beAWARE

Enhancing decision support and management services in extreme weather climate events

700475

D5.3

Advanced techniques for emergency report generation and their performance

Dissemination level:	Public
Contractual date of delivery:	Month 34, 30 October 2019
Actual date of delivery:	Month 35, 30 November 2019
Workpackage:	WP5 Multilingual report generation
Task:	T5.3 - Content selection and report discourse structure planning T5.4 - Multilingual linguistic surface report generation T5.5 - Evaluation of emergency report generation
Type:	Report
Approval Status:	Final Version
Version:	V0.8
Number of pages:	58
Filename:	D5.3_beaware_ Advanced techniques for emergency report generation and their performance_2019-11-30_v0.8

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 700475

Abstract

This deliverable reports on developments in WP5 from month M18 to M34 towards the completion of the final version of multilingual report generation module. It details the final versions of (i) content planning strategy, (ii) the multilingual datasets used to train modules for linguistic processing, and (iii) the rule-based and statistical modules for linguistic generation. This report also presents scientific evaluations of the components of the report generation module.

Co-funded by the European Union

History

Version	Date	Reason	Revised by
0.1	25/11/2019	First version MRG final description	Simon Mille (UPF)
0.2	26/11/2019	Continued FORGe sections	Simon Mille (UPF)
0.3	27/11/2019	Completed FORGe sections	Simon Mille (UPF)
0.5	27/11/2019	Filled linearisation sections	Roberto Carlini (UPF)
0.6	28/11/2019	Integrated all sections, finalised intro and conclusions	Simon Mille (UPF)
0.7	29/11/2019	Minor changes and proofreading	Gerard Casamayor (UPF)
0.8	29/11/2019	Internal Review / Final Version	CERTH

Author list

Organisation	Name	Contact Information
UPF	Gerard Casamayor	gerard.casamayor@upf.edu
UPF	Simon Mille	simon.mille@upf.edu
UPF	Roberto Carlini	roberto.carlini@upf.edu

Reviewer List

Organisation	Name	Contact Information
CERTH	Michail Emmanouil	michem@iti.gr
CERTH	Ilias Koulalis	iliask@iti.gr

Executive Summary

This deliverable describes the advanced methods to produce multilingual emergency reports and their implementation as part of the final version of the Multilingual Report Generation (MRG) module. It reports advances during months M18 to M34 of the beAWARE project in tasks T5.3 (Content selection and report discourse structure planning), T5.4 (Multilingual linguistic surface report generation) and T5.5 (Evaluation of emergency report generation). The work presented here elaborates on the contents of deliverables D5.2 and D7.6.

The advances reported cover the advanced methods for mapping ontological representations onto linguistic structures, and for rule-based and statistical multilingual linguistic surface report generation. We also present the linguistic datasets used for the development and training of the linguistic models and resources applied to multilingual generation and also the linguistic analysis component reported in D3.4. UPF is responsible for all the work presented in this deliverable.

In addition to the description of advanced methods for T5.3 and T5.4, and as part of T5.5, this document also includes qualitative and quantitative evaluations of (i) the rule-based FORGe generator and (ii) the deep learning-based TLin lineariser, with state-of-the-art results for both approaches. We also list several publications and dissemination events related to the datasets and methods developed.

Abbreviations and Acronyms

ConS	Conceptual Structure
DnS	Description and Situation
DoA	Description of Action
DSyntS	Deep-Syntactic Structure
dul	DOLCE+DnS Ultralite
EL	Entity Linking
FORGe	Fabra Open-source Rule-based Generator
JSON	JavaScript Object Notation
KB	Knowledge Base
KBS	Knowledge Base System
LAS	Labelled Attachment Score
LG	Language Generation
LSTM	Long Short-Term Memory
MAP	Mean Average Precision
MorphS	Morphological Structure
MRG	Multilingual Report Generator
MTT	Meaning-Text Theory
NE	Named Entity
NLG	Natural Language Generation
NLP	Natural Language Processing
PCA	Principal Component Analysis
PoS	Part of Speech
PredArg	Predicate-Argument
RDF	Resource Description FrameWork
RNN	Recurrent Neural Network
RST	Rhetorical Structure Theory
SemS	Semantic Structure
SRL	Semantic Role Labelling
SSyntS	Surface-Syntactic Structure
SW	Semantic Web
UDs	Universal Dependencies

Table of Contents

1 INTRODUCTION 9

2 TYPES OF REPORTS GENERATED BY THE KNOWLEDGE BASE SYSTEM 11

2.1 Summaries of all incidents (*Sum*$_{ALL}$) 11

2.2 Summaries per time frame (*Sum*$_{TIME}$) 13

2.3 From KBS to MRG 14

3 FINAL DATASETS AND SURFACE REALISATION SHARED TASKS 16

3.1 Task overview 16

3.2 General specifications of the data 17

3.3 A converter for obtaining the Deep track structures 18

3.4 Dataset variety and sizes 21

3.5 Evaluation of the quality of deep datasets 22

4 MULTILINGUAL REPORT GENERATION 23

4.1 General approach and summary of advances 23

4.2 Main improvements in the grammars and lexicons of the rule-based generator 28

4.2.1 Extensions to language-independent rules 28

4.2.2 Extensions to the Spanish rules 31

4.2.3 Crafting the Spanish dictionaries 32

4.2.4 Overview of the evolution of the FORGe generator in beAWARE 32

4.3 Development of a new deep learning-based lineariser 34

4.3.1 Related work 34

4.3.2 Fundamentals 36

4.3.3 Model 39

5 INTRINSIC EVALUATION OF THE FINAL MODULES 41

5.1 Quantitative and qualitative evaluation of FORGe on a standard RDF dataset 41

5.1.1 Selection of triples for evaluation 41

5.1.2 Reference sentences 42

5.1.3 Automatic evaluation 42

5.1.4 Qualitative analysis of the results 43

5.2 Quantitative evaluations of FORGe on a large-scale generic dataset 45

5.3 Evaluation of the neural lineariser 46

6 DISSEMINATION 49

6.1 The Surface Realisation Shared Tasks 49

7 CONCLUSIONS AND SUMMARY 53

8 REFERENCES 54

List of Figures

Figure 1: internal architecture of the MRG 9
Figure 2: KBS sample for video- and image-detected events in JSON format 12
Figure 3: KBS sample for text- and audio-extracted events in JSON format........ 13
Figure 4: Partial KBS sample in the JSON format 14
Figure 5: Partial generator input sample (Predicate-Argument template); top in CoNLL format, bottom in graphical format 14
Figure 6: Original UD structure in the CoNLL-U format (top: graphical representation)........ 17
Figure 7: Shallow track input in the CoNLL-U format (top: graphical representation) 18
Figure 8: Deep track input in the CoNLL-U format (top: graphical representation) 18
Figure 9: A sample graph-transduction rule. '?' indicates a variable, '?Xl{}' is a node, '?s' is a relation, 'a=?b' is an attribute/value pair. 20
Figure 10: Start (input) 23
Figure 11: Aggregation 24
Figure 12: Con-Sem in Spanish 24
Figure 13: Con-Sem in English 24
Figure 14: Sem-DSynt in Spanish........ 25
Figure 15: Sem-DSynt in English........ 25
Figure 16: DSynt-SSynt in Spanish........ 26
Figure 17: DSynt-SSynt in English 26
Figure 18: SSynt-DMorph in Spanish (partial) 27
Figure 19: SSynt-DMorph in English (partial) 27
Figure 20: DMorph-SMorph in Spanish (partial) 28
Figure 21: DMorph-SMorph in English (partial) 28
Figure 22: A non-projective example from the Penn TreeBank. 35
Figure 23: An unfolded recurrent neural network. Each word is processed as one time-step and influences the next words 36
Figure 24: Transformer architecture 38
Figure 25: TLin architecture........ 40
Figure 26: The softmax on the output points to the 6th element of the input list, which is the word "eat". 40

List of Tables

Table 1: Semantic labels in the output of the UD-based pipeline........ 18
Table 2: Graph-transduction rules for UD-based deep parsing 20
Table 3: SR'18 dataset sizes and splits. 21

Table 4: SR'19 dataset sizes and splits 21

Table 5: Evaluation of the quality of the output structures (Labeled Attachment Scores - LAS). 22

Table 6: Comparison between Initial, P2 and P3 generators. The table summarises the improvements of the components of the generator during the course of the project. 32

Table 7: Summary of the different attention mechanisms (Weng, 2018) 36

Table 8: English and Spanish scores according to BLEU, METEOR and TER, with 1 and All references on the 200-triples test set. 43

Table 9: BLEU score comparison on the PTB test set for word-ordering setups in Schmaltz et al. (2016), that is, using only surface information. We show the best results reported for each configuration... 47

Table 10: BLEU score comparison on the PTB test set for the syntactic linearisation task. We show the best results reported for each configuration. 47

Table 11: BLEU score comparison on the PTB test set between our system results and word-ordering system results, quoted from (Hasler et al., 2017). We show the best results reported for each configuration. 47

Table 12: BLEU scores of TLin on the UD data 48

Table 13: SR'18 BLEU scores of the participating teams 49

Table 14: SR'18 human evaluations 50

Table 15: SR'19 Shallow Track BLEU scores of the participating teams 51

Table 16: SR'19 Deep Track scores of the participating teams 51

Table 17: SR'19 human evaluations: Meaning Similarity (left) and Readability (right) 52

1 Introduction

Report generation is implemented within the overall beAWARE system in the Multilingual Report Generation (MRG) service, a data analysis and processing component of the business layer. MRG receives requests for report generation, through the message bus, containing ontological contents in JSON format, issued by the Knowledge Base Service (KBS). The content received consists of a list of incidents belonging to nearby locations with distinct timestamps. Each incident description contains details about the type and the set of participating objects. These descriptions are created by using information extracted from the data analysis services (i.e. text analysis, speech recognition, image analysis, video analysis) and are semantically integrated by the KBS. Report generation processes the contents received in a request in two steps, first by addressing content selection, and then by applying multilingual linguistic report generation (see **Error! Not a valid bookmark self-reference.**).

During the reporting period, UPF worked as planned on the development of the aforementioned resources and tools towards the completion of the final beAWARE platform. Furthermore, UPF dedicated efforts towards making these resources reusable and in line with the current trends in the Natural Language Processing (NLP) field. Indeed, during the time frame of the beAWARE project, two important shifts have been taking place in NLP: first, in the usage of annotated data, where Universal Dependencies (UDs) have become increasingly prominent in NLP, replacing language-specific annotations. Second, deep learning techniques

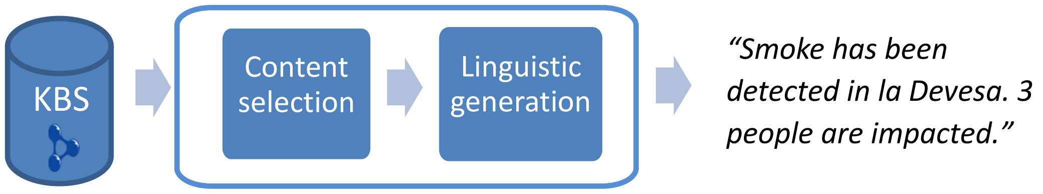

KBS
Content selection
Linguistic generation
"Smoke has been detected in la Devesa. 3 people are impacted."

Figure 1: internal architecture of the MRG

have become dominant in NLP in the past few years. However, in Natural Language Generation (NLG), little work has been done so far in these directions. In D5.2, UPF reported on NLG experiments carried out on language-specific datasets and by using Support Vector Machine techniques. Meanwhile, due to the new advances in the field, it was decided to interrupt these experiments and switch to UD-based datasets and neural techniques. For this reason, we tackled the creation of new datasets for the whole NLP community, and trained state-of-the-art neural models on them.

The current deliverable is structured as follows: Section 2 details the specificities of the conversion of Knowledge Base contents onto linguistic structures to be used as input by the beAWARE generator. Section 3 presents the final version of the Universal Dependency-based datasets created to train statistical generators. Section 4 describes the improvements made to the Fabra Open-source Rule-based Generator (FORGe) and reports on the development of a new neural lineariser. Following, Section 5 presents the results of various qualitative and quantitative evaluations of both systems. Finally, Section 6 summarises dissemination efforts made by UPF in terms of publications and event organisation, and Section 7 concludes the document.

2 Types of reports generated by the Knowledge Base System

For the second prototype, the report of incidents was simplistic: all incidents were stored in the Knowledge Base System (KBS) but only the last one was reported. For the final prototype, all detected and reported incidents are being used for the generation of the reports.

The Multilingual Report Generation (MRG) module generates two types of reports: (i) summaries of all incidents sent by the KBS (Sum_{ALL}) and (ii) summaries of incidents per time frame (Sum_{TIME}). In practice, the data are the same but the way they are being processed differs: for Sum_{ALL}, if an incident is exactly the same as a previous incident, it is not reported whereas, for Sum_{TIME} identical incidents can be reported more than once across time frames (but not within the same time frame). For instance, if the KBS sends four events of fire reported in Valencia in three different time frames, Sum_{ALL} will only return one "Fire in Valencia" report, while Sum_{TIME} will return three "Fire in Valencia" reports (one by time frame).

Both types of summaries consist of a title, which is used to label the overall incident as visualised in the Public Safety Answering Point map, and a description, which captures the system detected incidents. The content extraction process from KBS data in each case is described in the following.

2.1 Summaries of all incidents (Sum_{ALL})

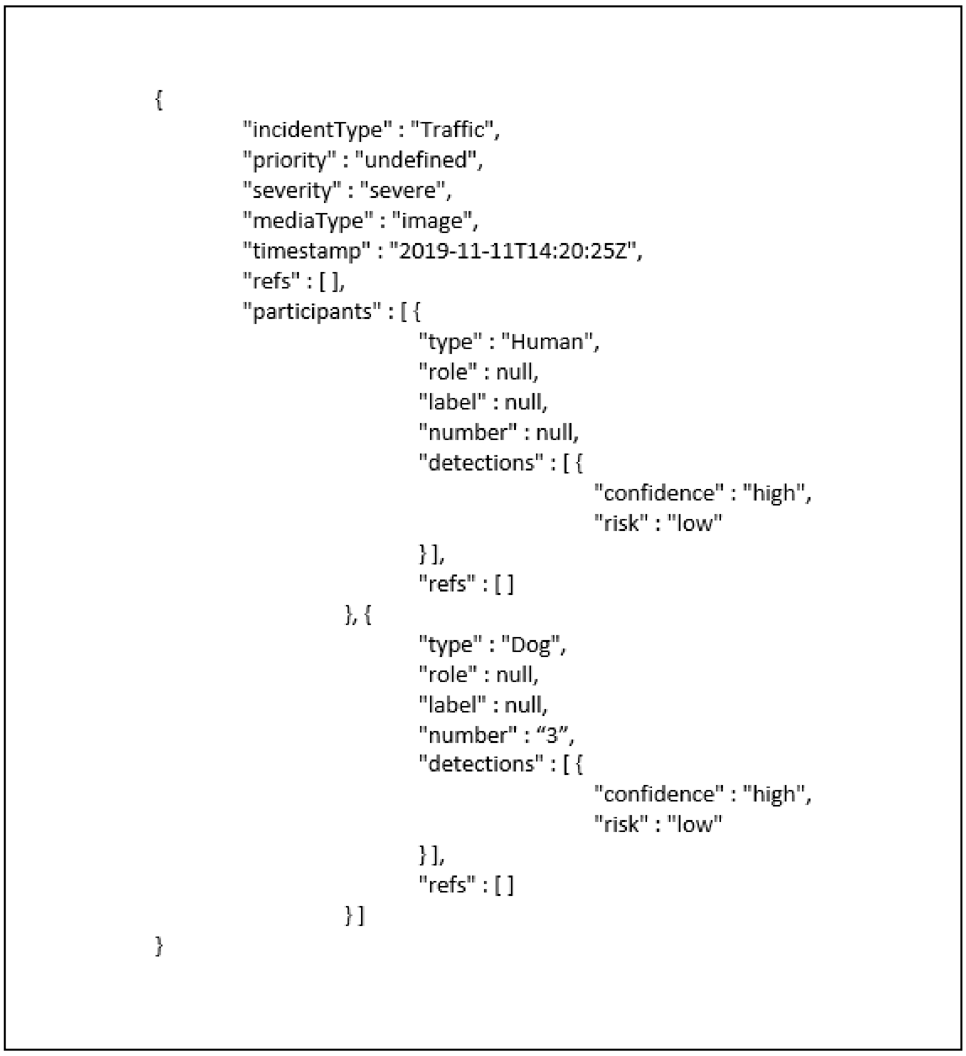
When a KBS report request is received, two different content extraction strategies are applied, due to the different data sources: one for video and image and another one for text and audio. For video and image, the incident type is identified, and the participants are collected.

Figure 2 shows a Traffic incident involving a human and 3 dogs.

