. 3) of a relation R as follows:

$$productivity(\langle a_{i,1}, a_{i,2}\rangle|R) = \frac{hits(\langle a_{i,1}, a_{i,2}\rangle|R)}{\sum_{j=1}^{A} hits(\langle a_{j,1}, a_{j,2}\rangle|R)} \quad (4)$$

where: $hits(\langle a_{i,1}, a_{i,2}\rangle|R)$ is the number of times a seed name pair of R matched $\langle a_{i,1}, a_{i,2}\rangle$ in the parsed sentences of R and contributed to the extraction of *any* template of R; and A is the total number of NP anchor pairs of R.[21] As with $productivity(\langle n_{i,1}, n_{i,2}\rangle|R)$, the intuition behind $productivity(\langle a_{i,1}, a_{i,2}\rangle|R)$ is that NP anchor pairs that match many seed name pairs of R are more indicative of R. When using the classifier to score a sentence plan p for a relation R, we compute the $productivity(\langle a_{i,1}, a_{i,2}\rangle|R)$ of *all* the NP anchor pairs of p, and we use the *maximum*, *minimum*, *average*, *total*, and *standard deviation* of these scores as five additional features of p.

Similarly to Eq. 3, the *productivity of an NP anchor* a_1 (considered on its own) that occurs as the first element of at least one NP anchor pair $\langle a_{i,1}, a_{i,2}\rangle = \langle a_1, a_{i,2}\rangle$ of R is defined as follows:

$$productivity(a_1|R) = \frac{\sum_{i=1}^{A_2} hits(\langle a_1, a_{i,2}\rangle|R)}{\sum_{j=1}^{A} hits(\langle a_{j,1}, a_{j,2}\rangle|R)} \quad (5)$$

where: A_2 is the number of NP anchor pairs $\langle a_1, a_{i,2}\rangle$ of R that have a_1 as their first element; $hits(\langle a_1, a_{i,2}\rangle|R)$ is the number of times a_1 (as part of an NP anchor pair $\langle a_1, a_{i,2}\rangle$ of R) matched any element of any seed name pair of R and contributed to the extraction of *any* template of R; and A, $hits(\langle a_{j,1}, a_{j,2}\rangle|R)$ are as in Eq. 4. Again, we calculate the $productivity(a_1|R)$ values of *all* the (distinct) NP anchors a_1 that occur as first elements in the NP anchor pairs of p, and we use the *maximum*, *minimum*, *average*, *total*, and *standard deviation* of these productivity scores as five more features of p. We define similarly $productivity(a_2|R)$ for a seed name a_2 that occurs as the *second* element in any NP anchor pair $\langle a_{i,1}, a_2\rangle$ of R, obtaining five more features for p.

The *productivity of a template* t (e.g., "S is made from O") of a relation R is defined as follows:

$$productivity(t|R) = \frac{hits(t|R)}{\sum_{k=1}^{T} hits(t_k|R)} \quad (6)$$

where: $hits(t|R)$ is the number of times the *particular* template t was extracted from any of the parsed sentences of R; T is the total number of templates of R; and t_k is the k-th template of R. The intuition is that templates that are produced more often for R are more indicative of R. Again, we calculate the $productivity(t|R)$ of *all* the templates t of p, and we use the *maximum*, *minimum*, *average*, *total*, and *standard deviation* of these productivity scores as five more features of p.

21. In the function $hits(\langle a_1, a_2\rangle|R)$, we actually multiply by $\frac{1}{2}$ any match of $\langle a_1, a_2\rangle$ with a seed name pair $\langle n_1, n_2\rangle$ if exactly one of n_1, n_2 is a secondary seed name, and by $\frac{1}{4}$ if both n_1, n_2 are secondary seed names.

The *productivity of a parsed sentence* s (e.g., "obviously Semillon is made from Semillon grapes in California") of a relation R is defined as follows:

$$productivity(s|R) = \frac{hits(s|R)}{\sum_{l=1}^{L} hits(s_l|R)} \tag{7}$$

where: $hits(s|R)$ is the number of times any template of R was extracted from the *particular* parsed sentence s; L is the total number of parsed sentences of R; and s_l is the l-th parsed sentence of R. The intuition is that sentences that produce more templates for R are more indicative of R. Again, we calculate the $productivity(s|R)$ of *all* the parsed sentences s of p, and we use the *maximum*, *minimum*, *average*, *total*, and *standard deviation* of these productivity scores as features of p.

The *joint productivity* of a seed name pair $\langle n_1, n_2 \rangle$ and a template t of a relation R is:

$$productivity(\langle n_1, n_2 \rangle, t|R) = \frac{hits(\langle n_1, n_2 \rangle, t|R)}{\sum_{j=1}^{N} \sum_{k=1}^{T} hits(\langle n_{j,1}, n_{j,2} \rangle, t_k|R)} \tag{8}$$

where: $hits(\langle n_1, n_2 \rangle, t|R)$ is the number of times the *particular* seed name pair $\langle n_1, n_2 \rangle$ matched any NP anchor pair of the parsed sentences of R and contributed to the extraction of the *particular* template t; and N, T are again the total numbers of seed name pairs and templates, respectively, of R. Again, when scoring a sentence plan p for a relation R, we calculate the $productivity(\langle n_1, n_2 \rangle, t|R)$ of *all* the combinations of seed name pairs $\langle n_1, n_2 \rangle$ and templates t that led to p, and we use the *maximum*, *minimum*, *average*, *total*, and *standard deviation* of these scores as features of p. We define very similarly $productivity(n_1, t|R)$, $productivity(n_2, t|R)$, $productivity(\langle a_1, a_2 \rangle, t|R)$, $productivity(a_1, t|R)$, $productivity(a_2, t|R)$, $productivity(\langle n_1, n_2 \rangle, \langle a_1, a_2 \rangle |R)$, $productivity(n_1, a_1|R)$, $productivity(n_2, a_2|R)$, obtaining five additional features of p from each one. We also define:

$$productivity(\langle n_1, n_2 \rangle, \langle a_1, a_2 \rangle, t|R) = \frac{hits(\langle n_1, n_2 \rangle, \langle a_1, a_2 \rangle, t|R)}{\sum_{i=1}^{N} \sum_{j=1}^{A} \sum_{k=1}^{T} hits(\langle n_{i,1}, n_{i,2} \rangle, \langle a_{j,1}, a_{j,2} \rangle, t_k|R)} \tag{9}$$

where: $hits(\langle n_1, n_2 \rangle, \langle a_1, a_2 \rangle, t|R)$ is the number of times $\langle n_1, n_2 \rangle$ matched the NP anchor pair $\langle a_1, a_2 \rangle$ in a parsed sentence of R contributing to the extraction of template t; N, A, T are the numbers of seed name pairs, NP anchor pairs, templates of R. We define similarly $productivity(n_1, a_1, t|R)$ and $productivity(n_2, a_2, t|R)$, obtaining five features from each type of productivity score.

4.3.2 PROMINENCE FEATURES

For each *productivity* version of Section 4.3.1, we define a *prominence* variant. For example, based on the productivity of a seed name pair $\langle n_{i,1}, n_{i,2} \rangle$ of a relation R (Eq. 2, repeated as Eq. 10),

$$productivity(\langle n_{i,1}, n_{i,2} \rangle |R) = \frac{hits(\langle n_{i,1}, n_{i,2} \rangle |R)}{\sum_{j=1}^{N} hits(\langle n_{j,1}, n_{j,2} \rangle |R)} \tag{10}$$

we define the *prominence of a candidate sentence plan* p with respect to the *seed name pairs* of R:

$$prominence_{\,seed\_name\_pairs}(p|R) = \frac{\sum_{i=1}^{N} 1\{hits(\langle n_{i,1}, n_{i,2} \rangle |p, R) > 0\}}{\sum_{j=1}^{N} 1\{hits(\langle n_{j,1}, n_{j,2} \rangle |R) > 0\}} \tag{11}$$

where: the $1\{\xi\}$ denotes 1 if condition ξ holds and 0 otherwise; $hits(\langle n_{i,1}, n_{i,2} \rangle |p, R)$ (in the numerator) is the number of times $\langle n_{i,1}, n_{i,2} \rangle$ matched any NP anchor pair of the parsed sentences of R, counting only matches that contributed to the extraction of a template of R that led to the

particular sentence plan p; by contrast, $hits(\langle n_{j,1}, n_{j,2}\rangle|R)$ (in the denominator) is the number of times $\langle n_{j,1}, n_{j,2}\rangle$ matched any NP anchor pair of the parsed sentences of R, counting only matches that contributed to the extraction of *any template* of R; and N is the total number of seed name pairs of R. In other words, we count how many (distinct) seed name pairs of R produced p, dividing by the number of (distinct) seed name pairs of R that produced at least one template of R. The intuition is that the more seed name pairs of R lead to the sentence plan p, the better p is.

In a similar manner, we define $prominence_{anchor\_pairs}(p|R)$ based on Eq. 4, and similarly for all the other productivity versions of Section 4.3.1. We obtain one feature for the candidate sentence plan p from each prominence variant, i.e., we do not compute any maximum, minimum, average, sum, standard deviation values, unlike the productivity versions, which lead to five features each.

4.3.3 PMI features

To estimate the extent to which two seed names n_1, n_2 of a relation R co-occur when they match NP anchors to produce templates of R, we use a Pointwise Mutual Information (PMI) score:

$$\text{PMI}(\langle n_1, n_2\rangle|R) = \frac{1}{-\log productivity(\langle n_1, n_2\rangle|R)} \log \frac{productivity(\langle n_1, n_2\rangle|R)}{productivity(n_1|R) \cdot productivity(n_2|R)} \quad (12)$$

The second factor of the right-hand side of Eq. 12 is the standard PMI definition, using productivity scores instead of probabilities. The first factor normalizes the PMI scores to $[-1, 1]$ (-1 if n_1, n_2 never co-occur when producing templates of R, 0 if they are independent, 1 if they always co-occur). Intuitively, if n_1, n_2 co-occur frequently when they produce templates of R, they are strongly connected and, hence, they are more indicative of R. Again, when using the classifier to score a sentence plan p for a relation R, we calculate $\text{PMI}(\langle n_1, n_2\rangle|R)$ for *all* the seed name pairs $\langle n_1, n_2\rangle$ of R, and we use the *maximum*, *minimum*, *average*, *total*, and *standard deviation* of these PMI scores as five more features of p. We define similarly $\text{PMI}(\langle n_1, n_2\rangle, t|R)$, $\text{PMI}(n_1, t|R)$, $\text{PMI}(n_2, t|R)$, $\text{PMI}(\langle a_1, a_2\rangle|R)$, $\text{PMI}(\langle a_1, a_2\rangle, t|R)$, $\text{PMI}(a_1, t|R)$, $\text{PMI}(a_2, t|R)$, $\text{PMI}(\langle n_1, n_2\rangle, \langle a_1, a_2\rangle|R)$, $\text{PMI}(n_1, a_1|R)$, $\text{PMI}(n_2, a_2|R)$, obtaining five features for p from each one.

4.3.4 Token-based features

These features view seed names, NP anchors, templates, and OWL identifiers as sequences of tokens.

For each seed name n_1 and NP anchor a_1 that matched (as first elements of a seed name pair $\langle n_1, n_{i,2}\rangle$ and NP anchor pair $\langle a_1, a_{j,2}\rangle$) to produce a particular sentence plan p, we calculate their *cosine similarity* $\cos(n_1, a_1)$ with $tf \cdot idf$ weights (defined as in Section 4.1).[22] We then use the *maximum*, *minimum*, *average*, *total*, and *standard deviation* of these cosine similarities as features of p. Intuitively, they show how good the matches that produced p were. We repeat for each seed name n_2 and NP anchor a_2 that matched (as second elements of their pairs) to produce p, this time computing $\cos(n_2, a_2)$, obtaining five additional features of p.

We do the same using $\text{AvgTokPmi}(n_1, a_1)$, defined below, instead of $\cos(n_1, a_1)$:

$$\text{AvgTokPmi}(n_1, a_1|R) = \frac{1}{|n_1| \cdot |a_1|} \sum_{\tau \in toks(n_1)} \sum_{\tau' \in toks(a_1)} \frac{1}{-\log P(\tau, \tau'|R)} \cdot \log \frac{P(\tau, \tau'|R)}{P(\tau|R) \cdot P(\tau'|R)} \quad (13)$$

where: $|n_1|$, $|a_1|$ are the lengths (in tokens) of n_1, a_1; $toks(n_1)$, $toks(a_1)$ are the token sequences of n_1, a_1, respectively; $P(\tau|R)$ is the probability of encountering token τ in a parsed sentence of R; and $P(\tau, \tau'|R)$ is the probability of encountering both τ and τ' in the same parsed sentence of

22. When computing the features of this section, all tokens are stemmed, stop-words are ignored, accents are removed, and numeric tokens are replaced by a particular pseudo-token, as in Section 4.1.

R; we use Laplace estimates for these probabilities. Again, we compute AvgTokPmi(n_1, a_1) for every seed name n_1 and NP anchor a_1 that matched to produce a particular sentence plan p, and we use the *maximum*, *minimum*, *average*, *total*, and *standard deviation* of these scores as features of p. We repeat using AvgTokPmi(n_2, a_2) instead of $\cos(n_2, a_2)$, obtaining five more features.

