Generating Natural Language from Linked Data: Unsupervised template extraction

Daniel Duma Ewan Klein
School of Informatics, University of Edinburgh
danielduma@gmail.com, ewan@inf.ed.ac.uk

Abstract

We propose an architecture for generating natural language from Linked Data that automatically learns sentence templates and statistical document planning from parallel RDF datasets and text. We have built a proof-of-concept system (LOD-DEF) trained on un-annotated text from the Simple English Wikipedia and RDF triples from DBpedia, focusing exclusively on factual, non-temporal information. The goal of the system is to generate short descriptions, equivalent to Wikipedia stubs, of entities found in Linked Datasets. We have evaluated the LOD-DEF system against a simple generate-from-triples baseline and human-generated output. In evaluation by humans, LOD-DEF significantly outperforms the baseline on two of three measures: non-redundancy and structure and coherence.

1 Introduction

In recent years, work on the Semantic Web has undergone something of a split. At one end of the continuum, considerable energy has been invested into the construction of detailed domain ontologies expressed in some variant of OWL,[1] with considerable attention paid to maintaining logical consistency. At the other end, the so-called Linked Data framework has given rise to the publication of quite large scale datasets, with relatively little concern for ensuring consistency. Although the language, namely RDF,[2] in which Linked Data is encoded can be regarded as a restricted form of first-order predicate logic, existing Linked Datasets are closer in many ways to large, distributed databases than the kind of carefully constructed knowledge base that is familiar from AI research. The work that we report here takes as its starting point the following question: can Linked Data be used as the input to a Natural Language Generation (NLG) system? There are at least a couple of reasons why a positive answer would be interest. First, it is still relatively hard for non-experts to browse unfamiliar Linked Data sets, and a natural language representation could potentially ameliorate this problem. Second, cultural heritage institutions (e.g., museums, art galleries, and libraries) are increasingly interested in publishing their data in the form of Linked Data.[3] Such institutions are typically committed to presenting information about their holdings in multiple forms, and consequently generation of natural language that utilises a single Linked Data source would be highly attractive. Moreover, the very nature of Linked Data should make it easier for an NLG system to supplement institution-specific data with encyclopaedic information drawn from sources such as DBpedia.[4]

In the light of these motivations, we propose an architecture for a trainable NLG system for Linked Data that can automatically learn sentence templates and document planning from parallel RDF data and text, with the communicative goal of describing Wikipedia-style factual descriptions of entities.

[1] http://www.w3.org/TR/owl2-overview/
[2] We describe RDF in more detail in the next section
[3] See http://museum-api.pbworks.com for many examples.
[4] http://dbpedia.org

2 Background

2.1 Natural Language Generation

Natural Language Generation is the task of producing natural language text from an input of domain-dependent semantic information. Reiter and Dale (2000) describe a modular architecture for a deep NLG system as a pipeline composed of three main modules: (1) document planning, where the information to include in the final text is selected (*content determination*) and ordered (*text structuring*), (2) *microplanning*, where lexicalisation, referring expression generation and aggregation (coordination and subordination of sentences) are performed, and (3) *surface realisation*, where possible verbalisations are typically over-generated, ranked and selected. This has been called "deep" NLG, in contrast to "shallow" methods based on templates.

We will adopt some of this terminology, but nevertheless propose a shallow approach, in which text realisation uses RDF data to fill slots in sentence templates. Templates are learned on the basis of parallel text-data — cf. the Text-Knowledge Resources (TKRs) proposed by Duboue and Mckeown (2003). They describe these as "a set of human written text and knowledge base pairs", where the knowledge base includes data that a concept-to-text system could use to generate output achieving the same communicative goals as the human-authored text. Wide-scale unsupervised learning of templates and document plans opens the prospect of designing NLG systems that are inexpensive to develop and deploy, easier to transfer to other domains, and potentially multilingual.

The system is trained from a TKR by performing four main actions. First it aligns the text and the data by matching data values from RDF statements with strings in the text. Second, it extracts and modifies sentences that express these values, so as to build a set of sentence templates. Third, it collects statistics about how the matched values are ordered in text. Finally, it determines the class of entity that a TKR pair describes and assembles a set of templates and associated information to form a 'model' for that class.

We describe LOD-DEF, a proof-of-concept system trained on text from the Simple English Wikipedia[5] and data from DBpedia (Mendes et al., 2012). Our goal is to demonstrate that such an architecture is feasible, and that a system trained in this way can generate texts which are perceived as intelligible and informative by human judges.