Some aggregations can take place at this level: for example, if two different incidents of the same type (e.g. *Traffic*) are reported in the same request, the participants are aggregated in a single, generic incident of that type.

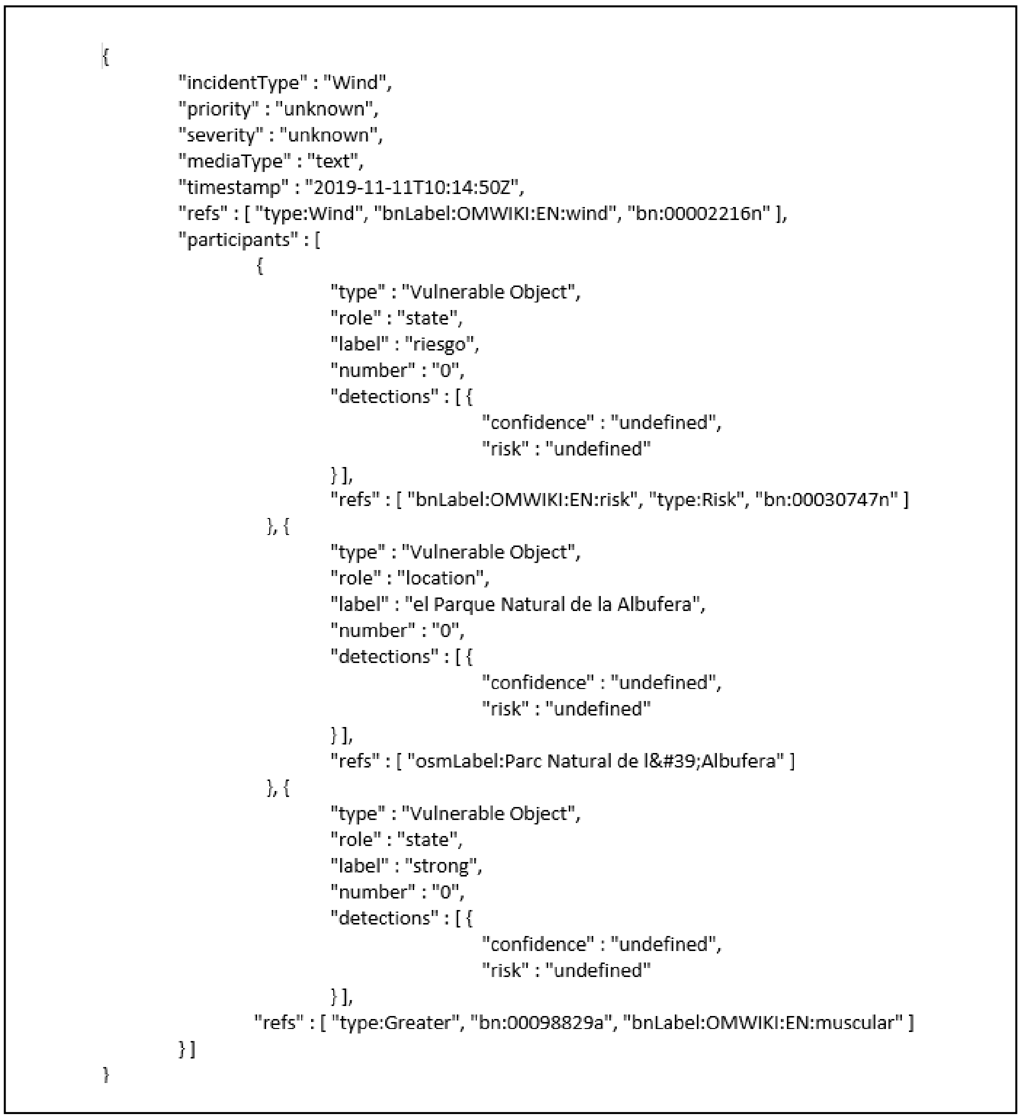
For text and audio, the content extraction is more complex, due to the richer relations between participants produced by the text analysis component. The content extraction takes into account these relations to identify the different parts of a generated report: incident type, object, location and qualitative modifiers (*strong, weak*, etc.). Once all the parts are identified, the same aggregation strategy is applied on top of it. Figure 3 shows a *Wind* incident, in which it is stated that there is a strong risk that wind will affect the Albufera Natural Park.

As final step, duplicated incidents are filtered out, in order to avoid repetition. The final set of contents extracted from the incidents is properly structured so it can be consumed by the text generator.



```
{
        "incidentType" : "Traffic",
        "priority" : "undefined",
        "severity" : "severe",
        "mediaType" : "image",
        "timestamp" : "2019-11-11T14:20:25Z",
        "refs" : [ ],
        "participants" : [ {
                        "type" : "Human",
                        "role" : null,
                        "label" : null,
                        "number" : null,
                        "detections" : [ {
                                        "confidence" : "high",
                                        "risk" : "low"
                        } ],
                        "refs" : [ ]
                }, {
                        "type" : "Dog",
                        "role" : null,
                        "label" : null,
                        "number" : “3”,
                        "detections" : [ {
                                        "confidence" : "high",
                                        "risk" : "low"
                        } ],
                        "refs" : [ ]
                } ]
}
```

Figure 2: KBS sample for video- and image-detected events in JSON format



```
{
        "incidentType" : "Wind",
        "priority" : "unknown",
        "severity" : "unknown",
        "mediaType" : "text",
        "timestamp" : "2019-11-11T10:14:50Z",
        "refs" : [ "type:Wind", "bnLabel:OMWIKI:EN:wind", "bn:00002216n" ],
        "participants" : [
                {
                        "type" : "Vulnerable Object",
                        "role" : "state",
                        "label" : "riesgo",
                        "number" : "0",
                        "detections" : [ {
                                        "confidence" : "undefined",
                                        "risk" : "undefined"
                        } ],
                        "refs" : [ "bnLabel:OMWIKI:EN:risk", "type:Risk", "bn:00030747n" ]
                 }, {
                        "type" : "Vulnerable Object",
                        "role" : "location",
                        "label" : "el Parque Natural de la Albufera",
                        "number" : "0",
                        "detections" : [ {
                                        "confidence" : "undefined",
                                        "risk" : "undefined"
                        } ],
                        "refs" : [ "osmLabel:Parc Natural de l&#39;Albufera" ]
                 }, {
                        "type" : "Vulnerable Object",
                        "role" : "state",
                        "label" : "strong",
                        "number" : "0",
                        "detections" : [ {
                                        "confidence" : "undefined",
                                        "risk" : "undefined"
                        } ],
                "refs" : [ "type:Greater", "bn:00098829a", "bnLabel:OMWIKI:EN:muscular" ]
        } ]
}
```

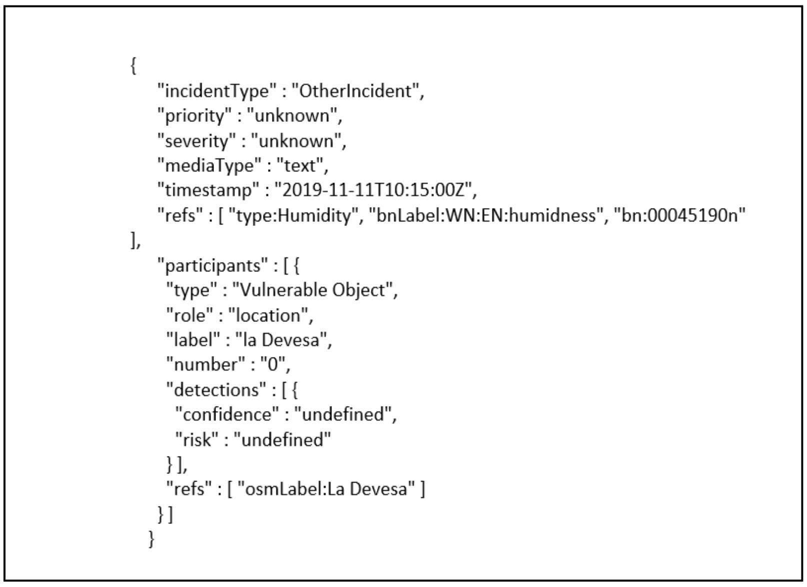
Figure 3: KBS sample for text- and audio-extracted events in JSON format

2.2 Summaries per time frame (*Sum$_{TIME}$*)

The *Sum$_{TIME}$* summaries are generated from an incoming set of multiple incidents, provided by the KBS. Those incidents are sorted by their timestamp and grouped in a given time frame (e. g. 30 minutes) in order to aggregate them for each time frame. For each group, incidents are processed in order to extract the necessary information for the grammar-based generation, discarding any duplicate incidents, as explained in the report description generation section. Once the information has been extracted, it is passed to the text generator and finally a start-end time header is added to each time frame group. Each event is stored in a JSON file with all its specifications.

2.3 From KBS to MRG

For both *Sum*$_{ALL}$ and *Sum*$_{TIME}$, each event is mapped to a linguistic structure ("conceptual" structure) for further processing. All events in *Sum*$_{ALL}$ and all events of a particular time frame *Sum*$_{TIME}$ are processed in one single structure, so that events can be grouped together and rendered as a cohesive paragraph (as opposed to individual sentences) by the generation module. Figure 4 shows a partial JSON structure for a *Humidity* event in la Devesa on 11 November 2019 at 10.15. Together with other events of the same time frame, this JSON is mapped to the corresponding Predicate-Argument template shown in Figure 5, used as input to the beAWARE generator (see Section 4).



```
{
   "incidentType" : "OtherIncident",
   "priority" : "unknown",
   "severity" : "unknown",
   "mediaType" : "text",
   "timestamp" : "2019-11-11T10:15:00Z",
   "refs" : [ "type:Humidity", "bnLabel:WN:EN:humidness", "bn:00045190n"
],
   "participants" : [ {
     "type" : "Vulnerable Object",
     "role" : "location",
     "label" : "la Devesa",
     "number" : "0",
     "detections" : [ {
       "confidence" : "undefined",
       "risk" : "undefined"
     } ],
     "refs" : [ "osmLabel:La Devesa" ]
   } ]
  }
```

Figure 4: Partial KBS sample in the JSON format

```
1     report       tem_constituency=PERF       0     ROOT
2     humidity     dpos=NN|definiteness=no     1     A2
3     La Devesa    dpos=NP                     1     Location
```

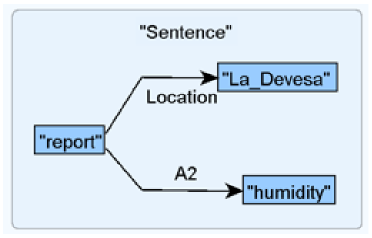


Figure 5: Partial generator input sample (Predicate-Argument template); top in CoNLL format, bottom in graphical format

For the time frame “10:10-10:20”, the following text is returned by the MRG pipeline for all events of that time frame:

> *Humidity and strong wind have been reported in La Devesa. Fire there. Risk of fire, strong wind and heat. Heat has been reported in Valencia.*

The illustration of the different steps performed by the MRG pipeline is provided in Section 4.1.

3 Final datasets and Surface Realisation Shared Tasks

As mentioned in the introduction, UPF tackled the large-scale annotation of Universal Dependency-based datasets to be used as training material for Natural Language Generation Systems. In order to validate these datasets and make the community aware of their existence, in 2018 and 2019, UPF co-organised the first two international shared tasks on multilingual surface realisation (SR'18 and SR'19), using as starting point data obtained from the official UD repository. In this section, we give an overview of (i) the tasks, (ii) the creation of the datasets and (iii) of the tools used to obtain them. The results obtained by the participating teams are presented in Section 6.1

3.1 Task overview

Universal Dependencies (UDs) is a generic framework for cross-lingual syntactico-semantic annotation that has been applied to over 80 languages so far, for a total of over 140 different treebanks.[1] Most treebanks have been obtained through automatic conversions of other treebanks, who themselves have been obtained via automatic annotation. The resulting annotations are known to lack consistency and quality but they have the advantage to provide a framework that reduces the differences across different languages. In beAWARE, we developed the first multilingual UD-based dataset for training statistical generators.

The annotated surface structures are syntactic trees with lemmas, part-of-speech tags, morphological and dependency information under the form of grammatical functions such as *subject*, *object*, *adverbial*, etc. We developed a converter for UD structures to obtain parallel "deep" data and thus serve as input for deep generators as part of WP5. By using the structures at these two levels, we have two different outputs for Natural Language Generation (in bold, the beAWARE languages, which were the only supported languages in D3.3):

- Shallow Track (T1)
    - Input: unordered UD dependency trees with lemmatised words that hold PoS tags and morphological information
    - Task: determine word order and inflect words
    - Languages with data: Arabic, Chinese, Czech, Dutch, **English**, Finnish, French, **Greek**, Hindi, Indonesian, **Italian**, Japanese, Korean, Portuguese, Russian, **Spanish**
- Deep track (T2)
    - Input: unordered predicate-argument tree with lemmatised content words that hold coarse-grained PoS tags and semantic information
    - Task: introduce functional words, resolve morphological agreements, determine word order and inflect words

[1] http://universaldependencies.org/

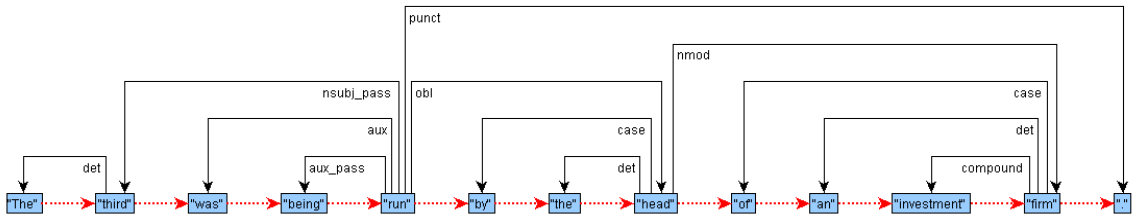
- Languages with data: Chinese, **English**, French, **Greek**, **Italian**, Portuguese, **Spanish**

3.2 General specifications of the data

The shallow track structures are obtained by simply removing the order and surface forms information from the original structures. The deep structures in this configuration consist of predicate-argument structures obtained through the application of graph-transduction grammars to the UD surface-syntactic structures. The deep and surface structures are aligned node to node. In the deep structures, we aim at removing all the information that is language-specific and oriented towards syntax:

- determiners and auxiliaries are replaced (when needed) by attribute/value pairs, as, e.g., Definiteness, Aspect, and Mood:
  - auxiliaries: *was built-> build*;
  - determiners: *the building-> building*;
- functional prepositions and conjunctions that can be inferred from other lexical units or from the syntactic structure are removed:
  - *built by X-> built X*
- edge labels are generalised into predicate argument (semantics-oriented) labels in the PropBank/NomBank fashion:
  - *subject*(*built, by X)-> FirstArgument(build, X)*

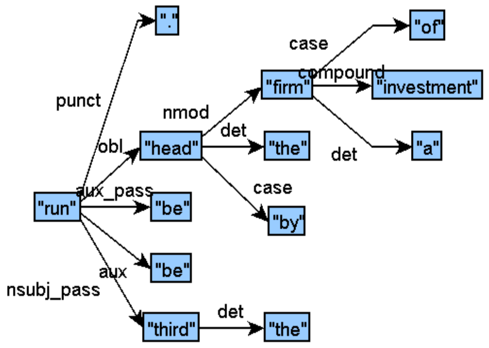
Figure 6, Figure 7, and Figure 8 show original, surface and deep structures respectively.



```
# sent_id = weblog-juancole.com_juancole_20051126063000_ENG_20051126_063000-0006
# text = The third was being run by the head of an investment firm.
```

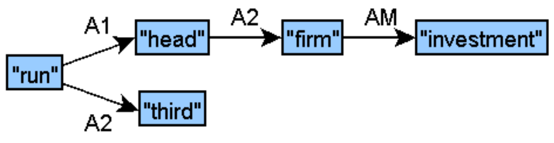
```
1   The         the         DET    DT   Definite=Def|PronType=Art                                 2   det
2   third       third       ADJ    JJ   Degree=Pos|NumType=Ord                                    5   nsubj_pass
3   was         be          AUX    VBD  Mood=Ind|Number=Sing|Person=3|Tense=Past|VerbForm=Fin     5   aux
4   being       be          AUX    VBG  VerbForm=Ger                                              5   aux_pass
5   run         run         VERB   VBN  Tense=Past|VerbForm=Part|Voice=Pass                       0   root
6   by          by          ADP    IN   _                                                         8   case
7   the         the         DET    DT   Definite=Def|PronType=Art                                 8   det
8   head        head        NOUN   NN   Number=Sing                                               5   obl
9   of          of          ADP    IN   _                                                         12  case
10  an          a           DET    DT   Definite=Ind|PronType=Art                                 12  det
11  investment  investment  NOUN   NN   Number=Sing                                               12  compound
12  firm        firm        NOUN   NN   Number=Sing                                               8   nmod
13  .           .           PUNCT  .    _                                                         5   punct
```

Figure 6: Original UD structure in the CoNLL-U format (top: graphical representation)



1	the	_	DET	DT	Definite=Def|PronType=Art	2	det
2	third	_	ADJ	JJ	Degree=Pos|NumType=Ord	3	nsubj_pass
3	run	_	VERB	VBN	Tense=Past|VerbForm=Part|Voice=Pass	0	root
4	be	_	AUX	VBD	Mood=Ind|Number=Sing|Person=3|Tense=Past|VerbForm=Fin	3	aux
5	be	_	AUX	VBG	VerbForm=Ger	3	aux_pass
6	head	_	NOUN	NN	Number=Sing	3	obl
7	.	_	PUNCT	.	_	3	punct
8	by	_	ADP	IN	_	6	case
9	the	_	DET	DT	Definite=Def|PronType=Art	6	det
10	firm	_	NOUN	NN	Number=Sing	6	nmod
11	an	_	DET	DT	Definite=Ind|PronType=Art	10	det
12	investment	_	NOUN	NN	Number=Sing	10	compound
13	of	_	ADP	IN	_	10	case

Figure 7: Shallow track input in the CoNLL-U format (top: graphical representation)



1	third	_	ADJ	_	Degree=Pos	2	A2
2	run	_	VERB	_	Tense=Past|Aspect=Progr	0	ROOT
3	head	_	NOUN	_	Number=Sing|Definiteness=Def	2	A1
4	firm	_	NOUN	_	Number=Sing|Definiteness=Indef	3	A2
5	investment	_	NOUN	_	Number=Sing	4	AM

Figure 8: Deep track input in the CoNLL-U format (top: graphical representation)

3.3 A converter for obtaining the Deep track structures

The beAWARE Deep UD grammars are used as an automatic converter to obtain the Deep UDs. The Deep UD grammars do not make any use of lexical resources; the predicate-argument relations are derived using syntactic cues only. The deep input is a compromise between correctness and adequacy in a generation setup. Indeed, the conversion of the UD structures into predicate-argument structures depends not only on the mapping process, but also on the availability of the information in the original annotation. Table 1 (repeated from D3.3) shows the different labels that the Deep UD grammars currently produce. More information can be found in D3.3.