Similarly, we compute AvgTokPmi$(a_1, t|R)$ and AvgTokPmi$(a_2, t|R)$ for each NP anchor a_1 or a_2 (left, or right element of an NP anchor pair) and template t (ignoring the S and O) that led to a particular sentence plan p, and we use their *maximum*, *minimum*, *average*, *total*, and *standard deviation* as ten additional features of p. We also compute $\cos(t, r)$ and AvgTokPmi$(t, r|R)$ for each template t and tokenized identifier r of a relation R (e.g., R = `:madeFrom` becomes r = "made from") that led to the sentence plan p, obtaining ten more features. Finally, we compute AvgTokPmi$(a_1, a_2|R)$, AvgTokPmi$(a_1, a_1|R)$, and AvgTokPmi$(a_2, a_2|R)$ for all the a_1 and a_2 NP anchors (first or second elements in their pairs) of p, obtaining fifteen more features of p. Although they may look strange, in effect AvgTokPmi$(a_1, a_1|R)$ and AvgTokPmi$(a_2, a_2|R)$ examine how strongly connected the words inside each NP anchor (a_1 or a_2) are.

4.3.5 Other features

Another group of features try to estimate the grammaticality of a candidate sentence plan p. Let us assume that p is for relation R. For every seed name pair of R (not only seed name pairs that led to p), we generate a sentence using p; we ignore only seed name pairs that produced no sentence plans at all, which are assumed to be poor. For example, for the seed name pair $<n_1$ ="Semillon", n_2 = "Semillon grape"> of the relation R = `:madeFrom` and the following candidate sentence plan:

$$[ref(S)]^1_{nom}\ [\text{make}]^2_{verb,\,passive,\,present,\,agr=1,\,polarity=+}\ [\text{from}]^3_{prep}\ [ref(O)]^4_{acc}$$

the sentence "Semillon is made from Semillon grapes" is generated. We do not generate referring expressions, even when required by the sentence plan (e.g., $[ref(S)]^1_{nom}$); we use the seed names instead. We obtain confidence scores for these sentences from the parser, and we normalize these scores dividing by each sentence's length. The *maximum*, *minimum*, *average*, and *standard deviation* of these scores are used as features of p.

Some additional features for a candidate sentence plan p follow:[23]

- True if p contains a present participle without an auxiliary; otherwise false.
- True if p has a main verb in active voice; otherwise false.
- True if p contains a referring expression for S before a referring expression for O; otherwise false. Sentence plans that refer to S before O are usually simpler and better.
- True if a referring expression of p is the *subject* of a verb of p; otherwise false. This information is obtained from the parsed sentences that led to p. We use the most frequent dependency tree, if p was derived from many sentences. Sentence plans with no subjects are often ill-formed.
- True if a referring expression of p is the *object* of a verb of p; otherwise false. Again, we consider the parsed sentences that led to p using the most frequent dependency tree. Sentence plans with no objects are often ill-formed, because most relations are expressed by transitive verbs.
- True if all the sentences p was derived from were well-formed, according to the parser.

23. All the non-Boolean features that we use are normalized to $[0, 1]$.

- True if p required a repair at the end of the sentence plan generation (Section 4.2); otherwise false. Repaired sentence plans can be poor.
- The number of slots of p, the number of slots before the slot for S, the number of slots after the slot for O, the number of slots between the slots for S and O (4 features).
- The *maximum*, *minimum*, *average*, *total*, *standard deviation* of the ranks of the Web pages (returned by the search engine, Section 4.1) that contained the sentences p was obtained from. Sentences from higher-ranked pages are usually more relevant to the seed name pairs we use as queries. Hence, sentence plans obtained from higher-ranked Web pages are usually better.
- The number of Web pages that contained the sentences from which p was obtained. Uncommon sentences often lead to poor sentence plans.

4.4 Ranking the candidate sentence plans

Each candidate sentence plan of a relation R is represented as a feature vector $\vec{v}$, containing the 251 features discussed above. Each vector is given to the MaxEnt classifier to obtain a probability estimate $P(c_+|\vec{v}, R)$ that it belongs in the positive class c_+, i.e., that the sentence plan is correct for R. The candidate sentence plans of each relation R are then ranked by decreasing estimated (by the classifier) $P(c_+|\vec{v}, R)$. We call SP our overall sentence plan generation method that uses the probability estimates of the classifier to rank the candidate sentence plans.

In an alternative configuration of our sentence plan generation method, denoted SP*, the probability estimate $P(c_+|\vec{v}, R)$ of each candidate sentence plan is multiplied by its *coverage*. To compute the coverage of a sentence plan for a relation R, we use the sentence plan to produce a sentence for each seed name pair of R (as when computing the grammaticality of a sentence plan in Section 4.3). Subsequently, we use each sentence as a phrase query in a Web search engine. The coverage of the sentence plan is the number of seed name pairs for which the search engine retrieved at least one document containing the search sentence (verbatim), divided by the total number of seed name pairs of R. Coverage helps avoid sentence plans that produce very uncommon sentences. Computing the coverage of *every* candidate sentence plan is time consuming, however, because of the Web searches; this is also why we do not include coverage in the features of the classifier. Hence, we first rank the candidate sentence plans of each relation R by decreasing $P(c_+|\vec{v}, R)$, and we then re-rank only the top ten of them (per R) after multiplying the $P(c_+|\vec{v}, R)$ of each one by its coverage.

In both SP and SP*, in a semi-automatic scenario we return to a human inspector the top five candidate sentence plans per relation. In a fully automatic scenario, we return only the top one.

5. Experiments

We now present the experiments we performed to evaluate our methods that generate NL names and sentence plans. We first discuss the ontologies that we used in our experiments.

5.1 The ontologies of our experiments

We used three ontologies: (i) the Wine Ontology, one of the most commonly used examples of OWL ontologies; (ii) the M-PIRO ontology, which describes a collection of museum exhibits, was originally developed in the M-PIRO project (Isard, Oberlander, Androutsopoulos, & Matheson, 2003), was later ported to OWL, and accompanies NaturalOWL (Androutsopoulos et al., 2013); and (iii) the Disease Ontology, which describes diseases, including their symptoms, causes etc.[24]

24. See also `http://www.w3.org/TR/owl-guide/wine.rdf/` and `http://disease-ontology.org/`.

The Wine Ontology involves a wide variety of OWL constructs and, hence, is a good test case for ontology verbalizers and NLG systems for OWL. The M-PIRO ontology has been used to demonstrate the high quality texts that NaturalOWL can produce, when appropriate manually authored linguistic resources are provided (Galanis, Karakatsiotis, Lampouras, & Androutsopoulos, 2009). We wanted to investigate if texts of similar quality can be generated with automatically or semi-automatically acquired NL names and sentence plans. The Disease Ontology was developed by biomedical experts to address real-life information needs; hence, it constitutes a good real-world test case.

The Wine Ontology contains 77 classes, 161 individuals, and 14 relations (properties). We aimed to produce NL names and sentence plans for the 49 classes, 146 individuals, and 7 relations that are directly involved in non-trivial definitions of wines (43 definitions of wine classes, 52 definitions of wine individuals), excluding classes, individuals, and relations that are only used to define wineries, wine-producing regions etc. By "non-trivial definitions" we mean that we ignored definitions that humans understand as repeating information that is obvious from the name of the defined class or individual (e.g., the definition of `:RedWine` in effect says that a red wine is a wine with red color).

The M-PIRO ontology currently contains 76 classes, 508 individuals, and 41 relations. Many individuals, however, are used to represent canned texts (e.g., manually written descriptions of particular types of exhibits) that are difficult to generate from symbolic information. For example, there is a pseudo-individual `:aryballos-def` whose NL name is the fixed string "An aryballos was a small spherical vase with a narrow neck, in which the athletes kept the oil they spread their bodies with". Several properties are also used only to link these pseudo-individuals (in effect, the canned texts) to other individuals or classes (e.g., to link `:aryballos-def` to the class `:Aryballos`); and many other classes are used only to group pseudo-individuals (e.g., pseudo-individuals whose canned texts describe types of vessels all belong in a common class). In our experiments, we ignored pseudo-individuals, properties, and classes that are used to represent, link, and group canned texts, since we focus on generating texts from symbolic information. We aimed to produce NL names and sentence plans for the remaining 30 classes, 127 individuals, and 12 relations, which are all involved in the definitions (descriptions) of the 49 exhibits of the collection the ontology is about.

The Disease Ontology currently contains information about 6,286 diseases, all represented as classes. Apart from IS-A relations, synonyms, and pointers to related terms, however, all the other information is represented using strings containing quasi-English sentences with relation names used mostly as verbs. For example, there is an axiom in the ontology stating that the Rift Valley Fever (DOID_1328) is a kind of viral infectious disease (DOID_934). All the other information about the Rift Valley Fever is provided in a string, shown below as 'Definition'. The tokens that contain underscores (e.g., `results_in`) are relation names. The ontology declares all the relation names, but uses them only inside 'Definition' strings. Apart from diseases, it does not define any of the other entities mentioned in the 'Definition' strings (e.g., symptoms, viruses).

Name: Rift Valley Fever (DOID_1328)
IS-A: viral infectious disease (DOID_934)
Definition: A viral infectious disease that `results_in` infection, `has_material_basis_in` Rift Valley fever virus, which is `transmitted_by` Aedes mosquitoes. The virus affects domestic animals (cattle, buffalo, sheep, goats, and camels) and humans. The infection `has_symptom` jaundice, `has_symptom` vomiting blood, `has_symptom` passing blood in the feces, `has_symptom` ecchymoses (caused by bleeding in the skin), `has_symptom` bleeding from the nose or gums, `has_symptom` menorrhagia and `has_symptom` bleeding from venepuncture sites.

We defined as individuals all the non-disease entities mentioned in the 'Definition' strings, also adding statements to formally express the relations mentioned in the original 'Definition' strings.

For example, the resulting ontology contains the following definition of Rift Valley Fever, where :infection, :Rift_Valley_fever_virus, :Aedes_mosquitoes, :jaundice etc. are new individuals.

```
SubClassOf(:DOID_1328
   ObjectIntersectionOf(:DOID_934
      ObjectHasValue(:results_in :infection)
      ObjectHasValue(:has_material_basis_in :Rift_Valley_fever_virus)
      ObjectHasValue(:transmitted_by :Aedes_mosquitoes)
      ObjectHasValue(:has_symptom :jaundice)
      ObjectHasValue(:has_symptom :vomiting_blood)
      ObjectHasValue(:has_symptom :passing_blood_in_the_feces)
      ObjectHasValue(:has_symptom :ecchymoses_(caused_by_bleeding_in_the_skin))
      ObjectHasValue(:has_symptom :bleeding_from_the_nose_or_gums)
      ObjectHasValue(:has_symptom :menorrhagia)
      ObjectHasValue(:has_symptom :bleeding_from_venepuncture_sites)))
```

The new form of the ontology was produced automatically, using patterns that searched the definition strings for relation names (e.g., results_in), sentence breaks, and words that introduce secondary clauses (e.g., "that", "which").[25] Some sentences of the original definition strings that did not include declared relation names, like the sentence "The virus affects... and humans" in the 'Definition' string of Rift Valley Fever above, were discarded during the conversion, because it was not always possible to reliably convert them to appropriate OWL statements.

The new form of the Disease Ontology contains 6,746 classes, 15 relations, and 1,545 individuals. We aimed to automatically produce NL names and sentence plans for the 94 classes, 99 individuals, and 8 relations that are involved in the definitions of 30 randomly selected diseases. We manually authored NL names and sentence plans for the same classes, individuals, and relations, to be able to compare the quality of the resulting texts. Manually authored NL names and sentence plans for the Wine and M-PIRO ontologies are also available (they are included in the software of NaturalOWL).

We note that the relations (properties) R of our experiments are all used in message triples $\langle S, R, O \rangle$, where O is an individual or class, i.e., they are *object properties* in OWL's terminology. Datatype properties, where O is a datatype value (e.g., integer, string, date), can in principle be handled using the same methods, but appropriate recognizers may be needed to obtain appropriate anchors, instead of NP anchors. For example, a datatype property may map persons to dates of birth; then a recognizer of dates would be needed to extract (and possibly normalize) appropriate date anchors from Web pages, since a parser may not treat dates as NPs.

5.2 Experiments with automatically or semi-automatically produced NL names

We now present experiments we performed with our method that generates NL names (Section 3).

5.2.1 Anonymity experiments

In a first experiment, we measured how well our NL names method determines which individuals and classes should be anonymous (Sections 2.1 and 3.1.1). We compared the decisions of our method against the corresponding anonymity declarations in the manually authored NL names of the three ontologies. Table 1 summarizes the results of this experiment. *Precision* is the total number of individuals and classes our NL names method *correctly* (in agreement with the manually authored NL names) declared as anonymous, divided by the total number of individuals and classes our method declared as anonymous. *Recall* is the total number of individuals and classes our NL names method correctly declared as anonymous, divided by the total number of individuals and classes

25. The new form of the Disease Ontology that we produced is available upon request.

	WINE	M-PIRO	DISEASE
precision	1.00 (38/38)	1.00 (49/49)	undefined (0/0)
recall	0.73 (38/52)	1.00 (49/49)	undefined (0/0)
accuracy	0.93 (184/198)	1.00 (157/157)	1.00 (195/195)

Table 1: Results of the anonymity experiments.

(among those we aimed to produce NL names for) that the manually authored NL names declared as anonymous. For the Disease Ontology, the manually authored NL names and our NL names method agreed that no individuals and classes (that we aimed to produce NL names for) should be anonymous, which is why precision and recall are undefined. *Accuracy* is the number of correct decisions (individuals and classes correctly declared, or correctly not declared as anonymous), divided by the total number of individuals and classes (that we aimed to produce NL names for).