2.2 Linked Data and RDF

The term *Linked Data* refers to a set of best practices for publishing and interlinking structured data on the Web (Heath and Bizer, 2011). These so-called "Linked Data Principles" mandate the use of a number of web-based open formats for publishing data, such as HTTP as a transport layer and the Resource Description Framework (RDF)[6] for representing and linking datasets. The central claim of Linked Data is that the general architecture of the World Wide Web can be generalised to the task of sharing structured data on global scale. The rate of publication of Linked Open Data (i.e., data freely accessible without restrictions on use) has been steadily increasing over last years, forming a big data cloud (Heath and Bizer, 2011) which includes information about an extensive variety of topics. DBpedia is the *de facto* central hub of this Web of Data. It is a broad-purpose knowledge base containing over a billion RDF statements, which have been extracted by mining Wikipedia "infoboxes", the tables of attribute-value pairs appearing alongside an article. These contain much factual information, such as birth and death dates for people, names of capitals and political leaders for countries, and so on.

RDF uses a graph-based data model for representing knowledge. Statements in RDF are expressed as so-called triples of the form `(subject predicate object)`, where `predicate` is a binary relation taking `subject` and `object` as arguments. RDF subjects and predicates are Uniform Resource Identifiers (URIs) and objects are either URIs or literals. For example, the following (using Turtle[7] syntax for serialising triples) is intended to say that the J. S. Bach's date of birth is 1685-03-21 and his place of birth is Eisenach:

(1)
```
:Johann_Sebastian_Bach dbont:birthDate "1685-03-21" .
:Johann_Sebastian_Bach dbont:birthPlace :Eisenach .
```

[5] http://dumps.wikimedia.org/simplewiki/latest/simplewiki-latest-pages-articles.xml.bz2

[6] http://www.w3.org/RDF/

[7] http://www.w3.org/TeamSubmission/turtle/

The notation `dbont:birthDate` makes use of an abbreviatory syntax for URIs involving the prefix `dbont` followed by a colon and a "local name". A prefix can be thought of as determining a namespace within which a given identifier, such as `:Johann_Sebastian_Bach`, is unique. To be complete, we also need to declare what the prefix `dbont` stands for. If, say, it is defined to be `http://dbpedia.org/ontology/`, then the unabbreviated version of `dbont:birthDate` would be `http://dbpedia.org/ontology/birthDate`. In the remainder of the paper, we use prefixed names and often use the empty prefix (e.g., as in `:Eisenach`) to stand for the default namespace. Example (1) illustrates another important feature of RDF, namely that the objects of predicates can be literals (like `"1685-03-21"`) or URIs (like `:Eisenach`). Although literals are often just strings, they can also be assigned to other XML Schema datatypes, such as `xsd:float`.

We will also adopt an abbreviatory convention from Turtle which allows a sequence of triples that share the same subject to be condensed with a semi-colon, as shown in (2).

(2)
```
:Johann_Sebastian_Bach dbont:birthDate "1685-03-21" ;
                       dbont:birthPlace :Eisenach .
```

This is equivalent to (1).

Sets of RDF triples can also be thought of as graphs, where the subject and object provide labels for nodes in the graph, and the predicate labels the edge between the nodes. This is illustrated in Fig. 1.

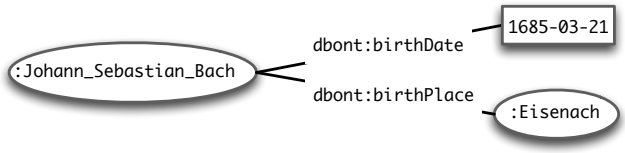

Figure 1: RDF Graph

RDF statements of the kind shown in (1) are equivalent to unquantified atomic sentences of first order logic.[8] For example, the graph in Fig. 1 translates to

(3) $birthDate(Johann\_Sebastian\_Bach, 1685\text{-}03\text{-}21) \wedge$
$birthPlace(Johann\_Sebastian\_Bach, Eisenach)$

3 Related Work

NLG for the Semantic Web Most previous approaches to Natural Language Generation from Semantic Web formalisms have been concerned with verbalising OWL ontologies (e.g. Stevens et al., 2011; Hewlett et al., 2005; Liang et al., 2012)). By contrast, the textual realisation of factual data in RDF has received relatively little attention. One interesting exception is Sun and Mellish (2007), whose Triple-Text system generates strings directly from RDF with a minimum of hand-coded rules. Sun and Mellish note that RDF predicates typically encode useful linguistic information. That is, while URIs are officially expected to be semantically opaque, in practice they often provide good clues as to their intended interpretation; examples are the predicates `ex:hasProperty` and `ex:industryOfArea`. Triple-Text exploits this information to lexicalise URIs and the triples they occur in without using domain knowledge. That is, since "camel-case" is often used to indicate implicit word boundaries in RDF predicates, it is straightforward to split up a predicate such as `hasProperty` into a sequence of tokens `[has, property]`, as a prelude to further processing. In Triple-Text, the tokens are POS-tagged and classified into categories via pattern-matching, and the categories then trigger the application of an appropriate realization rule to derive an output sentence. For example, given the triple (4), the system generates the sentence (5).

