Producing Compact Texts with Integer Linear Programming in Concept-to-Text Generation

Gerasimos Lampouras*
Department of Informatics, Athens
University of Economics and Business

Ion Androutsopoulos**
Department of Informatics, Athens
University of Economics and Business

Concept-to-text generation typically employs a pipeline architecture, which often leads to suboptimal texts. Content selection, for example, may greedily select the most important facts, which may require, however, too many words to express, and this may be undesirable when space is limited or expensive. Selecting other facts, possibly only slightly less important, may allow the lexicalization stage to use much fewer words, or to report more facts in the same space. Decisions made during content selection and lexicalization may also lead to more or fewer sentence aggregation opportunities, affecting the length and readability of the resulting texts. Building upon on a publicly available state of the art natural language generator for Semantic Web ontologies, this article presents an Integer Linear Programming model that, unlike pipeline architectures, jointly considers choices available in content selection, lexicalization, and sentence aggregation to avoid greedy local decisions and produce more compact texts, i.e., texts that report more facts per word. Compact texts are desirable, for example, when generating advertisements to be included in Web search results, or when summarizing structured information in limited space. An extended version of the proposed model also considers a limited form of referring expression generation and avoids redundant sentences. An approximation of the two models can be used when longer texts need to be generated. Experiments with three ontologies confirm that the proposed models lead to more compact texts, compared to pipeline systems, with no deterioration or with improvements in the perceived quality of the generated texts.

1. Introduction

The Semantic Web (Berners-Lee, Hendler, and Lassila 2001; Shadbolt, Berners-Lee, and Hall 2006) and the growing popularity of Linked Data (data that are published using Semantic Web technologies) have renewed interest in concept-to-text natural language generation (NLG), especially text generation from ontologies (Bontcheva 2005; Mellish and Sun 2006; Galanis and Androutsopoulos 2007; Mellish and Pan 2008; Schwitter et al. 2008; Schwitter 2010; Liang et al. 2011; Williams, Third, and Power 2011; Androutsopoulos, Lampouras, and Galanis 2013). An ontology provides a conceptualization of a knowledge domain (e.g., consumer electronics, diseases) 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, and van Harmelen 2003; Grau et al. 2008), a formal language based

* Department of Informatics, Athens University of Economics and Business, Patission 76, 104 34 Athens, Greece. E-mail: lampouras06@aueb.gr
** Department of Informatics, Athens University of Economics and Business, Patission 76, 104 34 Athens, Greece. E-mail: ion@aueb.gr

on description logics (Baader et al. 2002), RDF, and RDF SCHEMA (Antoniou and van Harmelen 2008).[1] Given an OWL ontology for a knowledge domain, one can publish on the Web machine-readable statements about the domain (e.g., available products, known diseases, their features or symptoms), with the statements having formally defined semantics based on the ontology. NLG can then produce texts describing classes or individuals of the ontology (e.g., product descriptions, information about diseases) from the same statements.[2] This way the same information becomes more easily accessible to both computers (which read the machine-readable statements) and end-users (who read the texts), which is one of the main goals of the Semantic Web.

NLG systems typically employ a pipeline architecture (Reiter and Dale 2000). Firstly, content selection chooses the logical facts (axioms, in the case of an OWL ontology) to be expressed in the text to be generated. The purpose of the next stage, text planning, ranges from simply ordering the facts to be expressed, in effect also ordering the sentences that will express them, to making more complex decisions about the rhetorical structure of the text. Lexicalization then selects the words and syntactic structures to express each fact as a single sentence. Sentence aggregation may then combine shorter sentences into longer ones. Another component generates appropriate referring expressions (pronouns, noun phrases etc.), and surface realization produces the final text, based on internal representations of the previous decisions. Each stage of the pipeline in effect performs a local optimization, constrained by decisions of the previous stages, and largely unaware of the consequences of its own decisions on the subsequent stages.

The pipeline architecture has engineering advantages (e.g., it is easier to specify and monitor the input and output of each stage), but produces texts that may be suboptimal, since the decisions of the generation stages are actually co-dependent (Danlos 1984; Marciniak and Strube 2005; Belz 2008). Content selection, for example, may greedily select the most important facts among those that are relevant to the purpose of the text, but these facts may require too many words to express, which may be undesirable when space is limited or expensive. Selecting other facts, possibly only slightly less important, may allow the lexicalization stage to use much fewer words, or to report more facts in the same space. Decisions made during content selection and lexicalization (facts to express, words and syntactic structures to use) may also lead to more or fewer sentence aggregation opportunities, affecting the length and readability of the texts. Some of these issues can be addressed by over-generating at each stage (e.g., producing several alternative sets of facts at the end of content selection, several alternative lexicalizations etc.) and employing a final ranking component to select the best combination (Walker, Rambow, and Rogati 2001). This over-generate and rank approach, however, may also fail to find an optimal solution, and generates an exponentially large number of candidate solutions when several components are pipelined.

In this article, we present an Integer Linear Programming (ILP) model that, unlike pipeline architectures, jointly considers choices available in content selection, lexicalization, and sentence aggregation to avoid greedy local decisions and produce more compact texts, i.e., texts that report more facts per word. Compact texts are desirable, for example, when generating short product descriptions to be included as advertisements

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 Following common practice in Semantic Web research, we often use the term 'ontology' to refer jointly to terminological knowledge (TBox statements) that establishes a conceptualization of a knowledge domain, and assertional knowledge (ABox statements) that describes particular individuals.

in Web search results (Thomaidou et al. 2013; Thomaidou 2014).[3] Question answering may also involve generating a natural language summary of facts (e.g., RDF triples) related to a question, without exceeding a maximum text length (Tsatsaronis et al. 2012); the more compact the summary, the more facts can be reported in the available space, increasing the chances of reporting the information sought by the user.[4] Compact texts are also desirable when showing texts on devices with small screens (Corston-Oliver 2001) or as subtitles (Vandeghinste and Y. 2004).[5]

If an importance score is available for each fact, our model can take it into account to maximize the total importance (instead of the total number) of the expressed facts per word. The model itself, however, does not produce importance scores; we assume that the scores are produced by a separate process (Barzilay and Lapata 2005; Demir, Carberry, and McCoy 2010), not included in our content selection. For simplicity, in the experiments of this article we treat all the facts as equally important. An extended version of our ILP model also considers a limited form of referring expression generation, where the best name must be chosen per individual or class among multiple alternatives. The extended model also avoids sentences that report information that is obvious (to humans) from the names of the individuals and classes (e.g., "A red wine is a kind of wine with red color"). Experiments with three OWL ontologies from very different knowledge domains (wines, consumer electronics, diseases) confirm that our models lead to more compact texts, compared to pipeline systems with the same components, with no deterioration or with improvements in the perceived quality of the generated texts. Although solving ILP problems is in general NP-hard (Karp 1972), off-the-shelf ILP solvers can be used. The available solvers guarantee finding a globally optimum solution, and they are very fast in practice in the ILP problems we consider, when the the number of available facts (per individual or class being described) is small. We also present an approximation of our ILP models, which is more efficient when the number of available facts is larger and longer texts need to be generated.

Our ILP models (and approximations) have been embedded in NaturalOWL (Androutsopoulos, Lampouras, and Galanis 2013), an NLG system for OWL, as alternatives to the system's original pipeline architecture. We base our work on NaturalOWL, because it is the only open-source NLG system for OWL that implements all the processing stages of a typical NLG system (Reiter and Dale 2000), it is extensively documented, and has been tested with several ontologies. The processing stages and linguistic resources of NaturalOWL are typical of NLG systems (Mellish et al. 2006). Hence, we believe that our work is, at least in principle, also applicable to other NLG systems. Our ILP models do not directly consider text planning, but rely on the (external to the ILP model) text planner of NaturalOWL. We hope to include more text planning and referring expression generation decisions directly in our ILP model in future work. We also do not consider surface realization, since it is not particularly interesting in NaturalOWL; all the decisions have in effect already been made by the time this stage is reached.

The remainder of this article is structured as follows. Section 2 below provides background information about NaturalOWL. Section 3 defines our ILP models. Section 4 discusses the computational complexity of our ILP models, along with the more ef-

3 See also http://www.google.com/ads/.
4 Consult also http://nlp.uned.es/clef-qa/ and http://www.bioasq.org/.
5 See also the smartphone application Acropolis Rock (http://acropolisrock.com/), which uses NaturalOWL to describe historical monuments; a video is available at https://www.youtube.com/watch?v=XMzdTir6Gas. The subtitles of the virtual museum guide of Galanis et al. (2009) are also generated by NaturalOWL; see the video at http://vimeo.com/801099.

ficient approximation that can be used when then number of available facts is large. Section 5 presents our experiments. Section 6 discusses previous related work. Section 7 concludes and proposes future work.

2. Background Information about NaturalOWL

NaturalOWL produces texts describing classes or individuals (entities) of an OWL ontology (e.g., descriptions of types of products or particular products). Given an OWL ontology and a particular target class or individual to describe, NaturalOWL first scans the ontology for OWL statements relevant to the target. If the target is the class `StEmilion`, for example, a relevant OWL statement may be the following.[6]

```
SubclassOf(:StEmilion
  ObjectIntersectionOf(:Bordeaux
    ObjectHasValue(:locatedIn :stEmilionRegion)
    ObjectHasValue(:hasColor :red)
    ObjectHasValue(:hasFlavor :strong)
    ObjectHasValue(:madeFrom :cabernetSauvignonGrape)
    ObjectMaxCardinality(1 :madeFrom)))
```

The statement above defines `StEmilion` as the intersection of: (i) the class of `Bordeaux` wines; (ii) the class of all individuals whose `locatedIn` property has (for each individual) `stEmilionRegion` among its values (OWL properties are generally many-valued); (iii)–(v) the classes of individuals whose `hasColor`, `hasFlavor`, and `madeFromGrape` property values include `red`, `strong`, and `cabernetSauvignonGrape`, respectively, without excluding wines that have additional values in these properties; and (vi) the class of individuals whose `madeFromGrape` property has exactly one value; hence, a St. Emilion wine is made *exclusively* from Cabernet Sauvignon grapes.

NaturalOWL 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.[7] 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. Message triples may also contain conjunctions or disjunctions as their O, as in `<:ColoradoTickFever, :hasSymptom, and(:fatigue, :headache, :myalgia)>`.[8] We use the terms 'fact' and 'message triple' as synonyms in the remainder of this article.

Having produced the message triples, NaturalOWL consults a user model to select the most important ones, 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* (NL names) for

6 This example is based on the Wine Ontology, one of the ontologies of our experiments (see Section 5).
7 For simplicity, we omit some details about message triples. Consult Androutsopoulos et al. (2013) for more information about message triples and their relation to RDF triples.
8 This example is from the Disease Ontology, another ontology used in our experiments.

the individuals and classes of the ontology. Rougly speaking, a sentence plan specifies how to generate a sentence to express a message triple involving a particular relation (R), whereas an NL name specifies how to generate a noun phrase to refer to a class or individual by name. We provide more information about sentence plans and NL names in the following subsections. If sentence plans and NL names are not supplied, NaturalOWL automatically produces them by tokenizing the OWL identifiers of the relations, individuals, and classes of the ontology, acting as a simple *ontology verbalizer* (Cregan, Schwitter, and Meyer 2007; Kaljurand and Fuchs 2007; Schwitter et al. 2008; Halaschek-Wiener et al. 2008; Schutte 2009; Power and Third 2010; Power 2010; Schwitter 2010; Stevens et al. 2011; Liang et al. 2011). The resulting texts, however, are of much lower quality (Androutsopoulos, Lampouras, and Galanis 2013). For example, the resulting text from the OWL statement above would be:

> 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.

By contrast, when appropriate sentence plans and NL names are provided, NaturalOWL produces the following text:

> 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.

In this article, we assume that appropriate sentence plans and NL names are supplied for each ontology. They can be manually constructed using a Protégé plug-in that accompanies NaturalOWL (Androutsopoulos, Lampouras, and Galanis 2013).[9] Semi-automatic methods can also be used to extract and rank candidate sentence plans and NL names from the Web, with a human selecting the best among the most highly ranked ones; in this case, it has been shown that high quality sentence plans and NL names can be constructed in a matter of a few hours (at most) per ontology (Lampouras 2015).

2.1 The Natural Language Names of NaturalOWL

In NaturalOWL, an NL name is a sequence of slots. The contents of the slots are concatenated to produce a noun phrase that names 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.[10]

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

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`,

9 Consult `http://protege.stanford.edu/`. NaturalOWL and its Protégé plug-in are available from `http://nlp.cs.aueb.gr/software.html`.

10 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.

`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 the triples `<:wine32, instanceOf, :TraditionalWinePiemonte>` and `<:wine32, instanceOf, :FamousWine>` were 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 "It 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 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. An 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:[11]

(1) *An article*, definite or indefinite, possibly to agree with a noun slot.
(2) *A noun flagged as the head*. The number of the head noun must also be specified.
(3) *An adjective flagged as the head*. For example, the NL name of the individual `:red` may consist of a single slot, to be filled in with the adjective 'red', which will also be the head of the NL name. The number and gender of the head adjective must be specified.
(4) *Any other noun or adjective*, (5) *a preposition*, or (6) *any fixed (canned) 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.[12] Multiple NL names can also be provided for the same individual or class, to produce more varied texts.

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 particular 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.

11 NaturalOWL also supports Greek. The possible annotations for Greek NL names (and sentence plans, see below) are slightly different, but in this article we consider only English NL names (and sentence plans).
12 We use SIMPLENLG (Gatt and Reiter 2009) to generate the inflectional forms of nouns, adjectives, 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`.

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

For 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 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 in plural), produce appropriate referring expressions (e.g., pronouns in the correct cases and genders), and aggregate shorter sentences into longer ones. For example, the 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 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:[13]

(1) *A referring expression for the S* (a.k.a. the *owner*) of the triple, in a particular case.

(2) *A verb* in a particular polarity and inflectional form (e.g., tense, voice), possibly to agree with another slot.

(3) *A noun or adjective* in a particular form, possibly to agree with another slot.

(4) *A preposition*, or (5) *a fixed string*.

(6) *A referring expression for the O* (a.k.a. the *filler*) of the triple, in a particular case.

Multiple sentence plans can be provided per relation, to produce more varied texts and increase sentence aggregation opportunities. Sentence plans for message triples that involve relation modifiers (e.g., <:StEmilion, maxCardinality(:madeFrom), 1>) are automatically produced from the sentence plans for the corresponding relations without modifiers (e.g., <:StEmilion, :madeFrom, :cabernetSauvignonGrape>).