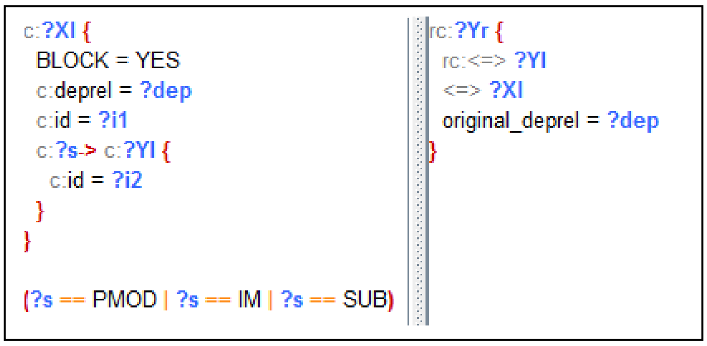
Table 1: Semantic labels in the output of the UD-based pipeline

Semantic label	Type	Description	Example
A1/A1INV	Core	1st argument of a predicate	build-> an architect
A2/A2INV	Core	2nd argument of a predicate	build-> a building

A3/A3INV	Core	3rd argument of a predicate	inaugurate-> on March 15
A4, A5, A6	Core	4th to 6th arguments	*Very uncommon*
AM	Non-Core	None of governor or dependent are argument of the other	build-> next to the museum
LIST	Coordinative	List of elements	built-> and-> inaugurated
NAME	Lexical	Part of a name	Chrysler-> Building
DEP	UKN	Undefined dependent	N/A

As described in D3.3, our Deep UD grammars are rules that apply to a subgraph of the input structure and produce a part of the output structure. During the application of the rules, both the input structure (covered by the left side of the rule) and the current state of the output structure at the moment of application of a rule (i.e., the right side of the rule) are available as context. The output structure in one transduction is built incrementally: all the rules are evaluated and the ones that match a part of the input graph are applied and a first piece of the output graph is built. Subsequently, the rules are evaluated again, this time with the right-side context as well, and another part of the output graph is built, and so on. The transduction is over when no rule is left that matches the combination of the left-side and the right-side. As an example, we present a sample rule from the SSynt-DSynt mapping in Figure 9. This rule, in which we can see the left-side and the right-side fields, collapses the functional prepositions (*?Xl*, identified during the pre-processing stage with the *BLOCK=YES* attribute/value pair) with their dependent (*?Yl*). That is, a functional preposition such as *'by'* in *'built by Y'* is removed from the output structure and made to correspond to the right-side node *Y* (i.e., the dependent). The right-side context is indicated by the prefix *'rc:'* before a variable or a correspondence[2]. In practice, this means that the rule looks for the *'rc:'*-marked elements in the current state of the output structure and builds the elements that are not *'rc:'*-marked; in this case, the correspondence between the right-side *'Y'* and the left-side *'by'* and the new feature *'original_deprel'*, which stores the left-side incoming dependency relation. A similar rule would apply to *'firm'* and *'of'*, with *'of'* being the dependent in this configuration (see Figure 7). As a result of the application of this rule, only *'firm'* is left in Figure 8, which has a correspondence with both *'firm'* and *'of'* from Figure 7.

[2] Correspondences are meta-information used during the transduction. They are not explicit as such in the output structure. In order to maintain the alignments between surface and deep nodes, attribute/value pairs can be used: e.g. if *"by"* has a surface identifier "id=2", and *"Y"* id = "3", the deep *"Y"* node could have two identifiers "id=2,3" to mark the correspondence.



```
c:?Xl {                                   rc:?Yr {
  BLOCK = YES                               rc:<=> ?Yl
  c:deprel = ?dep                           <=> ?Xl
  c:id = ?i1                                original_deprel = ?dep
  c:?s-> c:?Yl {                          }
    c:id = ?i2
  }
}

(?s == PMOD | ?s == IM | ?s == SUB)
```

Figure 9: A sample graph-transduction rule. ‘?’ indicates a variable, ‘?Xl{}’ is a node, ‘?s’ is a relation, ‘a=?b’ is an attribute/value pair.

*Includes rules that simply copy node features (~50 per grammar)

sums up the current state of the graph-transduction grammars and rules for the mapping between surface-syntactic structures and UD-based semantic structures. The main improvements since D3.3 are

i. the support for three additional languages, namely Chinese, French and Portuguese,
ii. an increase of the number of rules, to account mainly for the coverage of the new languages and of new phenomena present in the latest version of the UD data,
iii. the addition of two new rule sets for allowing different downstream applications to use the knowledge encoded in the UD trees (Graph UD and Deep-Syntax in the sense of the Meaning Text Theory; see D3.3 Section 6.2.2).

Table 2: Graph-transduction rules for UD-based deep parsing

Grammars	# rules D3.3*	#rules D3.4*	Description
Pre-processing	76	93	Identify nodes to be removed Identify verbal finiteness and tense
SSynt->Deep UD	120	147	Remove idiosyncratic nodes Establish correspondences with surface nodes Predict predicate-argument dependency labels Replace determiners, modality and aspect markers by attribute-value feature structures Identify duplicated core dependency labels below one predicate
Post-processing	60	73	Replace duplicated argument relations by best educated guess Identify remaining duplicated core dependency labels (for posterior debugging)
Deep UD->Graph UD	-	70	Converts the Deep UD tree into a graph
Deep UD->DSynt	-	72	Converts the Deep UD tree into a Deep-Syntactic tree

*Includes rules that simply copy node features (~50 per grammar)

3.4 Dataset variety and sizes

Table 3 shows the properties of the respective SR'18 datasets in terms of number of sentences (*Mille et al., 2018*).

Table 4 shows the dataset variety and sizes at SR'19. For SR'19 (*Mille et al., 2019*), new languages and data features were introduced, in particular:

- Evaluation: Both in-domain and out-of-domain data were used in the test data. For some languages, automatically parsed texts needed to be generated.
- Data alignment: In the training sets, the Shallow data were fully aligned with the original UD structures and the Deep data were fully aligned with both the original and Shallow structures.
- Relative word order information: Relative word order information in Multiword Expressions, punctuations and multiple coordinations was available in the input.
- Multiple datasets: For some languages, there were two or more UD datasets. The teams were allowed to choose which dataset(s) they want to use for training their models.

Table 3: SR'18 dataset sizes and splits.

	ar	cs	en	es	fi	fr	it	nl	pt	ru
train	6,016	66,485	12,375	14,289	12,030	14,529	12,796	12,318	8,325	48,119
dev	897	9,016	1,978	1,651	1,336	1,473	562	720	559	6,441
test	676	9,876	2,061	1,719	1,525	416	480	685	476	6,366

Table 4: SR'19 dataset sizes and splits

Data type	Dataset	Track	train	dev	test
In-domain	arabic_padt (ar)	T1	6,075	909	680
	chinese_gsd (zh)	T1	3,997	500	500
	english_ewt (en)	T1, T2	12,543	2,002	2,077
	english_gum (en)	T1, T2	2,914	707	778
	english_lines (en)	T1, T2	2,738	912	914
	english_partut (en)	T1, T2	1,781	156	153
	french_gsd (fr)	T1, T2	14,450	1,476	416
	french_partut (fr)	T1, T2	803	107	110
	french_sequoia (fr)	T1, T2	2,231	412	456
	hindi_hdtb (hi)	T1	13,304	1,659	1,684
	indonesian_gsd (id)	T1	4,477	559	557
	japanese_gsd (ja)	T1	7,133	511	551
	korean_gsd (ko)	T1	4,400	950	989
	korean_kaist (ko)	T1	23,010	2,066	2,287
	portuguese_bosque (pt)	T1	8,328	560	477
	portuguese_gsd (pt)	T1	9,664	1,210	1,204
	russian_gsd (ru)	T1	3,850	579	601
	russian_syntagrus (ru)	T1	48,814	6,584	6,491
	spanish_ancora (es)	T1, T2	14,305	1,654	1,721
	spanish_gsd (es)	T1, T2	14,187	1,400	426
Out-of-domain	english_pud (en)	T1, T2	-	-	1,000
	japanese_pud (ja)	T1	-	-	1,000
	russian_pud (ru)	T1	-	-	1,000
Automatically parsed	english_ewt-HIT (en)	T1, T2	-	-	1,795
	english_pud-LAT (en)	T1, T2	-	-	1,032
	hindi_hdtb-HIT (hi)	T1	-	-	1,675
	korean_kaist-HIT (ko)	T1	-	-	2,287
	portuguese_bosque-Sta (pt)	T1	-	-	471
	spanish_ancora-HIT (es)	T1, T2	-	-	1,723

3.5 Evaluation of the quality of deep datasets

Since the processing applied to the Shallow inputs is very straightforward and is limited to removing information, it does not call for an evaluation. For the Deep Track, however, the changes are much more complex, and the quality of the conversion needs to be assessed. We evaluated the quality of the Deep inputs as follows. We manually annotated around 900 deep tokens (75 sentences) in each language (English and Spanish), by post-editing the automatically converted structures, correcting any mistakes. Since the same person post-edited all three datasets, the resulting gold-standard is consistent across the languages, even though it does not allow for calculating inter-annotator agreement. Once post-edited, the reference structures are compared to the ones produced by the automatic mapping from UD structures. Since part of the mapping consists of adding and/or removing nodes, it often happens that the gold-standard and predicted structures end up with a different number of nodes, which makes evaluation scripts based on a strict node-to-node comparison unusable. Thus, we use the LAS evaluation method of Ballesteros et al. (2015), specifically designed to handle the comparison between non-isomorphic trees.

Language	**LAS**
English	79.83
Spanish	67.28

shows the results of the evaluation.

Quality is not the same across the two languages; while English structures obtain a LAS of 79.83, Spanish is more than 12 points lower. Since the mapping grammars are largely language-independent, and since roughly the same efforts have been dedicated to each language, it is likely that the LAS numbers reflect the quality of the original UD annotation. Note that during the evaluation, Part of Speech (PoS) errors and lemmatisation errors are not corrected but structural errors due to original tagging/lemmatising errors are counted. In other words, what is being evaluated is how correct the outputs are in terms of dependencies and labelling, rather than how well the transduction grammars perform. Error analysis showed that most dependency errors come from the AM relation, which is usually A1, A2, A1INV or A2INV in the reference structures. The systematic replacement of AM by one of these four labels always results in a drop of the LAS score. That is, in order to improve the quality of the structures, an improvement of the UD structures or a more fine-grained processing (which would imply a large number of rules and the use of detailed lexicons) would be needed.

Table 5: Evaluation of the quality of the output structures (Labeled Attachment Scores - LAS).

Language	**LAS**
English	79.83
Spanish	67.28

4 Multilingual report generation

This section focuses on the extension of UPF's multilingual discourse generators developed for multilingual report generation in a series of European projects to an incremental expressive generator that fits the needs of the beAWARE MRG pipeline.

4.1 General approach and summary of advances

As detailed in D5.2, discourse generation starts from the ontological assertions that comprise the content inferred through the interplay of the interaction manager and the knowledge base. Ontology substructures are mapped onto minimal Predicate-Argument (PredArg) templates, which are then aggregated together into more complex semantic graphs, which, in turn, are realised into well-formed sentences in several steps. The generation is performed step by step, by successively mapping one level of representation onto the adjacent one. There are in practice 14 successive mappings (for instance, the aggregation takes place in 4 sub-steps) summarised in the following 6 transitions, to be used as references for Table 6 (Sub-section 4.2.4), with illustrations in Spanish and English.

Start (input): *Ontology mapped to individual Predicate-Argument Templates (Con)*

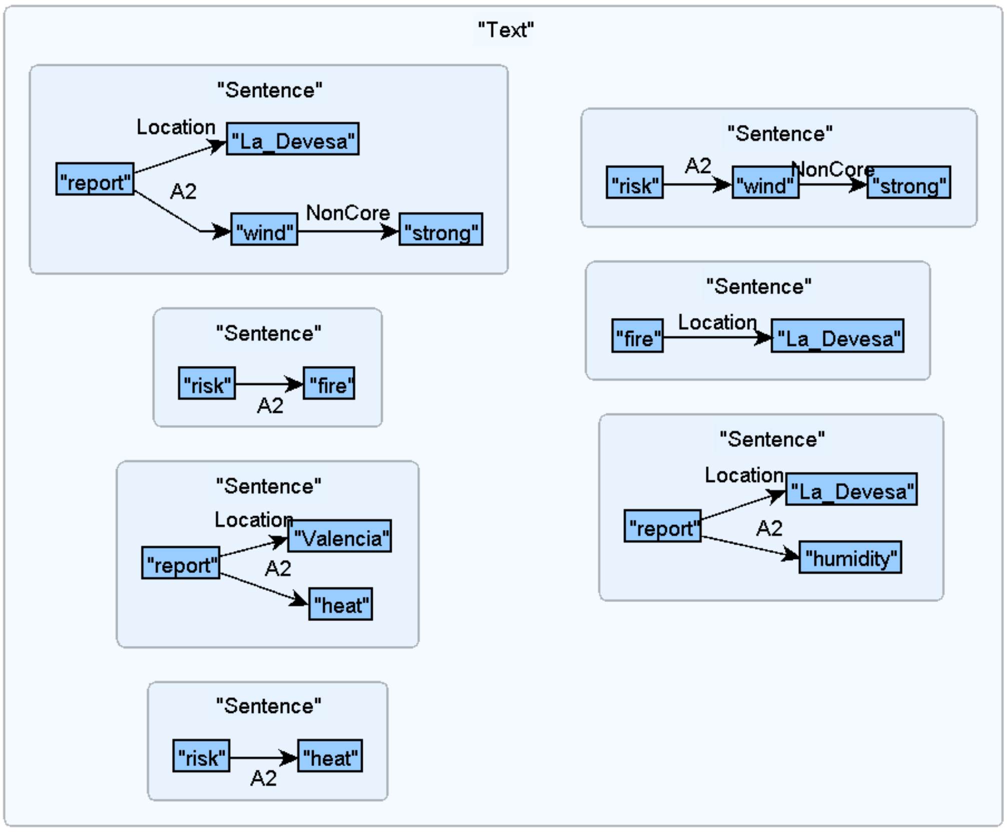


Figure 10: Start (input)

1) ***Aggregation***: *Aggregation of conceptual Predicate-Argument Templates (Con)*