The anonymity decisions of our method were perfect in the M-PIRO ontology and Disease Ontology. In the Wine Ontology, the precision of our method was also perfect, i.e., whenever our method decided to declare an individual or class as anonymous, this was a correct decision; but recall was lower, i.e., our method did not anonymize all the individual and classes that the manually authored NL names did. The latter is due to the fact that the manually authored NL names of the Wine ontology also anonymize 14 individuals and classes with complex identifiers (e.g., `:SchlossVolradTrochenbierenausleseRiesling`) to produce more readable texts. By contrast, our method declares individuals and classes as anonymous only to avoid redundancy in the generated texts (Section 3.1.1), hence it does not anonymize the 14 individuals and classes.

5.2.2 Inspecting the produced natural language names

We then invoked our NL name generation method for the individuals and classes it had not declared as anonymous (160 in the Wine Ontology, 108 in the M-PIRO ontology, 195 in the Disease Ontology), using the top 10 returned documents per Web search (or top 20, when the search engine proposed spelling corrections – see Section 3.1.3). We ranked the produced NL names (as in Sections 3.1.3 and 3.2), and kept the top 5 NL names per individual or class. The first author then inspected the resulting NL names and marked each one as correct or incorrect. An NL name was considered correct if and only if: (i) it would produce morphologically, syntactically, and semantically correct and unambiguous noun phrases (e.g., "Cabernet Sauvignon grape" is correct for `:CabernetSauvignonGrape`, but "Cabernet Sauvignon wine", "Cabernet Sauvignon", or "grape" are incorrect); and (ii) its slot annotations (e.g., POS tags, gender, agreement) were all correct.

Table 2 shows the results of this experiment. The "*1-in-1*" score is the ratio of individuals and classes for which the top returned NL name was correct. The "*1-in-3*" score is the ratio of individuals and classes for which there was at least one correct NL name among the top three, and similarly for "*1-in-5*". The "1-in-1" score corresponds to a fully automatic scenario, where the top NL name is used for each individual or class, without human intervention. By contrast, the "1-in-3" and "1-in-5" scores correspond to a semi-automatic scenario, where a human inspects the top three or five, respectively, NL names per individual or class, looking for a correct one to select. The *mean reciprocal rank* (MRR) is the mean (over all the individuals and classes we asked our method to produce NL names for) of the reciprocal rank $r_i = 1/k_i$, where k_i is the rank (1 to 5) of the top-most correct NL name returned for the i-th individual or class; if no correct NL name exists among the top five, then $r_i = 0$. MRR rewards more those methods that place correct NL names towards the top of the five returned ones. The *weighted* scores of Table 2 are similar, but they weigh each individual or class by the number of OWL statements that mention it in the ontology.

	WINE	M-PIRO	DISEASE
1-in-1	0.69 (110/160)	0.77 (83/108)	0.74 (145/195)
1-in-3	0.93 (145/160)	0.94 (102/108)	0.89 (173/195)
1-in-5	0.95 (152/160)	0.95 (103/108)	0.90 (176/195)
MRR	0.79	0.85	0.80
weighted 1-in-1	0.69	0.77	0.76
weighted 1-in-3	0.93	0.94	0.91
weighted 1-in-5	0.96	0.95	0.93
weighted MRR	0.80	0.85	0.82

Table 2: Results of the inspection of the produced NL names.

The results of Table 2 show that our NL names method performs very well in a semi-automatic scenario. In a fully automatic scenario, however, there is large scope for improvement. We note, though, that our definition of correctness (of NL names) in the experiment of this section was very strict. For example, an NL name with only a single error in its slot annotations (e.g., a wrong gender in a noun slot) was counted as incorrect, even if in practice the error might have a minimal effect on the generated texts that would use the NL name. The experiment of Section 5.2.4 below, where NL names are considered in the context of generated texts, sheds more light on this point.

By inspecting the produced NL names, we noticed that our method is very resilient to spelling errors and abbreviations in the OWL identifiers of individuals and classes. For example, it returns NL names producing "a Côte d'Or wine" for `:CotesDOr`, and "the Naples National Archaeological Museum" for `:national-arch-napoli`. Several wrongly produced NL names are due to errors of tools that our method invokes (e.g., parser, NER). Other errors are due to over-shortened `altTokNames` (Section 3.1); e.g., one of the `altTokNames` of `:CorbansDryWhiteRiesling` was simply "Dry White", which leads to an NL name that does not identify the particular wine clearly enough. Finally, in the Disease Ontology, our automatic conversion of the 'Description' strings produced many individuals whose identifiers are in effect long phrases (see, for example, the OWL description of `:DOID_1328` in Section 5.1). Our NL names method manages to produce appropriate NL names (with correct slot annotations etc.) for some of them (e.g., `:mutation_in_the_SLC26A2_gene`), but produces no NL names in other cases (e.g., `:infection_of_the_keratinized_layers`). Some of these errors, however, may not have a significant effect on the generated texts (e.g., using the tokenized identifer "infection of the keratinized layers", which is the default when no NL name is provided, may still lead to a reasonable text). Again, the experiment of Section 5.2.4 below sheds more light on this point.

5.2.3 Annotator agreement and effort to semi-automatically author NL names

The top five automatically produced NL names of each individual and class were also shown to a second human judge. The second judge was a computer science researcher not involved in NLG, fluent in English, though not a native speaker. For each individual or class, and for each one of its top five NL names, the judge was shown a phrase produced by the NL name (e.g., "Cabernet Sauvignon"), an automatically generated sentence about the individual or class (expressing a message triple of the ontology) illustrating the use of the NL name (e.g., "Cabernet Sauvignon is a kind of wine."), and a sentence where the NL name had been automatically replaced by a pronoun (e.g., "It is a kind of wine.") to check the gender of the NL name. The judge was asked to consider the phrases and sentences, and mark the best correct NL name for each individual or class. The judge could also mark more than one NL names for the same individual or class, if more than one seemed correct and equally good; the judge was instructed not to mark any of the five NL names, if none seemed correct. The judge completed this task in 49, 45, and 75 minutes for the Wine, M-PIRO, and Disease

	WINE	M-PIRO	DISEASE
micro-precision	0.97	0.96	0.97
macro-precision	0.98	0.94	0.98
J_1 1-in-5	0.95	0.95	0.90
J_2 1-in-5	0.95	0.93	0.90
pseudo-recall	1.00	0.96	1.00
Cohen's Kappa	0.77	0.80	0.98

Table 3: Inter-annotator agreement in the semi-automatic authoring of NL names.

Ontology (727, 540, 965 candidate NL names), respectively; by contrast, manually authoring the NL names of the three ontologies took approximately 2, 2, and 3 working days, respectively. These times and the fact that the second judge was not aware of the internals of NaturalOWL and its resources suggest that the semi-automatic authoring scenario is viable and very useful in practice.

Table 3 compares the decisions of the second judge, hereafter called J_2, to those of the first author, hereafter called J_1. J_1 was able to view the full details of the NL names using NaturalOWL's Protégé plug-in, unlike J_2 who viewed only phrases and example sentences. For the purposes of this study, J_1 marked *all* the correct NL names (not only the best ones) among the top five of each individual or class. In Table 3, *micro-precision* is the number of NL names (across all the individuals and classes) that were marked as correct by both J_1 and J_2, divided by the number of NL names marked as correct by J_2, i.e., we treat the decisions of J_1 as gold. *Macro-precision* is similar, but we first compute the precision of J_2 against J_1 separately for each individual or class, and we then average over all the individuals and classes. J_1 *1-in-5* is the percentage of individuals and classes for which J_1 marked at least one NL name among the top five as correct, and similarly for J_2 *1-in-5*. *Pseudo-recall* is the number of individuals and classes for which both J_1 and J_2 marked at least one NL name as correct, divided by the number of individuals and classes for which J_1 marked at least one NL name as correct; this measure shows how frequently J_2 managed to find at least one correct (according to J_1) NL name, when there was at least one correct NL name among the top five. Computing the true recall of the decisions of J_2 against those of J_1 would be inappropriate, because J_2 was instructed to mark only the best NL name(s) of each individual and class, unlike J_1 who was instructed to mark all the correct ones. We also calculated Cohen's Kappa between J_1 and J_2; for each individual or class, if J_1 had marked more than one NL names as correct, we kept only the top-most one, and similarly for J_2, hence each judge had six possible choices (including marking no NL name) per individual and class. The results of Table 3 indicate strong inter-annotator agreement in the semi-automatic authoring of NL names in all three ontologies.

5.2.4 Evaluating natural language names in generated texts

In order to examine how the produced NL names affect the perceived quality of the generated texts, we showed automatically generated texts describing individuals and classes of the three ontologies to six computer science students not involved in the work of this article; they were all fluent, though not native, English speakers. We generated texts using NaturalOWL configured in four ways. The NO-NLN configuration uses no NL names; in this case, NaturalOWL uses the tokenized OWL identifiers of the individuals and classes as their names.[26] MANUAL-NLN uses manually authored NL names. AUTO-NLN uses the top-ranked NL name that our NL names method produces for each individual and class. Finally, SEMI-AUTO-NLN uses the NL name (of each individual or class) that a human inspector (the first author of this article) selected among the top five NL names produced by our method.

26. If the ontology provides an `rdfs:label` for an individual or class, NO-NLN uses its tokens.

Additionally, both AUTO-NLN and SEMI-AUTO-NLN use the methods of Sections 3.1.1 and 3.3 to anonymize individuals or classes and to infer interest scores from NL names, whereas MANUAL-NLN uses the anonymity declarations and interest scores of the manually authored linguistic resources, and NO-NLN uses no anonymity declarations and no interest scores. Apart from the NL names, anonymity declarations, and interest scores, all four configurations use the same, manually authored other types of linguistic resources (e.g., sentence plans, text plans to order the message triples). Below are example texts generated from the three ontologies by the four configurations.

MANUAL-NLN: This is a moderate, dry Zinfandel. It has a medium body. It is made by Saucelito Canyon in the city of Arroyo Grande.
SEMI-AUTO-NLN: This is a moderate, dry Zinfandel wine. It has a medium body. It is made by the Saucelito Canyon Winery in the Arroyo Grande area.
AUTO-NLN: This is a dry Zinfandel and has the medium body. It is the moderate. It is made by Saucelito Canyon in Arroyo Grande.
NO-NLN: Saucelito Canyon Zinfandel is Zinfandel. It is Dry. It has a Medium body. It is Moderate. It is made by Saucelito Canyon. It is made in Arroyo Grande Region.

MANUAL-NLN: This is a statue, created during the classical period and sculpted by Polykleitos. Currently it is exhibited in the National Archaeological Museum of Napoli.
SEMI-AUTO-NLN: This is a statue, created during the classical period and sculpted by the sculptor polyclitus. Currently it is exhibited in the Naples National Archaeological Museum.
AUTO-NLN: This is a statue, created during classical and sculpted by the polyclitus. Currently it is exhibited in national arch napoli.
NO-NLN: Exhibit 4 is statue, created during classical period and sculpted by polyclitus. Today it is exhibited in national arch napoli.

MANUAL-NLN: Systemic mycosis is a kind of fungal infectious disease that affects the human body. It results in infection of internal organs. It is caused by fungi.
SEMI-AUTO-NLN: A systemic mycosis is a kind of fungal infectious disease that affects human body. It results in infections of internal organs and it is caused by the fungi.
AUTO-NLN: A systemic mycosis is fungal that affects human body. It results in infections of internal organs and it is caused by the fungi.
NO-NLN: Systemic mycosis is a kind of fungal infectious disease. It affects human body. It results in infection of internal organs. It is caused by Fungi.

We note that some NL names of SEMI-AUTO-NLN and AUTO-NLN can be easily improved using the Protégé plug-in of NaturalOWL. For example, the NL name of the human body can be easily modified to include a definite article, which would improve the texts of SEMI-AUTO-NLN and AUTO-NLN in the Disease Ontology examples above ("affects the human body" instead of "affects human body").[27] Nevertheless, we made no such improvements.

Recall that there are $43 + 52 = 95$ non-trivial definitions of wine classes and wine individuals in the Wine Ontology, 49 exhibits in the M-PIRO ontology, and that we randomly selected 30 diseases from the Disease Ontology (Section 5.1). Hence, we generated $95 \times 4 = 380$ texts from the Wine Ontology (with the four configurations of NaturalOWL), $49 \times 4 = 196$ texts from the M-PIRO ontology, and $30 \times 4 = 120$ texts from the Disease ontology. For each individual or class, the message triples of its definition (regardless of their interest scores) along with the corresponding texts were given to exactly one student. The four texts of each individual or class were randomly ordered and the students did not know which configuration had generated each one of the four texts. For each individual or class, the students were asked to compare the four texts to each other

27. The missing article is due to the fact that the on-line dictionary we used marks "body" as (possibly) non-countable.

and to the message triples, and score each text by stating how strongly they agreed or disagreed with statements S_1–S_4 below. A scale from 1 to 5 was used (1: strong disagreement, 3: ambivalent, 5: strong agreement). Examples and more detailed guidelines were also provided to the students.

(S_1) *Sentence fluency*: Each sentence of the text (on its own) is grammatical and sounds natural.

(S_2) *Clarity*: The text is easy to understand, provided that the reader is familiar with the terminology and concepts of the domain (e.g., historical periods, grape varieties, virus names).

(S_3) *Semantic correctness*: The text accurately conveys the information of the message triples.

(S_4) *Non-redundancy*: There is no redundancy in the text (e.g., stating the obvious, repetitions).