2.3 Importance Scores

Some message triples can lead to sentences that sound redundant, because they report relations that are obvious (to humans) from the NL names of the individuals or classes, as in the sentence "A red wine is a kind of wine with red color". The sentence of our example reports the following two message triples:

<: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. 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 allows message triples to be assigned *importance scores* indicating how important (or interesting) it is to convey each message triple to different user types or particular users. Assigning a zero importance score to a message triple instructs NaturalOWL to avoid expressing it. The importance scores can be constructed manually or by invoking an external user modeling component (Androutsopoulos, Lampouras, and Galanis 2013). An additional mechanism of NaturalOWL assigns zero importance scores to message triples like the ones above, which report relations that are obvious from the NL names; this is achieved by using heuristics discussed elsewhere (Lampouras 2015). In the experiments of this article, we use the zero importance scores that NaturalOWL automatically assigns to some message triples, but we treat all the other message triples as equally important for simplicity.

3. Our Integer Linear Programming Models

We now discuss our Integer Linear Programming (ILP) models, starting from the first, simpler version, which considers choices available in content selection, lexicalization,

13 For simplicity, we omit some details and functionality of sentence plans that are not relevant to the work of this article. More details can be found elsewhere (Androutsopoulos, Lampouras, and Galanis 2013).

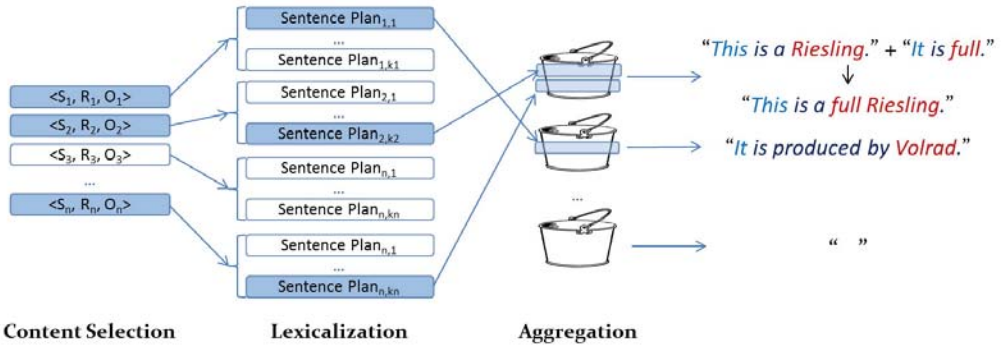

Figure 1
Illustration of the main decisions of our first ILP model.

and sentence aggregation. Figure 1 illustrates the main decisions of the model. For content selection, the model decides which of the available facts (message triples) should be expressed. For lexicalization, it decides which sentence plan should be used for each fact that will be expressed, assuming that multiple sentence plans are available per fact. For sentence aggregation, it decides which simple sentences (each reporting a single fact) should be aggregated to form longer sentences, by partitioning the simple sentences (or equivalently the message triples they express) into groups (shown as buckets in Fig. 1). After using the ILP model, the aggregation rules of NaturalOWL (Androutsopoulos, Lampouras, and Galanis 2013) are applied separately to the simple sentences of each group (bucket) to obtain a single aggregated sentence per group.[14] To keep the ILP model simpler, the model itself does not control which particular aggregation rules will be applied to each group. The number of groups (buckets) is fixed, equal to the maximum number of (aggregated) sentences that the model can generate per text. To avoid generating very long aggregated sentences, the number of simple sentences that can be placed in each group (bucket) cannot exceed a fixed upper limit (the same for all groups). Groups left empty produce no sentences.

Our second, extended ILP model is very similar, but also performs a limited form of referring expression generation by selecting among multiple alternative NL names; it also takes into account that using a particular NL name may make expressing some other facts redundant (Section 2.3). By contrast, the first, simpler ILP model assumes that a single NL name is available per individual and class (hence, no choice of NL names is needed) and does not try to avoid expressing redundant facts. In both models, a single (selected, or the only available one) NL name is picked per individual or class (unless the individual or class is marked as anonymous, see Section 2.1), and it is used throughout the particular text being generated. Neither of the two models considers other referring expression generation decisions (e.g., whether to use a pronoun or a demonstrative noun phrase like "this wine", as opposed to repeating the NL name of a wine). The existing referring expression generation component of NaturalOWL (Androutsopoulos, Lampouras, and Galanis 2013) is subsequently invoked (after using the ILP models)

14 The sentences of each group can always be aggregated, since they describe the same individual or class. If no better aggregation rule applies, a conjunction of the sentences in the group can be formed.

to decide if the picked NL name, a pronoun, or a demonstrative noun phrase should be used wherever a reference to an individual or class is needed in the text being generated.

A further limitation of our models is that they do not directly consider text planning, relying on the (external to the ILP models) text planner of NaturalOWL instead. The text planner is invoked (before using the ILP models) to partition the available message triples (the triples about the individual or class to be described) into *topical sections*; for example, message triples about the size, weight, and material of an electronic product may be placed in one section, and triples about the functions and features of the product in another one. This step is needed, because our ILP models never aggregate together sentences expressing facts from different topical sections, to avoid producing aggregated sentences that sound unnatural. The text planner is also invoked after using one of the ILP models, to order each group of simple sentences that the model has decided to aggregate. As already noted, each aggregated sentence is produced by applying the aggregation rules of NaturalOWL to a group (bucket) of simple sentences, but the rules presuppose that the simple sentences to be aggregated are already ordered, which is why the text planer is invoked at this point. After applying the aggregation rules to each group of (ordered) simple sentences, the text planner is also used to order the topical sections, and the (now aggregated) sentences within each section.

3.1 Our First ILP Model

Let us now focus on our first ILP model. As already noted, this model assumes that there is a single NL name per individual and class (excluding anonymous ones). Furthermore, the model assumes that all the NL names are short and approximately equally long.

Let $F = \{f_1, \ldots, f_n\}$ be the set of all the available facts f_i about the target individual or class S to be described. Recall that we use the term 'fact' as a synonym of 'message triple'. For each fact $f_i = \langle S, R_i, O_i \rangle$, we assume that a set $P_i = \{p_{i1}, p_{i2}, \ldots\}$ of alternative sentence plans is available; facts with the same relation (R_i) have the same set of sentence plans (P_i). Recall, also, that each sentence plan p_{ik} specifies how to express f_i as an alternative single sentence, and that a sentence plan is a sequence of slots, along with instructions specifying how to fill the slots in.

We call *elements* the unordered slots of a sentence plan along with their instructions, but with S_i and O_i accompanied by the individuals, classes, or datatype values they refer to. In the first example of Section 2.2, there are four elements: [*ref*(S = `:StEmilion`)], [make]$_{\textit{present, passive}}$, [from], [*ref*($O$ = `:cabernetSauvignonGrape`)]. When all the NL names are short and approximately equally long, we can roughly estimate the length (in words) of a sentence that will be produced to report a single fact, before actually producing the sentence, by counting the elements of the sentence plan that will be used to produce the sentence. Furthermore, we can roughly estimate the length of an aggregated sentence, i.e., a sentence that will be obtained by aggregating the simpler sentences (each reporting a single fact) of a group (bucket of Fig. 1), by counting the *distinct* elements (no duplicates) of the sentence plans that will be used to produce the simple sentences of the group, because duplicate elements (originating from more than one simple sentences) are typically expressed only once in the aggregated sentence.

In the following aggregation example, there are initially two simple sentences, produced by sentence plans identical to the first one of Section 2.2, except for the different prepositions. The sentence plans of the two simple sentences have four elements each: [*ref*(S = `:BancroftChardonnay`)], [make]$_{\textit{present, passive}}$, [by], [*ref*($O$ = `:Mountadam`)] and [*ref*(S = `:BancroftChardonnay`)], [make]$_{\textit{present, passive}}$, [in], [*ref*($O$ = `:Bancroft`)]. The

distinct elements of the two sentence plans are only six, indicating that the aggregated sentence will be shorter than the two initial sentences together (eight elements in total).

Bancroft Chardonnay is made by Mountadam. It is made in Bancroft. ⇒
Bancroft Chardonnay is made by Mountadam in Bancroft.

By contrast, if a slightly different sentence plan involving the verb 'produce' is used in the first simple sentence, the aggregated sentence will be longer, as shown below. The sentence plans of the two simple sentences again have eight elements in total, but their distinct elements are seven ([$ref(S$ = :BancroftChardonnay)], [produce]$_{present,\ passive}$, [by], [$ref(O$ = :Mountadam)], [make]$_{present}$, [in], [$ref(O$ = :Bancroft)]), correctly predicting that the aggregated sentence will now be longer.

Bancroft Chardonnay is produced by Mountadam. It is made in Bancroft. ⇒
Bancroft Chardonnay is produced by Mountadam and made in Bancroft.

The number of distinct elements is only an approximate estimate of the length of the aggregated sentence, because some of the names of the classes and individuals (e.g., 'Bancroft Chardonnay') and some of the verb forms (e.g., 'is made') are multi-word, but it allows the ILP model to roughly predict the length of an aggregated sentence by considering only sentence plans, before actually producing or aggregating any sentences.

The previous examples also show that selecting among alternative sentence plans affects the length of the generated text, not only because different sentence plans may require more or fewer words to express the same fact, but also because different combinations of sentence plans may produce more or fewer aggregation opportunities (e.g., shared verbs). Content selection also affects the length of the text, not only because different facts may require more or fewer words to report, but also because the selected facts may or may not have combinations of sentence plans that provide aggregation opportunities, and the aggregation opportunities may allow saving fewer or more words. For example, consider the following facts. Let us assume that all four facts are equally important, and that we want to generate a text expressing only four of them.

```
<:MountadamRiesling, isA, :Riesling>
<:MountadamRiesling, :hasBody, :Medium>
<:MountadamRiesling, :hasMaker, :Mountadam>
<:MountadamRiesling, :hasFlavor, :Delicate>
<:MountadamRiesling, :hasSugar, :Dry>
```

A pipeline approach to generation, where the content selection decisions are made greedily without considering their effects on the later stages of lexicalization (in our case, sentence plan selection) and aggregation, might select the first four of the facts (perhaps randomly, since all facts are equally important). Assuming that lexicalization also does not consider the effects of its choices (selected sentence plans) on sentence aggregation, we may end up with the following text, before and after aggregation.

This is a Riesling. It is medium. It is produced by Mountadam. It has a delicate flavor. ⇒
This is a medium Riesling, produced by Mountadam. It has a delicate flavor.

On the other hand, a global approach that jointly considers the decisions of content selection, lexicalization, and aggregation might prefer to express the fifth fact instead of the fourth, and to use sentence plans that allow more compressive aggregations, leading to a much shorter text, as shown below.

This is a Riesling. It is medium. It is dry. It is delicate. ⇒ This is a medium dry delicate Riesling.

The length of the resulting text is important when space is limited or expensive, as already discussed, which is why we aim to produce compact texts, i.e., texts that report as many facts per word as possible (or texts that maximize the importance of the reported facts divided by the words used, when facts are not equally important). More precisely, given an individual or class of an OWL ontology and a set of available facts about it, we aim to produce a text that:

Goal 1: expresses as many of the available facts as possible (or a text that maximizes the total importance of the reported facts, when facts are not equally important),
Goal 2: using as few words as possible.

By varying weights associated with Goals 1 and 2, we obtain different compact texts, aimed towards expressing more of the available facts at the expense of possibly using more words, or aimed towards using fewer words at the expense of possibly expressing fewer of the available facts.

We can now formally define our first ILP model. Let $s_1, \ldots, s_m$ be disjoint subsets (buckets of Fig. 1) of $F = \{f_1, \ldots, f_n\}$ (the set of available facts), each containing 0 to n facts. A single aggregated sentence is generated from each subset s_j by aggregating the simple sentences (more precisely, their selected sentence plans) that express the facts of s_j. An empty s_j generates no sentence. Hence, the resulting text can be at most m aggregated sentences long. Let us also define:

$$a_i = \begin{cases} 1, \text{ if fact } f_i \text{ is selected} \\ 0, \text{ otherwise} \end{cases} \tag{1}$$

$$l_{ikj} = \begin{cases} 1, \text{ if sentence plan } p_{ik} \text{ is used to express fact } f_i, \text{ and } f_i \text{ is in subset } s_j \\ 0, \text{ otherwise} \end{cases} \tag{2}$$

$$b_{tj} = \begin{cases} 1, \text{ if element } e_t \text{ is used in subset } s_j \\ 0, \text{ otherwise} \end{cases} \tag{3}$$

and let B be the set of all the distinct elements (no duplicates) from all the available sentence plans p_{ik} that can express the facts of F. As already noted, the length of an aggregated sentence resulting from a subset s_j can be roughly estimated by counting the distinct elements of the sentence plans that were chosen to express the facts of s_j.

The objective function of our first ILP model (Eq. 4 below) maximizes the total importance of the selected facts (or simply the number of selected facts, if all facts are equally important), and minimizes the number of distinct elements in each subset s_j, i.e., the approximate length of the corresponding aggregated sentence; an alternative explanation is that by minimizing the number of distinct elements in each s_j, we favor subsets that aggregate well. By a and b we jointly denote all the a_i and b_{tj} variables. $|\sigma|$ denotes the cardinality of a set σ. The two parts of the objective function are normalized to $[0, 1]$ by dividing by the total number of available facts $|F|$ and the number of subsets m times the total number of distinct elements $|B|$. We multiply α_i with the importance score $imp(f_i)$ of the corresponding fact f_i. We assume that the importance scores range in $[0, 1]$; in our experiments, all the importance scores are set to 1, with the exception of redundant message triples that are assigned zero importance scores (Section 2.3). The parameters λ_1, λ_2 are used to tune the priority given to expressing many important facts vs. generating shorter texts; we set $\lambda_1 + \lambda_2 = 1$.

Constraint 5 ensures that for each selected fact, exactly one sentence plan is selected and that the fact is placed in exactly one subset; if a fact is not selected, no sentence plan

for the fact is selected and the fact is placed in no subset. In Constraint 6, B_{ik} is the set of distinct elements e_t of the sentence plan p_{ik}. This constraint ensures that if p_{ik} is selected in a subset s_j, then all the elements of p_{ik} are also present in s_j. If p_{ik} is not selected in s_j, then some of its elements may still be present in s_j, if they appear in another selected sentence plan of s_j. In Constraint 7, $P(e_t)$ is the set of sentence plans that contain element e_t. If e_t is used in a subset s_j, then at least one of the sentence plans of $P(e_t)$ must also be selected in s_j. If e_t is not used in s_j, then no sentence plan of $P(e_t)$ may be selected in s_j. Constraint 8 limits the number of elements that a subset s_j can contain to a maximum allowed number B_{max}, in effect limiting the maximum (estimated) length of an aggregated sentence. Constraint 9 ensures that facts from different topical sections will not be placed in the same subset s_j, to avoid unnatural aggregations.

$$\max_{a,b} \lambda_1 \cdot \sum_{i=1}^{|F|} \frac{a_i \cdot imp(f_i)}{|F|} - \lambda_2 \cdot \sum_{j=1}^{m} \sum_{t=1}^{|B|} \frac{b_{tj}}{m \cdot |B|} \tag{4}$$

subject to:

$$a_i = \sum_{j=1}^{m} \sum_{k=1}^{|P_i|} l_{ikj}, \text{for } i = 1, \ldots, n \tag{5}$$

$$\sum_{e_t \in B_{ik}} b_{tj} \geq |B_{ik}| \cdot l_{ikj}, \text{for} \begin{cases} i = 1, \ldots, n \\ j = 1, \ldots, m \\ k = 1, \ldots, |P_i| \end{cases} \tag{6}$$

$$\sum_{p_{ik} \in P(e_t)} l_{ikj} \geq b_{tj}, \text{for} \begin{cases} t = 1, \ldots, |B| \\ j = 1, \ldots, m \end{cases} \tag{7}$$

$$\sum_{t=1}^{|B|} b_{tj} \leq B_{max}, \text{for } j = 1, \ldots, m \tag{8}$$

$$\sum_{k=1}^{|P_i|} l_{ikj} + \sum_{k'=1}^{|P_{i'}|} l_{i'k'j} \leq 1, \text{for} \begin{cases} j = 1, \ldots, m, \ i = 2, \ldots, n \\ i' = 1, \ldots, n-1; i \neq i' \\ \textit{section}(f_i) \neq \textit{section}(f'_i) \end{cases} \tag{9}$$

3.2 Our Extended ILP Model

The ILP model of the previous section assumes that a single NL name is available for each individual or class (excluding anonymous ones). By contrast, our extended ILP model assumes that multiple alternative NL names are available. The reader is reminded that an NL name specifies how to generate a noun phrase naming an individual or class, and that it is a sequence of slots, along with instructions specifying how to fill them in.