(4) `ex:North ex:industryOfArea ex:manufacturing_sector.`

(5) The industry of the North is manufacturing sector.

Although Triple-Text is fast and inexpensive to deploy, its expressiveness is limited by the fact that triples are processed in isolation from each other. As a result, only binary relations can be +expressed (since that is all that can be stated by a single RDF statement). A related consequence is that the

[8] Note that there is no way of expressing negation in RDF. Although existential quantification is allowed, we shall not have need of it in this paper.

information contained in a set of RDF statements cannot be aggregated or placed within the context of a larger discourse structure. Finally, the output is not always grammatically correct and cannot easily be tailored to a specific domain, since lexical choice is determined by the form of URIs in the triples.

Trainable NLG As mentioned above, we learn templates from parallel text-knowledge base pairs (TKRs) as originally proposed by Duboue and Mckeown (2003). Data in a TKR is aligned with the text (the "matching" stage) in a two-step process. First, the system identifies spans of text that are string-identical to values in the data. Second, it builds a statistical language model and uses the entropy of these to determine if other spans of text are indicative of similar values being expressed. The output of Duboue and Mckeown's system is dependent on hand-written rules and is specifically targeted at content determination for the constrained domain of biography generation. The approach we are adopting is less sophisticated, and is similar to the first matching step in their algorithm, where values in the data are matched to identical or near-identical strings in the text.

Automatic summarisation Our approach is related to multi-document summarisation and text-to-text generation insofar as they deal with the extraction and arrangement of sentences to create a new document. Text-to-text NLG only deals with processing documents that are about the same entity or topic and extracts the most relevant sentences from those documents to create a new document. In contrast, we want to generate natural language describing an instance in an ontology for which there may be no prior text available. To achieve this goal, we need to identify sentences about an entity that will be "transferable", that is, will be true of other entities of the same type (see § 4.2). Where such sentences are not directly derivable from the text, we can try to modify them to make them transferable. This is similar to the task of sentence compression in automatic summarisation (e.g. Cohn and Lapata, 2009; Filippova and Strube, 2008). Compression is often effected by tree operations over a parse tree, with the most frequent operation being the deletion of constituents. For summarisation, constituents are removed because they are deemed of lesser importance, while in our approach they are deleted when there is no evidence for them in the data. We adopt a syntactic pruning approach inspired by Gagnon and Da Sylva (2006), where sentences are first parsed and then the resulting structures are simplified by applying hand-built rules and filters.

4 The Task

4.1 Overview

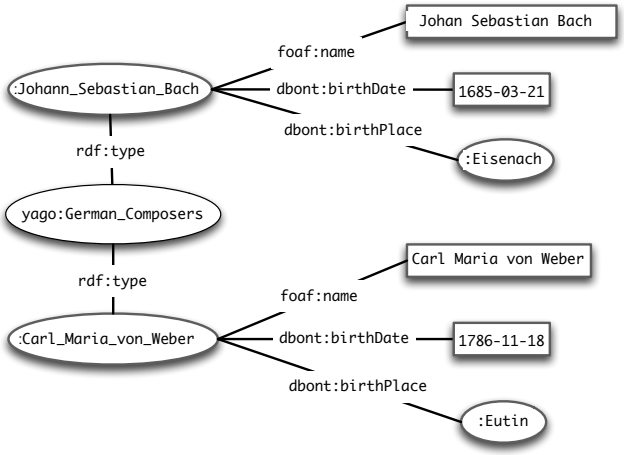

Figure 2: RDF Graph of Two Composers

We begin this section with a simplified example of how templates are extracted and then filled to produce a new sentence. Let's assume that our TKR is a pairing of the text

(6) Johan Sebastian Bach (b. Eisenach, 21 March 1685) was a German composer of the Baroque period.

with the RDF graph in Fig. 2. Note that a triple using the predicate `rdf:type` is the standard way of expressing class membership in RDF. By comparing the text with the part of the graph rooted in the node `:Johann_Sebastian_Bach`, we can figure out that certain substrings of (6) correspond to values of predicates in the RDF. For example, the string *Johan Sebastian Bach* matches the value of the predicate `foaf:name`. If we delete any string in (6) that can be mapped to the value of an RDF predicate in Fig. 2, then the result is a sentence template like that shown in (7), where we've use an underscore (___) to indicate 'slots' in the template.