Criteria	NO-NLN	AUTO-NLN	SEMI-AUTO-NLN	MANUAL-NLN
Sentence fluency	3.99	3.50	**4.70**[1]	**4.83**[1]
Clarity	4.41	3.43	**4.79**[1]	**4.79**[1]
Semantic correctness	4.44[1]	3.54	**4.66**[1,2]	**4.85**[2]
Non-redundancy	3.17	3.93[1]	**4.31**[1,2]	**4.56**[2]

Table 4: Human scores for the Wine Ontology with different methods to obtain NL names.

Criteria	NO-NLN	AUTO-NLN	SEMI-AUTO-NLN	MANUAL-NLN
Sentence fluency	4.14[1]	4.22[1]	**4.90**[2]	**4.98**[2]
Clarity	4.00[1]	3.69[1]	**4.82**[2]	**4.98**[2]
Semantic correctness	4.06[1]	4.04[1]	**4.82**[2]	**4.98**[2]
Non-redundancy	3.18	4.06	**4.86**[1]	**4.96**[1]

Table 5: Human scores for the M-PIRO ontology with different methods to obtain NL names.

Criteria	NO-NLN	AUTO-NLN	SEMI-AUTO-NLN	MANUAL-NLN
Sentence fluency	**4.27**[1,2]	4.03[1]	**4.40**[1,2]	**4.73**[2]
Clarity	**4.73**[1]	3.80	**4.57**[1]	**4.90**[1]
Semantic correctness	**4.83**[1]	4.23[2]	**4.47**[1,2]	**4.87**[1]
Non-redundancy	**4.23**[1]	**4.30**[1]	**4.43**[1]	**4.33**[1]

Table 6: Human scores for the Disease Ontology with different methods to obtain NL names.

Tables 4–6 show the scores of the four configurations of NaturalOWL, averaged over the texts of each ontology. For each criterion, the best scores are shown in bold. In each criterion (row), we detected no statistically significant differences between scores marked with the same superscript; all the other differences (in the same row) were statistically significant.[28] Overall, the manually authored NL names led to the best (near-perfect) scores, as one might expect. The scores of SEMI-AUTO-NLN were overall slightly lower, but still high (always $\geq 4.3/5$) and no statistically significant differences to the corresponding scores of MANUAL-NLN were detected. These findings confirm that our NL names method performs very well in a semi-automatic scenario, where a human inspects and selects among the top-ranked automatically produced NL names. By contrast, AUTO-NLN performed overall much worse than SEMI-AUTO-NLN and MANUAL-NLN, and often worse than NO-NLN, which again indicates that our NL names method cannot be used in a fully automatic manner.

The NO-NLN configuration, which uses tokenized identifiers of individuals and classes, performed overall much worse than MANUAL-NLN and SEMI-AUTO-NLN in the Wine and M-PIRO ontologies, which shows the importance of NL names in the perceived quality of generated texts. The differences between NO-NLN, SEMI-AUTO-NLN, and MANUAL-NLN were smaller in the Disease Ontology, where

28. We performed Analysis of Variance (ANOVA) and post-hoc Tukey tests ($a = 0.05$) in the four scores of each criterion in each ontology. A post-hoc power analysis of the ANOVA values resulted in power values greater or equal to 0.98, with the exception of the non-redundancy scores of the Disease Ontology, where power was 0.25.

no statistically significant differences between the three configurations were detected. These smaller differences are due to the fact that the conversion of the Disease Ontology (Section 5.1) produced many individuals whose OWL identifiers are in effect long phrases, easily readable, and sometimes better than then top-ranked NL names of our methods; furthermore, our NL names method does not manage to produce any NL names for many of these individuals and, hence, SEMI-AUTO-NLN ends up using their tokenized identifiers, like NO-NLN. We also note that there are very few redundant message triples and no anonymous individuals or classes in the Disease Ontology, which explains the higher non-redundancy scores of NO-NLN in the Disease Ontology, compared to the much lower non-redundancy scores of NO-NLN in the other two ontologies.

5.3 Experiments with automatically or semi-automatically produced sentence plans

We now present the experiments we performed to evaluate our method that generates sentence plans (Section 4). Recall that our method employs a MaxEnt classifier to predict the probability that a candidate sentence plan is correct (positive class) or incorrect (negative class).[29]

5.3.1 Training the classifier of the sentence plan generation method

To create training instances for the MaxEnt classifier, we used our sentence plan generation method without the classifier to obtain candidate sentence plans (as in Sections 4.1 and 4.2) from Wikipedia for the seven relations of the Wine Ontology (Section 5.1). We used the manually authored NL names of the Wine Ontology to obtain seed names, and the top 50 Wikipedia articles of each search query.[30] We searched Wikipedia exclusively at this stage, as opposed to querying the entire Web, to obtain high quality texts and, hence, hopefuly more positive training examples (correct sentence plans). The first author then manually tagged the resulting 655 candidate sentence plans as positive or negative training instances, depending on whether or not they were correct. A candidate sentence plan was considered correct if and only if: (i) it would produce morphologically, syntactically, and semantically correct sentences; and (ii) the annotations of its slots (e.g., POS tags, voice, tense, agreement) were all correct. To compensate for class imbalance in the training set (16% positive vs. 84% negative candidate sentence plans), we replicated all the positive training instances (over-sampling) to obtain an equal number of positive and negative training instances.

Figure 4 shows the error rate of the classifier on (i) unseen instances (test error) and (ii) on the instances it has been trained on (training error). To obtain the curves of Fig. 4, we performed a leave-one-out cross validation on the 655 instances (candidate sentence plans) we had constructed, i.e., we repeated the experiment 655 times, each time using a different instance as the only test instance and the other 654 instances as the training dataset. Within each repetition of the cross-validation, we iteratively trained the classifier on 10%, 20%, ..., 100% of the training dataset (654 instances, with over-sampling applied to them). The *training error* counts how many of the instances that were used to train the classifier were also correctly classified by the classifier. The *test error* counts how many of the test (unseen) instances (one in each repetition of the cross-validation) were correctly classified by the classifier (trained on the corresponding percentage of the training dataset). The error rates of Fig. 4 are averaged over the 655 repetitions of the cross-validation.[31] The training error curve can be thought of as a lower bound of the test error curve, since a classifier typically performs better on the instances it has been trained on than on unseen instances. The two curves indicate that the classifier might perform slightly better with more training data, though

29. We also experimented with SVMs and different kernels, but saw no significant improvements compared to MaxEnt.
30. We used Google's Custom Search API (`developers.google.com/custom-search/`) to search Wikipedia.
31. We tuned the similarity threshold T (Section 4.1) to 0.1, based on additional cross-validation experiments.

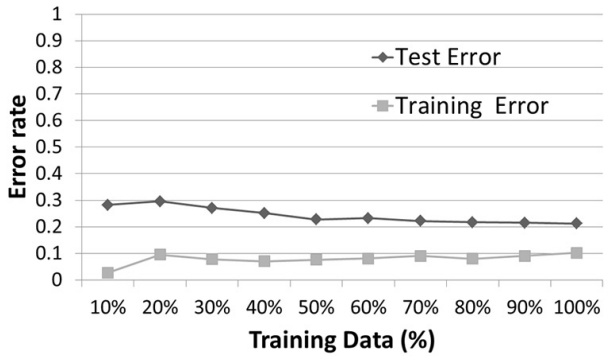


Figure 4: Test and training error rate of the classifier of our sentence plan generation method.

	Avg. IG	Max. IG	Min. IG
Productivity features	0.29	0.58	0.06
Prominence features	0.13	0.29	0.05
PMI features	0.50	0.62	0.21
Token-based features	0.48	0.61	0.03
Other features	0.20	0.62	0.00

Table 7: Information Gain of different groups of features of our sentence plan generation method.

the test error rate would remain above 0.1. The relatively small distance of the right ends of the two curves indicates only mild overfitting when the entire training dataset is used.

To assess the contribution of the 251 features (Section 4.3), we ranked them by decreasing Information Gain (IG) (Manning & Schutze, 2000) computed on the 655 instances. Table 7 shows the maximum, minimum, and average IG scores of the features in each group (subsection) of Section 4.3. On average, the PMI and token-based features are the best predictors, whereas the prominence features are the worst. The maximum scores, however, show that there is at least one good feature in each group, with the prominence features being again the worst in this respect. The minimum scores indicate that there is also at least one weak feature in every group, with the exception of the PMI features, where the minimum IG score (0.21) was much higher. Figure 5 shows the IG scores of all the features in each group, in ascending order. There are clearly many good features in every group, with the prominence features again being the weakest group overall.

We then iteratively trained the classifier on the entire training dataset (100%), removing at each iteration the feature (among the remaining ones) with the smallest IG score. Figure 6 shows the resulting test and training error rate curves, again obtained using a leave-one-out cross-validation. As more features are removed, the distance between the training and test error decreases, because of reduced overfitting. When very few features are left (far right), the performance of the classifier on unseen instances becomes unstable. The best results are obtained using all (or almost all) of the features, but the test error is almost stable from approximately 50 to 200 removed features, indicating that there is a lot of redundancy (e.g., correlated features) in the feature set. Nevertheless, we did not remove any features in the subsequent experiments, since the overfitting was reasonably low and the training and test times of the MaxEnt classifier were also low (performing a leave-one-out cross-validation on the 655 instances with all the features took approximately 6 minutes). We hope to explore dimensionality reduction further (e.g., via PCA) in future work.

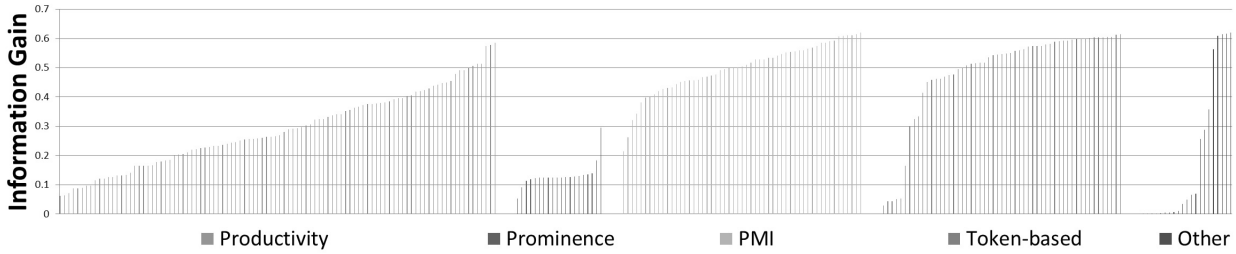


Figure 5: Ascending IG in each group of features used by our sentence plan generation method.

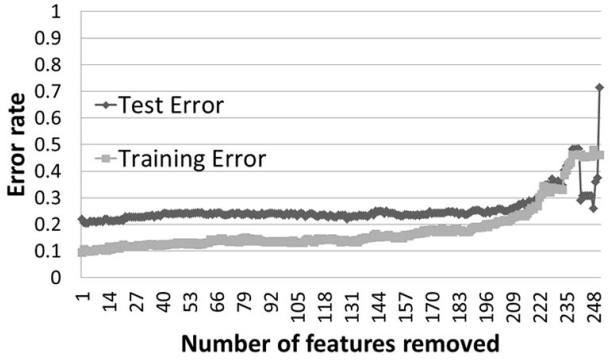


Figure 6: Error rate of the classifier of our sentence plan method, for different numbers of features.

5.3.2 Inspecting the produced sentence plans

In a subsequent experiment, the classifier was trained on the 655 instances (candidate sentence plans) of the previous section; recall that those instances were obtained from Wikipedia articles for Wine Ontology relations. The classifier was then embedded (without retraining) in our overall sentence plan generation method (SP or SP*, see Section 4.4). The sentence plan generation method was then invoked to produce sentence plans from the entire Web, not just Wikipedia, and for the relations of all three ontologies, not just those of the Wine Ontology.[32] We kept the top 10 returned documents per Web search (Section 4.1), to reduce the time to complete them. The first author inspected the top 5 sentence plans per relation (as ranked by SP or SP*), marking them as correct or incorrect (as in Section 5.3.1). We then computed the *1-in1*, *1-in-5*, and MRR scores of the produced sentence plans per ontology, along with weighted variants of the three measures. All six measures are defined as in Section 5.2.2, but for sentence plans instead of NL names; the weighted variants weigh each relation by the number of OWL statements that mention it in the ontology.

Tables 8–10 show the results for the three ontologies. The configurations "with seeds of MANUAL-NLN" use the NL names from the manually authored linguistic resources to obtain seed names (Section 4.1); by contrast, the configurations "with seeds of SEMI-AUTO-NLN" use the semi-automatically produced NL names to obtain seed names. Recall that SP* reranks the candidate sentence plans using their coverage (Section 4.4). Tables 8–10 also include results for a bootstrapping baseline (BOOT), described below. For each measure, the best results are shown in bold.

32. To avoid testing the classifier on sentence plans it had encountered during its training (more likely to happen with sentence plans for Wine Ontology relations), we actually excluded from the training data of the classifier any of the 655 instances that had the same feature vector with any test candidate sentence plan of the experiments of this section, when classifying that particular test candidate sentence plan.