For an individual or class acting as the O of a fact $\langle S, R, O \rangle$ to be expressed, the extended ILP model always selects the shortest available NL name. It takes, however, into account the length of the (shortest) NL name of O when estimating the length of a sentence that will express $\langle S, R, O \rangle$. By contrast, the model of the previous section ignored the lengths of the NL names when estimating sentence lengths, assuming that all the NL names are short and approximately equally long, an assumption that does not always hold. For example, the Disease Ontology, one of the ontologies of our experiments, includes an individual with an NL name that produces the noun phrase

"paralysis of the legs due to thrombosis of spinal arteries", and another individual with an NL name that produces simply "inflammation". Hence, a sentence that uses the former NL name to express a fact whose O is the former individual will be much longer than a sentence that uses the latter NL name to express another fact whose O is the latter individual, even if both sentences are produced by the same sentence plan.

The extended model also considers the possibility of O being a conjunction or disjunction of classes, individuals, datatype values (Section 2), as in the last fact below.

```
<:BrazilianHemorrhagicFever, :isA, :ViralInfectiousDisease>
<:BrazilianHemorrhagicFever, :hasMaterialBasisIn, :SabiaVirus>
<:BrazilianHemorrhagicFever, :transmittedBy, :rodents>
<:BrazilianHemorrhagicFever, :hasSymptom,
                             and(:fatigue, :muscleAches, :dizziness)>
```

In the ILP model of the previous section, we made no distinction between Os that are single classes, individuals, or datatype values, and Os that are conjunctions or disjunctions, assuming that the number of conjuncts or disjuncts, respectively, is always small and does not affect much the length of the resulting sentence. In some ontologies, though, the number of conjuncts or disjuncts varies greatly. In the Disease Ontology, the number of conjuncts in the `hasSymptom` relation ranges from 1 to 14. Let us assume that we wish to generate a text for `BrazilianHemorrhagicFever`, that we are limited to expressing two facts, and that all facts are equally important. The model of the previous section might, for example, select the first and last of the facts above, possibly because their sentence plans are short (in elements), leading to the following sentence.

> The Brazilian hemorrhagic fever is a viral infectious disease that causes fatigue, muscle aches and dizziness.

By contrast, the extended ILP model takes into account that the conjunction in the O of the last fact above requires five words. Hence, it might select the first and third facts instead, producing the following shorter sentence.

> The Brazilian hemorrhagic fever is a viral infectious disease transmitted by rodents.

Note, also, that selecting the first and second facts, which only have single individuals or classes as Os, would lead to the following sentence, which is longer, because of the length of "the Sabia virus".

> The Brazilian hemorrhagic fever is a viral infectious disease caused by the Sabia virus.

Selecting among the alternative NL names of the S of a fact $\langle S, R, O \rangle$ is more complicated, because a longer NL name (e.g., producing "the Napa Region Bancroft Chardonay wine") may also convey some of the other available facts, without requiring separate sentences for them, thus saving words. Consider, for example, the following facts and assume that we wish to generate a text expressing all of them.

```
<:BancroftChardonnay, isA, :Chardonnay>
<:BancroftChardonnay, :locatedIn, :NapaRegion>
<:BancroftChardonnay, :hasMaker, :Bancroft>
<:BancroftChardonnay, :hasFlavor, :Moderate>
<:BancroftChardonnay, :hasSugar, :Dry>
```

Let us also assume that `BancroftChardonnay` has three alternative NL names, which produce "Bancroft Chardonnay", "the Napa Region Bancroft Chardonnay wine", and

"the moderate tasting and dry Bancroft Chardonnay wine", respectively.[15] For each alternative NL name of S, we invoke the mechanism of NaturalOWL (Section 2.3) that detects redundant facts (message triples with zero importance scores). In our example, if we choose to refer to S as "Bancroft Chardonnay", we do not need to produce separate sentences for the first and third facts above, since they are already indirectly expressed by the NL name of S, and similarly for the other two NL names of S, as shown below.

S called "Bancroft Chardonnay":
Bancroft Chardonnay is moderate and dry. It is produced in the Napa Region.
~~It is a Chardonnay. It is produced by Bancroft.~~

S called "the Napa Region Bancroft Chardonnay wine":
The Napa Region Bancroft Chardonnay wine is moderate and dry.
~~It is a Chardonnay. It is produced by Bancroft in the Napa Region.~~

S called "the moderate tasting and dry Bancroft Chardonnay wine":
The moderate tasting and dry Bancroft Chardonnay wine is produced in the Napa Region.
~~It is a moderate, dry Chardonnay. It is produced by Bancroft.~~

Selecting the NL name that produces the shortest noun phrase ("Bancroft Chardonnay") does not lead to the shortest text. The shortest text is obtained when the second NL name is selected. Selecting the third NL name above, which leads to the largest number of facts made redundant (meaning facts that no longer need to be expressed as separate sentences), also does not lead to the shortest text, as shown above.

To further increase the range of options that the extended ILP model considers and help it to produce more compact texts, when using the extended ILP model we allow alternative NL names to be provided also for individuals or classes declared as 'anonymous' (Section 2.1); *possibly anonymous* is now a better term. In other words, the system can refer to an individual or class declared to be possibly anonymous, by using a demonstrative pronoun ("this") or a demonstrative noun phrase mentioning the parent class (e.g., "this Chardonnay"), as with anonymous individuals and classes before, but it can also use an NL name of the individual or class (if provided), i.e., declaring an individual or class as possibly anonymous licenses the use of a demonstrative or demonstrative noun phrase, without excluding the use of an NL name.[16] Continuing our example, let us assume that `BancroftChardonnay` has been declared as possibly anonymous. Then the following texts are also possible.

Demonstrative used for S:
This is a moderate, dry Chardonnay. It is produced by Bancroft in the Napa Region.

Demonstrative noun phrase used for S:
This Chardonnay is moderate and dry. It is produced by Bancroft in the Napa Region.
~~It is a Chardonnay.~~

As illustrated above, a demonstrative noun phrase that mentions the ancestor class (e.g., "this Chardonnay") is also taken to express the corresponding fact about the ancestor class (e.g., `<:BancroftChardonnay, isA, :Chardonnay>`). Notice, also, that using a

15 Some of the NL names of this article, like the first two of this example, were semi-automatically constructed by the methods of Lampouras (2015). The other NL names, like the third one of this example, were manually authored to provide more choices to the extended ILP model.

16 A pronoun can also be used, but pronouns are generated by the (external to our ILP models) referring expression generation component of NaturalOWL, after invoking the ILP models, as already discussed.

demonstrative or demonstrative noun phrase does not necessarily lead to the shortest text. In our example, the shortest text is still obtained using the second NL name.

Before moving on to the formulation of the extended ILP model, let us discuss how it estimates the lengths of (possibly aggregated) sentences. In the ILP model of the previous section, we roughly estimated the length of an aggregated sentence resulting from a subset (bucket) s_j by counting the distinct elements of the sentence plans chosen to express the facts of s_j. For example, let us assume that the distinct elements [*ref*(S = :StEmilion)], [make]$_{present,\ passive}$, [from], and [*ref*(O = :cabernetSauvignonGrape)] are used in a single subset s_j. The ILP model of the previous section did not consider the lengths of the noun phrases that will be produced by the NL names of :StEmilion and :cabernetSauvignonGrape of the elements [*ref*(S = :StEmilion)] and [*ref*(O = :cabernetSauvignonGrape)]. Also, it did not take into account that the element [make]$_{present,\ passive}$ actually produces two words ("is made").

The extended model defines a function $length(e_t)$ that maps each distinct element e_t to the length (in words) of the text it produces (e.g., "is made"). More specifically, if e_t is an element referring to a single individual or class acting as the O of a message triple (e.g., [*ref*(O = :cabernetSauvignonGrape)]), then $length(e_t)$ is the length (in words) of the (shortest) NL name of O; if O is a conjunction or disjunction, then $length(e_t)$ is the sum of the lengths of the (shortest) NL names of all the conjuncts or disjuncts. However, if e_t is an element referring to S (e.g., [*ref*(S = :StEmilion)]), then $length(e_t) = 1$, because the NL name of S will be used only once at the beginning of the text, and each subsequent reference to S will be via a pronoun of length 1 (e.g., "St. Emilion is red and strong. *It* is made from Cabernet Sauvignon grapes."); the first occurrence of the NL name is counted separately, directly in the objective function discussed below. The estimated length of a (possibly aggregated) sentence is the sum of the estimated lengths ($\sum_t length(e_t)$) of the distinct elements of the sentence plan(s) that produced it. Overall, the extended model estimates more accurately the length of the text that will be produced, though the actual text length may still be slightly different; for example, connectives or complementizers (e.g., 'and', 'that') may be added during aggregation.

We can now formally define our extended ILP model. As in the simpler model of Section 3.1, F is the set of available facts f_i about the individual or class S we wish to generate a text for, and $s_1, \ldots, s_m$ are disjoint subsets of F (buckets of Fig. 1) showing which simple sentences (each expressing a single fact of F) will be aggregated together. Let $N = \{n_1, n_2, \ldots\}$ be a set of alternative NL names for S. Recall that we model only the choice of NL name for S, assuming that the shortest NL name is always used for the O_i of each fact $f_i = \langle S, R_i, O_i \rangle$. Each a_i variable now indicates if the corresponding fact f_i is explicitly expressed by generating a sentence:

$$a_i = \begin{cases} 1, \text{ if the fact } f_i \text{ is expressed as a sentence} \\ 0, \text{ otherwise} \end{cases} \tag{10}$$

By contrast, d_i is more general; $d_i = 1$ if the corresponding fact f_i is conveyed either explicitly (by generating a sentence for f_i) or implicitly (via an NL name):

$$d_i = \begin{cases} 1, \text{ if the fact } f_i \text{ is expressed as a sentence or via an NL name} \\ 0, \text{ otherwise} \end{cases} \tag{11}$$

The distinction between a_i and d_i is necessary, because when a fact f_i is expressed as a sentence, a sentence plan for f_i is also selected. For example, a fact $f_i =$

<:BancroftChardonnay, :hasMaker, :Bancroft> can be expressed as a sentence in the final text (e.g., "This *is produced by Bancroft*. It comes from the Napa Region.") or through an NL name (e.g., "*Bancroft Chardonnay* is produced in the Napa Region."). In both texts, f_i is expressed ($d_i = 1$), but in the former text $a_i = 1$, whereas in the latter one $a_i = 0$.

The l_{ikj} and b_{tj} variables are as in the ILP model of the previous section (Eq. 2 and 3). For the extended model, we also define:

$$m_r = \begin{cases} 1, \text{ if the NL name } n_r \text{ is used for } S \\ 0, \text{ otherwise} \end{cases} \tag{12}$$

Similarly to the previous model's objective function (4), the extended model's objective function (13) maximizes the total importance of the expressed facts (or simply the number of expressed facts, if all facts are equally important), and minimizes the length of the distinct elements in each subset s_j and the length of the (single, initial occurrence of the) NL name used to express S, i.e., the approximate length of the resulting text. By d, b, and m we jointly denote all the d_i, b_{tj}, and m_r variables. The left part of the objective is the same as in the previous model, with the variables a_i replaced by d_i. In the right part, we multiply the b_{tj} and m_r variables with the functions $length(e_t)$ and $length(n_r)$, which calculate the lengths (in words) of the corresponding element (e_t) and NL name (n_r), respectively. The two parts of the objective function are normalized to $[0, 1]$ by dividing by the total number of available facts $|F|$ and the number of subsets m times the total length of distinct elements $|B|$ plus the total length of the R available NL names. Again, the parameters λ_1, λ_2 are used to tune the priority given to expressing many important facts vs. generating shorter texts; we set $\lambda_1 + \lambda_2 = 1$.

$$\max_{d,b,m} \lambda_1 \cdot \sum_{i=1}^{|F|} \frac{d_i \cdot imp(f_i)}{|F|} - \lambda_2 \cdot \left(\frac{\sum_{j=1}^{m}\sum_{t=1}^{|B|} b_{tj} \cdot length(e_t) + \sum_{r=1}^{|R|} m_r \cdot length(n_r)}{m \cdot \sum_{t=1}^{|B|} length(e_t) + \sum_{r=1}^{|R|} length(n_r)}\right) \tag{13}$$

subject to:

$$a_i = \sum_{j=1}^{m}\sum_{k=1}^{|P_i|} l_{ikj}, \text{for } i = 1, \ldots, n \tag{14}$$

$$\sum_{e_t \in B_{ik}} b_{tj} \geq |B_{ik}| \cdot l_{ikj}, \text{for} \begin{cases} i = 1, \ldots, n \\ j = 1, \ldots, m \\ k = 1, \ldots, |P_i| \end{cases} \tag{15}$$

$$\sum_{p_{ik} \in P(e_t)} l_{ikj} \geq b_{tj}, \text{for} \begin{cases} t = 1, \ldots, |B| \\ j = 1, \ldots, m \end{cases} \tag{16}$$

$$\sum_{t=1}^{|B|} b_{tj} \cdot length(e_t) \leq W_{max}, \text{for } j = 1, \ldots, m \tag{17}$$

$$\sum_{k=1}^{|P_i|} l_{ikj} + \sum_{k'=1}^{|P_{i'}|} l_{i'k'j} \leq 1, \text{for} \begin{cases} j = 1, \ldots, m, \ i = 2, \ldots, n \\ i' = 1, \ldots, n-1; i \neq i' \\ section(f_i) \neq section(f'_i) \end{cases} \tag{18}$$

$$\sum_{r=1}^{|N|} m_r = 1 \tag{19}$$

$$d_i = a_i + \sum_{m_r \in R(f_i)} m_r, \text{for } i = 1, \ldots, n \tag{20}$$

Constraints 14–18 serve the same purpose as in the previous model (Eq. 5–9), except that Constraint 17 now limits the number of words (instead of elements) that a subset s_j can contain to a maximum allowed number W_{max}. Constraint 19 ensures that exactly one NL name is selected from the available NL names of S. In Constraint 20, $R(f_i)$ is the set of NL names that (indirectly) express the fact f_i. If f_i is to be expressed (i.e., $d_i = 1$), then either one of the NL names in $R(f_i)$ must be selected, or a sentence for f_i must be generated ($a_i = 1$), not both. If f_i is not to be expressed, then none of the NL names in $R(f_i)$ may be selected, nor should a sentence be generated for f_i.