(7) ___ (b. ___, ___) was a ___ of the Baroque period.

Although (7) is a template, it is not MINIMAL, since it contains information (*of the Baroque period*) that fails to correspond to any values in the graph. However, once we have pruned away non-minimal text, we are left with a transferable template which can be used to generate new sentences. In particular, we can use the subgraph of Fig. 2 rooted in the node `Carl_Maria_von_Weber` to fill the slots in the template, giving us the result in (8).

(8) Carl Maria von Weber (b. Eutin, 18 November 1786) was a German composer.

4.2 Extracting Templates

Automatically aligning the RDF data with the text can be viewed a special case of Named Entity Recognition (NER), or as a similar task to that of Wikification, which aims to annotate keywords in the document with their most relevant Wikipedia page (Mihalcea and Csomai, 2007). In our more restricted scenario, we only need to match spans of text with literal values that occur as objects of triples in the RDF. For instance, we want to be able to match the text string *Eisenach* with the literal that occurs as object of the predicate `dbont:birthPlace`. Rather than labelling named entity mentions with classes, we label them with the predicate of the triple in which the corresponding data value occurs. Let's write $\alpha \sim \mathtt{o}$ to mean that a string α is matched to an RDF object `o`. Then we say that α *has object-type* `p` if $\alpha \sim \mathtt{o}$ and for some subject `s`, a triple of the form `(s p o)` occurs in the relevant RDF graph.

To illustrate, let's consider the slightly more complex example in Fig. 3.

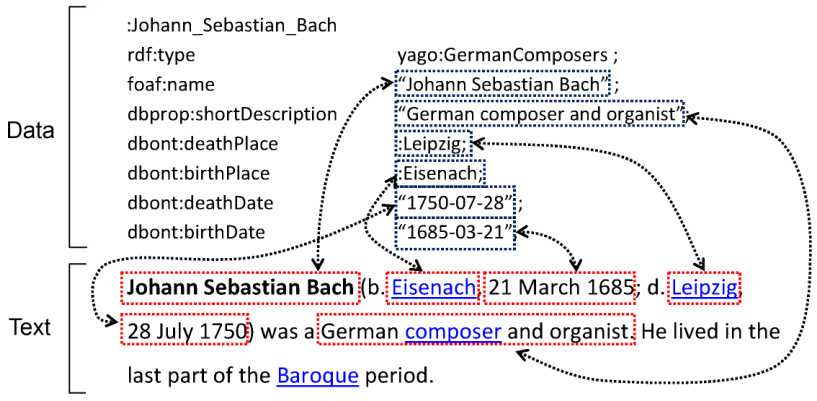

Figure 3: Aligning Data and Text

The template extracted from this alignment is represented as a text string with interpolated slots of the form `[p]`, where `p` is an RDF predicate. This is illustrated in (9).

(9) `[foaf:name]` (b. `[dbont:birthPlace]`, `[dbont:birthDate]`, d. `[dbont:deathPlace]`, `[dbont:deathDate]`) was a `[dbprop:shortDescription]`.

Selecting the RDF predicates whose values are to be matched in the text is dependent on the domain of the text and on the way data is encoded in RDF. We can think of the domain in terms of a subgraph of a given depth within the overall RDF graph in our dataset. In the present approach, we only retrieve the triples whose subject corresponds to the main topic of the Wikipedia article being processed. This restriction to subgraphs of depth 1 turned out to be sufficient for our task, and retrieving longer paths through the graph was found to significantly increase the complexity of the extraction without corresponding benefit.

Following Grice's Maxim of Quality, the output of the system should be constrained to be "truthful", in the sense that the textual output should be supported by evidence in the input data. As mentioned earlier, the sentence templates extracted for a given subject `s` should be minimal with respect to `s`, in that they do not contain substantive information unless it is licensed by a triple `(s p o)`. If a template $\mathcal{T}$ is minimal for `s`, and there is some `s'` which has the same class and (at least) the same attributes as `s`, then the result of filling $\mathcal{T}$'s slots with property values of `s'` should be a true sentence. As we pointed out earlier, we adopt a similar approach to (Gagnon and Da Sylva, 2006) by first parsing[9] the source sentences and then pruning away any constituents which fail to be licensed by the RDF data.