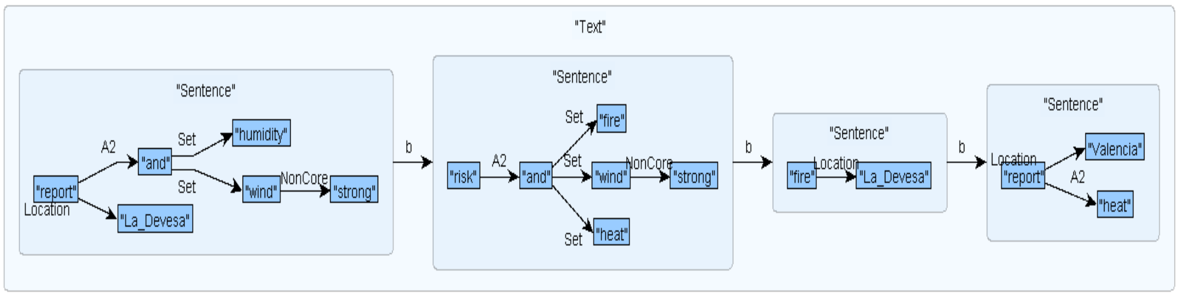


Figure 11: Aggregation

*2) **Con-Sem**: Mapping to language-specific Semantic Structures (Sem)*

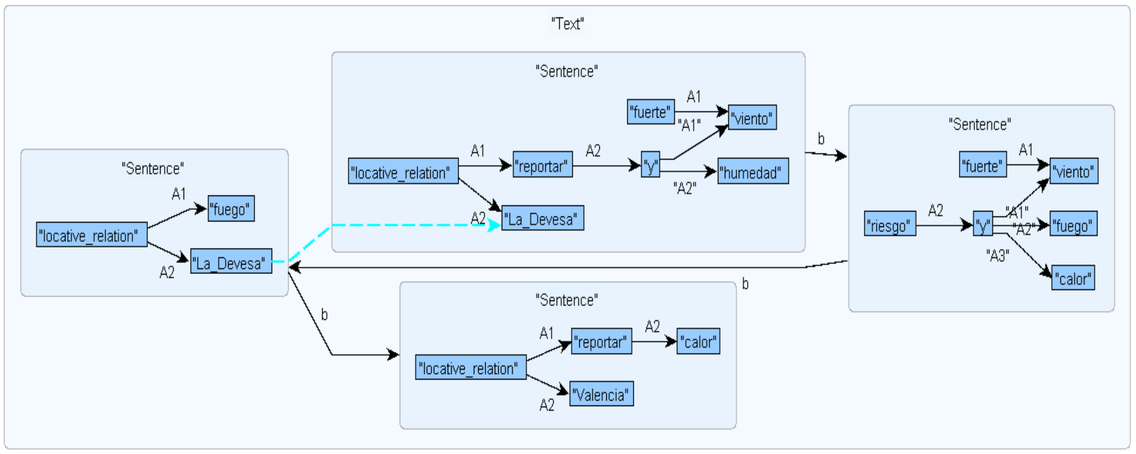


Figure 12: Con-Sem in Spanish

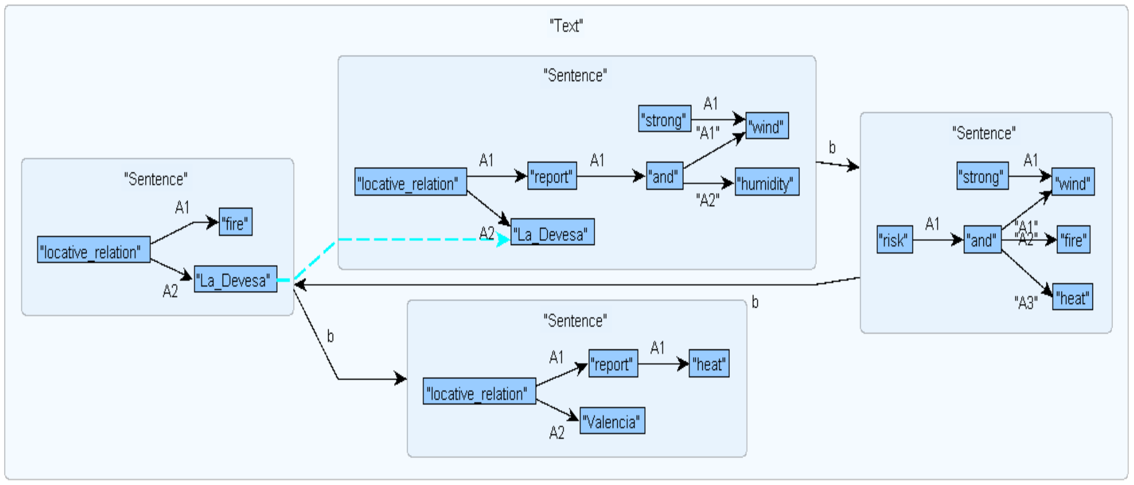


Figure 13: Con-Sem in English

3) ***Sem-DSynt**: Sentence structure determination (mapping to Deep-Syntactic Structures - DSynt)*

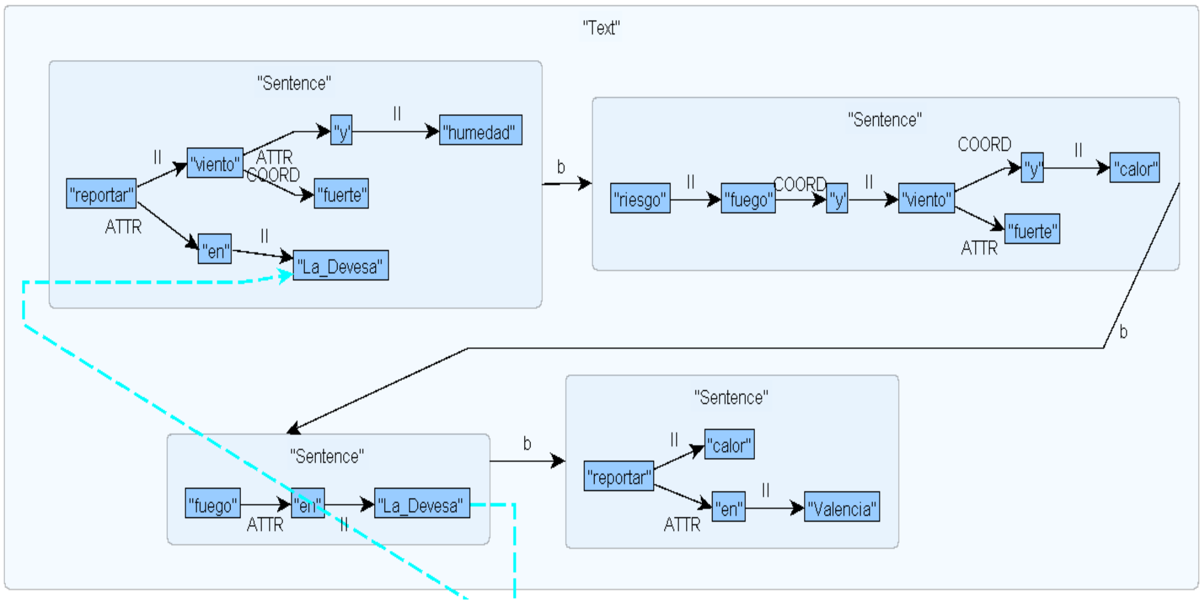


Figure 14: Sem-DSynt in Spanish

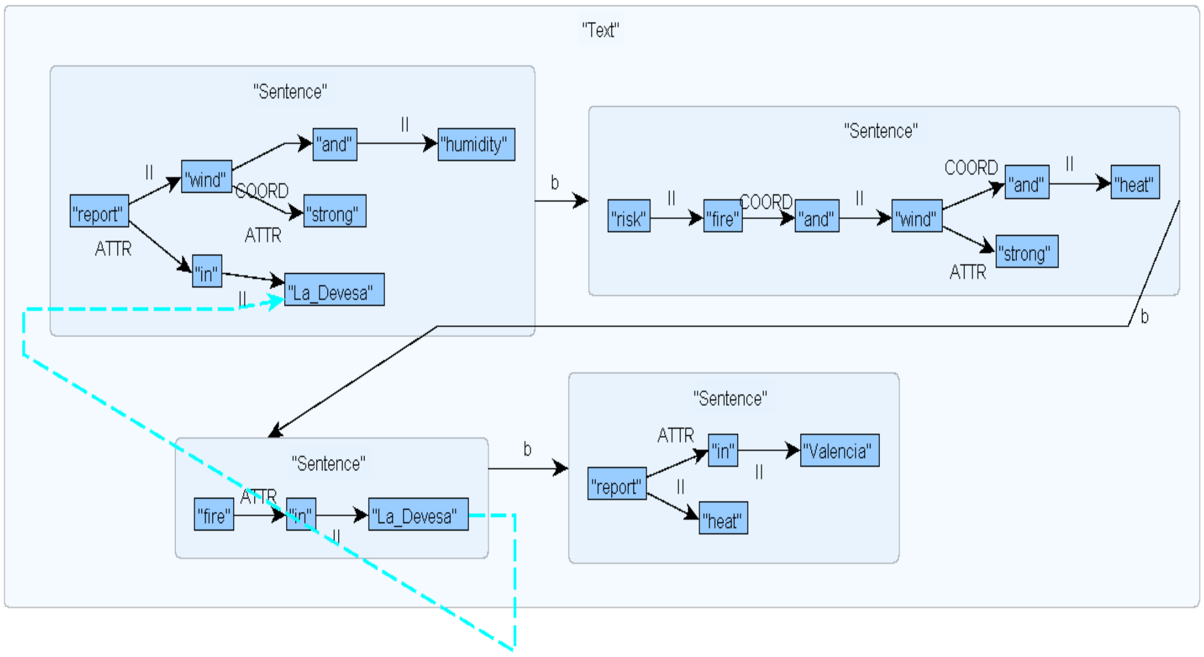


Figure 15: Sem-DSynt in English

4) ***DSynt-SSynt**: Introduction of idiosyncratic words and relations (mapping to Surface-Syntactic Structures - SSynt)*

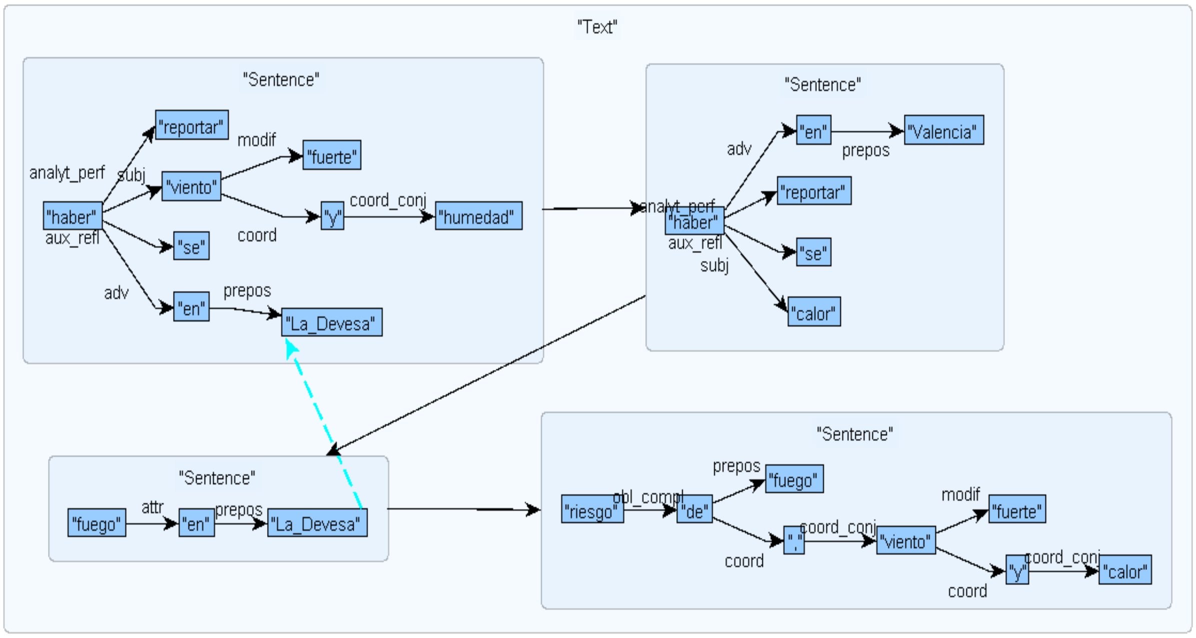


Figure 16: DSynt-SSynt in Spanish

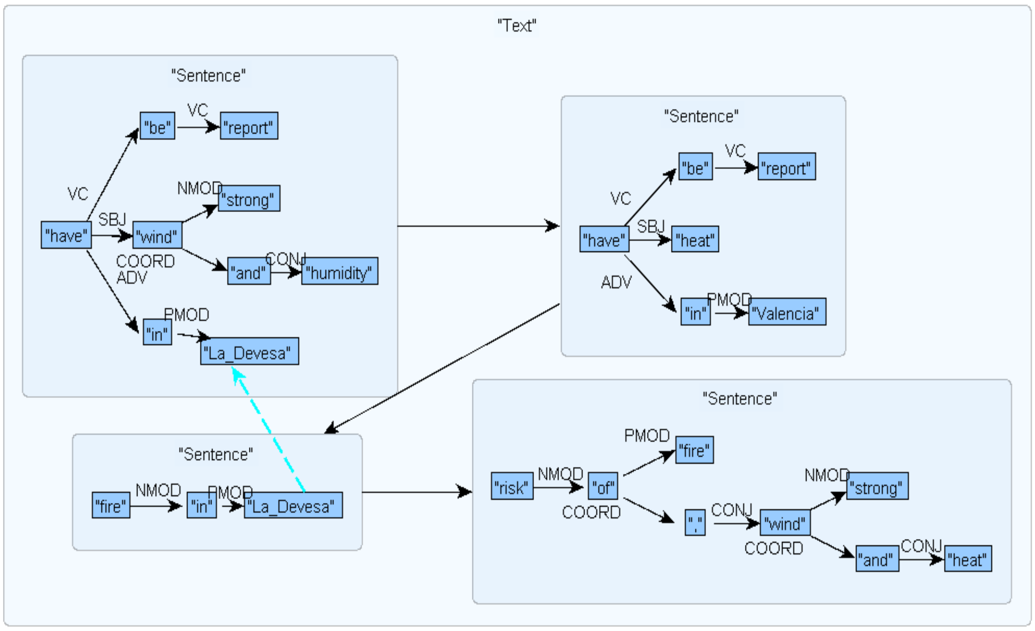


Figure 17: DSynt-SSynt in English

5) ***SSynt-DMorph**: Linearisation and morphological agreement resolution (mapping to Deep-Moprhologic Structures - DMorph)*

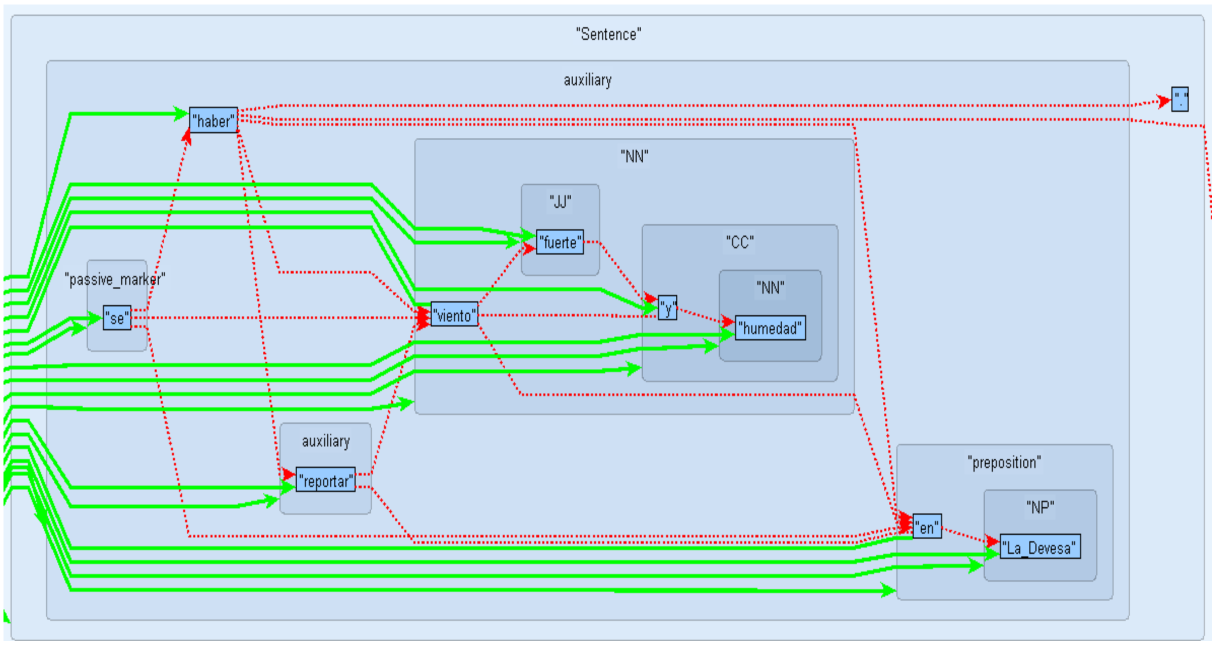


Figure 18: SSynt-DMorph in Spanish (partial)