4. Computational Complexity and Approximations

The models of Sections 3.1 and 3.2 are formulated as ILP problems, more precisely binary ILP problems since all their variables are binary. Solving binary ILP problems is in general NP-hard (Karp 1972). We also note that content selection, as performed by our models, is similar to the 0-1 multiple Knapsack problem, which is also NP-hard. In both cases, we have n items (facts), m knapsacks (fact subsets, buckets) of a certain capacity, and we wish to fill the knapsacks with m disjoint subsets of the available items, so that the total importance of the selected items (items placed in the knapsacks) is maximum. However, in our models each item (fact) is further associated with a set of (sentence plan) elements, subsets of which are possibly shared (in a subset, bucket) with other items (facts), and the capacity of the knapsacks is specified in distinct elements. Furthermore, the elements of each item depend on the selected sentence plans, there are additional constraints to comply with topical sections, and the objective function of our models also tries to minimize the total length of the resulting text. Hence, our models do not correspond directly to the 0-1 multiple Knapsack problem.

A possible approach to solve ILP models in polynomial time is to relax the constraint that variables are integer (or binary) and solve the resulting Linear Programming model (LP relaxation) using, for example, the Simplex algorithm (Dantzig 1963). The resulting values of the variables are then rounded to the closest integral values. The solution is not guaranteed to be optimal for the original ILP problem, nor feasible (some constraints of the original problem may be violated). The solution of the LP relaxation, though, is the same as the solution of the original ILP problem if the problem can be formulated as $\max_x c^T x$ with constraints $Ax = b$, where c, A, m have integer values and the matrix A is totally unimodular (Schrijver 1986; Roth and Yih 2004). An integer matrix is totally unimodular if every square, nonsingular submatrix is unimodular (i.e., its determinant is 0, 1, or -1). Unfortunately, this is not the case in our ILP models.

In practice, off-the-shelf solvers that solve the original ILP problem (not the LP relaxation) are very fast when the number of variables is small.[17] Our experiments show that solving the first ILP model is reasonably fast, provided that the number of fact subsets (buckets) is $m \leq 4$. Indeed, m seems to be the greatest factor to the model's

17 We use the branch-and-cut implementation of GLPK with mixed integer rounding, mixed cover, and clique cuts; see `http://sourceforge.net/projects/winglpk/`.

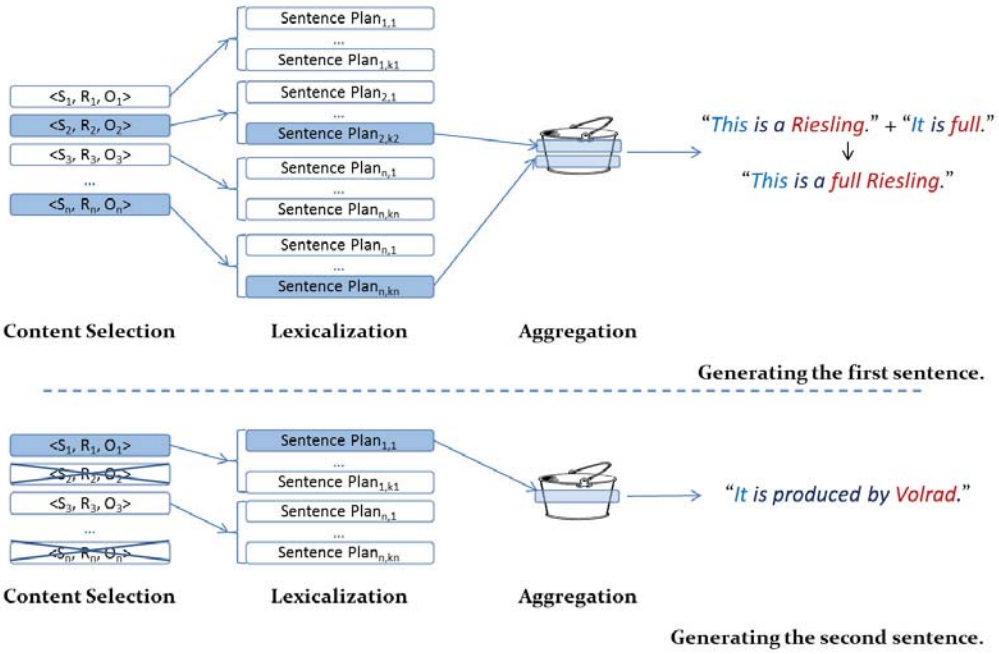

Figure 2
Illustration of the approximation of the first ILP model.

complexity; the number of variables in the model grows exponentially to m, while the effect of the other parameters (e.g., number of available facts $|F|$) is weaker. We did not examine experimentally how the solving times of the extended ILP model relate to the number of subsets m; however, the variables in the extended model also grow exponentially to the number of fact subsets m.

When the number of variables is too large to solve the first ILP model efficiently, we use an approximation of the model, which considers each fact subset (bucket, aggregated sentence of the final text) separately (Fig. 2). We start with the full set of available facts (F) and use the first ILP model with $m = 1$ to produce the first (aggregated) sentence of the final text. We then remove the facts expressed by the first (aggregated) sentence from F, and use the ILP model, again with $m = 1$, to produce the second (aggregated) sentence etc. This process is repeated until we produce the maximum number of allowed aggregated sentences, or until we run out of available facts.

Since the approximation of the first ILP model does not consider all the fact subsets jointly, it does not guarantee finding a globally optimal solution for the entire text. Nevertheless, experiments (presented below) that compare the approximation to the original first ILP model show no apparent decline in text quality nor in the ability to produce compact texts. Solving times now grow almost linearly to both the number of subsets m and the number of available facts $|F|$. Furthermore, $|F|$ decreases in every subsequent solving of the model (to produce the next aggregated sentence of the text), which reduces the time needed by the solver. Our experiments indicate that the approximation can guarantee practical running times even for $m \geq 5$, while still outperforming the pipeline approach in terms of producing more compact texts.

The same approximation (considering each fact subset separately) can be applied to our extended ILP model. We did not experiment with the approximation of the extended model, however, because the only ontology we considered that required $m \geq 5$ and, hence, an approximation (Consumer Electronics Ontology) did not require the extended model (the lengths of the NL names did not vary significantly, and we could not think of alternative NL names for the products being described).

5. Experiments

We now present the experiments we performed to evaluate our ILP models. We first discuss the ontologies and systems that were used in our experiments.

5.1 The Ontologies of our Experiments

We experimented with three OWL ontologies: (1) the Wine Ontology, which provides information about wines, wine producers etc.; (2) the Consumer Electronics Ontology, intended to help exchange information about consumer electronics products; and (3) the Disease Ontology, which describes diseases, including their symptoms, causes etc.[18] The Wine Ontology is one of the most commonly used examples of OWL ontologies and involves a wide variety of OWL constructs; hence, it is a good test case for systems that produce texts from OWL. The Consumer Electronics and Disease Ontologies were constructed by biomedical and e-commerce experts to address real-life information needs; hence, they constitute good real-world test cases from different domains.

The Wine Ontology contains 63 wine classes, 52 wine individuals, a total of 238 classes and individuals (including wineries, regions, etc.), and 14 relations (properties). Manually authored, high-quality domain-dependent generation resources (text plans, sentence plans, NL names etc.) for NaturalOWL are available for this ontology from our previous work (Androutsopoulos, Lampouras, and Galanis 2013).

The Consumer Electronics Ontology comprises 54 classes and 441 individuals (e.g., printer types, paper sizes, manufacturers), but no information about particular products. In previous work (Androutsopoulos, Lampouras, and Galanis 2013), we added 60 individuals (20 digital cameras, 20 camcorders, 20 printers). The 60 individuals were randomly selected from a publicly available dataset of 286 digital cameras, 613 camcorders, and 58 printers that complies with the Consumer Electronics Ontology.[19] From these 60 individuals, we generate texts for the 30 'development' individuals (10 cameras, 10 camcorders, 10 printers), for which high-quality manually authored domain-dependent generation resources are available from our previous work.

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.,

18 Consult `http://www.w3.org/TR/owl-guide/wine.rdf/`, `http://www.ebusinessunibw.org/ontologies/consumerelectronics/v1`, and `http://disease-ontology.org/`.

19 The dataset was obtained from `http://rdf4ecommerce.esolda.com/`. A list of similar datasets is available at `http://wiki.goodrelations-vocabulary.org/Datasets`.

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 introducing secondary clauses (e.g., "that", "which").[20] Some sentences of the original definition strings that did not include declared relation names (e.g., "The virus affects…and humans" in the 'Definition' string of Rift Valley Fever) were discarded, because they could not be automatically converted to appropriate OWL statements.

The new form of the Disease Ontology contains 6,746 classes, 15 relations, and 1,545 individuals. From the 6,746 classes (all describing diseases), 5,014 classes participate only in IS-A and synonym relations; hence, texts for them would not be particularly interesting. From the remaining 1,732 classes, we generate texts for the 200 randomly selected 'development' classes of Evaggelakaki (2014), for which manually authored domain-dependent generation resources for NaturalOWL are available.

5.2 The Systems of our Experiments

We call PIPELINE the original NaturalOWL, which uses a pipeline architecture. Two modified versions of NaturalOWL, called ILPNLG and ILPNLGEXTEND, use our first

20 The new form of the Disease Ontology that we produced is available upon request and will be made publicly available when this article is published.

and extended ILP models, respectively. All the systems of our experiments share the same linguistic resources (e.g., text plans, sentence plans, NL names, aggregation rules), ontologies, and importance scores; all facts are assigned an importance of 1, except for facts that are automatically assigned zero importance scores (Section 2.3).

PIPELINE has a parameter M specifying the number of facts to report per generated text. During content selection, PIPELINE ranks all the available facts (F) by decreasing importance, and selects the M most important ones (or all of them if $M > |F|$) selecting randomly among facts with the same importance when needed. In the experiments that follow, we generated texts with PIPELINE for different values of M. For each M value, the texts of PIPELINE were generated T times, each time using a different (randomly selected) alternative sentence plan of each relation, and a different (randomly selected) NL name of each individual or class (when multiple alternative NL names were available). For the PIPELINE model, we assume that the sentence plans and NL names are uniformly distributed with each being equally probable to be selected. For the aggregation of the selected facts, PIPELINE uses the text planner from the original NaturalOWL. The text planner is invoked after content selection to partition the selected facts into topical sections, and to order the topical sections and the facts within each topical section. The aggregation rules are then applied to all the facts of each topical section (also considering their selected sentence plans). From the T generated texts, PIPELINE returns the one which is estimated to have the highest facts per word ratio. Rather than use the actual length of each produced text to calculate the facts per words ratio, the number of words is instead estimated as the sum of distinct elements in each sentence of the text, to better align the objective of PIPELINE to that of ILPNLG.

We also generated the texts (for different values of M) using a variant of PIPELINE, dubbed PIPELINESTOCH, which selects randomly amongst available facts, in addition to sentence plans and NL names. However, unlike PIPELINE, the probability of each sentence plan or NL name is based on their respective length (in distinct elements), with the shortest ones being more probable to be selected. The fact's probabilities are similarly estimated by the length of the shortest sentence plan and NL name available to them. In regards to aggregation, PIPELINESTOCH constructs fact subsets (corresponding to sentences in the final text) with the objective of minimizing the number of distinct elements in each subset, similarly to ILPNLG. Each subset is initialized with random facts (sampled based on the length of their available resources) and subsequent facts are randomly placed in each subset, with probabilities estimated on the number of elements each fact has in common with the facts already in that particular subset. As with PIPELINE, for each M the texts are generated T times, and the one with the highest facts per word ratio is used for the evaluation.

A greedier variant of PIPELINE, PIPELINESHORT always selects the shortest (in elements) sentence plan among the available ones and the shortest (in words) NL name. In PIPELINESHORT, if a subset of facts has the same importance, they are additionally ranked by increasing length of the shortest sentence plan and NL name available to each; this way the fact with the potential to generate the shortest sentence will be selected first.

Our final baseline, PIPELINEBEAM extends the output of PIPELINESHORT by employing beam search to select alternative facts, sentence plans, NL names and fact subsets. During content selection, PIPELINEBEAM selects the subset of M facts with the shortest sentence plans and NL names available to them (similarly to PIPELINESHORT), and subsequently replaces a single random (based on the length of the available resources) fact from this subset with a random non-selected fact. This process is repeated until $K - 1$ additional fact subsets are constructed; all differing from the initial subset by one (replaced) fact. In a similar way, K different sentence plan assignments, K different

NL name assignment and K different fact subset assignments are also constructed, differing from the respective assignments of PIPELINESHORT by one substitution each. The combination of these assignments result in $K \times K \times K \times K$ different texts for each M. As in the other baselines, the text amongst these with the highest estimated facts per words ratio is used for the evaluation.

To better compare the output of the pipeline baselines, we set the number of generated texts T that PIPELINE, PIPELINESTOCH and PIPELINESHORT generate to $K \times K \times K \times K$ as PIPELINEBEAM.

All the systems use the same text planner (from the original NaturalOWL) which is invoked before content selection to partition the facts into topical sections, and to order the topical sections and the facts within each topical section. Each of the systems described above have different strategies to partition the selected facts after content selection in sentences. The selected facts retain the order given from the text planner, and the sentences inherit the minimum order of their included facts. Afterwards, aggregation rules are applied to all the facts of each fact subset (also considering their selected sentence plans). the text planner is first invoked (before using the ILP models) to partition all the available facts (F) into topical sections. It is also invoked after using one of the ILP models, to order the sentences in each group (bucket) that the ILP model has decided to aggregate; as already noted, the aggregation rules presuppose that the sentences to be aggregated are already ordered, which is why the text planer is invoked at this point. After applying the aggregation rules to each group of (ordered) sentences, ILPNLG and ILPNLGEXTEND invoke the text planner again to order the topical sections and the (now aggregated) sentences within each topical section.