We will call the following grammatical categories PRUNABLE: noun, adjective, adverb and numeral. Pruning proceeds in three stages. First, if a word w belongs to a prunable category, then we delete w if it fails to match a data value in the relevant graph. Second, we delete any subtree T whose head has been deleted; for example, if w of category N has been deleted, then we also delete the NP (and any remaining daughters) which w headed. Finally, templates are discarded unless they contain slots corresponding to both the subject of the triple set and at least one property value, and remaining templates undergo post-processing to ensure correct punctuation by deleting empty brackets, duplicate commas, and so on.

The template extraction algorithm was informally evaluated by running it on 268 articles. 199 candidate templates were extracted, of which 124 were discarded after pruning and other filtering, leading to 74 templates. We judged 60 of these to be minimal, and 43 (58%) to be grammatically acceptable.

4.3 Document Planning

For content determination, we implement the baseline of (Duboue and Mckeown, 2003). We collect unigrams of matched RDF predicates in the text, and if the frequency of a predicate is below a threshold in the articles for a given class of entity, even if an instance of this class has this property in the data, the system should not output it. According to Reiter and Dale (2000), document structuring carries out more complex selection and ordering of information than just sequencing; it treats the text as a tree structure and clusters related items of information. Given that the sentences templates extracted from the text contain several properties expressed, we can think of them as partial trees, part of the bigger tree required for document structuring, so we expect that extracting these templates and ordering them in the right way will yield good results. We never attempt to assess the relative importance of a sentence to a text; we aim to extract as many templates as possible and treat them as equally important.

We need to deal with the fact that, for a given subject, an RDF predicate can have more than one value and that more than one property can have the same value. Indeed this is often the case and LOD tends to show very high redundancy. To deal with this, we cluster predicates into equivalence classes, or PREDICATE POOLS, when they appear to take the same value for the subject with frequency over a given threshold. For example, `foaf:name`, `rdfs:label` and `dbprop:name` are clustered together in a single pool for all classes used in the experiments.

Sentence templates that have been selected for inclusion in the final text are ordered using trigram probabilities of the predicate pools they verbalise. For each step, the probability score of each template is computed using the trigram probability of the last two verbalised pools and the first one in the template and the one with the highest probability is chosen.

4.4 Class Models

The NLG system has a single communicative goal: to describe an instance of a class (for example, J. S. Bach as an instance of `GermanComposer`). However, attributes that are relevant to, say, a rock band, are unlikely to be relevant to an antelope or other animal species. Similarly, given predicates need not be expressed in the same order for all classes. Finally, sentence templates can also be expected to depend on the class of the entity, because of the RDF predicates they realise and also because of their lexical and syntactic composition. In order to capture this dependence, we associate with classes a bundle of information including the relevant RDF predicate pools, a preferred ordering

[9] We used the Stanford Parser version 1.6.5, from 30/11/2010 with the pre-trained PCFG English model (Klein and Manning, 2003).

for realising the corresponding object-types, and the appropriate sentence templates. This bundle is termed a CLASS MODEL.

This means that when we want to generate text about a given entity, we need to choose, from those available, the class that it would be most prototypical of (cf. Rosch, 1973). This class must have the right granularity and level of detail, both for training and generating. It must not be so general that its attributes are too generic, or so specific that the extracted templates do not generalise to other entities of the same class. This task is surprisingly nontrivial, since not only can instances be members of multiple classes (via triples of the form `s rdf:type C`), but classes also allow multiple inheritance (via triples of the form `C rdfs:subClassOf SC`). An initial exploratory analysis of the Wikipedia data shows that a single entity typically belongs to a large number of classes. This is illustrated by Figure 4, which shows the classes to which J. S. Bach belongs according to DBPedia.[10]

```
AnglicanSaints              ComposersForViolin
Person                      ComposersForCello
GermanComposers             GermanClassicalOrganists
ComposersForPipeOrgan       PeopleCelebratedInTheLutheranLiturgicalCalendar
ComposersForLute            OrganistsAndComposersInTheNorthGermanTradition
18th-centuryGermanPeople    PeopleFromSaxe-Eisenach
BaroquEComposers            ClassicalComposersOfChurchMusic
PeopleFromEisenach
```

Figure 4: Class membership of `Johann_Sebastian_Bach`

Choosing the "right" class for an entity is a challenging problem. For the present approach we have adopted baseline algorithm which rests on the heuristic that if a substring occurs frequently in the terms used as class names for some entity, then it is likely to correspond to a core class for that entity. For example, inspection of Figure 4 shows that *Composers* occurs as a substring in 8 of the 15 class names (and will also occur in the literal value of the `rdfs:label` predicate for those classes). The algorithm has the following steps:

1. Collect the `rdfs:label` values for each of the classes as a bag-of-words, and combine this with the words from the first sentence of the Wikipedia article. Create a word vector *tf* (term frequency) whose values are each word's raw frequency in the bag.
2. Compute a score for every class label, which is the normalized sum of *tf* scores of every work in it, using the formula:

$$score(w) = \frac{1}{M} \sum_{i=0}^{N} tf(w_i)$$

   where w is the list of tokens derived from the class label string (e.g. *["People", "from", "Eisenach"]*), w_i is the i^{th} element in this list and N is the total number of elements in w. M is set to be N if $N > 1$, otherwise it is set to an arbitrary higher value to reflect a dispreference against one-word class names.
3. Select the classes with the n-highest scores. We train for several models at the same time, given that we cannot be confident the class we chose is the only one that the entity is prototypical of. During the experiments, we set the value of n to 5.

As an example, the 5-best class list (with their scores) for `:Johann_Sebastian_Bach` is shown in (10). For each of these classes, a class model is created (or updated if already present).

(10) `yago:GermanComposers (6.0), yago:BaroquEComposers (3.3)`
`yago:ComposersForViolin (3.0), yago:ComposersForCello (3.0),`
`yago:ComposersForLute (3.0)`

4.5 Generation

The LOD-DEF generation algorithm takes as input a collection of class models, and the URI of the entity to be described. The five best classes are chosen (using the approach described above, but without the addition of article text to the bag of words) and the corresponding model classes are

[10] To ease presentation, we have omitted the namespace qualifiers of these classes.

scored according to the number of predicate pools that would be instantiated via the model's sentence templates.

In order to check instantiation, we need to access the RDF properties of the subject entity, and this is carried out by running a sequence of SPARQL queries against the DBPedia endpoint. That is, we execute a query for each predicate pool in the class model. As mentioned earlier, a predicate pool may contain a number of different RDF predicates, but since these are considered to be semantically equivalent, as soon as the value of one of them has been successfully returned by the query, this value is taken as the representative for the whole pool and querying is terminated. Where a single pool has multiple values, it is treated as a list for the purpose of aggregation, e.g. if ":X dbont:birthPlace :A" and ":X dbont:birthPlace :B" then "X was born in A and B".

To illustrate the scoring procedure, let's suppose that we have selected the classes shown in (10) for the topic `:Johann_Sebastian_Bach`. The scores for two class models are shown in Table 1.

	Models	
	`GermanComposers`	`ComposersForCello`
Number of templates	2	3
Predicates with values in the data	7	6
Predicates instantiated in templates	5	6

Table 1: Scoring class models

The chosen model would be `ComposersForCello`, even though it received a lower score in the first step, because a higher number of values would be (potentially) expressed through templates. The motivation behind this choice is that an extracted sentence template is expected to generate higher quality text, so a model instantiating more predicates through extracted templates is preferred.

We use chart generation: all sentence templates in the model for which there are enough triples in the data are put on a chart and combinations of them are generated. The following steps are taken: (1) We discard templates whose object-types involve predicate pools for which no data values have been found and put the remaining ones on the chart. (2) For each pool in the model, a default sentence template of the form `[possessive] [property] is [value]` is generated and added to the chart. This deals with the situation where a retrieved data value would failed to be expressed for a lack of an appropriate template.

We select and order sentence templates from the chart to produce a text. Ideally, we would want to find the combination of sentences that expresses all the values of the predicate pools in the model, while employing as many extracted templates and as few default templates as possible. We also want to order the templates in the most plausible order according to the RDF predicate trigrams collected during training. In order to deal with the combinatorial explosion we compute scores for all the options at every step, select the combination with the highest score and discard all the others, thus only ever keeping one possible combination. One important constraint is that a given predicate pool should only be instantiated once per generated article. When a template is chosen that requires the value of a predicate pool, any other template on the chart also requiring that information will not be considered for the combination. This is not guaranteed to be the optimal solution to the requirements outlined above, but it is a satisfactory trade-off between quality and speed and keeps the algorithm to $O(n^2 log(n))$.

5 Evaluation

5.1 Method and Materials

Given the exploratory nature of this project, the evaluation relies on human evaluation of the system's output compared to output from two other systems: the baseline and expert human output. We adopt a two-panel (i.e., two separate groups of subjects) approach to compare the three outputs, similarly to Sun and Mellish (2007). Subjects in Panel I generate descriptions of the same 12 entities [11] while subjects in Panel II rate the different outputs of System A (baseline), System B (LOD-DEF) and System C (human generation) across a number of dimensions. The hypothesis is that subjects will rate LOD-DEF higher on average than a baseline system generating exclusively from English words in RDF predicate labels. The system is also ranked against human-generated text for the same data.