	SP with seeds of MANUAL-NLN	SP* with seeds of MANUAL-NLN	SP with seeds of SEMI-AUTO-NLN	SP* with seeds of SEMI-AUTO-NLN	BOOT with seeds of MANUAL-NLN
1-in-1	0.71 (5/7)	**0.86** (6/7)	0.71 (5/7)	0.71 (5/7)	0.57 (4/7)
1-in-3	0.86 (6/7)	0.86 (6/7)	**1.00** (7/7)	**1.00** (7/7)	0.71 (5/7)
1-in-5	**1.00** (7/7)	**1.00** (7/7)	**1.00** (7/7)	**1.00** (7/7)	0.86 (6/7)
MRR	0.82	**0.89**	0.83	0.86	0.68
weighted 1-in-1	0.73	**0.86**	0.71	0.71	0.58
weighted 1-in-3	0.86	0.86	**1.00**	**1.00**	0.71
weighted 1-in-5	**1.00**	**1.00**	**1.00**	**1.00**	0.86
weighted MRR	0.83	**0.89**	0.83	0.86	0.68

Table 8: Results of the inspection of the produced sentence plans for the Wine Ontology.

	SP with seeds of MANUAL-NLN	SP* with seeds of MANUAL-NLN	SP with seeds of SEMI-AUTO-NLN	SP* with seeds of SEMI-AUTO-NLN	BOOT with seeds of MANUAL-NLN
1-in-1	0.58 (7/12)	**0.67** (8/12)	**0.67** (8/12)	**0.67** (8/12)	0.17 (2/12)
1-in-3	0.67 (8/12)	0.67 (8/12)	**0.75** (9/12)	**0.75** (9/12)	0.33 (4/12)
1-in-5	0.67 (8/12)	0.67 (8/12)	**0.83** (10/12)	**0.83** (10/12)	0.33 (4/12)
MRR	0.62	0.67	**0.73**	**0.73**	0.24
weighted 1-in-1	0.73	**0.85**	0.61	0.72	0.04
weighted 1-in-3	**0.85**	**0.85**	0.73	0.73	0.19
weighted 1-in-5	0.85	0.85	**0.98**	**0.98**	0.45
weighted MRR	0.79	**0.85**	0.73	0.79	0.22

Table 9: Results of the inspection of the produced sentence plans for the M-PIRO ontology.

Overall, SP* performs better than SP, though the scores of the two methods are very close or identical in many cases, and occasionally SP performs better. Also, SP and SP* occasionally perform better when semi-automatically produced NL names are used to obtain seed names, than when manually authored NL names are used. It seems that manually authored NL names occasionally produce seeds that are uncommon on the Web and, hence, do not help produce good sentence plans, unlike semi-automatically produced NL names, which are extracted from the Web (and then manually filtered). The 1-in-5 results of Tables 8–10 show that our sentence plan generation method (especially SP*) performs very well in a semi-automatic scenario, especially if the weighted measures are considered. By contrast, our method does not always perform well in a fully automatic scenario (1-in-1 results); the Disease Ontology was the most difficult in that respect. Overall, SP* with seeds of SEMI-AUTO-NLN seems to be the best version. The MRR scores of our method (all versions) were higher in the Wine Ontology and lower in the other two ones, which may be due to the fact that the classifier was trained for Wine Ontology relations (but with texts from Wikipedia).

While inspecting the sentence plans, we noticed several cases where the OWL identifier of the relation was poor (e.g., the `:locatedIn` relation of the Wine Ontology connects wines to the regions producing them, but our method produced good sentence plans (e.g., [*ref*(*S*)] [is produced] [in] [*ref*(*O*)]). On the other hand, our method (all versions) produced very few (or none) sentence plans for relations with fewer than 10 seed name pairs. Also, the most commonly produced sentence plans are [*ref*(*S*)] [is] [*ref*(*O*)] and [*ref*(*O*)] [is] [*ref*(*S*)]. While the former may be appropriate for a message triple $\langle S, R, O \rangle$, the latter is almost never appropriate, so we always discard it.

5.3.3 Agreement and effort to semi-automatically author sentence plans

The top five sentence plans of SP* for each relation were also shown to the second human judge (J_2) who had examined the automatically produced NL names in the experiments of Section 5.2.3. For

	SP with seeds of MANUAL-NLN	SP* with seeds of MANUAL-NLN	SP with seeds of SEMI-AUTO-NLN	SP* with seeds of SEMI-AUTO-NLN	BOOT with seeds of MANUAL-NLN
1-in-1	0.00 (0/8)	0.37 (3/8)	0.12 (1/8)	**0.50** (4/8)	0.12 (1/8)
1-in-3	0.75 (6/8)	**0.87** (7/8)	0.75 (6/8)	**0.87** (7/8)	0.37 (3/8)
1-in-5	**1.00** (8/8)	**1.00** (8/8)	**1.00** (8/8)	**1.00** (8/8)	0.75 (6/8)
MRR	0.41	0.65	0.47	**0.70**	0.33
weighted 1-in-1	0.00	0.16	0.01	**0.17**	0.00
weighted 1-in-3	0.42	**0.89**	**0.89**	**0.89**	0.15
weighted 1-in-5	**1.00**	**1.00**	**1.00**	**1.00**	0.85
weighted MRR	0.35	**0.55**	0.45	0.48	0.22

Table 10: Results of the inspection of the produced sentence plans for the Disease Ontology.

	WINE	M-PIRO	DISEASE
micro-precision	1.00	1.00	1.00
macro-precision	1.00	1.00	1.00
J_1 1-in-5	1.00	0.83	1.00
J_2 1-in-5	1.00	0.75	1.00
pseudo-recall	1.00	0.75	1.00
Cohen's Kappa	1.00	0.62	0.74

Table 11: Inter-annotator agreement in the semi-automatic authoring of sentence plans.

each relation, and for each one of its top five sentence plans, the second judge was shown a template view of the sentence plan (e.g., "S is made from O"), and an automatically generated sentence illustrating its use (e.g., "Cabernet Sauvignon is made from Cabernet Sauvignon grapes."). The judge was asked to consider the templates and sentences, and mark the best correct sentence plan for each relation. The judge could also mark more than one sentence plans for the same relation, if more than one seemed correct and equally good; the judge was instructed not to mark any of the five sentence plans, if none seemed correct. The second judge completed this task (inspecting 40, 29, and 40 candidate sentence plans of the Wine, M-PIRO, and Disease Ontology, respectively) in approximately 5 minutes per ontology (15 minutes for all three ontologies); by contrast, manually authoring the sentence plans took approximately one working day per ontology. Again, these times suggest that the semi-automatic authoring scenario is viable and very useful in practice.

We also measured the agreement between the second judge (J_2) and the first author (J_1) in the semi-automatic authoring (selection) of sentence plans, as in Section 5.2.3. The results, reported in Table 11, show perfect agreement in the Wine and Disease Ontologies. In the M-PIRO ontology, the agreement was lower, but still reasonably high; the pseudo-recall score shows J_2 did not select any sentence plan for some relations where J_1 believed there was a correct one among the top five.

5.3.4 The sentence plan generation baseline that uses bootstrapping

As a baseline, we also implemented a sentence plan generation method that uses a bootstrapping template extraction approach. Bootstrapping is often used in information extraction to obtain templates that extract instances of a particular relation (e.g., `:makeFrom`) from texts, starting from seed pairs of entity names (e.g., <"cream", "milk">) for which the relation is known to hold (Riloff & Jones, 1999; Muslea, 1999; Xu, Uszkoreit, & Li, 2007; Gupta & Manning, 2014). The seed pairs are used as queries in a search engine to obtain documents that contain them in the same sentence (e.g., "cream is made from milk"). Templates are then obtained by replacing the seeds with slots in the retrieved sentences (e.g., "X is made from Y"). The templates (without their slots, e.g.,

"is made from") are then used as phrasal search queries to obtain new sentences (e.g., "gasoline is made from petroleum"), from which new seed pairs (<"gasoline", "petroleum">) are obtained. A new iteration can then start with the new seed pairs, leading to new templates, and so on.

Given a relation R, our baseline, denoted BOOT, first constructs seed name pairs using the ontology and the NL names, as in Section 4.1; we used only manually authored NL names in the experiments with BOOT. Then, BOOT uses the seed name pairs to obtain templates ("X is made from Y") from the Web, again as in Section 4.1. If the number of obtained templates is smaller than L, the templates (without their slots) are used as phrasal search queries to obtain new documents and new sentences (from the documents) that match the templates. For each new sentence (e.g., "gasoline is made from petroleum"), BOOT finds the NPs ("gasoline", "petroleum") immediately before and after the search phrase, and treats them as a new seed name pair, discarding pairs that occur in only one retrieved document. The new pairs are then used to obtain new templates, again discarding templates occurring in only one document. This process is repeated until we have at least L templates for R, or until no new templates can be produced. In our experiments, we set $L = 150$ to obtain approximately the same number of templates as with SP and SP*.

At the end of the bootstrapping, instead of using a MaxEnt classifier (Sections 4.3 and 4.4), BOOT scores the templates of each relation R using the following confidence function:[33]

$$conf(t) = \frac{hits(t)}{hits(t) + misses(t)} \cdot \log finds(t) \tag{14}$$

where t is a template being scored, $hits(t)$ is the number of (distinct) NP anchor pairs of R extracted by t (from the documents retrieved by all the seed name pairs of R), $misses(t)$ is the number of (distinct) NP anchor pairs of R *not* extracted by t (from the same documents), and $finds(t)$ is the number of sentences (of the same documents) that match t. The five templates with the highest $conf(t)$ (in a semi-automatic scenario) or the single template with the highest $conf(t)$ (in a fully automatic scenario) are then converted to sentence plans, as in Section 4.2.

Functions like Eq. 14 can also be applied *within* each iteration of the bootstrapping, not only at the end of the entire bootstrapping, to keep only the best new templates of each iteration. This may help avoid concept drift, i.e., gradually obtaining templates that are more appropriate for other relations that share seed name pairs with the relation we wish to generate templates for. We did not use Eq. 14 within each iteration, because in our experiments very few iterations (most often only the initial one) were needed. Also, using a function like Eq. 14 within each iteration requires a threshold d, to discard templates with $conf(t) < d$ at the end of each iteration, which is not trivial to tune. Similar functions can be used to score the new seed name pairs within each iteration or at the end of the bootstrapping.[34] Since very few iterations (most often only the initial one) were needed in our experiments, we ended up using mostly (and most often only) the initial seed name pairs, which are known to be correct; hence, scoring the seed name pairs seemed unnecessary.

Tables 8–10 show that the results of BOOT are consistently worse than the results of SP and SP*. As already noted, for most relations more than L templates had been produced at the end of the first iteration (with the initial seed name pairs) of BOOT. Additional iterations were used only for 5 relations of the M-PIRO ontology. Hence, the differences in the performance of BOOT compared to SP and SP* are almost entirely due to the fact that BOOT uses the confidence function

33. This function is from Jurafsky & Marting (2008), and is based on a similar function of Riloff & Jones (1999). Unlike Jurafsky & Martin, we count NP anchor pairs rather than seed name pairs in $hits(t)$ and $misses(t)$, placing more emphasis on extracting all the NPs (of the retrieved documents of R) that correspond to the seed names.

34. Consult Chapter 22 of Jurafsky & Martin (2008) for a broader discussion of bootstrapping template extraction methods. See also `http://nlp.stanford.edu/software/patternslearning.shtml` for a publicly available bootstrapping template extraction system (SPIED), which supports however only templates with a single slot.

of Eq. 14 instead of the MaxEnt classifier (and the coverage of the sentence plans, in the case of SP*). Hence, the MaxEnt classifier has an important contribution in the performance of SP and SP*. Tables 8–10 show that this contribution is large in all three ontologies, despite the fact that the classifier was trained on Wine Ontology relations only (but with texts from Wikipedia).

5.3.5 Evaluating sentence plans in generated texts

To examine how sentence plans produced by different methods affect the perceived quality of generated texts, we showed automatically generated texts describing individuals and classes of the three ontologies to six computer science students, the same students as in the experiments of Section 5.2.4. We used six configurations of NaturalOWL in this experiment. The NO-SP configuration is given no sentence plans; in this case, NaturalOWL automatically produces sentence plans by tokenizing the OWL identifiers of the relations, acting like a simple verbalizer.[35] MANUAL-SP uses manually authored sentence plans. AUTO-SP* uses the SP* method (Section 4.4) with no human selection of sentence plans, i.e., the top-ranked sentence plan of each relation. We did not consider SP in this experiment, since the previous experiments indicated that SP* was overall better. In SEMI-AUTO-SP*, a human inspector (the first author) selected the best sentence plan of each relation among the five top-ranked sentence plans of SP*. Similarly, AUTO-BOOT and SEMI-AUTO-BOOT use the BOOT baseline of Section 5.3.4 with no human selection or with a human selecting among the top five, respectively. Apart from the sentence plans, all six configurations use the same, manually authored other types of linguistic resources (e.g., NL names, interest scores, text plans to order the message triples). Below are example texts generated from the three ontologies by the six configurations.

MANUAL-SP: This is a moderate, dry Zinfandel. It has a full body. It is made by Elyse in the Napa County.
SEMI-AUTO-SP*: This is a full, dry Zinfandel. It is moderate. It is made at Elyse in the Napa County.
SEMI-AUTO-BOOT: This is a full, dry Zinfandel. It is moderate. Elyse produced it and the Napa County is Home to it.
AUTO-SP*: This is a full, dry Zinfandel. It is moderate. It is made at Elyse and it is the Napa County.
AUTO-BOOT: This is a full, dry Zinfandel. It is moderate. It is Elyse and the Napa County.
NO-SP: This is a Zinfandel. It has sugar dry, it has body full and it has flavor moderate. It has maker Elyse and it located in the Napa County.