ILPNLG assumes that there is a single NL name per individual or class (excluding anonymous ones) and, hence, cannot be used when multiple alternative NL names are available. By contrast, ILPNLGEXTEND can handle multiple alternative NL names. For each text, it selects a single NL name per individual and class (as discussed in Section 3.2), which is then replaced by a demonstrative, demonstrative noun phrase, or pronoun, whenever the referring expression generation component of the original NaturalOWL decides to. PIPELINE and PIPELINESHORT can also handle multiple NL names, but PIPELINE selects randomly among the alternative NL names, and PIPELINESHORT selects always the shortest one. Like ILPNLGEXTEND, for each text PIPELINE and PIPELINESHORT select a single NL name per individual and class, which is then replaced by a demonstrative, demonstrative noun phrase, or pronoun, whenever the referring expression generation component of the original NaturalOWL decides to.

A variant of PIPELINESHORT, called PIPELINESHORT*, always selects the shortest (now in words) sentence plan among the available ones, and the NL name of S (the individual or class the text is generated for) that indirectly expresses the largest number of available facts $f_i = \langle S, R_i, O_i \rangle$ (Section 2.3), thus not requiring sentences to express them.[21] For O_i, PIPELINESHORT* selects the same (shortest in words) NL name as ILPNLGEXTEND and PIPELINESHORT. Otherwise, PIPELINESHORT* is identical to PIPELINESHORT. PIPELINESHORT* is a more appropriate baseline for ILPNLGEXTEND than PIPELINESHORT, because like ILPNLGEXTEND it estimates the lengths of sentences and NL names in words, and it takes into account that NL names may indirectly express some of the available facts.

21 Selecting the NL name of S that expresses the largest number of available facts usually leads to better facts per word ratios than simply selecting the shortest (in words) NL name of S. If several NL names of S express the same number of available facts, PIPELINESHORT* selects the shortest (in words) NL name.

Finally, ILPNLGAPPROX denotes a system that is identical to ILPNLG (it uses our first ILP model), but with the approximation of Section 4, whereby each (possibly aggregated) sentence of the text is generated separately.

5.3 Overview of the Experiments

Before presenting the details of our experiments, let us first provide an overview. We started by comparing ILPNLG to PIPELINE and PIPELINESHORT on the Wine Ontology, where experiments showed that ILPNLG leads to more compact texts, i.e., texts with higher facts per word ratios, with no deterioration in the perceived quality of the resulting texts, compared to the texts of PIPELINE and PIPELINESHORT.

We then tried to repeat the same experiments on the Consumer Electronics Ontology, but ILPNLG was too slow in many cases, because of the larger number of available facts per product ($|F|$) and the larger ($m = 10$) number of subsets (buckets) required to express all (or many) of the available facts. To address this problem, we developed the approximation (Section 4) of ILPNLG, which is used in ILPNLGAPPROX. The approximation was much more efficient and achieved higher facts per word ratios than PIPELINE and PIPELINESHORT, with no deterioration in the perceived quality of the texts. In texts expressing many facts, the perceived quality of the texts of ILPNLGAPPROX was actually higher, comparing to the texts of PIPELINE and PIPELINESHORT.

We then moved on to the Disease Ontology, to experiment with an additional domain. Since the Disease Ontology only required $m = 4$ fact subsets to express all the available facts per disease, ILPNLGAPPROX was not required, and ILPNLG was used instead. We found that ILPNLG did not always perform better than PIPELINE and PIPELINESHORT (in terms of facts per word ratios), because the lengths of the NL names of the Disease Ontology vary a lot, and there are also several facts $\langle S, R, O \rangle$ whose O is a conjunction, sometimes with many conjuncts. To address these issues, we extended ILPNLG to ILPNLGEXTEND, which consistently produced more compact texts than PIPELINE and PIPELINESHORT* on the Disease Ontology.

Lastly, we returned to the Wine Ontology to see how ILPNLGEXTEND performs with multiple alternative NL names. For this experiment, we created alternative NL names for the individuals and classes of the Wine Ontology; we could not do the same for the Consumer Electronics and Disease Ontologies, because the names of electronic products tend to be unique and we did not have the expertise to create alternative names of diseases. Indeed, ILPNLGEXTEND produced more compact texts than PIPELINE and PIPELINESHORT* from the Wine Ontology, when multiple NL names were available.

5.4 Experiments with the Wine Ontology

In a first set of experiments, we used the Wine Ontology, along with the manually authored domain-dependent generation resources (e.g., text plans, NL names, sentence plans) we had constructed for this ontology in previous work (Androutsopoulos, Lampouras, and Galanis 2013). We added more sentence plans to ensure that three sentence plans were available per relation.[22] A single NL name was available per individual and class in these experiments. We generated English texts for the 52 wine individuals of the ontology; we did not experiment with texts describing classes, because we could

22 The domain-dependent generation resources of NaturalOWL that we used in all the experiments of this article are available upon request and will be made publicly available when this article is published.

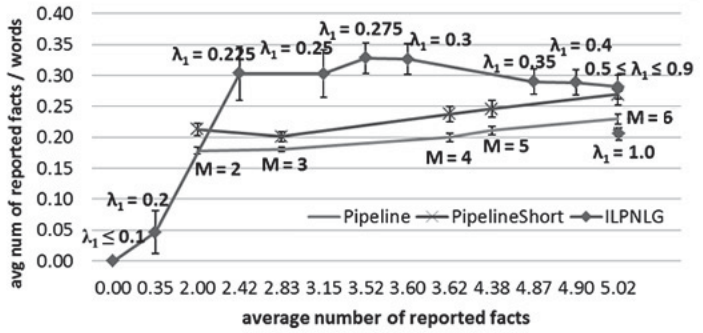

Figure 3
Facts per word ratios for the Wine Ontology (grouped by M or λ_1 values).

not think of multiple alternative sentence plans for many of their axioms. For each wine individual, there were 5 available facts on average and a maximum of 6 facts.

We generated texts with ILPNLG, PIPELINE, and PIPELINESHORT for the 52 individuals. With PIPELINE and PIPELINESHORT, we generated texts for $M = 2, 3, 4, 5, 6$; recall that M is the number of selected facts per text, and that for each M value the texts of PIPELINE and PIPELINESHORT are generated three times, with randomly selected sentence plans (Section 5.2). With ILPNLG, we repeated the generation of the texts of the 52 individuals using different λ_1 values ($\lambda_2 = 1 - \lambda_1$), which led to texts expressing from zero to all of the available facts. We set the maximum number of fact subsets to $m = 3$, which was the maximum number of sentences (after aggregation) in the texts of PIPELINE and PIPELINESHORT. All three systems were allowed to form aggregated sentences with up to $B_{max} = 22$ distinct elements; this was the number of distinct elements of the longest aggregated sentence in our previous experiments (Androutsopoulos, Lampouras, and Galanis 2013), where PIPELINE was allowed to combine up to three simple (expressing one fact each) sentences to form an aggregated one.[23]

For each M value (in the case of PIPELINE and PIPELINESHORT) and for each λ_1 value (in the case of ILPNLG), we measured the average (over the 52 texts) number of facts each system reported per text (horizontal axis of Fig. 3), and the average (again over the 52 texts) number of facts each system reported per text divided by the average (over the 52 texts) number of words (vertical axis of Fig. 3, with error bars showing 95% confidence intervals).[24] As one would expect, PIPELINESHORT expressed on average more facts per word (Fig. 3) than PIPELINE, but the differences were small.

For $\lambda_1 \leq 0.1$ (far left of Fig. 3), ILPNLG produces empty texts, because it focuses on minimizing the number of distinct elements of each text. For $\lambda_1 \geq 0.2$, it performs better than PIPELINE and PIPELINESHORT. For $\lambda_1 \approx 0.3$, it obtains the highest average facts per word ratio by selecting the facts and sentence plans that lead to the most compressive aggregations. For greater values of λ_1, it selects additional facts whose sentence plans do not aggregate that well, which is why the ratio declines. When M is small, the two pipeline systems often select facts and sentence plans that offer few aggregation opportunities; as the number of selected facts increases, some more aggregation opportunities arise, which is why the facts per word ratio of the two systems improves.

23 We modified PIPELINE and PIPELINESHORT to count distinct elements during aggregation.
24 For $0.5 \leq \lambda_1 \leq 0.9$, ILPNLG's results were identical.

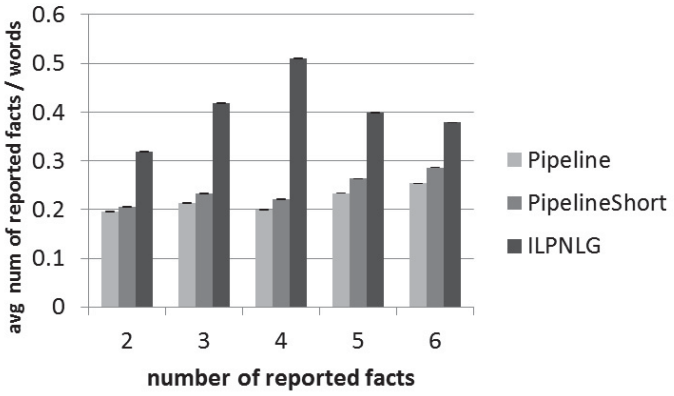

Figure 4
Facts per word ratios for the Wine Ontology (grouped by numer of reported facts).

Figure 4 provides an alternative view of the behavior of the three systems. In this case, we group together all the texts of each system (regardless of the M or λ_1 values that were used to generate them) that report 2, 3, 4, 5, or 6 facts (horizontal axis of Fig. 4). For each group (and each system), we show (vertical axis of Fig. 4) the average number of reported facts per text, divided by the average number of words of the texts in the group.[25] Again, Fig. 4 shows that ILPNLG produces clearly more compact texts than PIPELINE and PIPELINESHORT, with the difference between the latter two systems being very small.[26] In all the experiments of this section, the ILP solver (used in ILPNLG) was very fast (average: 0.08 sec, worst: 0.14 sec per text).

We show below sample texts generated by PIPELINE and PIPELINESHORT (both with $M = 4$) and ILPNLG (with $\lambda_1 = 0.3$).

PIPELINE: This Sauternes has strong flavor. It is made from Sauvignon Blanc and Semillon grapes and it is produced by Chateau D'ychem.
PIPELINESHORT: This is a strong Sauternes. It is made from Sauvignon Blanc and Semillon grapes and it is produced by Chateau D'ychem.
ILPNLG: This is a strong Sauternes. It is made from Sauvignon Blanc and Semillon grapes by Chateau D'ychem.

PIPELINE: This Riesling has sweet taste and it is full bodied. It is made by Schloss Volrad.
PIPELINESHORT: This is a full sweet Riesling. It is produced by Schloss Volrad.
ILPNLG: This is a full sweet moderate Riesling.

In the first group of generated texts above, PIPELINE and PIPELINESHORT use different verbs for the grapes and producer, whereas ILPNLG uses the same verb, which leads to

25 We remove from each group duplicate texts of the same system (for different M or λ_1 values). If we still have more than one texts (of the same system) for the same individual or class in the same group, we keep only the text with the best facts per word ratio, to avoid placing too much emphasis on individuals and classes with many texts in the same group. Especially for PIPELINE, whose texts are generated three times per individual and class (for each M value), we keep the three texts (regardless of M value, excluding duplicates) with the highest facts per word ratios per individual or class in each group.
26 Figure 4 and all the similar figures in the remainder of this article include error bars corresponding to 95% confidence intervals, but the intervals are so small that they can hardly be seen.

Table 1
Human scores for Wine Ontology texts.

Criteria	PIPELINESHORT	ILPNLG
Sentence fluency	4.75 ± 0.21	**4.85** ± 0.10
Text structure	**4.94** ± 0.06	4.88 ± 0.14
Clarity	**4.77** ± 0.18	4.75 ± 0.15
Overall	4.52 ± 0.20	**4.60** ± 0.18

a more compressive aggregation; all the texts of the first group describe the same wine and report four facts each. In the second group of generated texts above, ILPNLG has chosen to report the (moderate) flavor of the wine instead of the producer, and uses the same verb ('is') for all the facts, leading to a shorter sentence; again all the texts of the second group describe the same wine and report four facts each. Recall that we treat all (non-redundant) facts as equally important in our experiments. In both groups of texts, some facts are not aggregated because they belong in different topical sections.

We also wanted to investigate the effect of the higher facts per word ratio of ILPNLG on the perceived quality of the generated texts, compared to the texts of the pipeline systems. We were concerned that the more compressive aggregations of ILPNLG might lead to sentences sounding less fluent or unnatural, though aggregation is often used to produce more fluent texts. We were also concerned that the more compact texts of ILPNLG might be perceived as being more difficult to understand (less clear) or less well-structured. To investigate these issues, we showed the $52 \times 2 = 104$ texts of PIPELINESHORT ($M = 4$) and ILPNLG ($\lambda_1 = 0.3$) to 6 computer science students (undergraduates and graduates), who were not involved in the work of this article; they were all fluent, though not native English speakers. We did not use PIPELINE in this experiment, since its facts per word ratio was similar to that of PIPELINESHORT. Each one of the 104 texts was given to exactly one student. Each student was given approximately 9 randomly selected texts of each system. The OWL statements that the texts were generated from were not shown, and the students did not know which system had generated each text. Each student was shown all of his/her texts in random order, regardless of the system that generated them. The students were asked to score each text by stating how strongly they agreed or disagreed with statements S_1–S_3 below. A scale from 1 to 5 was used (1: strong disagreement, 3: ambivalent, 5: strong agreement).

(S_1) *Sentence fluency*: The sentences of the text are fluent, i.e., each sentence on its own is grammatical and sounds natural. When two or more smaller sentences are combined to form a single, longer sentence, the resulting longer sentence is also grammatical and sounds natural.

(S_2) *Text structure*: The order of the sentences is appropriate. The text presents information by moving reasonably from one topic to another.

(S_3) *Clarity*: The text is easy to understand, if the reader is familiar with basic wine terms.

The students were also asked to provide an overall score (1–5) per text. We did not score referring expressions, since both systems use the same component for them. We note that although both systems use the same text planner, in PIPELINESHORT (and all the pipeline variants) the text planner is invoked once, whereas in ILPNLG (and ILPNLGEXTEND) it is invoked at different stages before and after using the ILP model (Section 5.2), which is why we collected text structure scores too.

Table 1 shows the average scores of the two systems with 95% confidence intervals. For each criterion, the best score is shown in bold. The sentence fluency and overall scores of ILPNLG are slightly higher than those of PIPELINESHORT, whereas PIPELI-

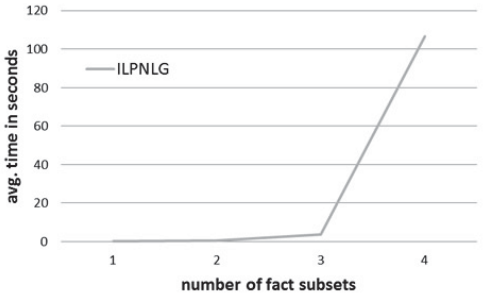

Figure 5
Average solver times for ILPNLG with different maximum numbers of fact subsets (m), for the Consumer Electronics ontology.