[11] For a full list of the entities used and more in-depth details of our implementation, see Duma (2012).

Human-generated text need not always be an upper bound in subjective evaluation, but given the simplicity of the two NLG systems, this is a reasonable expectation.

Subjects were asked to complete an online survey. For this survey, the same 12 entities were described by the three systems, which produced 36 short texts, rated by 25 subjects. The participants were self-identified as having an upper-intermediate or above level of English. The texts were presented to the subjects in pseudo-random order, to avoid texts about the same entity occurring within a page of each other (four texts were presented on every page). Subject were asked to rate each text on a measure of 1 (lowest) to 5 (highest) on the following three criteria, adapted from the DUC 2007 criteria: grammaticality; non-redundancy; and structure and coherence.[12] No direct evaluation of content determination is carried out, but this is assumed to evaluated implicitly through the dimension of *non-redundancy*, given that its main effect in the implementation is filtering out redundant and unnecessary information. LOD-DEF does not select more relevant triples, but it omits irrelevant ones. It was not disclosed to the subjects that humans generated the texts of one of the systems being tested.

Data One of the aims was to evaluate the effectiveness of extracted sentence templates used by the LOD-DEF system. Consequently, classes for evaluation were not chosen at random, but were selected with a bias towards maximising the number of RDF predicates matched in text and the number of templates; this correlated with classes for which more factual information (strings and quantities) was available on DBpedia. Subject to this constraint, an attempt was made to vary the selected classes as much as possible.

Baseline	Jennifer Saunders is an English television actor. Her birth date is 6 July 1958. Her description is British comedienne. Her place of birth is Sleaford, Lincolnshire, England.
Humans	Jennifer Saunders (Born 06/07/1958) is an English comedienne, originally from Sleaford, Lincolnshire.
LOD-DEF	Jennifer Saunders (6 July 1958, Sleaford and Lincolnshire) is a British comedienne.

Figure 5: Sample outputs of baseline, human and LOD-DEF

Baseline Our baseline NLG system employs direct generation from RDF triples in the style of Sun and Mellish (2007). Each triple is realised as a single sentences and a shallow linguistic analysis of the words in the RDF predicate determines the structure of these sentences. The label values (from `rdfs:label`) of predicates are used for this when available, otherwise a tokenisation of the predicate URI was used. The initial sentence created by the baseline is realises the class of the entity, formed by the name of the entity (its `rdfs:label`) followed by *is a* and the `rdfs:label` of the class of the entity, e.g., *Johann Sebastian Bach is a German composer*. The relevant class is chosen using the class selection algorithm detailed earlier.

The collection of triples we are dealing with encodes information about a single entity, and we wish to present this information as a coherent text made up of several sentences. To aid coherence, the baseline implements a very simple Referring Expression Generation algorithm, where subsequent references to the topic entity use a personal pronouns rather than the full name. The system selects the correct personal and possessive pronoun on the basis of the value of the `foaf:gender` predicate. This baseline makes no attempt at document planning, but simple heuristics filter out properties with inappropriate values (e.g., values containing more than ten words).

Human Upper Bound Panel I, formed by two native speakers of English (linguistics postgraduate students), were given triples related to the chosen entities and instructions to write summary descriptions of the entities the data is about, expressing in text as much of this data as possible. The information was raw but presented in a human-friendly format as illustrated in (11).

(11)
```
birth date = 1958-07-06
place of birth = Sleaford, Lincolnshire, England
```

The triples were based on the same data as that used by by the LOD-DEF system except that they were manually filtered to remove redundancy and to randomize their order. We avoided giving the subjects examples of what kind of output we expected, thus taking care not to prime them. Fig. 5 shows short examples of each kind of output.

[12] `http://www-nlpir.nist.gov/projects/duc/duc2007/quality-questions.txt`

5.2 Results

An exploratory analysis of the data collected showed clear differences in means between for the rating of the three systems (Table 2). We ran a One-Way ANOVA for each of the three criteria the texts were rated on. All three ANOVAs were statistically significant: for grammaticality $F(2, 72) = 119.001, p < 0.001$, for non-redundancy $F(2, 72) = 129.053, p < 0.001$ and for structure and coherence $F(2, 72) = 129.053, p < 0.001$. Tukey's Post-Hoc test established which comparisons were significant for each; Tables 3 and 4 show the differences in mean and their significance.