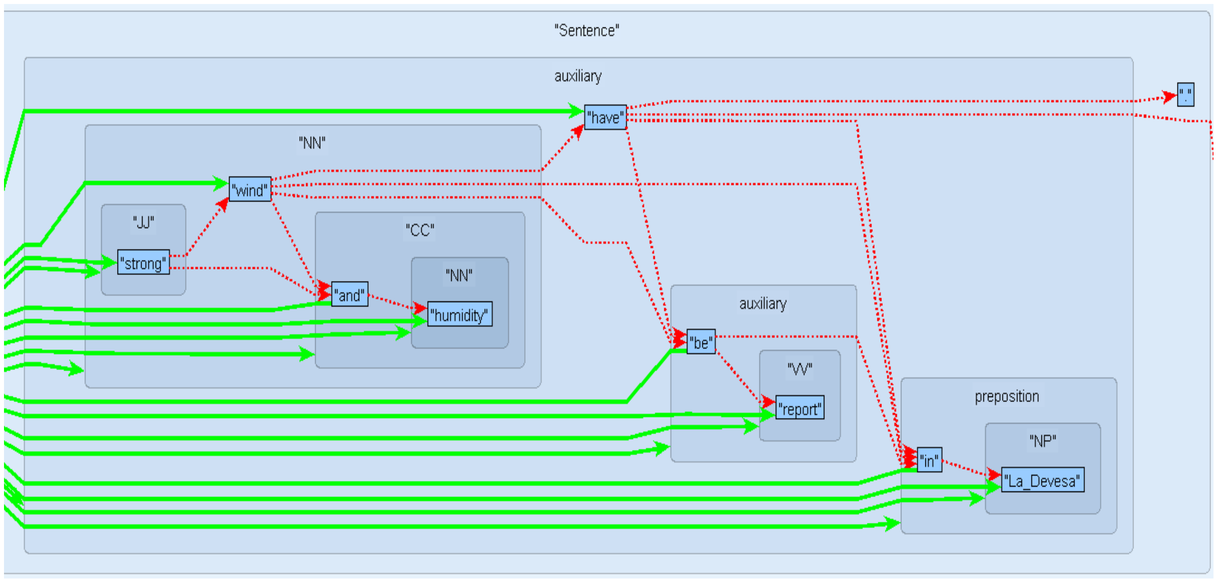


Figure 19: SSynt-DMorph in English (partial)

6) ***DMorph-SMorph**: Retrieval of surface forms (mapping to Surface-Morphologic Structures - SMorph)*

Spanish (partial):

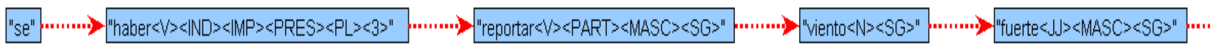

"se"
"haber<V><IND><IMP><PRES><PL><3>"
"reportar<V><PART><MASC><SG>"
"viento<N><SG>"
"fuerte<JJ><MASC><SG>"

Figure 20: DMorph-SMorph in Spanish (partial)

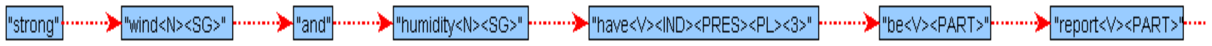


Figure 21: DMorph-SMorph in English (partial)

***End (output):** Full-fledged report:*

> ***Spanish**: Se han reportado viento fuerte y humedad en La Devesa. Riesgo de fuego, viento fuerte y calor. Fuego en La Devesa. Se ha reportado calor en Valencia.*
>
> ***English**: Strong wind and humidity have been reported in La Devesa. Risk of fire, strong wind and heat. Fire in La Devesa. Heat has been reported in Valencia.*

The underlying linguistic framework is the Meaning-Text Theory (*Mel'čuk, 1988*), which foresees a very similar stratification in language description. Each transition is performed with graph-transduction grammars, using the MATE environment (*Bohnet and Wanner, 2010*) and for some of the transitions we have developed advanced techniques. In this section, we mainly report the following improvements since D5.2:

- Basic techniques:
  - Increase of the coverage of the language-specific generation grammars, in particular for English and Spanish,
  - Increase of the coverage and quality of the language-independent grammars; in particular of the semantic aggregation module to package the beAWARE summaries coherently into small texts.
- Advanced techniques:
  - Development of new deep learning linearisers for English, Spanish, Italian and Greek.

4.2 Main improvements in the grammars and lexicons of the rule-based generator

4.2.1 Extensions to language-independent rules

In D5.2, we reported on the development of the rule-based generator, which performs all the transitions between two consecutive layers using graph-transduction grammars. For the specific needs of beAWARE, since D5.2, the main improvement needed to complete the final prototype was the development of better aggregation grammars that combine the PredArg structures together into what will be realised as a single sentence, in a way that supports the summaries that need to be provided by the beAWARE platform.

With the extension of the aggregation grammars, there are 5 main types of aggregation currently supported. Each type includes several subtypes based on the context in which the contents to be aggregated appear. We describe here the 5 types of aggregations that take place within the Predicate-Argument templates, which involve the following types of components:

- Predicates P_n, which usually correspond to a main verb: *report, detect, impact, overflow*, etc.
- Arguments A_n, which are the participants of the predicates: *fire, flood, person, building, river*, etc.
- Argument slots S_n, which are the type of relation that links a predicate with an argument: first slot (*someone detects*), second slot (*something is detected*), etc. In the beAWARE context, so far there is only one argument slot for each predicate, but the aggregation rules support also multiple argument slots. The examples to illustrate these cases are made up and do not correspond to actual outputs produced by the MRG pipeline.
- Locations L_n, which specify the place where the event denoted by the predicate or the argument took place: *el Saler, Mateotti Square, Valencia*, etc.
- Times T_n, which specify the time when the event denoted by the predicate or the argument took place: *2PM, midnight*, etc. These are usually not verbalised by the MRG pipeline since the reports are sent together with the corresponding timestamp.

The aggregation rules examine all Predicate-Argument templates, and group two or more of them (the same value for the index n indicates sameness):

1) **Fusion of predicate and one argument**: if two or more predicates, one of their arguments and the relation between them (the argument slot) are the same, we bring the other arguments/locations together, either by coordinating or juxtaposing them, depending on the configuration.
   Coordination*: [Fire]A_1S_1 [detected]P_1 [in Valencia]L_1 + [Fire]A_1S_1 [detected]P_1 [in el Saler]L_2 = Fire detected in Valencia and el Saler*.
   Juxtaposition*: [Fire]A_1S_1 [detected]P_1 [in Valencia]L_1 + [Fire]A_1S_1 [detected]P_1 [at 2PM]T_1 = Fire detected in Valencia at 2PM*.
2) **Fusion of predicate and location (or time)**: if a predicate and the location (or time) is the same, but one argument is different (but with the same slot), then the arguments are coordinated.
   [Two persons]A_1S_1 [are impacted]P_1 [in Valencia]L_1 + [Three dogs]A_2S_1 [are impacted]P_1 [in Valencia]L_1 = Two persons and three dogs are impacted in Valencia.
3) **Fusion of predicate only**: if two predicates have different arguments with the same argument slot, we coordinate the arguments.

[Fire]A_1S_1 [detected]P_1 + [Smoke]A_2S_1 [detected]P_1 = Fire and smoke detected.

4) **Coordination of predicates**: if two predicates are the same but have no argument slot in common, or they have one argument slot in common but the argument is not the same, the predicates are coordinated. In this case, the second occurrence of the verb is elided in order to avoid repetitions.
   [Two persons]A_1S_1 [are impacted]P_1 [in Valencia]L_1 + [Three persons]A_2S_1 [are impacted]P_1 [in el Saler]L_2 = Two persons are impacted in Valencia and three persons in el Saler.
5) **Fusion of one argument only**: if two arguments are the same, but the surrounding elements are different, we introduce an embedded clause (relative or participial clause). This type of aggregation covers what is referred to as *subject progression* and *object progression* in the literature.
   [Fire]A_1S_1 [detected]P_1 [in Valencia]L_1 + [Fire]A_1S_1 [reported]P_2 [in el Saler]L_2 = Fire, which was detected in Valencia, is reported in el Saler.

Note that the rules sometimes find more than one possible aggregations, but only one type applies at a time. However, there are two rounds of aggregation, which means that complex aggregations are possible:

Fire, which was detected in Valencia, is reported in el Saler and la Devesa.

In order to foresee the packaging of the sentences, starting from complex semantic graphs instead of minimal triple-based structures, we developed a statistical sentence packaging module based on community detection algorithms. Since the work was not part of the DoA, we do not report in detail on this tool; the experiments are described in (Shvets et al., 2018).

In addition to adding functionalities to generator, some detected issues after the evaluation of the first prototype were fixed. Indeed, even though FORGe received good evaluation marks at the WebNLG challenge, as shown in D5.2, especially in the human assessments, according to which it was close to the quality of human-written text, after an error analysis of FORGe's outputs, we found a series of general problems impairing the quality of the generated texts in terms of contents and grammaticality. In particular: (i) some properties were not verbalised due to the failure to produce relative clauses in some specific cases, (ii) the aggregations were at times excessive, erroneously merging verbs with different tenses (e.g. *X impacted Y, which was impacted by Z*, instead of *X and Z were impacted by Y*), failing to merge (e.g. *X was impacted by Y. Z was impacted by Y*), or leading to an ungrammatical outcome, with, for instance, the presence of several *'also'*, (iii) the construction of some relative clauses were faulty, as e.g. *X can a variation of which be Y*, instead of *X, which can be a variation of Y*; (iv) the referring expression module was applying excessively, resulting in ambiguous pronouns, and sometimes incorrectly pronominalising non-human entities with *'he'*, or failing to pronominalise locations, such as in *'A fire has been detected in Valencia. Fire in Valencia'*, the second sentences being more naturally rendered by *'Fire there'*, (v) some agreements were not solved (e.g. *'the main ingredient are'*), (vi) some determiners were erroneously introduced, and some others not in the correct form (*a* instead of *an*). All these issues could potentially affect beAWARE outputs, in case such cases would be found in the inputs to the generator. Many occurrences of these issues were fixed in the grammars, by modifying and adding rules, and some new features were added, as for instance new rules to cover more cases of embedded clauses generation. For developing the grammars, we used the collection of beAWARE inputs, 6 and 7 triple inputs from the WebNLG training data, and the whole WebNLG development set. A qualitative evaluation of the new outputs is provided in Section 5.1.

4.2.2 **Extensions to the Spanish rules**

The final pilot in beAWARE was on the Spanish use case. Since the necessary coverage of the generator had been previously ensured for the Greek and Italian use cases, we took advantage of the Valencia pilot to increase significantly the coverage of the rules for the Spanish part of the generator.

The most important rules added for Spanish are: (i) rules introducing the surface-syntactic relations, based on which linear order and morphological agreements are resolved, (ii) rules for gender and number agreements in noun groups and auxiliary constructions, and (iii) word ordering rules. Note that the rules for Spanish also apply to other Romance languages with similar features (e.g. French, Italian, etc.).

For designing the rules, we followed the approach of AnCora-UPF (*Mille et al., 2013*), a Spanish dataset in which each dependency relation is associated with a set of syntactic properties. For instance, a *subject* is characterised by being linearised to the left of its governing verb (by default), by being removable, by triggering the number and person agreements on the verb, etc. During the linguistic generation stage, 27 out of the 47 relations proposed in AnCora-UPF -namely *adjunct, adv, agent, analyt_fut, analyt_pass, analyt_perf, analyt_progr, aux_phras, appos, attr, compar, coord, coord_conj, copul, det, dobj, iobj, modal, modif, obl_compl, obl_obj, prepos, punc, quant, relat, sub_conj, and subj*- are currently supported.

In order to generalise the ordering rules across languages, the dependencies were introduced in the lexicon with details about how they are linearised with respect to their governor (*vertical* ordering). Generic linearisation rules also apply. For instance, for the *copul* dependency (such as between *be* and *detected*), pronominal dependents are linearised BEFORE the finite verb, and the other dependents AFTER it. If several dependents end up at the same height with respect to their governor, they need to be ordered with each other. 21 rules were added to manage these *horizontal* orderings. They facilitate the ordering of, for instance, determiners before the adjectives, or small adverbial groups before the objects. Finally, 18 rules for resolving the agreements between verb and subject, adjective/determiner and noun, copulatives and subjects, etc. were implemented.

For instance, in the structure Un fuego$_{3\text{-MASC-SING}}$ y viento$_{3\text{-MASC-SING}}$ <-subj *ser* copul-> *detectado*, will be linearised and inflected as follows: *Un fuego y viento son detectados* (lit. `A fire and wind are$_{3\text{-PL}}$ detected$_{\text{MASC-PL}}$').

4.2.3 Crafting the Spanish dictionaries

Several types of dictionaries are needed for generation of: (i) a dictionary that maps the input meanings/concepts onto lexical units of a particular language (called *concepticon*), (ii) a dictionary that contains the combinatorial properties of each lexical unit (*lexicon*), (iii) a dictionary with the full forms of the words (called *morphologicon*). Some other information, such as linearisation properties of dependencies are also better stored in the lexicon in order to allow for more generic (hence less numerous) rules.

In our generation architecture, the input structured data (ontology substructures or DBpedia properties) are mapped to PredArg structures. For the WebNLG challenge, English was the only language to generate, so the labels of the nodes in the PredArg templates were in English. In order to take advantage of the templates developed for FORGe in 2017, we also used these structures with English vocabulary as input to the generator. Thus, we manually crafted the concepticon (255 entries), in which the keys are the predicates from the templates, and the values are lexical units in Spanish; for instance, the predicate *locate* is mapped to the Spanish verb *estar_VB_04* ('be').

In the lexicon, lexical units such as *estar_VB_04* are described; this fourth entry for *estar* corresponds to a verb that has two arguments, the second being an adverb or a prepositional group. *estar_VB_01* is the simple copula, *estar_VB_02* is the existential *be*, which has only one argument, and *estar_VB_03* is the auxiliary. Each lexical unit contained in the concepticon is a key in the lexicon. A fine-grained lexicon has been crafted manually for the beAWARE and WebNLG experiments, and we developed an automatic conversion of the large-scale AnCora-Verb (Aparicio 2008) to obtain a large coverage resource. Finally, in order to store the surface forms of the inflected words, we crafted a very small morphological dictionary of about 450 entries to cover the needed forms in the experiments.

4.2.4 Overview of the evolution of the FORGe generator in beAWARE

Table 6: Comparison between Initial, P2 and P3 generators. The table summarises the improvements of the components of the generator during the course of the project.

		beAWARE Start	beAWARE D5.2	beAWARE D5.3
Languages		EN (+very basic ES)	EN, IT, ES, EL	EN, IT, ES, EL
Number of rules and % of language-independent rules	ALL	**Con-SMorph** (1060) : 66%	**Con-SMorph** (1,373) : 70%	**Con-SMorph** (1,742) : 70%
		1) Con-Sem (358) : 93% 2) Aggregation (0) : N/A 3) Sem-DSynt (157) : 73% 4) DSynt-SSynt (331) : 42% 5) SSynt-DMorph (130) : 54% 6) DMorph-SMorph (35) : 51%	1) Con-Sem (416) : 94% 2) Aggregation (212) : 91% 3) Sem-DSynt (177) : 75% 4) DSynt-SSynt (307) : 39% 5) SSynt-DMorph (172) : 48% 6) DMorph-SMorph (89) : 55%	1) Aggregation (251) : 100% 2) Con-Sem (429) : 98% 3) Sem-DSynt (220) : 71% 4) DSynt-SSynt (451) : 46% 5) SSynt-DMorph (230) : 53% 6) DMorph-SMorph (161) : 40%
Main linguistic phenomena supported	EN	• Advanced sentence structure ○ Argumental dependents ○ Circumstantials • Lexicon-based introduction of functional elements: ○ Functional prep/conj ○ Modals and auxiliaries	• Improved sentence structure: ○ Coordinations ○ Embedded clauses ○ Complex syntactic structures • Advanced linearisation	• Improved linearisation • Improved aggregation strategies • Improved referring expressing expression generation

		• Verbal agreements	• Basic semantic and syntactic aggregations • Nominal compositionality	• Basic structure well-formedness checking • Complex support verb constructions • Complex relative clause construction • Advanced verb agreements (coordinations)
	IT	N/A	• Basic sentence structures: ○ Argumental dependents ○ Locative circumstancials • Lexicon-based introduction of functional elements: ○ Functional prep/conj ○ Auxiliaries ○ Determiners • Verbal, adjectival and det. agreements • Basic linearisation	
	ES	• Minimal sentence structure	• Basic sentence structures: ○ Argumental dependents ○ Locative circumstancials • Lexicon-based introduction of functional elements: ○ Functional prep/conj ○ Auxiliaries ○ Determiners • Verbal, adjectival and det. agreements • Basic linearisation	
	EL	N/A	• Basic sentence structures: ○ Argumental dependents ○ Locative circumstancials • Lexicon-based introduction of functional elements: ○ Functional prep/conj ○ Auxiliaries ○ Determiners • Verbal and det. agreements • Basic linearisation	• Improved aggregation strategies • Embedded clauses
Number of lexical units in lexicon	EN	41,838	41,955	41,895
	IT	0	63	75
	ES	324	1,852	12,332
	EL	0	20	20
BLEU score	**EN**	**31.78**	**35.53 (+11.80%)**	**39.84 (+25.36%)**

The above table summarises the extensions to the generator components, first with a count of the rules and lexical entries and a description of the covered phenomena in the different languages, and then with an objective evaluation of the quality of the outputs generated in English (last row). First of all, the aggregation now takes place before the mapping to language-specific structures, but the transition is essentially the same. The rule sets have been extended, with 70% more rules than at the beginning of the project (over 1,700 against about 1,000 initially), and generally been made more language independent (70% VS 66% of language-independent rules). The Sem-DSynt and DMorph-SMorph transitions contain a higher proportion of language-specific rules. For the Sem-DSynt transition, this is because some rules are currently substituting information that should be in the lexicon. On the long term, these rules will be replaced by (fewer) language-independent rules and the language-specific information will be stored in the respective lexicons. The reason is different for the

DMorph-SMorph transition, which is language-specific by nature, with the modelling of phenomena that are often highly idiosyncratic. Increasing the coverage of this grammar usually means adding language-specific rules, which makes their proportion increase. The last row of Table 6 shows the improvements of the performance of the generator on a generic dataset through the course of the project. Details on this evaluation are provided in Section 5.2, after an evaluation of the generator on the most challenging beAWARE-like dataset publicly available.