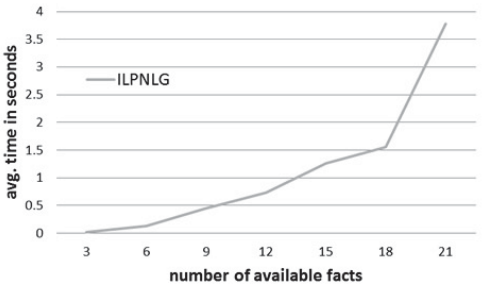

Figure 6
Average solver times for ILPNLG with different numbers of available facts ($|F|$) and $m = 3$, for the Consumer Electronics ontology.

NESHORT obtained a slightly higher score for text structure and clarity. The differences, however, are very small, especially in clarity, and we detected no statistically significant difference between the two systems in any of the criteria.[27] Hence, there was no evidence in these experiments that the higher facts per word ratio of ILPNLG comes at the expense of lower perceived text quality. We investigated these issues further in a second set of experiments, discussed in the next section, where the generated texts were longer.

5.5 Experiments with the Consumer Electronics Ontology

In the second set of experiments, we used the Consumer Electronics Ontology, with the manually authored domain-dependent generation resources (e.g., text plans, NL names, sentence plans) of our previous work (Androutsopoulos, Lampouras, and Galanis 2013). As in the previous section, we added more sentence plans to ensure that three sentence plans were available for almost every relation; for some relations we could not think of enough sentence plans. Again, a single NL name was available per individual and class.

We generated English texts with ILPNLG, PIPELINE, PIPELINESHORT for the 30 development individuals (Section 5.1), using $M = 3, 6, 9, \ldots, 21$ in the two pipeline systems, and different values of λ_1 ($\lambda_2 = 1 - \lambda_1$) in ILPNLG. All three systems were allowed to form aggregated sentences with up to $B_{max} = 39$ distinct elements; this was the number of distinct elements of the longest aggregated sentence in the experiments of our previous work (Androutsopoulos, Lampouras, and Galanis 2013), where PIPELINE was allowed to combine up to three simple (expressing one fact each) sentences to form an aggregated one. There are 14 available facts ($|F|$) on average and a maximum of 21 facts for each one of the 30 development individuals, compared to the 5 available facts on average and the maximum of 6 facts of the Wine Ontology. Hence, the texts of the Consumer Electronics Ontology are much longer, when they report all the available facts. In ILPNLG, we would have to set the maximum number of fact subsets to $m = 10$, which was the maximum number of (aggregated) sentences in the texts of PIPELINE and

27 We performed Analysis of Variance (ANOVA) and post-hoc Tukey tests to check for statistically significant differences. A post-hoc power analysis of the ANOVA values resulted in power values greater or equal to 0.95. We also note that in similar previous experiments (Androutsopoulos, Lampouras, and Galanis 2013), inter-annotator agreement was strong (sample Pearson correlation $r \geq 0.91$).

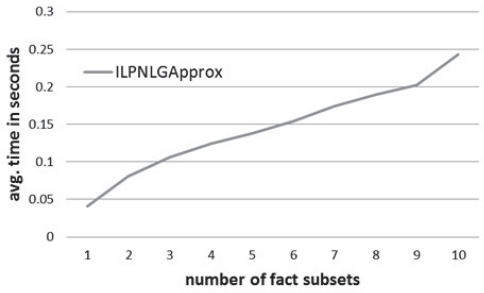

Figure 7
Average solver times for ILPNLGAPPROX with different numbers of fact subsets (m), for the Consumer Electronics ontology.

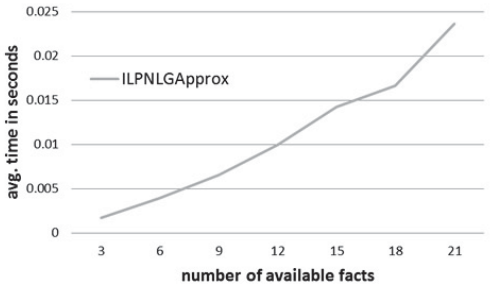

Figure 8
Average solver times for ILPNLGAPPROX with different numbers of available facts ($|F|$) and $m = 3$, for the Consumer Electronics ontology.

PIPELINESHORT. The number of variables of our ILP model, however, grows exponentially to m and $|F|$ (Fig. 5–6), though the effect of $|F|$ is weaker.

Figure 5 shows the average time the ILP solver took for different values of m in the experiments with the Consumer Electronics ontology; the results are averaged over the 30 development individuals and also for $\lambda_1 = 0.4, 0.5, 0.6$. For $m = 4$, the solver took 1 minute and 47 seconds on average per text; recall that $|F|$ is also much larger now, compared to the experiments of the previous section. For $m = 5$, the solver was so slow that we aborted the experiment. Figure 6 shows the average solver times for different numbers of available facts $|F|$, for $m = 3$; in this case, we modified the set of available facts (F) of every individual to contain $3, 6, 9, 12, 15, 18, 21$ facts. The results are again averaged over the 30 development individuals and for $\lambda_1 = 0.4, 0.5, 0.6$. Although the times of Fig. 6 also grow exponentially to $|F|$, they remain under 4 seconds, showing that the main factor to the complexity of ILPNLG is m, the number of fact subsets, i.e., the maximum allowed number of (aggregated) sentences of each text.

To efficiently generate texts with larger m values, we developed ILPNLGAPPROX, the approximation of ILPNLG that considers each fact subset separately (Section 4). Figures 7–8 show the average solver times of ILPNLGAPPROX for different values of m and $|F|$, respectively; all the other settings are as in Fig. 5–6. The solver times now grow approximately linearly to m and $|F|$ and are under 0.3 seconds in all cases.

In Figure 9, we compare ILPNLG to ILPNLGAPPROX, by showing their average fact per word ratios, computed as in Fig. 3 (Section 5.4). We set $m = 3$ in ILPNLG to keep the solving times low; in ILPNLGAPPROX we experimented with both $m = 3$ (the value used in ILPNLG) and $m = 10$ (the value that was actually needed). In all cases, $B_{max} = 39$. The facts per word ratios of all three systems are very similar. We conclude that ILPNLGAPPROX achieves very similar results to ILPNLG in much less time.

Figures 10 and 11 show the facts per word ratios of ILPNLGAPPROX ($m = 10$), PIPELINE, and PIPELINESHORT, computed in two ways, as in Section 5.4, for the texts of the 30 development individuals. Again, PIPELINESHORT achieves slightly better results than PIPELINE. The behavior of ILPNLGAPPROX in Figure 10 is very similar to the behavior of ILPNLG on the Wine Ontology (Fig. 3); for $\lambda_1 \leq 0.3$ it produces empty texts, while for $\lambda_1 \geq 0.4$ it performs better than the other systems. ILPNLGAPPROX obtains the highest facts per word ratio for $\lambda_1 = 0.45$, where it selects the facts and sentence plans that lead to the most compressive aggregations. For greater values of λ_1, it selects additional facts whose sentence plans do not aggregate that well, which is why the

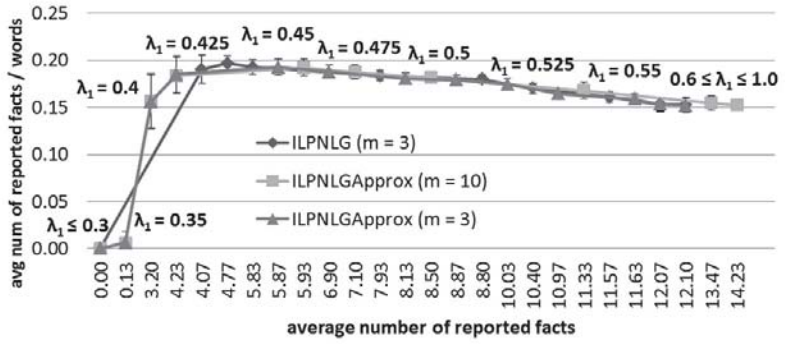

Figure 9
Comparing the facts per word ratios of ILPNLGAPPROX and ILPNLG in texts generated from the Consumer Electronics ontology.

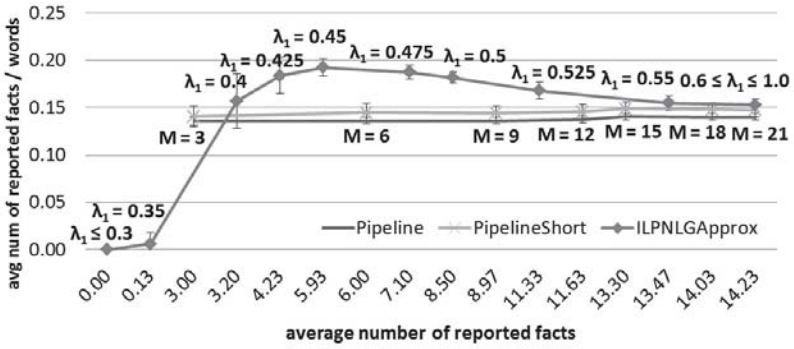

Figure 10
Facts per word ratios for the Consumer Electronics Ontology (grouped by M or λ_1 values).

ratio declines. The two pipeline systems select facts and sentence plans that offer very few aggregation opportunities; as the number of selected facts increases, some more aggregation opportunities arise, which is why the facts per word ratio of the two systems improves (more clearly in Fig. 11). Figure 11 also shows that ILPNLGAPPROX generates more compact texts than PIPELINE and PIPELINESHORT.

We show below three example texts produced by PIPELINE, PIPELINESHORT (both with $M = 6$), and ILPNLGAPPROX ($\lambda_1 = 0.45$, $m = 10$). Each text reports six facts, but ILPNLGAPPROX has selected facts and sentence plans that allow more compressive aggregations. Recall that we treat all the facts as equally important. If importance scores are also available (e.g., if dimensions are less important), they can be added as multipliers $imp(f_i)$ of α_i in the objective function (Eq. 4) of the ILP model.

PIPELINE: SonySony DCR-TRV270 requires minimum illumination of 4.0 lux and its display is 2.5 in. It features a Sports scene mode, it includes a microphone and an IR remote control. Its weight is 780.0 grm.

PIPELINESHORT: Sony DCR-TRV270 requires minimum illumination of 4.0 lux and its display is 2.5 in. It features a Sports scene mode, it includes a microphone and an IR remote control. It weighs 780.0 grm.

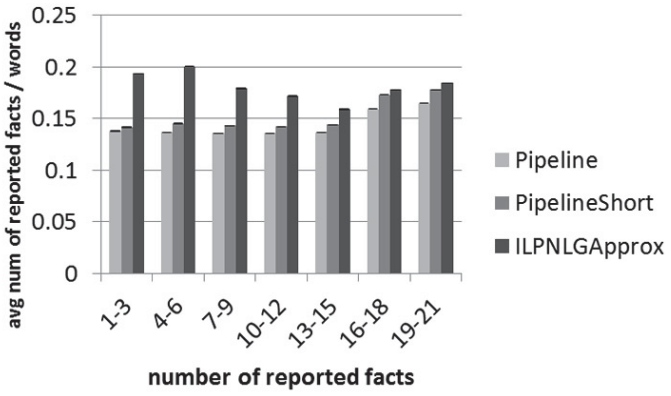

Figure 11
Facts per word ratios for the Consumer Electronics Ontology (grouped by reported facts).

ILPNLGAPPROX: Sony DCR-TRV270 has a microphone and an IR remote control. It is 98.0 mm high, 85.0 mm wide, 151.0 mm deep and it weighs 780.0 grm.

We showed the $30 \times 2 = 60$ texts of PIPELINESHORT ($M = 6$) and ILPNLGAPPROX ($\lambda_1 = 0.45$, $m = 10$) to the same six students that participated in the experiments with the Wine Ontology (Section 5.4). Again, each text was given to exactly one student. Each student was given approximately 5 randomly selected texts of each system. The OWL statements were not shown, and the students did not know which system had generated each text. Each student was shown all of his/her texts in random order, regardless of the system that generated them. The students were asked to score each text by stating how strongly they agreed or disagreed with statements S_1–S_3, as in Section 5.4. They were also asked to provide an overall score (1–5) per text.

Table 2 shows the average scores of the two systems with 95% confidence intervals. For each criterion, the best score is shown in bold; the confidence interval of the best score is also shown in bold, if it does not overlap with the confidence interval of the other system. Unlike the Wine Ontology experiments (Table 1), the scores of our ILP approach (with the approximation of ILPNLGAPPROX) are now higher than those of PIPELINESHORT in all of the criteria, and the differences are also larger, though we found the differences to be statistically significant only for clarity and overall quality.[28] We attribute these larger differences, compared to the Wine Ontology experiments, to the fact that the texts are now longer and the sentence plans more varied, which often makes the texts of PIPELINESHORT sound verbose and, hence, more difficult to follow, compared to the more compact texts of ILPNLGAPPROX, which sound more concise.

Overall, the human scores of the experiments with the Wine and Consumer Electronics ontologies suggest that the higher facts per word ratios of our ILP approach do not come at the expense of lower perceived text quality. On the contrary, the texts of the

28 When two confidence intervals do not overlap, the difference is statistically significant. When they overlap, the difference may still be statistically significant; we performed Analysis of Variance (ANOVA) and post-hoc Tukey tests to check for statistically significant differences in those cases. A post-hoc power analysis of the ANOVA values resulted in power values greater or equal to 0.95.

Table 2
Human scores for Consumer Electronics texts.

Criteria	PIPELINESHORT	ILPNLGAPPROX
Sentence fluency	4.50 ± 0.30	$\mathbf{4.87} \pm 0.12$
Text structure	4.33 ± 0.36	$\mathbf{4.73} \pm 0.22$
Clarity	4.53 ± 0.29	$\mathbf{4.97 \pm 0.06}$
Overall	4.10 ± 0.31	$\mathbf{4.73 \pm 0.16}$

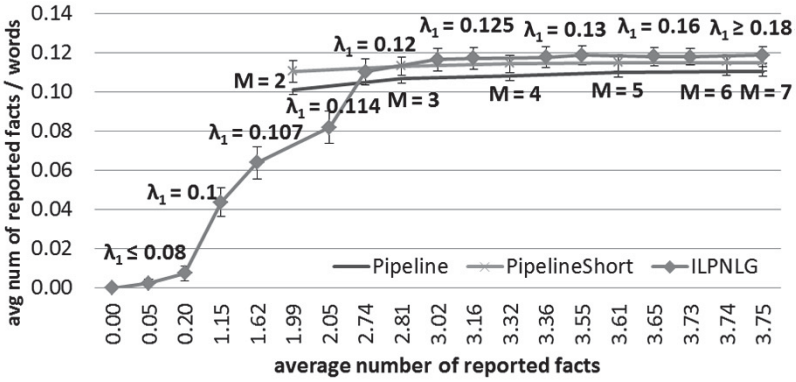

Figure 12
Facts per word ratios (grouped by M or λ_1 values) of ILPNLG, PIPELINE, and PIPELINESHORT for texts generated from the Disease Ontology.

ILP approach may be perceived as clearer and overall better than those of the pipeline, when the texts report many facts.