As expected, human generation was judged to be consistently superior to the other two systems. LOD-DEF does not improve on the perception of grammaticality of the baseline, but it does significantly outperform the baseline on non-redundancy and structure and coherence. The most significant improvement of LOD-DEF over the baseline is on the non-redundancy metric, with a difference of 1.14.

System	Grammaticality	Non-redundancy	Structure & coherence
A (baseline)	2.29	1.89	1.95
B (LOD-DEF)	2.58	3.03	2.70
C (humans)	4.48	4.66	4.49

Table 2: Means

Baseline vs. LOD-DEF	Grammaticality	Non-redundancy	Structure & coherence
Difference	0.29	1.14	0.75
Significance	p = 0.151	p < 0.001	p < 0.001
Significant	No	Yes	Yes

Table 3: Differences and significance

LOD-DEF vs. Humans	Grammaticality	Non-redundancy	Structure & coherence
Difference	1.14	1.63	1.79
Significance	p < 0.001	p < 0.001	p < 0.001
Significant	Yes	Yes	Yes

Table 4: Differences and significance

6 Conclusion

We have implemented and tested a trainable shallow Natural Language Generation system for verbalising factual RDF data based on the extraction of sentence templates and document planning via content n-grams. We trained this system on text from the Simple English Wikipedia and RDF triples from DBpedia, and also implemented a baseline system, based on direct generation from triples.

We conducted human evaluation of the two systems, together with text generated by humans from the same information, where LOD-DEF significantly outperformed the baseline on *non-redundancy* and *structure and coherence*. These are encouraging results that suggest shallow systems like this one can be easily built and trained from Text-Knowledge Resources. While any structured representation of meaning could be used as the "knowledge" resource, we have dealt specifically with Linked Open Data, which required specific solutions to some of its inherent challenges.

It is conceivable that most, if not all, components of an NLG system could be trained from these TKR. More sophisticated approaches applying a deeper understanding of natural language and deeper NLG would be required for this, together with much reasoning and inference to connect the data and text. Our results, preliminary as they are, suggest that this vision is worth pursuing.

7 Acknowledgements

This research was partially supported by European Commission FP7 project ROBOT-ERA (grant agreement ICT-288899).

References

Cohn, T. and M. Lapata (2009). Sentence compression as tree transduction. *Journal of Artificial Intelligence Research 34*, 637–674.

Duboue, P. A. and K. R. Mckeown (2003). Statistical acquisition of content selection rules for natural language generation. In *Proceedings of the 2003 conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 121–128.

Duma, D. (2012). Natural language generation for the semantic web: Unsupervised template extraction. MSc Dissertation. http://www.scribd.com/doc/111024287/Natural-Language-Generation-for-the-Semantic-Web-Unsupervised-Template-Extraction.

Filippova, K. and M. Strube (2008). Dependency tree based sentence compression. In *Proceedings of the Fifth International Natural Language Generation Conference*, pp. 25–32. Association for Computational Linguistics.

Gagnon, M. and L. Da Sylva (2006). Text compression by syntactic pruning. *Advances in Artificial Intelligence 1*, 312–323.

Heath, T. and C. Bizer (2011). *Linked Data: Evolving the Web into a Global Data Space*. Synthesis Lectures on the Semantic Web: Theory and Technology. Morgan & Claypool.

Hewlett, D., A. Kalyanpur, V. Kolovski, and C. Halaschek-Wiener (2005). Effective NL paraphrasing of ontologies on the Semantic Web. In *Workshop on End-User Semantic Web Interaction, 4th Int. Semantic Web conference, Galway, Ireland*.

Klein, D. and C. Manning (2003). Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pp. 423–430. Association for Computational Linguistics.

Liang, S., R. Stevens, D. Scott, and A. Rector (2012). OntoVerbal: a Protegé plugin for verbalising ontology classes. In *Proceedings of the Third International Conference on Biomedical Ontology*.

Mendes, P., M. Jakob, and C. Bizer (2012). DBpedia: A multilingual cross-domain knowledge base. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2012)*.

Mihalcea, R. and A. Csomai (2007). Wikify!: linking documents to encyclopedic knowledge. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pp. 233–242. ACM.

Reiter, E. and R. Dale (2000). *Building natural language generation systems*. Cambridge Univ Press.

Rosch, E. (1973). Natural categories. *Cognitive psychology 4*(3), 328–350.

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

Sun, X. and C. Mellish (2007). An experiment on free generation from single RDF triples. In *Proceedings of the Eleventh European Workshop on Natural Language Generation*, pp. 105–108. Association for Computational Linguistics.