MANUAL-SP: This is a kantharos, created during the Hellenistic period and it originates from Amphipolis. Today it is exhibited in the Archaeological Museum of Kavala.
SEMI-AUTO-SP*: This is a kantharos, produced during the Hellenistic period and almost certainly from Amphipolis. It is found in the Archaeological Museum of Kavala.
SEMI-AUTO-BOOT: This is a kantharos, produced during the Hellenistic period and almost certainly from Amphipolis. It is among the Archaeological Museum of Kavala.
AUTO-SP*: This is a kantharos that handles from the Hellenistic period and is almost certainly from Amphipolis. It derives from the Archaeological Museum of Kavala.
AUTO-BOOT: This is a kantharos that is the Hellenistic period and is Amphipolis. It is the Archaeological Museum of Kavala.
NO-SP: This is a kantharos. It creation period the Hellenistic period, it original location Amphipolis and it current location the Archaeological Museum of Kavala.

MANUAL-SP: Molluscum contagiosum is a kind of viral infectious disease that affects the skin. It results in infections and its symptom is lesions. It is transmitted by fomites and contact with the skin, and it is caused by the molluscum contagiosum virus.
SEMI-AUTO-SP*: Molluscum contagiosum is a kind of viral infectious disease that can occur in the skin.

35. If the ontology provides an `rdfs:label` for a relation, NO-SP uses its tokens.

Infections are caused by molluscum contagiosum. Molluscum contagiosum often causes lesions. It is transmissible by fomites and contact with the skin, and it is caused by the molluscum contagiosum virus.
SEMI-AUTO-BOOT: Molluscum contagiosum is a kind of viral infectious disease that occurs when the skin. Infections are caused by molluscum contagiosum. Molluscum contagiosum can cause lesions. Fomites and contact with the skin can transmit molluscum contagiosum. Molluscum contagiosum is caused by the molluscum contagiosum virus.
AUTO-SP*: Molluscum contagiosum is a kind of viral infectious disease that is the skin. It is infections. It is lesions. It is fomites and contact with the skin, and it is caused by the molluscum contagiosum virus.
AUTO-BOOT: Molluscum contagiosum is a kind of viral infectious disease that is the skin. It is infections. It is lesions. It is fomites, the molluscum contagiosum virus and contact with the skin.
NO-SP: Molluscum contagiosum is a kind of viral infectious disease and it located in the skin. It results in infections. It has symptom lesions. It transmitted by fomites and contact with the skin, and it has material basis in the molluscum contagiosum virus.

As in the corresponding experiment with NL names (Section 5.2.4), we generated $95 \times 6 = 570$ texts from the Wine ontology (this time with six configurations), $49 \times 6 = 294$ texts from the M-PIRO ontology, and $30 \times 6 = 180$ texts from the Disease ontology.[36] The students were asked to score each text by stating how strongly they agreed or disagreed with statements S_1–S_3 below; the non-redundancy criterion was not used in this experiment, because all six configurations used the same (manually authored) NL names and interest scores. Otherwise, the experimental setup was the same as in the corresponding experiment with NL names (Section 5.2.4).

(S_1) *Sentence fluency*: Each sentence of the text (on its own) is grammatical and sounds natural.

(S_2) *Clarity*: The text is easy to understand, provided that the reader is familiar with the terminology and concepts of the domain (e.g., historical periods, grape varieties, virus names).

(S_3) *Semantic correctness*: The text accurately conveys the information of the message triples.

Tables 12–14 show the scores of the six configurations of NaturalOWL, averaged over the texts of each ontology. For each criterion, the best scores are shown in bold. In each criterion (row), we detected no statistically significant differences between scores marked with the same superscript; all the other differences (in the same row) were statistically significant.[37] MANUAL-SP was the best overall configuration, as one would expect, but SEMI-AUTO-SP* performed only slightly worse, with no detected statistically significant difference between the two configurations in most cases. The only notable exception was the semantic correctness of the M-PIRO ontology, where the difference between SEMI-AUTO-SP* and MANUAL-SP was larger, because for some relations SEMI-AUTO-SP* produced sentence plans that did not correctly express the corresponding message triples, because of too few seeds. These findings confirm that SP* performs very well in a semi-automatic scenario. SEMI-AUTO-BOOT performed clearly worse than SEMI-AUTO-SP* in this scenario.

In the fully automatic scenario, with no human selection of sentence plans, AUTO-SP* was overall better than AUTO-BOOT, but still not good enough to be used in practice. The NO-SP configuration, which uses sentence plans constructed by tokenizing the identifiers of the OWL relations, obtained much lower sentence fluency and clarity scores than MANUAL-SP, which shows the importance of sentence plans in the perceived quality of the texts. The semantic correctness scores of NO-SP were also much lower than those of MANUAL-SP in the Wine and M-PIRO ontologies, but the difference was smaller (with no detected statistically significant difference) in the Disease Ontology, because

36. Compared to the Wine and Disease ontologies, the M-PIRO ontology provides fewer seeds, which leads to no sentence plans for some of its relations. For those relations, we relaxed the constraint of Section 4.1 that requires templates to be extracted from at least two sentences, but this did not always produce good sentence plans.

37. We performed ANOVA and post-hoc Tukey tests ($a = 0.05$) in the six scores of each criterion in each ontology. A post-hoc power analysis of the ANOVA values resulted in power values greater or equal to 0.95 in all cases.

tokenizing the OWL identifiers of the relations of the Disease Ontology (e.g., `:has_symptom`) leads to sentences that convey the correct information in most cases, even if the sentences are not particularly fluent and clear. The sentence fluency and clarity scores of NO-SP in the Disease Ontology were also higher, compared to the scores of NO-SP in the other two ontologies, for the same reason.

Criteria	NO-SP	AUTO-BOOT	SEMI-AUTO-BOOT	AUTO-SP*	SEMI-AUTO-SP*	MANUAL-SP
Sentence fluency	2.69	3.67^{1}	4.18^{2}	$3.88^{1,2}$	$\mathbf{4.71}^{3}$	$\mathbf{4.75}^{3}$
Clarity	$3.26^{1,2}$	2.89^{1}	4.28	3.47^{2}	$\mathbf{4.81}^{3}$	$\mathbf{4.90}^{3}$
Semantic correctness	$3.02^{1,2}$	2.71^{1}	4.27	3.21^{2}	$\mathbf{4.77}^{3}$	$\mathbf{4.93}^{3}$

Table 12: Human scores for the Wine Ontology with different methods to obtain sentence plans.

Criteria	NO-SP	AUTO-BOOT	SEMI-AUTO-BOOT	AUTO-SP*	SEMI-AUTO-SP*	MANUAL-SP
Sentence fluency	1.18	2.73	3.63^{1}	4.08^{1}	$\mathbf{4.67}^{2}$	$\mathbf{4.98}^{2}$
Clarity	2.39	1.67	3.12^{1}	3.28^{1}	$\mathbf{4.65}^{2}$	$\mathbf{5.00}^{2}$
Semantic correctness	2.96^{1}	1.90	2.90^{1}	2.86^{1}	4.08	**5.00**

Table 13: Human scores for the M-PIRO ontology with different methods to obtain sentence plans.

Criteria	NO-SP	AUTO-BOOT	SEMI-AUTO-BOOT	AUTO-SP*	SEMI-AUTO-SP*	MANUAL-SP
Sentence fluency	3.40^{1}	3.67^{1}	$\mathbf{4.07}^{1,2}$	3.77^{1}	$\mathbf{4.77}^{2}$	$\mathbf{4.83}^{2}$
Clarity	$3.80^{1,2}$	2.37^{1}	3.13	2.53^{2}	$\mathbf{4.73}^{3}$	$\mathbf{4.83}^{3}$
Semantic correctness	$\mathbf{4.10}^{1,2}$	2.40^{3}	$3.27^{1,3}$	2.63^{2}	$\mathbf{4.57}^{2}$	$\mathbf{4.83}^{2}$

Table 14: Human scores for the Disease Ontology with different methods to obtain sentence plans.

5.4 Joint experiments with extracted NL names and sentence plans

In a final set of experiments, we examined the effect of combining our methods that produce NL names and sentence plans. We experimented with four configurations of NaturalOWL. The AUTO configuration produces NL names using the method of Section 3; it then uses the most highly ranked NL name of each individual or class to produce seed name pairs, and invokes the SP* method of Section 4 to produce sentence plans; it then uses the most highly ranked sentence plan for each relation. SEMI-AUTO also produces NL names using the method of Section 3, but a human selects the best NL name of each individual or class among the five most highly ranked ones; the selected NL names are then used to produce seed name pairs, and the SP* method is invoked to produce sentence plans; a human then selects the best sentence plan of each relation among the five most highly ranked ones. The MANUAL configuration uses manually authored NL names and sentence plans. In the VERBALIZER configuration, no NL names and sentence plans are provided to NaturalOWL; hence, acting like a simple verbalizer, NaturalOWL produces NL names and sentence plans by tokenizing the OWL identifiers of individuals, classes, and relations.[38] Furthermore, MANUAL uses manually authored intrest scores, VERBALIZER uses no interest scores, whereas AUTO and SEMI-AUTO use interest scores obtained from the (top-ranked or selected) NL names using the method of Section 3.3. All the other linguistic resources (most notably, text plans) are the same (manually authored) across all four configurations. We did not experiment with the BOOT and SP sentence plan generation methods in this section, since SP* performed overall better in the previous experiments. Below are example texts generated from the three ontologies by the four configurations we considered.

38. Whenever the ontology provides an `rdfs:label` for an individual, class, or relation, VERBALIZER uses its tokens.

MANUAL: This is a moderate, dry Chenin Blanc. It has a full body. It is made by Foxen in the Santa Barbara County.
SEMI-AUTO: This is a full, dry Chenin Blanc wine. It is moderate. It is made at the Foxen Winery in the Santa Barbara region.
AUTO: This is dry Chenin Blanc and is the full. It is the moderate. It is made at Foxen and it is Santa Barbara.
VERBALIZER: Foxen Chenin Blanc is Chenin Blanc. It has sugar Dry, it has maker Foxen and it has body Full. It located in Santa Barbara Region and it has flavor Moderate.

MANUAL: This is a portrait that portrays Alexander the Great and it was created during the Roman period. It is made of marble and today it is exhibited in the Archaeological Museum of Thassos.
SEMI-AUTO: This is a portrait. It is thought to be Alexander the Great, it is produced during the Roman period and it was all hand carved from marble. It is found in the Archaeological Museum of Thasos.
AUTO: This is a portrait. It is thought to be Alexander the Great, it handles from Roman and it is marble. It derives from the Thasos Archaeological Museum.
VERBALIZER: Exhibit 14 is portrait. It exhibit portrays alexander the great. It creation period roman period. It made of marble. It current location thasos archaelogical.

MANUAL: Ebola hemorrhagic fever is a kind of viral infectious disease. Its symptoms are muscle aches, sore throat, fever, weakness, stomach pain, red eyes, joint pain, vomiting, headaches, rashes, internal and external bleeding, hiccups and diarrhea. It is transmitted by infected medical equipment, contact with the body fluids of an infected animal, and contaminated fomites and it is caused by Bundibugyo ebolavirus, Cote d'Ivoire ebolavirus, Sudan ebolavirus and Zaire ebolavirus.
SEMI-AUTO: An Ebola hemorrhagic fever is a kind of viral Infectious disease. It often causes a muscle ache, a sore throat, a fever, weakness, stomach pain, a red eye symptom, joint pain, vomiting, a headache, rash, a severe internal bleeding, hiccups and diarrhea. It is transmissible by contaminated medical equipment, the direct contact with infected animals, and the contaminated fomite and it is caused by the species Bundibugyo ebolavirus, the Côte d'Ivoire ebolavirus, the Sudan ebolavirus and the Zaire ebolavirus.
AUTO: An Ebola hemorrhagic fever is viral. It is a muscle aches, contaminated medical equipment, a sore throat, a fever, weakness, stomach pain, a red eye, joint pain, a vomiting, a headache, rash, a content internal bleeding symptom, a hiccups and diarrhea. It is caused by the Bundibugyo ebolavirus, a Côte d'Ivoire ebolavirus, the Sudan ebolavirus and the Zaire ebolavirus.
VERBALIZER: Ebola hemorrhagic fever is a kind of viral infectious disease. It has symptom muscle aches, sore throat, fever, weakness, stomach pain, red eyes, joint pain, vomiting, headache, rash, internal and external bleeding, hiccups and diarrhea. It transmitted by infected medical equipment, contact with the body fluids of an infected animal, and contaminated fomites and it has material basis in Bundibugyo ebolavirus, Cote d'Ivoire ebolavirus, Sudan ebolavirus and Zaire ebolavirus.

Again, some NL names and sentence plans of SEMI-AUTO and AUTO can be easily improved using the Protégé plug-in of NaturalOWL. For example, the sentence plan that reports the historical period of the exhibit in the second SEMI-AUTO example above can be easily modified to use the simple past tense ("was produced" instead of "is produced"). Nevertheless, we made no such improvements.

Apart from the configurations of NaturalOWL, the experimental setup was the same as in Sections 5.2.4 and 5.3.5. We generated $95 \times 4 = 380$ texts from the Wine ontology, $49 \times 4 = 196$ texts from the M-PIRO ontology, and $30 \times 4 = 120$ texts from the Disease ontology. The students were now asked to score each text for sentence fluency, clarity, semantic correctness, and non-redundancy, by stating how strongly they agreed or disagreed with statements S_1–S_4 of Section 5.2.4.