5.6 Experiments with the Disease Ontology

In a third set of experiments, we generated texts for the 200 'development' classes (Section 5.1) of the Disease Ontology, using the manually authored domain-dependent generation resources (e.g., text plans, NL names, sentence plans) of Evaggelakaki (2014), but with additional sentence plans we constructed to ensure that there were three alternative sentence plans per relation. We generated texts with ILPNLG, PIPELINE, and PIPELINESHORT, for $M = 2, 3, 4, \ldots, 7$ in the two pipeline systems, and different values of λ_1 ($\lambda_2 = 1 - \lambda_1$) in ILPNLG. All three systems were allowed to form aggregated sentences with up to $B_{max} = 30$ distinct elements; this was the number of distinct elements of the longest aggregated sentence in the experiments of Evaggelakaki (2014), where PIPELINE was allowed to combine up to three simple (expressing one fact each) sentences to form an aggregated one. There are 3.7 available facts ($|F|$) on average and a maximum of 7 facts for each one of the 200 classes. In ILPNLG, we set $m = 4$, which was the maximum number of (aggregated) sentences in the texts of PIPELINE and PIPELINESHORT. We did not use ILPNLGAPPROX, since ILPNLG was reasonably fast (average solver time: 0.11 sec per text, worst: 0.90 sec per text), because of the smaller values of m and $|F|$, compared to the experiments of the Consumer Electronics ontology.

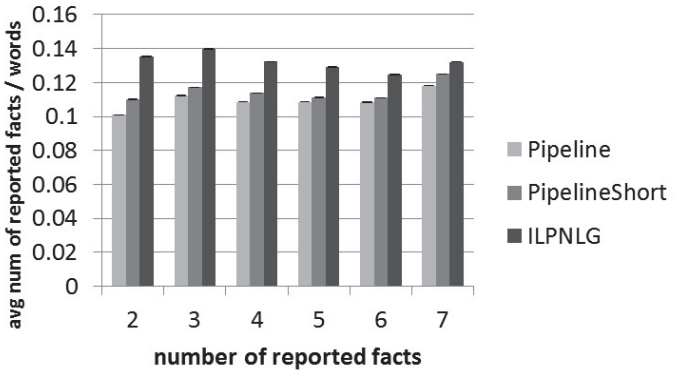

Figure 13
Facts per word ratios (grouped by reported facts) of ILPNLG, PIPELINE, and PIPELINESHORT for texts generated from the Disease Ontology.

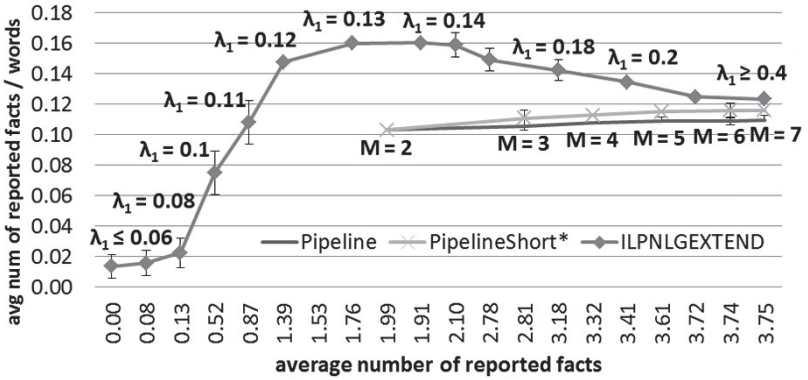

Figure 14
Facts per word ratios (grouped by M or λ_1 values) of ILPNLGEXTEND, PIPELINE, and PIPELINESHORT* for texts generated from the Disease Ontology.

Figures 12 and 13 show the facts per word ratios of ILPNLG, PIPELINE, and PIPELINESHORT, computed in two ways, as in Section 5.4, for the texts of the 200 classes. PIPELINESHORT achieves only slightly better results than PIPELINE in both figures. Also, Fig. 13 shows that ILPNLG produces more compact texts than the two pipeline systems. In Figure 12, however, the difference between ILPNLG and the two pipeline systems is less clear. For small λ_1 values, ILPNLG produces empty texts, because it focuses on minimizing the number of distinct elements of each text. For $\lambda_1 \geq 0.125$, it performs only marginally better than PIPELINESHORT, unlike previous experiments (cf. Fig. 3 and 10). We attribute this difference to the fact that ILPNLG does not take into account the lengths of the NL names, which vary a lot in the Disease Ontology; nor does it take into account that the O of many facts $\langle S, R, O \rangle$ is a conjunction. These issues were addressed in our extended ILP model (Section 3.2), which is used in ILPNLGEXTEND.

We then generated texts for the 200 classes again, this time with PIPELINE, PIPELINESHORT* (both with $M = 2, 3, \ldots, 7$, $W_{max} = 54$) and ILPNLGEXTEND ($m = 4$, $W_{max} =$

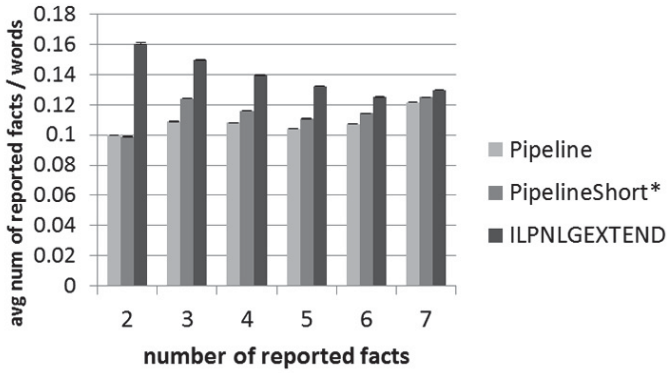

Figure 15
Facts per word ratios (grouped by reported facts) of ILPNLGEXTEND, PIPELINE, and PIPELINESHORT* for texts generated from the Disease Ontology.

54); we modified PIPELINE and PIPELINESHORT to count words (instead of elements) when comparing against ILPNLGEXTEND, which is why we report W_{max} in all three systems. Similarly to how B_{max} was previously selected, $W_{max} = 54$ was the number of words of the longest aggregated sentence in the experiments of Evaggelakaki (2014). Figures 14 and 15 show the new facts per word ratios, for the texts of the 200 classes. In Figure 14, for $\lambda_1 \leq 0.06$, ILPNLGEXTEND produces empty texts, because it focuses on minimizing the lengths of the texts. For $\lambda_1 \geq 0.12$, ILPNLGEXTEND now performs clearly better than the pipeline systems, obtaining the highest facts per word ratio for $\lambda_1 = 0.14$; notice that we now compare to PIPELINESHORT*, which is a better baseline for ILPNLGEXTEND than PIPELINESHORT (Section 5.2). Figure 15 also confirms that ILPNLGEXTEND outperforms the pipeline systems. The ILP solver was actually slightly faster with ILPNLGEXTEND (average: 0.09 sec, worst: 0.65 sec per text) compared to ILPNLG (average: 0.11 sec, worst: 0.90 sec per text).

We show below three example texts produced by PIPELINE, PIPELINESHORT* (both with $M = 3$), and ILPNLGEXTEND ($\lambda_1 = 0.14$). Each text reports three facts, but ILPNLGEXTEND has selected facts with fewer and shorter NL names, and sentence plans that lead to better sentence aggregation. Recall that we treat all facts as equally important in these experiments, but that our ILP models can also handle importance scores (e.g., treating facts reporting symptoms as more important than facts about is-a relations).

PIPELINE: Nephropathia epidemica can be found in the kidneys. It can often cause myalgia, nausea, renal failure, vomiting, abdominal pain, headaches, internal hemorrhage and back pain, and it results in infections.
PIPELINESHORT*: Nephropathia epidemica is a kind of hemorrhagic fever with renal syndrome. It originates from bank voles and it is caused by the puumala virus.
ILPNLGEXTEND: Nephropathia epidemica results in infections. It often originates from bank voles from the puumala virus.

5.7 Further Experiments with the Wine Ontology

The experiments of the previous section tested the ability of ILPNLGEXTEND to take into account the different lengths of NL names and the fact that some facts $\langle S, R, O \rangle$

involve conjunctions (or disjunctions) in their O. They did not, however, test the ability of ILPNLGEXTEND to cope with multiple alternative NL names per individual or class. The Consumer Electronics and Disease Ontologies were inappropriate in this respect, because the names of electronic products tend to be unique and we did not have the expertise to create alternative names of diseases, as already noted. Instead, we returned to the Wine Ontology, which had been used in Section 5.4 with a single NL name per individual and class. We now added more NL names to the Wine Ontology to ensure that approximately three NL names on average (with a minimum of 2 and a maximum of 5) were available for each one of the individual and classes we generated texts for. We generated texts for the 52 wine individuals and 24 of the wine classes of the Wine Ontology, using PIPELINE, PIPELINESHORT*, and ILPNLGEXTEND.[29]

All three systems were allowed to form aggregated sentences with up to $W_{max} = 26$ words; again, we modified PIPELINE and PIPELINESHORT to count words (instead of elements) when comparing against ILPNLGEXTEND, which is why we report W_{max} for all three systems. Similarly to Section 5.6, W_{max} was set to the number of words of the longest aggregated sentence in the experiments of our previous work (Androutsopoulos, Lampouras, and Galanis 2013), where PIPELINE was allowed to combine up to three simple (expressing one fact each) sentences to form an aggregated one. In ILPNLGEXTEND, we used different values for λ_1 ($\lambda_2 = 1 - \lambda_1$), setting $m = 3$, as in Section 5.4. In PIPELINE and PIPELINESHORT*, we used $M = 2, 3, \ldots, 7$.[30] For each M value, the texts of PIPELINE for the 76 individuals and classes were generated 10 times (not 3, unlike all the previous experiments with PIPELINE); each time, we used one of the different alternative sentence plans for each relation and one of the different alternative NL names for the individual or class the text was being generated for, since PIPELINE cannot select among alternative NL names (and sentence plans) by itself.

Figures 16 and 17 show the facts per word ratios, computed in two ways, as in Section 5.4. In Fig. 16, for $\lambda_1 < 0.04$, ILPNLGEXTEND produces empty texts, because it focuses on minimizing the length of each text. For $\lambda_1 \geq 0.08$, it performs clearly better than the other systems. For $\lambda_1 = 0.12$, it obtains the highest facts per word ratio by selecting the facts and sentence plans that lead to the shortest (in words) aggregated sentences, and NL names that indirectly express facts (not requiring separate sentences). For greater values of λ_1, ILPNLGEXTEND selects additional facts whose sentence plans do not aggregate that well or that cannot be indirectly expressed via NL names, which is why the ratio of ILPNLGEXTEND declines. We note that the highest average facts per word ratio of ILPNLGAPPROX (0.37, for $\lambda_1 = 0.12$) of Fig. 16 is higher than the highest average ratio (0.33, for $\lambda_1 = 0.3$) we had obtained in Section 5.4 with ILPNLG (Fig. 3). Also, the overall values of λ_1 are now smaller. This is due to the larger number of factors in the right part of the objective function (Eq. 13) of ILPNLGEXTEND. Figure 17 confirms that ILPNLGEXTEND outperforms the pipelines. In the experiments of this section with ILPNLGEXTEND, the ILP solver was very fast (average: 0.06 sec, worst: 0.64 sec per text).

We show below texts produced by PIPELINE, PIPELINESHORT* (both with $M = 4$), and ILPNLGEXTEND ($\lambda_1 = 0.12$). All texts describe the same wine and report four facts.

29 In the experiments of Section 5.4, we had not generated texts for wine classes, because we could not think of alternative sentence plans for their axioms. In the experiments of this section, we generated texts for 24 (out of 63) wine classes, because we were able to provide alternative NL names for them.

30 Generating texts for the additional 24 classes required raising the maximum M value to 7, unlike the experiments of Section 5.4, where it was 6.

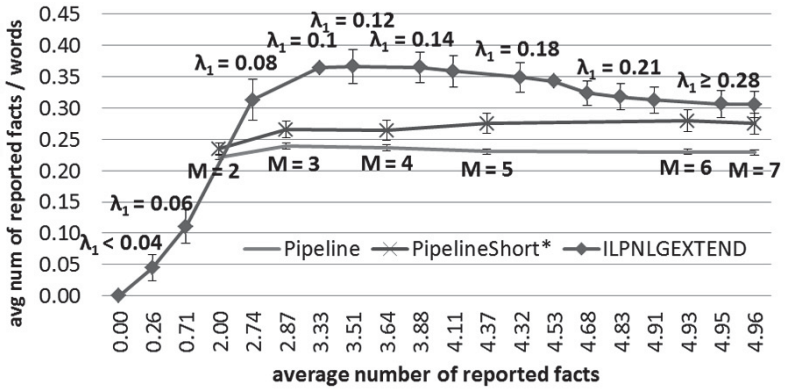

Figure 16
Fact per word ratios (grouped by M or λ_1 values) of the additional Wine Ontology experiments.

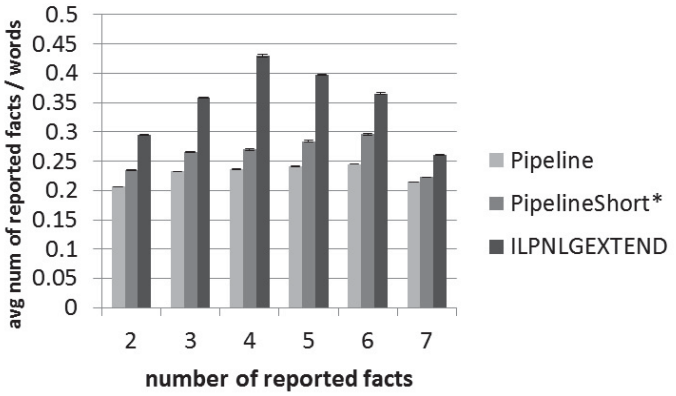

Figure 17
Fact per word ratios (grouped by reported facts) of the additional Wine Ontology experiments.

PIPELINE: This Sauvignon Blanc is dry and medium. It is made by Stonleigh and it is produced in New Zealand.
PIPELINESHORT*: This delicate tasting and dry Sauvignon Blanc wine originates from New Zealand.
ILPNLGEXTEND: This Stonleigh Sauvignon Blanc is dry, delicate and medium.

ILPNLGEXTEND chose an NL name that avoids expressing the maker as a separate sentence, and used the same verb ("is") to express the other three facts, allowing a single aggregated sentence to be formed. It also avoided expressing the origin (New Zealand), which would require a long sentence that would not aggregate well with the others.

6. Related Work

Marciniak and Strube (2005) proposed an ILP approach to language processing problems where the decisions of classifiers that consider different, but co-dependent, sub-

tasks need to be combined. They applied their approach to the generation of multi-sentence route directions, by training classifiers (whose decisions affect the generated text) on a parallel corpus consisting of semantic representations and route directions. The classifiers control the ordering and lexicalization of phrases and a simple form of aggregation (mainly the choice of connectives between the phrases). Marciniak and Strube aimed to generate fluent and grammatically correct texts; by contrast, our ILP models employ manually authored linguistic resources that guarantee fluent and grammatical texts (as also confirmed by our experiments), and make no decisions directly affecting fluency or grammaticality. Instead, our models make decisions related to content selection, lexicalization, aggregation (using more complex rules than Marciniak and Strube), and a limited form of referring expression generation (in the case of our extended model), aiming to produce more compact texts, without invoking classifiers.