4.3 Development of a new deep learning-based lineariser

As mentioned in the introduction, UPF changed the approach followed for the advanced modules during the course of the project. In this deliverable, we report on new experiments on the application of deep learning methods for multilingual syntactic linearisation (step 5 in the previous sections). Our objectives are: (i) to develop a novel set-to-sequence syntactic linearisation model in order to be able to (ii) compare it to the state-of-the-art neural linearisation systems (provided here) and (iii) analyse and discuss the hypothesis in the context of the result comparison.

4.3.1 Related work

Linearisation can be classified as word ordering (*de Gispert et al., 2014; Schmaltz et al., 2016; Hasler et al., 2017*), when no syntactic information is available, or as decoding and syntactic tree linearisation, when syntactic information is available (*Bohnet et al., 2012; Liu et al., 2015; Puduppully et al., 2016; Song et al., 2018*).

In the recent years and due to the resurface of the neural networks, there have been some efforts in applying neural-based approaches to the task but all of these works fall short with respect to the state of the art in linearisation, represented by Puduppully et al. (2016) on English data and Bohnet et al. (2012) in multilingual data. One of our hypotheses is that most of the recent works in neural linearisation are based on sequence-to-sequence architectures, transferred from other tasks like tagging or parsing, but the linearisation problem input is a set, not a sequence, so the problem should be approached as a set-to-sequence problem (for word ordering) or a tree-to-sequence (for syntactic tree linearisation). We want to test that hypothesis by exploring a novel linearisation approach based on Gu et al. (2019).

In the syntactic linearisation task, like in any AI task, there are two main approaches: rule-based and statistical. Rule-based approaches map phrase or dependency structures to an ordered word sequence using manually crafted grammars that encode linguistic knowledge. Many different works have been produced, from the early works on phrase structure grammars (*Kathol and Pollard, 1995, Langkilde and Knight, 1998, Rambow and Joshi, 1997*) to the more recent on dependency-based grammars (*Gerdes and Kahane, 2001, Bohnet, 2004*). However, most of the research nowadays is carried out with statistical approaches, as shown by the state-of-the-art results obtained by Bohnet et al. (2012), which take a statistical tree traversal strategy, or Puduppully et al. (2016), who opt for a transition-based algorithm.

Moreover, in the two editions of the Shared Task on Surface Realisation – SR'11 (*Belz et al., 2011*) and SR'18 (*Mille et al., 2018*) – the top performing systems were all statistical: in SR'11, Bohnet et al. (2011) presented a stack of SVM-based decoders, Guo et al. (2011) proposed a n-gram language model, and Stent (2011) also participated with a n-gram language model. In SR'18 Ferreira et al. (2018) used two maximum entropy classifiers and Moses Machine Translation toolkit (*Koehn et al., 2007*) and King and White (2018) combined a neural encoder-decoder for morphological inflection with a passive-aggressive classifier for linearisation. And looking among the participant systems in the SR'18, it is manifested that neural networks have been widely adopted in the NLP community: 6 out of 8 systems have a neural network component.

Syntactic linearisation has also been addressed as word ordering, which assumes that only surface information – i.e. words, as opposed to morpho-syntactic information – is available for linearisation. Hasler et al. (2017) perform a comparison of the recent advances in word ordering, proposing a bag-to-sequence model and comparing it to other language model approaches. Results are compared to those of Schmaltz et al. (2016) and de Gispert et al. (2014) systems.

One of the main challenges of linearisation is to deal with non-projective sentences. Projective sentences are those sentences in which dependencies can be drawn without any crossing edges on ordered words. Let's see an example in order to better understand the issue applying a tree-traversal bottom-to-top linearisation strategy. In Figure 22, we are going to process first *[federal]*. As a leaf in the dependency tree, we order it with respect to its father *[support]*, and it should be placed at the left of *[support]*. We continue with the next step and *[federal support]* should be placed at the left of its parent *[should]*. Now let us see what happens with the other side of *[should]* subtree. As a leaf, *[what]* is ordered with respect to *[achieve]*, which is its parent and should be placed at the left of it. Now, when ordering *[what achieve]* with respect to its parent *[to]*, it should be placed at the right side of *[to]*. But we are going to end up with a wrong linearisation, since the *[what achieve]* is not the correct order.

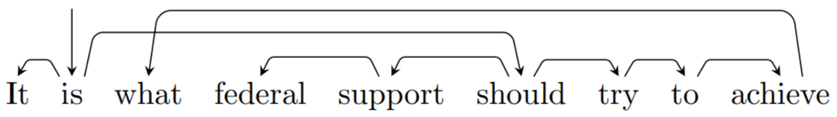


Figure 22: A non-projective example from the Penn TreeBank.

Many of the statistical approaches that work by traversing the tree structure (*Guo et al., 2011; Bohnet et al., 2011*) or rely on transition-based algorithms, like Song et al. (2018), do not deal with non-projective word orders. But non-projectivity has received considerable attention in relation to data-driven parsing and several strategies have been developed to deal with it. For example, Bohnet et al. (2012) trained a model to predict when to apply a reversible pseudo-projectivisation strategy in order to be able to process non-projective sentences. King and White (2018) introduced a specific "discontinuity feature" in order to allow the model to deal with limited non-projectivities, relaxing the requirement of the next predicted word being a children of the previous word (the authors do not present much more detail about it). In contrast, many of the algorithms for neural linearisation do not impose such restrictions (*Hasler et al., 2017; Schmaltz et al., 2016*) but the results are still far from the state of the art.

4.3.2 Fundamentals

Encoder-decoder architecture

The encoder-decoder architecture is a widely used neural network architecture where the input is encoded, condensing the useful information in a fixed size representation, and used by the decoder to produce a result. In NLP, dealing with sequence is quite common, as many of the structures NLP works with are sequences (sentences are sequences of words, documents are sequences of sentences, speech is a sequence of sounds, etc.). Thus, the main component of an NLP neural encoder-decoder system usually is a recurrent neural network (RNN). RNNs are a type of neural network that processes a temporal sequence (see Figure 23). Each time-step receives a new word and information from previous processed words. As this information is passed time-step to time-step, degrades over time (vanishing gradient). Thus, long range dependencies cannot be captured properly by basic RNNs. In order to overcome this degradation problem, improved RNNs like LSTMs (*Sutskever et al., 2014*) or GRUs (*Cho et al., 2014*) were developed. The main difference between basic RNNs and LSTMs/GRUs is that the later allow the error to be back-propagated untouched.

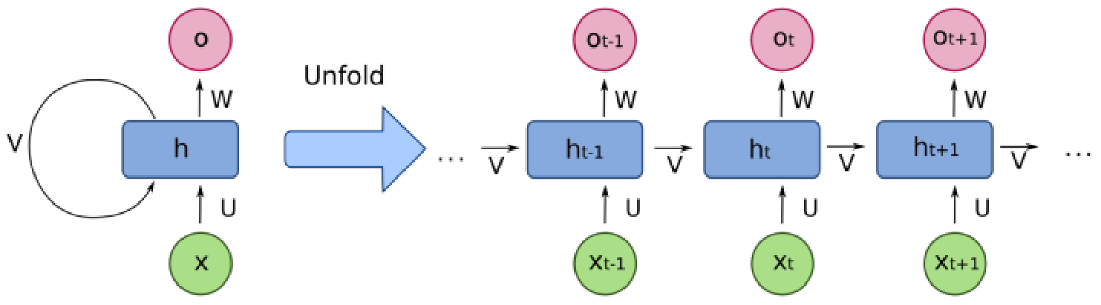


Figure 23: An unfolded recurrent neural network. Each word is processed as one time-step and influences the next words

Attention

An attention mechanism shows the importance of each element of a set/sequence for the prediction of another element. In particular, if such an element belongs to the same set/sequence being evaluated, it is considered to be self-attention mechanism. Some popular attention mechanisms are summarised in Table 7:

Table 7: Summary of the different attention mechanisms (Weng, 2018)

Name	Alignment score function	Citation
Content-base attention	$score(s_t, h_i) = cosine[s_t, h_i]$	(Graves et al., 2014)
Additive	$score(s_t, h_i) = v_a^\top tanh(W_a[s_t; h_i])$	(Bahdanau et al., 2014)
Location-Base	$\alpha_{t,i} = softmax(W_a s_t)$	(Luong et al., 2015)
General	$score(s_t, h_i) = s_t^\top W_a h_i$	(Luong et al., 2015)
Dot-Product	$score(s_t, h_i) = s_t^\top h_i$	(Luong et al., 2015)
Scaled Dot-Product	$score(s_t, h_i) = \frac{s_t^\top h_i}{\sqrt{n}}$	(Vaswani et al., 2017)

Auto-regressive system

An auto-regressive system conditions the prediction on previous predictions, that is, at each step the model consumes the previously generated symbols as additional input for the next prediction. Auto-regressive systems are used in many NLP generation tasks, like summarisation or translation.

The Transformer

The Transformer is an encoder-decoder with attention architecture, introduced by Vaswani et al. (2017). The main contribution of the Transformer is the replacement of the recurrent layers, commonly used in encoder-decoder architectures, with multi-head self-attention and feed-forward layers. The self-attention layer allows the system to access previous words, acting as the memory of a RNN. Specifically, the Transformer uses both forms of attention: self-attention layers in the encoder and in the decoder to emulate the RNN memory, and attention layers over the encoder output in the decoder, so the decoder can access the encoded sequence. The model shows state-of-the-art results and improved training times. Transformer's architecture is depicted in Figure 24.

Transformer-INDIGO

Gu et al (2019) extended Transformer in order to be able to process sequences and sets of words (randomised sequences), capturing the generation order through self-attention. They conducted a set of experiments using different order setups (random, left-to-right, right-to-left, syntactic, etc.) on four different tasks (word order recovery, machine translation, image caption and code generation), reporting competitive performance results. Gu et al. (2019) applied two techniques worth mentioning: adaptation of relative positions in self-attention from (*Shaw et al. 2018*) and insertion operations applying the ideas of Vinyals et al. (2015).

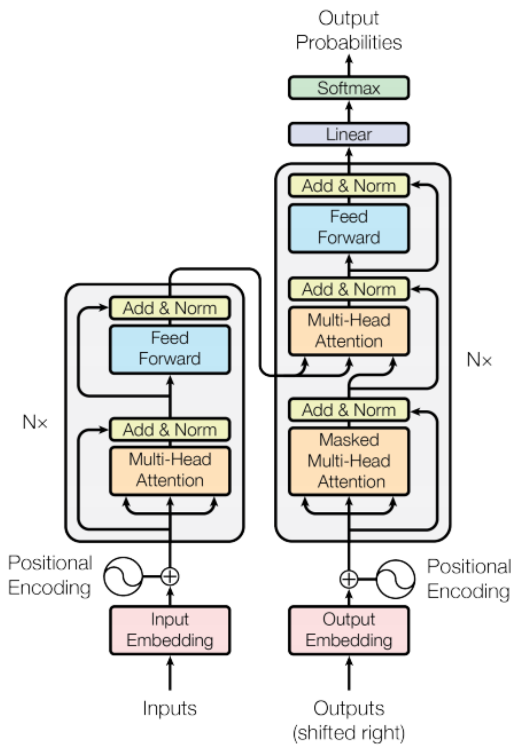


Figure 24: Transformer architecture

Encoded relative positions

Non-recurrent models do not guarantee encoding positions, so adding position information to the input helps the model when dealing with sequences. Specifically, Gu et al. (2019) implemented a variation of the relative position proposed by Shaw et al. (2018) that allows for inserting new positions without the necessity of updating/recomputing previous positions. Their model uses a ternary vector $r_{i,j} \in \{-1, 0, 1\}$ where the j-th element is defined as:

$$r_{ij} = \begin{cases} -1 & z_j \geq z_i \\ 0 & z_j = z_i \\ 1 & z_j \leq z_i \end{cases}$$

where the elements in r_i show the relative positions of word i with respect to all other words in the sequence. This relative representation can be mapped back to absolute position by:

$$z_i = \sum_{j=0}^{n} max(0, r_{ij})$$

where n is the length of the sequence.

Pointer networks

Vinyals et al. (2015) introduced an interesting neural model: a sequence-to-sequence model that computes the conditional probability of a sequence of indexes given a sequence of vectors, by estimating the terms of a probability chain rule:

$$p(\mathcal{C}^{\mathcal{P}}|\mathcal{P};\theta) = \prod_{i=1}^{m(\mathcal{P})} p_\theta(C_i|C_1, ..., C_{i-1}, \mathcal{P};\theta)$$

where P = {P_1, ..., P_n } is a sequence of n vectors and C^P = {P_1, ..., $P_{m(P)}$ } is a sequence of m(P) indices, each between 1 and n. In other words, this technique allows for computing the conditional probability of a permutation with repetition of the input. Also, in order to improve the model, it adds an attention mechanism to both the encoder and the decoder.

4.3.3 Model

We developed an extension of the Transformer (*Vaswani et al., 2017*) based on the ideas of Gu et al. (2019). To the best of our knowledge, this is the first fully attention-based set-to-sequence auto-regressive linearisation model. The previous works addressed the task as a sequence-to-sequence task (Schmaltz et al., 2016, Song et al., 2018) but linearisation is a tree-to-sequence or set-to-sequence task, as pointed out by Hasler et al. (2017). Other works modify the sequence-to-sequence encoder by replacing "the recurrent layer with non-recurrent transformations" and by adding an attention mechanism. Also, they constrain the beam decoder by limiting the output vocabulary to the input bag and by reducing it at each step, thus guaranteeing that the output is always a valid permutation of the input. Similarly, we removed from the Transformer encoder the position encoding, removing from the transformer the position information for each input. Also, we trained TLin unconstrained on the encoded set, but we constrained the decoder on testing by masking the already chosen position in the encoded set, in order to guarantee a permutation of the input as output. We hypothesise that by training on a harder task should make the prediction better, but we need to run some experiments to prove it.

We model the training input and output as follows. Given an ordered sequence w = (w_1, ..., w_T) and its corresponding sequence of correlative indexes i = (i_1, ..., i_T), we randomise the indexes r = (r_1, ..., r_T) and use them as the expected output of the system. By using these randomised indexes, we generate the input set z = (z_1, ..., z_T) where z_t is the word w of the original sequence w at position r_t . For example, given a sentence, e.g. "I like to eat apples .", after tokenisation it would be a sequence of words: w = ["I", "like", "to", "eat", "apples", "."]. Then, we create a correlative index sequence, i = [1, 2, 3, 4, 5, 6] and randomise it, r = [2, 5, 1, 6, 3, 4]. Now, we can create a randomised word sequence by taking each word from the ordered sequence and putting it in the position indicated by r, e. g., the 2nd ordered element is the 5th element of the randomised sequence: ["to", "I", "apples", ".", "like", "eat"]. This way, we have created the expected output r and the randomised input z. By taking the representation from the last layer of the Transformer we predict the next word y_{t+1} modeling the problem similarly to Pointer Networks (*Vinyals et al., 2015*).