Tables 15–17 show the scores of the four configurations of NaturalOWL, averaged over the texts of each ontology. For each criterion, the best scores are shown in bold. In each criterion (row), we detected no statistically significant differences between scores marked with the same superscript;

Criteria	VERBALIZER	AUTO	SEMI-AUTO	MANUAL
Sentence fluency	2.77	3.90	**4.67**[1]	**4.81**[1]
Clarity	3.57	2.79	**4.87**[1]	**4.93**[1]
Semantic correctness	3.57	2.86	**4.68**[1]	**4.97**[1]
Non-redundancy	3.20[1]	3.47[1]	**4.57**[2]	**4.79**[2]

Table 15: Human scores for the Wine Ontology, different methods for NL names, sentence plans.

Criteria	VERBALIZER	AUTO	SEMI-AUTO	MANUAL
Sentence fluency	2.10	4.33	**4.73**[1]	**4.96**[1]
Clarity	3.02[1]	3.49[1]	4.43	**5.00**
Semantic correctness	3.33[1]	2.90[1]	3.96	**5.00**
Non-redundancy	2.67	4.16[1]	**4.59**[1,2]	**5.00**[2]

Table 16: Human scores for the M-PIRO ontology, different methods for NL names, sentence plans.

all the other differences (in the same row) were statistically significant.[39] MANUAL had the best overall scores, as one would expect, but the scores of SEMI-AUTO were close, in most cases with no detected statistically significant difference, despite the combined errors of the methods that produce NL names and sentence plans. The biggest difference between SEMI-AUTO and MANUAL was in the semantic correctness criterion of the M-PIRO ontology. This difference is mostly due to the fact that SEMI-AUTO did not always manage to produce sentence plans to convey correctly the semantics of the message triples, because of too few seeds, as in the experiments of the previous section. This also affected the clarity score of SEMI-AUTO in the M-PIRO ontology. The scores of AUTO were much lower, again indicating that our methods cannot be used in a fully automatic scenario.

The scores of VERBALIZER were overall much lower than those of MANUAL, again showing the importance of linguistic resources when generating texts from ontologies. The high non-redundancy score of VERBALIZER in the Disease Ontology is due to the fact that there are very few redundant message triples and no anonymous individuals or classes in the Disease Ontology. Hence, VERBALIZER, which treats all the message triples as important and does not anonymize any individuals or classes performs well in terms of non-redundancy. We made a similar observation in Section 5.2.4.

6. Related work

Simple ontology verbalizers (Cregan et al., 2007; Kaljurand & Fuchs, 2007; Schwitter et al., 2008; Halaschek-Wiener et al., 2008; Schutte, 2009; Power & Third, 2010; Power, 2010; Schwitter, 2010; Liang et al., 2011) typically produce texts describing individuals and classes without requiring manually authored domain-dependent linguistic resources. They usually tokenize the OWL identifiers or labels (e.g., `rdfs:label`) of the individuals or classes to obtain NL names and sentence plans. Androutsopoulos et al. (2013) showed that the texts of the SWAT verbalizer (Stevens et al., 2011; Williams et al., 2011), one of the best publicly available verbalizers, are perceived as being of significantly lower quality compared to texts generated by NaturalOWL with domain-dependent linguistic resources; NL names, sentence plans, and (to a lesser extent) text plans were found to contribute most to this difference. Without domain-dependent linguistic resources, NaturalOWL was found to generate texts of the same quality as the SWAT verbalizer.

NaturalOWL is based on ideas from ILEX (O'Donnell, Mellish, Oberlander, & Knott, 2001) and M-PIRO (Isard et al., 2003; Androutsopoulos et al., 2007). Excluding simple verbalizers, it is the only publicly available NLG system for OWL, which is why we based our work on it. Nevertheless, its

39. Again, we performed ANOVA and post-hoc Tukey tests ($a = 0.05$) in the four scores of each criterion in each ontology. A post-hoc power analysis of the ANOVA values resulted in power values equal to 1.0 in all cases.

Criteria	VERBALIZER	AUTO	SEMI-AUTO	MANUAL
Sentence fluency	3.20[1]	3.00[1]	**4.33**[2]	**4.86**[2]
Clarity	3.77	2.13	**4.47**[1]	**4.90**[1]
Semantic correctness	3.77[1]	2.43	**4.20**[1,2]	**4.93**[2]
Non-redundancy	**4.20**[1,2]	3.43[1]	**4.37**[2]	**4.70**[2]

Table 17: Human scores for the Disease Ontology, different methods for NL names, sentence plans.

processing stages and linguistic resources are typical of NLG systems (Reiter & Dale, 2000; Mellish et al., 2006). Hence, we believe that our work is also applicable, at least in principle, to other NLG systems. For example, ONTOSUM (Bontcheva, 2005), which generates natural language descriptions of individuals, but apparently not classes, from RDF SCHEMA and OWL ontologies, uses similar processing stages, and linguistic resources corresponding to NL names and sentence plans. Reiter et al. (2003) discuss the different types of knowledge that NLG systems require and the difficulties of obtaining them (e.g., by interviewing experts or analyzing corpora). Unlike Reiter et al., we assume that domain knowledge is already available, in the form of OWL ontologies. The domain-specific linguistic resources of NaturalOWL belong in the 'domain communication knowledge' of Reiter et al., who do not describe particular corpus-based algorithms to acquire knowledge.

Ngonga Ngomo et al. (2013) discuss SPARQL2NL, a system that translates SPARQL queries to English. SPARQL2NL uses techniques similar to those of simple ontology verbalizers. To express the RDF triples $\langle S, R, O\rangle$ that are involved in a SPARQL query, it assumes that the labels (e.g., `rdfs:label`, perhaps also identifiers) of the relations are verbs or nouns. It determines if a relation label is a verb or noun using hand-crafted rules and the POS tags of the label's synonyms in WordNet (Fellbaum, 1998). It then employs manually authored templates, corresponding to our sentence plans, to express the relation; e.g., the template "S writes O" is used for a triple involving the relation `:write`, since "write" is a verb, but "S's author is O" is used for the relation `:author`, since "author" is a noun. To express the S or O of a triple, SPARQL2NL tokenizes the label (or identifier) of the corresponding individual or class, pluralizing the resulting name if it refers to a class.

Ratnaparkhi (2000) aims to express a set of attribute-value pairs as a natural language phrase; e.g., {*city-from* = Athens, *city-to* = New York, *depart-day* = Wednesday} becomes "flights from Athens to New York on Wednesday". A parallel training corpus containing sets of attribute-value pairs, the corresponding phrases, and their dependency trees is required. A maximum entropy model is trained on the corpus, roughly speaking to be able to estimate the probability of a dependency tree given a set of attribute-value pairs. Then, given an unseen set of attribute-value pairs, multiple alternative dependency trees are constructed in a top-down manner, using beam search and the maximum entropy model to estimate the probabilities of the trees being constructed. The most probable tree that expresses all the attribute-value pairs is eventually chosen, and the corresponding phrase is returned. In later work (Ratnaparkhi, 2002), the generated dependency trees are further altered by a set of hand-crafted rules that add unmentioned attributes, and the trees are also ranked by language models. In our case, where we aim to express multiple message triples $\langle S, R_i, O_i\rangle$ all describing an individual or class S, we can think of the message triples as attribute-value pairs $R_i = O_i$. To apply the methods of Ratnaparkhi, however, a parallel training corpus with sets of attribute-value pairs (or message triples) and the corresponding target texts would be needed; and corpora of this kind are difficult to obtain. By contrast, our methods require no parallel corpus and, hence, can be more easily applied to ontologies of new domains. Furthermore, the methods of Ratnaparkhi aim to produce a single sentence per set of attribute-value pairs, whereas we produce linguistic resources that are used to generate multi-sentence texts (e.g., our NL names and sentence plans include annotations used in sentence aggregation and referring expression generation).

Angeli et al. (2010) generate multi-sentence texts describing database records. Their methods also require a parallel training corpus consisting of manually authored texts and the database records (and particular record fields) expressed by each text. The generative model of Liang et al. (2009) is applied to the training corpus to align the words of each text to the database records and fields it expresses. Templates are then extracted from the aligned texts, by replacing words aligned to record fields with variables. To generate a new text from a set of database records, the system generates a sequence of phrases. For each phrase, it first decides which records and fields to express, then which templates to generate the phrase with, and finally which template variables to replace by which record fields. These decisions are made either greedily or by sampling probability distributions learnt during training. This process is repeated until all the given record fields have been expressed. A language model is also employed to ensure that the transitions between phrases sound natural. As with the work of Ratnaparkhi, the methods of Angeli et al. could in principle be applied to express message triples describing an individual or class, but again a parallel training corpus containing texts and the database records and fields expressed by each text would be needed.

Wong and Mooney (2006, 2007) employ Statistical Machine Translation (SMT) methods to automatically obtain formal semantic representations from natural language sentences. They automatically construct a synchronous context-free grammar, by applying a statistical word alignment model to a parallel training corpus of sentences and their semantic representations. The grammar generates both natural language sentences and their semantic representations. Given a new sentence, the grammar produces candidate semantic representations, and a maximum-entropy model estimates the probability of each canidate representation. Chen and Mooney (2008) use the same methods in the reverse direction, to convert formal semantic representations to single sentences. In principle, similar SMT methods could be employed to generate sentences from message triples. However, a parallel corpus of texts and message triples would again be needed. Furthermore, SMT methods produce a single sentence at a time, whereas our work concerns multi-sentence texts.

Lu et al. (2009) generate natural language sentences from tree-structured semantic representations (Lu, Ng, Lee, & Zettlemoyer, 2008). Given a parallel training corpus of sentences and tree-structured semantic representations, hybrid trees are created by expanding the original semantic representation trees of the corpus with nodes standing for phrases of the corresponding sentences. To generate a new sentence from a tree-structured semantic representation, a set of candidate hybrid trees is initially produced based on predefined tree patterns and a CRF model trained on the hybrid trees of the parallel corpus. A sentence is then obtained from the most probable candidate hybrid tree. In later work, Lu and Ng (2011) extend their hybrid trees to support formal logic (typed lambda calculus) semantic representations. A synchronous context free grammar is obtained from the extended hybrid trees of the parallel corpus. The grammar is then used to map formal logic expressions to new sentences. We note that OWL is based on description logic (Baader et al., 2002) and, hence, methods similar to those of Lu et al. could in principle be used to map OWL statements to sentences, though the hybrid trees would have to be modified for description logic. A parallel training corpus of texts and description logic expressions (or corresponding OWL statements) would again be needed, however, and only single sentences would be obtained.

Konstas and Lapata (2012b) use a probabilistic context-free grammar to convert a given set of database entries to a single sentence (or phrase). Roughly speaking, in each parse tree of the grammar, the leaves are the words of a sentence, and the internal nodes indicate which database entries are expressed by each subtree. The grammar is constructed using hand-crafted templates of rewrite rules and a parallel training corpus of database entries and sentences; a generative model based on the work of Liang et al. (2009) is employed to estimate the probabilities of the grammar. Subsequently, all the parse trees of the grammar for the sentences of the training corpus and the corresponding database entries are represented as a weighted directed hypergraph (Klein &

Manning, 2002). The hypergraph's weights are estimated using the inside-outside algorithm (Li & Eisner, 2009) on the training corpus. Following Huang and Chiang (2007), the hypergraph nodes are then integrated with an n-gram language model trained on the sentences of the corpus. Given a new set of database entries, the most probable derivation is found in the hypergraph using a k-best Viterbi search with cube pruning (Chiang, 2007) and the final sentence is obtained from the derivation. In later work, Konstas and Lapata (2012a) find the most probable derivation in the hypergraph by forest reranking, using features that include the decoding probability of the derivation according to their previous work, the frequency of rewrite rules in the derivation, as well as lexical (e.g., word n-grams) and structural features (e.g., n-grams of record fields). The weights of the features are estimated with a structured perceptron (Collins, 2002) on the training corpus.

Apart from simple verbalizers, all the other related methods discussed above require a parallel training corpus of texts (or sentences, or phrases) and their semantic representations, unlike our work. A further difference from our work is that all the previous methods assume that the English names of the various entities (or individuals and classes) are already available in the semantic representations of the texts to be generated, or that they can be directly obtained from the identifiers of the entities in the semantic representations. By contrast, we also proposed methods to produce appropriate NL names for individuals and classes, and we showed experimentally (Section 5.2.4) that without NL names the perceived quality of the generated texts is significantly lower.

Our sentence plan generation method contains a template extraction stage (Section 4.1), which is similar to methods proposed to automatically obtain templates that extract instances of particular relations from texts. We discussed bootstrapping in Section 5.3.4. Xu et al. (2007) adopt a similar bootstrapping approach with templates obtained from dependency trees. Bootstrappoing has also been used to obtain paraphrasing and textual entailment rules (Szpektor, Dagan, & Coppola, 2004; Androutsopoulos & Malakasiotis, 2010). The sentence plans we produce are not just templates (e.g., "X bought Y"), but include additional annotations (e.g., POS tags, agreement, voice, tense, cases). Furthermore, they are not intended to capture *all* the alternative natural language expressions that convey a particular relation, unlike information extraction, paraphrase and textual entailment recognition; our goal is to obtain a single sentence plan per relation that leads to high quality texts.

Bootstrapping approaches have also been used to obtain templates that extract named entities of a particular semantic class (e.g., person names) from texts (Riloff, 1996; Patwardhan & Riloff, 2006). Methods of this kind aim to extract *all* the named entities of a particular class from a corpus. By contast, we aim to assign a single high quality NL name to each individual or class of a given ontology. Furthermore, our NL names are not simply strings, but contain additional information (e.g., head, gender, number, agreement) that helps produce high quality texts.