Barzilay and Lapata (2005) treated content selection as an optimization problem. Given a pool of facts (database entries) and scores indicating the importance of including or excluding each fact or pair of facts, their method selects the facts to express by solving an optimization problem similar to energy minimization. A solution is found by applying a minimal cut partition algorithm to a graph representing the pool of facts and the importance scores. The importance scores of single facts are obtained via supervised machine learning (AdaBoost) from a dataset of (sports) facts and news articles expressing them. The importance scores of pairs of facts depend on parameters tuned on the same dataset using Simulated Annealing. Our ILP models are simpler, in that they allow importance scores to be associated only with single facts, not pairs of facts. On the other hand, our models jointly perform content selection, lexicalization, aggregation, and (limited) referring expression generation, not just content selection.

In other work, Barzilay and Lapata (2006) consider sentence aggregation. Given a set of facts (again database entries) that a content selection stage has produced, aggregation is viewed as the problem of partitioning the facts into optimal subsets (similar to the buckets of our ILP models). Sentences expressing facts of the same subset are aggregated to form a longer sentence. The optimal partitioning maximizes the pairwise similarity of the facts in each subset, subject to constraints that limit the number of subsets and the number of facts in each subset. A Maximum Entropy classifier predicts the semantic similarity of each pair of facts, and an ILP model is used to find the optimal partitioning. By contrast, our ILP models aggregate sentences by minimizing the distinct elements of each subset, to maximize the aggregation opportunities in each subset, taking care not to aggregate together sentences expressing facts from different topics; an external text planner partitions the available facts into topical sections. Again, our models have broader scope, in the sense that they (jointly) perform content selection, lexicalization, aggregation, and (limited) referring expression generation, not just aggregation.

Althaus et al. (2004) show that the ordering of a set of sentences to maximize local (sentence-to-sentence) coherence is equivalent to the traveling salesman problem and, hence, NP-complete. They also provide an ILP formulation of the problem, which can be solved efficiently in practice using branch-and-cut with cutting planes. Our models do not order the sentences (or facts) of the generated texts, relying on an external text planner instead. It would be particularly interesting to add sentence (or fact) ordering to our models, along the lines of Althaus et al. in future work.

Kuznetsova et al. (2012) use ILP to generate image captions. They train classifiers to detect the objects in each image. Having identified the objects of a given image, they retrieve phrases from the captions of a corpus of images, focusing on the captions of objects that are similar (color, texture, shape) to the ones in the given image. To select which objects of the image to report (a kind of content selection) and in what order,

Kuznetsova et al. maximize (via ILP) the mean of the confidence scores of the object detection classifiers and the sum of the co-occurrence probabilities of the objects that will be reported in adjacent positions in the caption. The co-occurrence probabilities are estimated from a corpus of captions. Having decided which objects to report and their order, a second ILP model decides which phrases to use for each object (a kind of lexicalization) and orders the phrases. The second ILP model maximizes the confidence of the phrase retrieval algorithm and the local cohesion between subsequent phrases. Although generating image captions is very different to generating texts from ontologies, it may be possible to use ideas from the work of Kuznetsova et al. related to ordering objects (in our case, facts) and phrases in future extensions of our models.

Joint optimization ILP models have also been used in multi-document text summarization and sentence compression (McDonald 2007; Clarke and Lapata 2008; Berg-Kirkpatrick, Gillick, and Klein 2011; Galanis, Lampouras, and Androutsopoulos 2012; Woodsend and Lapata 2012), where the input is text, not formal knowledge representations. Statistical methods to jointly perform content selection, lexicalization, and surface realization have also been proposed in NLG (Liang, Jordan, and Klein 2009; Konstas and Lapata 2012a, 2012b), but they are currently limited to generating single sentences from flat records, as opposed to generating multi-sentence texts from ontologies.

To the best of our knowldge, our work is the first to consider content selection, lexicalization, sentence aggregation, and a limited form of referring expression generation as an ILP joint optimization problem in multi-sentence concept-to-text generation. An earlier form of our work has already been published (Lampouras and Androutsopoulos 2013b, 2013a), but without the extended version of our ILP model (Section 3.2), without the experiments on the Disease Ontology (Section 5.6), without the further experiments on the Wine Ontology (Section 5.7), with facts per word ratios grouped only by M and λ_1 values (without the results of Fig. 4, 11, 15, 17), and with much fewer details.

7. Conclusions and Future Work

We presented an ILP model that jointly considers decisions in content selection, lexicalization, and sentence aggregation to avoid greedy local decisions and produce more compact texts. An extended version of the ILP model predicts more accurately the lengths of the generated texts and also performs a limited form of referring expression generation, by considering alternative NL names and how they can indirectly express facts. We also defined an approximation of our models that generates separately each (possibly aggregated) sentence of the final text and is more efficient when longer texts are generated. The ILP models (and approximations) of this article were embedded in NaturalOWL, a state of the art publicly available NLG system for OWL ontologies that used a pipeline architecture in its original form. Experiments with three ontologies confirmed that our models can express more facts per word, with no deterioration in the perceived quality of the generated texts or with improved perceived quality, compared to texts generated by a pipeline architecture. Our experiments also showed that our ILP methods (or their approximations) are efficient enough to be used in practice.

The work of this article is the first to consider content selection, lexicalization, sentence aggregation, and a limited form of referring expression generation as an ILP joint optimization problem in multi-sentence concept-to-text generation. Previous work in NLG employed a pipeline architecture, considered fewer and different processing stages, was concerned with generating single sentences, or had very different inputs and goals. Our work could be extended to consider additional generation stages (e.g., text planning, or more referring expression generation decisions). It would also be interest-

ing to combine the ILP models with other user modeling components that would assign interest scores to message triples. Another valuable direction would be to combine ILP models for concept-to-text generation and multi-document summarization, to produce texts summarizing both structured and unstructured information.

References

Althaus, E., N. Karamanis, and A. Koller. 2004. Computing locally coherent discourses. In *42nd Annual Meeting of ACL*, pages 399–406, Barcelona, Spain.

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

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

Baader, F., D. Calvanese, D.L. McGuinness, D. Nardi, and P.F. Patel-Schneider, editors. 2002. *The Description Logic Handbook*. Cambridge University Press.

Barzilay, R. and M. Lapata. 2005. Collective content selection for concept-to-text generation. In *Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 331–338, Vancouver, British Columbia, Canada.

Barzilay, R. and M. Lapata. 2006. Aggregation via set partitioning for natural language generation. In *Conference on Human Language Technology and the Annual Conference of the North American Chapter of ACL*, pages 359–366, New York, NY.

Belz, A. 2008. Automatic generation of weather forecast texts using comprehensive probabilistic generation-space models. *Natural Language Engineering*, 14(4):431–455.

Berg-Kirkpatrick, T., D. Gillick, and D. Klein. 2011. Jointly learning to extract and compress. In *49th Annual Meeting of ACL: Human Language Technologies*, pages 481–490, Portland, Oregon.

Berners-Lee, T., J. Hendler, and O. Lassila. 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*, pages 531–545, Heraklion, Greece.

Clarke, J. and M. Lapata. 2008. Global inference for sentence compression: an integer linear programming approach. *Journal of Artificial Intelligence Research*, 1(31):399–429.

Corston-Oliver, S. 2001. Text compaction for display on very small screens. In *Workshop on Automatic Summarization of Annual Conference of the North American Chapter of ACL*, Pittsburgh, PA.

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

Danlos, L. 1984. Conceptual and linguistic decisions in generation. In *10th International Conference on Computational Linguistics*, pages 501–504, Stanford, CA.

Dantzig, G. B. 1963. *Linear Programming and Extensions*. Princeton University Press.

Demir, S., S. Carberry, and K.F. McCoy. 2010. A discourse-aware graph-based content-selection framework. In *6th International Natural Language Generation Conference*, pages 17–25, Trim, Co. Meath, Ireland.

Evaggelakaki, M. 2014. Generation of natural language texts from biomedical ontologies with the NaturalOWL system. BSc thesis, Department of Informatics, Athens University of Economics and Business, in Greek.

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

Galanis, D., G. Karakatsiotis, G. Lampouras, and I. Androutsopoulos. 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)*, pages 17–20, Athens, Greece.

Galanis, D., G. Lampouras, and I. Androutsopoulos. 2012. Extractive multi-document summarization with ILP and Support Vector Regression. In *24th International Conference on Computational Linguistics*, pages 911–926, Mumbai, India.

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

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

Halaschek-Wiener, C., J. Golbeck, B. Parsia, V. Kolovski, and J. Hendler. 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., P.F. Patel-Schneider, and F. van Harmelen. 2003. From SHIQ and RDF to OWL: the making of a Web Ontology Language. *Web Semantics*, 1(1):7–26.
Kaljurand, K. and N. Fuchs. 2007. Verbalizing OWL in Attempto Controlled English. In *3rd International Workshop on OWL: Experiences and Directions*, Innsbruck, Austria.
Karp, RichardM. 1972. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, The IBM Research Symposia Series. Springer US, pages 85–103.
Konstas, I. and M. Lapata. 2012a. Concept-to-text generation via discriminative reranking. In *50th Annual Meeting of ACL*, pages 369–378, Jeju Island, Korea.
Konstas, I. and M. Lapata. 2012b. Unsupervised concept-to-text generation with hypergraphs. In *Conference on Human Language Technology of the Annual Conference of the North American Chapter of ACL*, pages 752–761, Montréal, Canada.
Kuznetsova, P., V. Ordonez, A. Berg, T. Berg, and Y. Choi. 2012. Collective generation of natural image descriptions. In *50th Annual Meeting of ACL*, pages 359–368, Jeju Island, Korea.
Lampouras, G. 2015. *Natural Language Generation from Semantic Web Ontologies*. Ph.D. thesis, Department of Informatics, Athens University of Economics and Business, Greece.
Lampouras, G. and I. Androutsopoulos. 2013a. Using integer linear programming for content selection, lexicalization, and aggregation to produce compact texts from owl ontologies. In *14th European Workshop on Natural Language Generation*, pages 51–60, Sofia, Bulgaria.
Lampouras, G. and I. Androutsopoulos. 2013b. Using integer linear programming in concept-to-text generation to produce more compact texts. In *51st Annual Meeting of ACL*, pages 561–566, Sofia, Bulgaria.
Liang, P., M. Jordan, and D. Klein. 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*, pages 91–99, Suntec, Singapore.
Liang, S.F., R. Stevens, D. Scott, and A. Rector. 2011. Automatic verbalisation of SNOMED classes using OntoVerbal. In *13th Conference on Artificial Intelligence in Medicine*, pages 338–342, Bled, Slovenia.
Marciniak, T. and M. Strube. 2005. Beyond the pipeline: Discrete optimization in NLP. In *9th Conference on Computational Natural Language Learning*, pages 136–143, Ann Arbor, MI.
McDonald, R. 2007. A study of global inference algorithms in multi-document summarization. In *European Conference on Information Retrieval*, pages 557–564, Rome, Italy.
Mellish, C. and J.Z. Pan. 2008. Natural language directed inference from ontologies. *Artificial Intelligence*, 172(10):1285–1315.
Mellish, C., D. Scott, L. Cahill, D. Paiva, R. Evans, and M. Reape. 2006. A reference architecture for natural language generation systems. *Natural Language Engineering*, 12(1):1–34.
Mellish, C. and X. Sun. 2006. The Semantic Web as a linguistic resource: opportunities for natural language generation. *Knowledge Based Systems*, 19(5):298–303.
Power, R. 2010. Complexity assumptions in ontology verbalisation. In *48th Annual Meeting of ACL (short papers)*, pages 132–136, Uppsala, Sweden.
Power, R. and A. Third. 2010. Expressing OWL axioms by English sentences: Dubious in theory, feasible in practice. In *23rd International Conference on Computional Linguistics*, pages 1006–1013, Beijing, China.
Reiter, E. and R. Dale. 2000. *Building Natural Language Generation systems*. Cambridge University Press.
Roth, D. and W.T. Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *Conference on Human Language Technology and the Annual Conference of the North American Chapter of ACL*, Boston, MA.
Schrijver, Alexander. 1986. *Theory of Linear and Integer Programming*. John Wiley & Sons, Inc.
Schutte, N. 2009. Generating natural language descriptions of ontology concepts. In *12th European Workshop on Natural Language Generation*, pages 106–109, Athens, Greece.
Schwitter, R. 2010. Controlled natural languages for knowledge representation. In *23rd International Conference on Computational Linguistics (posters)*, pages 1113–1121, Beijing, China.
Schwitter, R., K. Kaljurand, A. Cregan, C. Dolbear, and G. Hart. 2008. A comparison of three controlled natural languages for OWL 1.1. In *4th OWL: Experiences and Directions Workshop*, Washington DC.

Shadbolt, N., T. Berners-Lee, and W. Hall. 2006. The Semantic Web revisited. *IEEE Intell. Systems*, 21(3):96–101.
Stevens, R., J. Malone, S. Williams, R. Power, and A. Third. 2011. Automatic generation of textual class definitions from OWL to English. *Biomedical Semantics*, 2(S 2:S5).
Thomaidou, S. 2014. *Automated Creation and Optimization of Online Advertising Campaigns*. Ph.D. thesis, Department of Informatics, Athens University of Economics and Business.
Thomaidou, S., I. Lourentzou, P. Katsivelis-Perakis, and M. Vazirgiannis. 2013. Automated snippet generation for online advertising. In *22nd ACM international conference on Conference on information & knowledge management*, pages 1841–1844, San Francisco, California, USA.
Tsatsaronis, G., M. Schroeder, G. Paliouras, Y. Almirantis, I. Androutsopoulos, E. Gaussier, P. Gallinari, T. Artieres, M. Alvers, M. Zschunke, and A. Ngonga. 2012. BioASQ: A challenge on large-scale biomedical semantic indexing and question answering. In *AAAI Fall Symposium on Information Retrieval and Knowledge Discovery in Biomedical Text*, pages 92–98, Arlington, VA, USA.
Vandeghinste, V. and Pan Y. 2004. Sentence compression for automated subtitling: A hybrid approach. In *Workshop on Text Summarization of the 42nd Annual Meeting of ACL*, pages 89–95, Barcelona, Spain.
Walker, M.A., O. Rambow, and M. Rogati. 2001. Spot: A trainable sentence planner. In *2nd Annual Conference of the North American Chapter of ACL*, pages 17–24, Pittsburgh, PA.
Williams, S., A. Third, and R. Power. 2011. Levels of organization in ontology verbalization. In *13th European Workshop on Natural Language Generation*, pages 158–163, Nancy, France.
Woodsend, K. and M. Lapata. 2012. Multiple aspect summarization using integer linear programming. In *Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 233–243, Jesu Island, Korea.