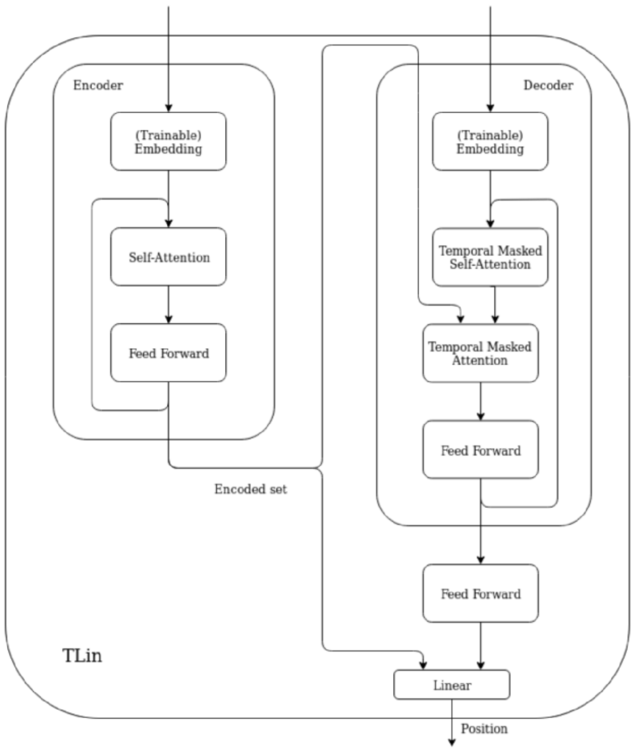


Figure 25: TLin architecture

Original sentence	I	like	to	eat	apples	.
Randomized sentence	to	I	apples	.	like	eat
Softmax	0.06	0.01	0.15	0.05	0.03	0.7

Figure 26: The softmax on the output points to the 6th element of the input list, which is the word "eat".

5 Intrinsic evaluation of the final modules

In this section, we present three different evaluations:

- a qualitative and quantitative evaluation of the FORGe rule-based system on a standard ontological dataset,
- a quantitative evaluation of the FORGe rule-based system on a large-scale generic dataset to compare it to the previous versions of the generator,
- a quantitative evaluation of the advanced linearisation system.

Note that the extrinsic evaluation of the generated reports have not been done since, at this point, there was no unseen test data to generate from.

5.1 Quantitative and qualitative evaluation of FORGe on a standard RDF dataset

The beAWARE use cases required the generation of texts of a specific domain with little variation in the inputs. In order to test the capacities of the beAWARE generator on a wider scale, we evaluated it again on the WebNLG dataset (in D5.2 we reported the best score obtained by FORGe at the WebNLG challenge). The reason for using as reference the WebNLG challenge dataset is that it is the most recent and comprehensive dataset with respect to text generation from RDF data (a standard ontology model as the one used in the beAWARE Knowledge Base), that has been specifically designed to promote data and text variety (*Pérez et al, 2016*). Moreover, it allows the direct comparison with the generators that participated in the challenge. In order to ensure future comparisons with machine learning-based systems in terms of their best obtained performance, only a subset of the original test set has been considered, which included the seen categories, i.e. only inputs with entities that belonged to DBpedia categories that were contained in the training data.

In this section, we detail how we built a new dataset for evaluating the outputs of the generator and describe the results of the automatic and human evaluations.

5.1.1 Selection of triples for evaluation

For evaluation purposes, we compiled a benchmark dataset of 200 inputs, i.e., sets of DBpedia triples, with sizes ranging from 1 to 7 triples, using as reference pool the WebNLG challenge test set. The compilation methodology for our benchmark dataset implements a twofold goal. On one hand, we want to ensure that all properties appearing in the seen categories subset are included. On the other hand, and unlike the WebNLG human evaluation test set, we aim towards a more balanced number of inputs of different sizes. In practice, since the inputs of size 6 and 7 in the original seen categories subset of the WebNLG test set are 24 and 21 respectively, we chose to include them all in the benchmark. Thirty one (31) inputs for each of the remaining input sizes were subsequently added, by iterating over the reference test set and opting for the inclusion of inputs that: i) contain different properties or properties combinations rather than different property values and ii) contain, if none, the least possible number of properties that have been already selected in a previous iteration. In this way, the different input sizes are represented in a better proportion, avoiding possible biases that may be introduced when favoring some input sizes over others (indicatively in the WebNLG seen categories test set, the ratio of inputs of size 6 and 7 over those of smaller sizes ranges from 1 to 6 up to 1 to 9). Inevitably, the small number of inputs of size 6 and 7 initially available, did not leave any space for selection, hence these inputs have a rather high degree of overlapping properties.

5.1.2 Reference sentences

The English reference texts are taken from the WebNLG dataset, for which there could be more than one reference per triple set. For Spanish, one single reference text was produced for each triple set, with natural and grammatical constructions containing all the entities and relations in the triples. The reference texts were written by a native Spanish speaker, having at hand the English references from the WebNLG challenge to serve as a potential model.

5.1.3 Automatic evaluation

The predicted outputs in English and Spanish were compared to the reference sentences in the corresponding language. Three metrics were used: BLEU (*Papineni et al., 2002*), which matches exact words, METEOR (*Banerjee and Lavie, 2005*), which matches also synonyms, and TER (*Snover et al., 2009*), which reflects the amount of edits needed to transform the predicted output into the reference output. Table 8 shows the results of the automatic evaluation on the English and Spanish extensions proposed in Section 4.2 using for each input its corresponding reference text(s). The first two rows show that in terms of automatic metrics, the extended FORGe and the 2017 FORGe have almost the same scores on the English data (which are also very close to the WebNLG scores: 40.88, 0.40, 0.55). In other words, the quality improvements in English are not reflected by these metrics. To compare English and Spanish results, we calculated the scores using one sentence as reference (only one reference per text is available in Spanish). The English scores drop (third row) due to the way the scores are calculated by the individual metrics (BLEU matches n-grams in all candidate references, and METEOR and TER consider the best scoring reference). In the last row of the table, the scores of the Spanish generator look contradictory: the Spanish BLEU is 10 points below the English BLEU with the same number of reference (1), but METEOR is 8 points above, which means that the predicted outputs do not match the exact word forms, but they do match similar words. One reason for the low BLEU score could be the higher morphological variation in Spanish. However, the METEOR score is surprisingly high, actually even higher than the highest METEOR score at WebNLG, obtained by ADAPT and calculated with multiple references (0.44).

Table 8: English and Spanish scores according to BLEU, METEOR and TER, with 1 and All references on the 200-triples test set.

Reference set	BLEU	METEOR	TER
EN (All$_{FORGe-2017}$)	39.87	0.40	0.58
EN (All$_{FORGe-Ext}$)	39.33	0.40	0.58
EN (1$_{FORGe-Ext}$)	29.18	0.38	0.65
ES (1$_{FORGe-Ext}$)	18.68	0.46	0.77

5.1.4 Qualitative analysis of the results

In the 200 outputs of the 2017 generator, 275 errors were detected, compared to 166 in the current one in English (170 in Spanish), whereas 26.5% of the texts were error-free, as opposed to 43.5% now (45.5% in Spanish). In this section, we report on the qualitative evaluation of both English and Spanish outputs, in order to identify the main issues of the grammars in both languages. Outputs are available as supplementary material below.

English errors

The qualitative analysis of the generated English texts showed that the resulting texts are of a higher grammaticality and fluency than the 2017 ones. Below, we discuss the observed remaining errors and their respective causes.

Determiners

Although determiners are handled overall correctly, there are cases that a definite determiner should precede the mentioned NE. In some of these cases, for example in '*acharya institute of technology was established in 2000*', the absence of the determiner can still be considered grammatically acceptable, while in others, for example in '*arabian sea is located to the west of karnataka* and *st. louis is part of kingdom of france*', the determiner's absence is unequivocally erroneous. The missing determiner is traced back to the PredArg template that implements the involved DBpedia property and in particular to the assumptions underlying the semantic types of its respective arguments. For example, properties capturing information about administrative divisions (e.g., canton, state, city, country) and their respective *part-of relations*, as well as cardinal and inter-cardinal directions (e.g., west, southwest) range over entities denoting such subdivisions (i.e., names of cities, countries, regions, etc.), that in the general case do not admit a determiner. As a result, when an argument belongs to the exceptional cases, the generated text misses the determiner.

Definite determiners are missed with the property 'language', when referring to the language of a written work. The reason of this error lies in the discrepancy between the respective PredArg template that was defined based on the premise that the object value of this property is a language name (i.e., English, Italian), hence not admitting a determiner, and the form of the DBpedia language entities that in practice concatenate the language name with the word *language* (cf., *English language*). This type of error is the most frequent, being found about 65 times in the test set and representing about 40% of the total amount of errors (166).

This underlies the need for further normalisation of the DBpedia property values, so that during the PredArg templates instantiation, consistent linguistic features will be ensured for argument values of the same type.

Tense

Errors are observed with respect to the verb tense selection (6% of the errors). More specifically, in some cases the present tense is used instead of the past, as, e.g., in *'Alan Shepard, who graduated from NWC in 1957 with a M.A., is deceased. [...] He is a test pilot.'*
This is a direct consequence of the fact that in the current implementation, tense selection does not take into account the temporal context as defined by the rest of the input triples.

Aggregations

Another type of error relates to the generation of unintuitive, yet still grammatical, constructs when aggregating the contents of more than one triple, when certain properties are involved (11% of the errors). More specifically, when the property 'occupation' is selected to be expressed as a relative clause, it fails to append the occupation information to the referring entity as shown in *Alan Bean, born in wheeler (Texas) on March 15, 1932, is from the United States (test pilot)*. Similar behaviour has been observed with the property 'category'. This is a result of the current implementation of aggregation that takes place in a single step and tries to avoid orphan clauses, by attaching them to the closest reference head. Introducing iterative aggregation steps and incorporating semantic coherence information would mitigate such effects.
A related issue is the way location information is verbalised in the presence of multiple subdivision references (15% of the errors), as for example, in *the 'Acharya Institute of Technology is in Bangalore, Karnataka and India'*, where the three involved location-denoting properties, namely 'city', 'state' and 'country' have been aggregated in a semantics-agnostic manner. Navigating DBpedia and obtaining information about their interrelations would enable more fluent verbalisations. Fluency and meaning accuracy are also impacted when the input triples capture in practice n-ary relations. This is the case with the 'leader' and 'leaderTitle' properties, which in the absence of any semantic pre-processing before the instantiation of the PredArg templates, results in verbalisations such as '*the leaders of Romania are the prime minister of Romania and Klaus Iohannis'*, which does not communicate the fact that Klaus Iohannis is the prime minister.

Subject/Object values

Last, a number of disfluent verbalisations is the direct result of idiosyncrasies in the involved DBpedia properties and/or the respective subject and object values (4% of the errors). There are properties that although meant to capture different types of information are not used consistently, thus impacting the resulting verbalisations. The properties 'mainIngredient(s)' and 'ingredient(s)' are such an example, e.g. in an input about the dish *Ayam Penyet*, which is described as having as main ingredient the fried chicken and as a further ingredient chicken. Some minor errors such as unnatural word ordering (11%) or lexicalisations (8%) were also detected.

Spanish errors

The aforementioned errors listed for English are mostly independent of the language and thus also apply to Spanish, except from the first aggregation error, which does not appear in

Spanish due to a difference in the templates. The determiner error represents 30% of the total number of detected errors (51/170), the location aggregation represents 12%, the values and word choices 7%, the ordering 6%, the verbal tense 5%. However, despite its overall good quality, Spanish has some additional specific issues.

English words

There are some not-translated nouns (*52 minutes*) or phrases (*está dedicado a Ottoman army soldiers killed in the battle of Baku*), which in addition of not being understandable, may produce subsequent morphological errors (21% of the errors).

Morphology

Morphological errors, mainly gender (invisible in English) and number disagreements, are found in the Spanish texts (5% of the errors). For example, in '*Dianne Feinstein es un senador de california'*, (lit. 'Dianne Feinstein is a$_{\mathrm{MASC}}$ senator$_{\mathrm{MASC}}$ of California'), both *a* and *senator* should be feminine, but there is no information that D. Feinstein is a woman in the input.

Complex relative clauses

The main syntactic error is related to the genitive relatives with *cuyo* ('of which'), in particular when the antecedent is a location (5% of the errors). For example, in the sentence '*Alba Iulia, en el cual está el 1 Decembrie 1918 University'*, lit. 'Alba Iulia, in the which is the 1 Decembrie 1918 University', the proper pronoun should be *donde* 'where' instead of *en el cual*. Even when grammatically correct, sentences with these relative clauses tend to lack naturalness.

Other series of errors that produce sub-optimal Spanish constructions include: occasional choice of a relative clause instead of a past participle modifier and various other constructions that lack naturalness (10% of the errors).

5.2 Quantitative evaluations of FORGe on a large-scale generic dataset

We also performed a second quantitative evaluation on a dataset that was already available at the beginning of the project, so as to assess the extent to which FORGe improved during the whole course of the project. As a metric, we use again the BLEU metric, as foreseen in the assessment plan. BLEU is an n-gram-based comparison score obtained by comparing a predicted output, produced by our generator, with the expected one. Single words, bigrams (sequences of two words), trigrams and quadrigrams in both outputs are compared and the similarity between them is calculated. As dataset, we use the whole evaluation section of the dependency version of the Penn Treebank *(Johansson and Nugues 2007*), converted to predicate-argument structures, using the semantic analyser described in (*Mille, et al. 2017*). In order to make the results fully comparable with the results at Month 0, the converted semantic structures are then sent to the FORGe generator, replacing the linearisation module by the same off-the-shelf linearisation tool used in the previous evaluation. As a result, the improvement of the score is due almost exclusively to the Sem-DSynt and DSynt-SSynt grammars, which are the core of the generation pipeline. The last row of Table 6 (Sub-section 4.2.4) shows that the BLEU score increased successively from 31.78 to 35.53 and then to 39.84, a score more than 25% higher than the baseline score obtained by the same generator at the beginning of the project. In terms of error reduction, the generator went from 68.22 (100 minus 31.78) to 60.16 (100 minus 39.84), an error reduction of about 12%, better than the highest expectation of 10% defined in D1.2.

5.3 Evaluation of the neural lineariser

In order to compare the lineariser with existing work, we evaluated the performance of the system on the Penn TreeBank (*Marcus et al., 1993*) sections used in (*Schmaltz et al., 2016*). We then trained the lineariser on the UD datasets presented in Section 3, in order to cover all the beAWARE languages.

For the Penn TreeBank evaluation, as is usual, we use sections 2-22 for training (40k sentences), section 23 for validation and section 24 for testing (2.5k sentences each). Gold-standard dependency trees are extracted using pennconverter, a supersede of Penn2Malt.

We test three different feature sets:

- Forms only, as a regular word-ordering system.
- Forms and Part-of-Speech (PoS), embedded and concatenated.
- Forms, Part-of-Speech (PoS) and dependency label, embedded and concatenated.

For evaluation, we use BLEU (*Papineni et al. 2002*), a precision metric that computes the geometric mean of the n-gram precisions between generated text and reference text and adds a brevity penalty for shorter sentences. We use the BLEU script ScoreBLEU.sh, which uses mteval-v13-ADRIA.pl script, both publicly available in the ZGEN repository.

Before introducing the results, it is important to note that we have contextualised our results with previous work that uses beam search. We are aware that the results with beam size 1 cannot be considered completely equivalent to a beam-less system like ours, since the beam at small sizes could be penalising the results. It remains as future work to introduce a beam search mechanism in the system. In Table 9, the results are compared with the beam setups of size 1 in Schmaltz et al. (2016). Our system outperforms Schmaltz et al. (2016) in any configuration with beam search size 1, even with additional training data (Gigaword). Results suggest that just by using a basic NGram model (NGRAM) is not enough to capture the order of the words of a sentence. Adding a future cost function, i.e. an estimate of the score contribution of the remaining tokens, improves the results but the score achieved is still lower than TLin's. The same happens for Long short-term memory (LSTM), suggesting that using a sequence-to-sequence system is not suitable for the task and adding more resources (Gigaword) just confuses the system.

Table 9: BLEU score comparison on the PTB test set for word-ordering setups in Schmaltz et al. (2016), that is, using only surface information. We show the best results reported for each configuration.