7. Conclusions and future work

Concept-to-text generation systems typically require domain-specific linguistic resources to produce high quality texts, but manually constructing these resources can be tedious and costly. Focusing on NaturalOWL, a publicly available state of the art natural language generator for OWL ontologies, we proposed methods to automatically or semi-automatically extract from the Web sentence plans and natural language names, two of the most important types of domain-specific generation resources.[40] We showed experimentally that texts generated using linguistic resources produced by our methods in a semi-automatic manner, with minimal human involvement, are perceived as being almost as good as texts generated using manually authored linguistic resources, and much better than texts

40. The software of this article has been embedded in NaturalOWL and will be available from `http://nlp.cs.aueb.gr/software.html`.

produced by using linguistic resources extracted from the relation and entity identifiers of the ontologies. Using our methods, constructing sentence plans and natural language names requires human effort of a few minutes or hours, respectively, per ontology, whereas constructing them manually from scratch is typically a matter of days. Also, our methods do not require any familiarity with the internals of NaturalOWL and the details of its linguistic resources. Furthermore, unlike previous related work, no parallel corpus of sentences and semantic representations is required. On the downside, our methods do not perform sufficiently well in a fully-automatic scenario, with no human involvement during the construction of the linguistic resources.

The processing stages and linguistic resources of NaturalOWL are typical of NLG systems. Hence, we believe that our work is also applicable, at least in principle, to other NLG systems. Our methods may also be useful in simpler ontology verbalizers, where the main concern seems to be to avoid manually authoring domain-specific linguistic resources, currently at the expense of producing texts of much lower quality. Future work could aim to improve our methods to allow using them in a fully automatic manner. Further work could also explore how other kinds of domain-specific linguistic resources for NLG, most importantly text plans, could be constructed automatically or semi-automatically. Another future goal might be to consider languages other than English.

References

Androutsopoulos, I., Lampouras, G., & Galanis, D. (2013). Generating natural language descriptions from OWL ontologies: the NaturalOWL system. *Journal of Artificial Intelligence Research*, *48*(1), 671–715.

Androutsopoulos, I., & Malakasiotis, P. (2010). A survey of paraphrasing and textual entailment methods. *Journal of Artificial Intelligence Research*, *38*(1), 135–187.

Androutsopoulos, I., Oberlander, J., & Karkaletsis, V. (2007). Source authoring for multilingual generation of personalised object descriptions. *Natural Language Engineering*, *13*(3), 191–233.

Angeli, G., Liang, P., & Klein, D. (2010). A simple domain-independent probabilistic approach to generation. In *Conference on Empirical Methods in Natural Language Processing*, pp. 502–512, Cambridge, MA.

Antoniou, G., & van Harmelen, F. (2008). *A Semantic Web primer* (2nd edition). MIT Press.

Baader, F., Calvanese, D., McGuinness, D., Nardi, D., & Patel-Schneider, P. (Eds.). (2002). *The Description Logic Handbook*. Cambridge University Press.

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The Semantic Web. *Scientific American*, *May*(1), 34–43.

Bontcheva, K. (2005). Generating tailored textual summaries from ontologies. In *2nd European Semantic Web Conference*, pp. 531–545, Heraklion, Greece.

Chen, D. L., & Mooney, R. J. (2008). Learning to sportscast: a test of grounded language acquisition. In *25th International Conference on Machine Learning*, pp. 128–135, Helsinki, Finland.

Chiang, D. (2007). Hierarchical phrase-based translation. *Computational Linguistics*, *33*(2), 201–228.

Collins, M. (2002). Discriminative training methods for Hidden Markov Models: Theory and experiments with Perceptron algorithms. In *Conference on Empirical Methods in Natural Language Processing*, pp. 1–8, Stroudsburg, PA, USA.

Cregan, A., Schwitter, R., & Meyer, T. (2007). Sydney OWL syntax – towards a controlled natural language syntax for OWL. In *OWL: Experiences and Directions Workshop*, Innsbruck, Austria.

Fellbaum, C. (Ed.). (1998). *WordNet: an electronic lexical database*. MIT Press.

Galanis, D., & Androutsopoulos, I. (2007). Generating multilingual descriptions from linguistically annotated OWL ontologies: the NaturalOWL system. In *11th European Workshop on Natural Language Generation*, pp. 143–146, Schloss Dagstuhl, Germany.

Galanis, D., Karakatsiotis, G., Lampouras, G., & Androutsopoulos, I. (2009). An open-source natural language generator for OWL ontologies and its use in Protégé and Second Life. In *12th Conference of the European Chapter of ACL (demos)*, pp. 17–20, Athens, Greece.

Gatt, A., & Reiter, E. (2009). SimpleNLG: A realisation engine for practical applications. In *12th European Workshop on Natural Language Generation*, pp. 90–93, Athens, Greece.

Grau, B., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P., & Sattler, U. (2008). OWL 2: The next step for OWL. *Web Semantics*, *6*(4), 309–322.

Gupta, S., & Manning, C. D. (2014). Improved pattern learning for bootstrapped entity extraction. In *18th Conference on Computational Natural Language Learning*, pp. 98–108, Ann Arbor, Michigan.

Halaschek-Wiener, C., Golbeck, J., Parsia, B., Kolovski, V., & Hendler, J. (2008). Image browsing and natural language paraphrases of semantic web annotations. In *1st International Workshop on Semantic Web Annotations for Multimedia*, Tenerife, Spain.

Horrocks, I., Patel-Schneider, P., & van Harmelen, F. (2003). From SHIQ and RDF to OWL: the making of a Web Ontology Language. *Web Semantics*, *1*(1), 7–26.

Huang, L., & Chiang, D. (2007). Forest rescoring: Faster decoding with integrated language models. In *45th Annual Meeting of ACL*, pp. 144–151, Prague, Czech Republic.

Isard, A., Oberlander, J., Androutsopoulos, I., & Matheson, C. (2003). Speaking the users' languages. *IEEE Intelligent Systems*, *18*(1), 40–45.

Jurafsky, D., & Martin, J. (2008). *Speech and Language Processing*. Prentice Hall.

Kaljurand, K., & Fuchs, N. (2007). Verbalizing OWL in Attempto Controlled English. In *3rd International Workshop on OWL: Experiences and Directions*, Innsbruck, Austria.

Klein, D., & Manning, C. (2002). Parsing and hypergraphs. In Bunt, H., Carroll, J., & Satta, G. (Eds.), *New Developments in Parsing Technology*, pp. 351–372. Kluwer Academic Publishers.

Konstas, I., & Lapata, M. (2012a). Concept-to-text generation via discriminative reranking. In *50th Annual Meeting of ACL*, pp. 369–378, Jeju Island, Korea.

Konstas, I., & Lapata, M. (2012b). Unsupervised concept-to-text generation with hypergraphs. In *Conference on Human Language Technology of Annual Conference of the North American Chapter of ACL*, pp. 752–761, Montréal, Canada.

Li, Z., & Eisner, J. (2009). First- and second-order expectation semirings with applications to minimum-risk training on translation forests. In *Conference on Empirical Methods in Natural Language Processing*, pp. 40–51, Singapore.

Liang, P., Jordan, M., & Klein, D. (2009). Learning semantic correspondences with less supervision. In *47th Meeting of ACL and 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*, pp. 91–99, Suntec, Singapore.

Liang, S., Stevens, R., Scott, D., & Rector, A. (2011). Automatic verbalisation of SNOMED classes using OntoVerbal. In *13th Conference on Artificial Intelligence in Medicine*, pp. 338–342, Bled, Slovenia.

Lu, W., & Ng, H. T. (2011). A probabilistic forest-to-string model for language generation from typed lambda calculus expressions. In *Conference on Empirical Methods in Natural Language Processing*, pp. 1611–1622, Edinburgh, UK.

Lu, W., Ng, H. T., & Lee, W. S. (2009). Natural language generation with tree conditional random fields. In *Conference on Empirical Methods in Natural Language Processing*, pp. 400–409, Singapore.

Lu, W., Ng, H. T., Lee, W. S., & Zettlemoyer, L. S. (2008). A generative model for parsing natural language to meaning representations. In *Conference on Empirical Methods in Natural Language Processing*, pp. 783–792, Honolulu, Hawaii.

Manning, C., & Schutze, H. (2000). *Foundations of Statistical Natural Language Processing*. MIT Press.

Mellish, C., & Pan, J. (2008). Natural language directed inference from ontologies. *Artificial Intelligence*, *172*(10), 1285–1315.

Mellish, C., Scott, D., Cahill, L., Paiva, D., Evans, R., & Reape, M. (2006). A reference architecture for natural language generation systems. *Natural Language Engineering*, *12*(1), 1–34.

Mellish, C., & Sun, X. (2006). The Semantic Web as a linguistic resource: opportunities for natural language generation. *Knowledge Based Systems*, *19*(5), 298–303.

Muslea, I. (1999). Extraction patterns for information extraction tasks. In *AAAI Workshop on Machine Learning for Information Extraction*, pp. 1–6, Orlando, Florida.

Ngonga Ngomo, A.-C., Bühmann, L., Unger, C., Lehmann, J., & Gerber, D. (2013). Sorry, I don't speak SPARQL: Translating SPARQL queries into natural language. In *22nd International Conference on World Wide Web*, pp. 977–988, Rio de Janeiro, Brazil.

O'Donnell, M., Mellish, C., Oberlander, J., & Knott, A. (2001). ILEX: an architecture for a dynamic hypertext generation system. *Natural Language Engineering*, *7*(3), 225–250.

Patwardhan, S., & Riloff, E. (2006). Learning domain-specific information extraction patterns from the Web. In *ACL Workshop on Information Extraction Beyond the Document*, pp. 66–73, Sydney, Australia.

Power, R. (2010). Complexity assumptions in ontology verbalisation. In *48th Annual Meeting of ACL (short papers)*, pp. 132–136, Uppsala, Sweden.

Power, R., & Third, A. (2010). Expressing OWL axioms by English sentences: Dubious in theory, feasible in practice. In *23rd International Conf. on Computional Linguistics*, pp. 1006–1013, Beijing, China.

Ratnaparkhi, A. (2000). Trainable methods for surface natural language generation. In *1st Annual Conference of the North American Chapter of ACL*, pp. 194–201, Seattle, WA.

Ratnaparkhi, A. (2002). Trainable approaches to surface natural language generation and their application to conversational dialog systems. *Computer Speech & Language*, *16*(3-4), 435–455.

Rector, A., Drummond, N., Horridge, M., Rogers, J., Knublauch, H., Stevens, R., Wang, H., & Wroe, C. (2004). OWL pizzas: Practical experience of teaching OWL-DL: Common errors & common patterns. In *14th Int. Conf. on Knowledge Eng. & Knowledge Management*, pp. 63–81, Northamptonshire, UK.

Reiter, E., & Dale, R. (2000). *Building Natural Language Generation systems*. Cambridge University Press.

Reiter, E., Sripada, S., & Robertson, R. (2003). Acquiring correct knowledge for natural language generation. *Journal of Artificial Intelligence Research*, *18*, 491–516.

Riloff, E. (1996). Automatically generating extraction patterns from untagged text. In *13th National Conference on Artificial Intelligence - Volume 2*, pp. 1044–1049, Portland, Oregon.

Riloff, E., & Jones, R. (1999). Learning dictionaries for information extraction by multi-level bootstrapping. In *16th National Conference on Artificial Intelligence and the 11th Innovative Applications of Artificial Intelligence Conference*, pp. 474–479, Orlando, Florida, USA.

Schutte, N. (2009). Generating natural language descriptions of ontology concepts. In *12th European Workshop on Natural Language Generation*, pp. 106–109, Athens, Greece.

Schwitter, R. (2010). Controlled natural languages for knowledge representation. In *23rd International Conference on Computational Linguistics (posters)*, pp. 1113–1121, Beijing, China.

Schwitter, R., Kaljurand, K., Cregan, A., Dolbear, C., & Hart, G. (2008). A comparison of three controlled natural languages for OWL 1.1. In *4th OWL: Experiences and Directions Workshop*, Washington DC.

Shadbolt, N., Berners-Lee, T., & Hall, W. (2006). The Semantic Web revisited. *IEEE Intelligent Systems*, *21*(3), 96–101.

Stevens, R., Malone, J., Williams, S., Power, R., & Third, A. (2011). Automatic generation of textual class definitions from OWL to English. *Biomedical Semantics*, *2*(S 2:S5).

Szpektor, I.and Tanev, H., Dagan, I., & Coppola, B. (2004). Scaling web-based acquisition of entailment relations. In *Conf. on Empirical Methods in Natural Language Processing*, pp. 41–48, Barcelona, Spain.

Williams, S., Third, A., & Power, R. (2011). Levels of organization in ontology verbalization. In *13th European Workshop on Natural Language Generation*, pp. 158–163, Nancy, France.

Wong, Y. W., & Mooney, R. J. (2006). Learning for semantic parsing with statistical machine translation. In *Conf. on Human Lang. Technology of the Annual Conf. of ACL*, pp. 439–446, New York, New York.

Wong, Y. W., & Mooney, R. J. (2007). Learning synchronous grammars for semantic parsing with lambda calculus. In *45th Annual Meeting of ACL*, pp. 960–967, Prague, Czech Republic.

Xu, F., Uszkoreit, H., & Li, H. (2007). A seed-driven bottom-up machine learning framework for extracting relations of various complexity. In *45th Annual Meeting of ACL*, pp. 584–591, Prague, Czech Republic.