Model	Beam size	Forms	PoS	Dep. Labels	BLEU
GYRO (de Gispert et al., 2014)	512	✓			42.20
NGRAM (Schmaltz et al., 2016)	512	✓			38.61
LSTM (Schmaltz et al., 2016)	512	✓			42.70
n-gram (Hasler et al., 2017)	512	✓			38.90
RNNLM (Hasler et al., 2017)	512	✓			**44.20**
bag2seq (Hasler et al., 2017)	512	✓			37.10
TLin	-	✓	✓	✓	43.45

As can be seen in Table 10, compared to Song et al. (2018), our form-only configuration outperforms all LSTM-based setup with BeamSize = 1. Moreover, when PoS and dependency labels are added, our system outperforms all setups, regardless of the beam size. This, again, suggests that using a sequence-to-sequence strategy for this task is not adequate. It should be noted that our system achieves better results even without dependency structure information.

Table 10: BLEU score comparison on the PTB test set for the syntactic linearisation task. We show the best results reported for each configuration.

System	Beam size	Forms	PoS	Dep. Labels	Dep. Struct.	BLEU
SYNxLSTM (Song et al., 2018)	1	✓	✓	✓	✓	24.91
SYN$_l$xLSTM (Song et al., 2018)	512	✓	✓	✓	✓	37.99
TLin (form-only)	-	✓				28.83
TLin	-	✓	✓	✓		**43.45**

Comparing our PoS+dependency labels system setup to the word-ordering task systems reported in (*Hasler et al., 2017*), our results prove to be competitive, just 0.75 BLEU points below the best result, as can be seen in Table 11. These results suggest that the morpho-syntactic information compensates the absence of beam search.

Table 11: BLEU score comparison on the PTB test set between our system results and word-ordering system results, quoted from (Hasler et al., 2017). We show the best results reported for each configuration.

System	Beam size	Future cost	GW	BLEU
NGRAM (Schmaltz et al., 2016)	1			15.40
NGRAM (Schmaltz et al., 2016)	1	✓		27.10
LSTM (Schmaltz et al., 2016)	1			14.60
LSTM (Schmaltz et al., 2016)	1	✓		25.00
LSTM (Schmaltz et al., 2016)	1	✓	✓	23.80
TLin (form-only)	-			**28.83**

Finally, in order to demonstrate that out lineariser can be used for other languages, in particular the ones addressed in beAWARE, we trained and evaluated TLin on the Universal Dependency annotations presented in this deliverable (Section 3). The results are not comparable to the ones obtained by the shared tasks participants (see Section 6.1), since at both SR'18 and SR'19, the teams were allowed to use external data and language models for training their systems. Table 12 shows the results obtained on the UD dataset for the four languages of beAWARE. It can be noticed that the results on the English dataset are comparable to the ones obtained on the Penn TreeBank (0.38 vs 0.43). The lower score on the

UD dataset is mainly due to the size of the training sets, 3 times bigger in the Penn TreeBank setup. The lower scores in Spanish and Italian (0.24 and 0.27) are likely due to the quality of the data: unlike the Spanish and Italian ones, the English annotations have been fully supervised (see the related comment in Section 3.5). Another reason for the drop may be that word order in Spanish and Italian could be harder to predict without using the syntactic tree information. Finally, the Greek dataset is very small, which is very harmful for neural systems and explains the important drop. The main lesson from this experiment is that the linearizer theoretically works for any language. We also trained it on German in order to see if it works on languages with a relatively free word order, and the BLEU score obtained is 31, which shows that it is the case.

Table 12: BLEU scores of TLin on the UD data

Language		**BLEU**
English	~13K sentences	0.38
Spanish	~14K sentences	0.24
Italian	~13K sentences	0.27
Greek	~2K sentences	0.16

6 Dissemination

In this section, we summarise the two shared tasks that took place using the datasets described in Section 3 and list the papers related to language generation published in the framework of the project.

6.1 The Surface Realisation Shared Tasks

In 2018 and 2019, UPF organised the first and second multilingual Surface Realisation Shared Task (SR'18 and SR'19). The UD data were processed as detailed in the previous sections, and a subset of the data shown above in this section was provided to the participants. Outputs were evaluated according to 3 automatic metrics (BLEU, NIST and Normalised Inverted Edit Distance, DIST) and 2 human evaluations (Meaning similarity and Readability). Human-produced outputs were also evaluated to serve: (i) as reference score in Readability, for which evaluators are asked to rate from 0 to 100, with a slider, the intrinsic quality of sentences, and (ii) as comparison for meaning similarity, for which evaluators were asked to rate from 0 to 100 (with a slider too) if the meanings of two sentences were the same or not, one being a system output and the other one being the human reference.

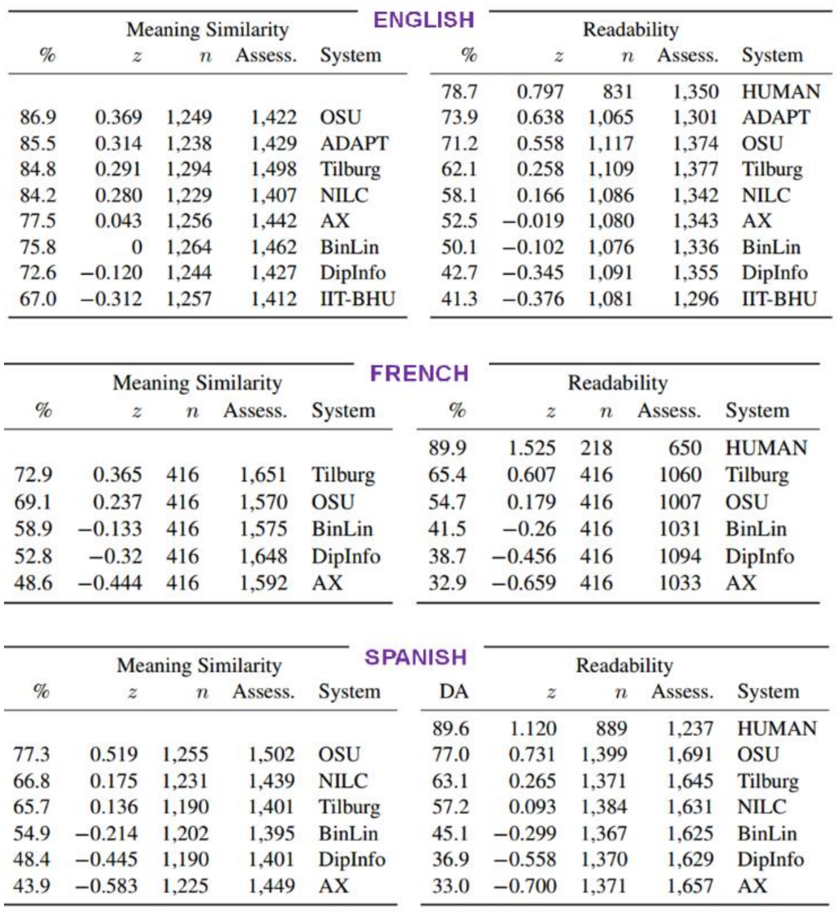
For SR'18, 21 international teams registered to the task and 8 teams submitted outputs. Table 13 shows the results according to the BLEU metric. It can be observed that 2 teams managed to tackle all languages, but only one team addressed the deep task. Table 14: SR'18 human evaluations

shows the results of the human evaluations in three languages. The results show a gap between the quality of human-produced outputs and system outputs in terms of readability, in which the *z* column indicates the main score used for the rankings: 0.16z difference in English and 0.39z difference in Spanish. The full task overview and results have been published in (*Mille et al, 2018*).

Table 13: SR'18 BLEU scores of the participating teams

	Shallow										Deep
	ar	cs	en	es	fi	fr	it	nl	pt	ru	en
AX	4.57	9.75	28.09	10.2	7.95	7.87	16.35	14.21	16.29	15.59	–
BinLin	16.2	**25.05**	29.6	32.15	23.26	20.53	23.55	22.69	24.59	**34.34**	–
OSU	**25.65***	–	66.33	**65.31**	**37.52***	38.24*	–	25.52*	–	–	–
Tilburg	–	–	55.29	49.47	–	**52.03**	**44.46**	**32.28**	**30.82**	–	–
DipInfo	–	–	23.2	26.9	–	23.12	24.61	–	–	–	–
NILC	–	–	50.74	51.58	–	–	–	–	27.12	–	–
ADAPT	–	–	**69.14**	–	–	–	–	–	–	–	**21.67**
IIT-BHU	–	–	8.04	–	–	–	–	–	–	–	–

Table 14: SR’18 human evaluations



ENGLISH

Meaning Similarity					Readability				
%	z	n	Assess.	System	%	z	n	Assess.	System
					78.7	0.797	831	1,350	HUMAN
86.9	0.369	1,249	1,422	OSU	73.9	0.638	1,065	1,301	ADAPT
85.5	0.314	1,238	1,429	ADAPT	71.2	0.558	1,117	1,374	OSU
84.8	0.291	1,294	1,498	Tilburg	62.1	0.258	1,109	1,377	Tilburg
84.2	0.280	1,229	1,407	NILC	58.1	0.166	1,086	1,342	NILC
77.5	0.043	1,256	1,442	AX	52.5	−0.019	1,080	1,343	AX
75.8	0	1,264	1,462	BinLin	50.1	−0.102	1,076	1,336	BinLin
72.6	−0.120	1,244	1,427	DipInfo	42.7	−0.345	1,091	1,355	DipInfo
67.0	−0.312	1,257	1,412	IIT-BHU	41.3	−0.376	1,081	1,296	IIT-BHU

FRENCH

Meaning Similarity					Readability				
%	z	n	Assess.	System	%	z	n	Assess.	System
					89.9	1.525	218	650	HUMAN
72.9	0.365	416	1,651	Tilburg	65.4	0.607	416	1060	Tilburg
69.1	0.237	416	1,570	OSU	54.7	0.179	416	1007	OSU
58.9	−0.133	416	1,575	BinLin	41.5	−0.26	416	1031	BinLin
52.8	−0.32	416	1,648	DipInfo	38.7	−0.456	416	1094	DipInfo
48.6	−0.444	416	1,592	AX	32.9	−0.659	416	1033	AX

SPANISH

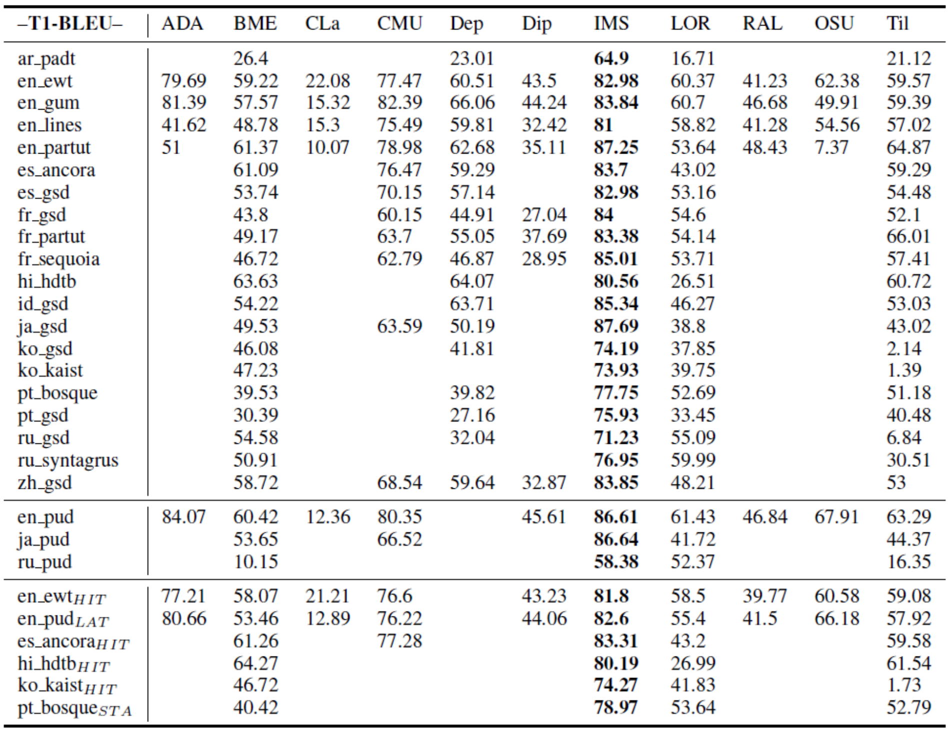
Meaning Similarity					Readability				
%	z	n	Assess.	System	DA	z	n	Assess.	System
					89.6	1.120	889	1,237	HUMAN
77.3	0.519	1,255	1,502	OSU	77.0	0.731	1,399	1,691	OSU
66.8	0.175	1,231	1,439	NILC	63.1	0.265	1,371	1,645	Tilburg
65.7	0.136	1,190	1,401	Tilburg	57.2	0.093	1,384	1,631	NILC
54.9	−0.214	1,202	1,395	BinLin	45.1	−0.299	1,367	1,625	BinLin
48.4	−0.445	1,190	1,401	DipInfo	36.9	−0.558	1,370	1,629	DipInfo
43.9	−0.583	1,225	1,449	AX	33.0	−0.700	1,371	1,657	AX

For SR'19 (*Mille et al., 2019*), 33 international teams (from 17 countries) registered to the task; 14 of these teams submitted outputs, two of which withdrew their submissions at the last minute. New languages and features were introduced (see Section 0). Table 15 shows the results of the 12 teams in the Shallow Track according to the BLEU metric. This time, one can see than 4 teams addressed all 29 datasets (11 languages) and that 4 other teams addressed 3 datasets and 9 languages. In Table 16, the scores for the Deep Track are presented, and again the increase in participation compared to SR’18 is clearly visible: 3 teams addressed the Deep Track, 2 of which for all datasets and languages.

Finally, Table 17: SR’19 human evaluations: Meaning Similarity (left) and Readability (right) shows the results of the human evaluation on English and Spanish (for which system outputs originating from gold-standard and silver-standard data were evaluated). One can notice here that the gaps seen in the SR’18 evaluations between human texts and system outputs are closing: 0.78z in English and 0.65z in Spanish. Among other notable trends, we can observe that there is a notable gap between human assessment (higher) and metric assessment (lower) of deep track systems, in particular for the best deep track systems. The biggest progress was made in SR’19 for Deep track systems: not only did we have multiple Deep Track systems to evaluate (compared to just one in 2018), but the best Deep Track system performed equally well or better than most Shallow Track systems for both Readability and Meaning similarity. Another notable development is the introduction of silver-standard data. Even though the quality of the texts obtained when generating from automatically parsed data is lower than when using gold-standard data, the high scores according to human evaluations suggest that the shallow inputs could be used as pivot representations in text-to-text systems such as paraphrasing, simplification or summarisation applications.

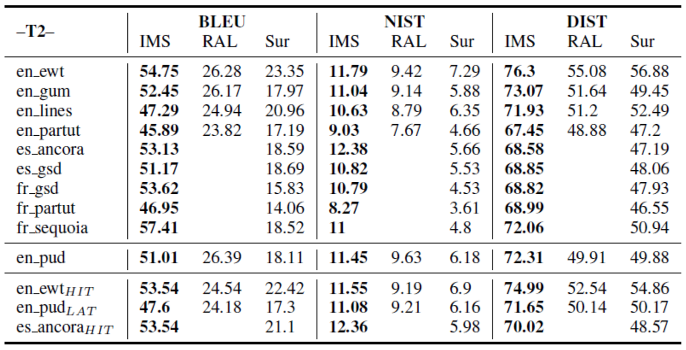
The participation level for such a task is rather high, and due to the success of these first two tasks, a third edition will take place in 2020.

Table 15: SR'19 Shallow Track BLEU scores of the participating teams



–T1-BLEU–	ADA	BME	CLa	CMU	Dep	Dip	IMS	LOR	RAL	OSU	Til
ar_padt		26.4			23.01		**64.9**	16.71			21.12
en_ewt	79.69	59.22	22.08	77.47	60.51	43.5	**82.98**	60.37	41.23	62.38	59.57
en_gum	81.39	57.57	15.32	82.39	66.06	44.24	**83.84**	60.7	46.68	49.91	59.39
en_lines	41.62	48.78	15.3	75.49	59.81	32.42	**81**	58.82	41.28	54.56	57.02
en_partut	51	61.37	10.07	78.98	62.68	35.11	**87.25**	53.64	48.43	7.37	64.87
es_ancora		61.09		76.47	59.29		**83.7**	43.02			59.29
es_gsd		53.74		70.15	57.14		**82.98**	53.16			54.48
fr_gsd		43.8		60.15	44.91	27.04	**84**	54.6			52.1
fr_partut		49.17		63.7	55.05	37.69	**83.38**	54.14			66.01
fr_sequoia		46.72		62.79	46.87	28.95	**85.01**	53.71			57.41
hi_hdtb		63.63			64.07		**80.56**	26.51			60.72
id_gsd		54.22			63.71		**85.34**	46.27			53.03
ja_gsd		49.53		63.59	50.19		**87.69**	38.8			43.02
ko_gsd		46.08			41.81		**74.19**	37.85			2.14
ko_kaist		47.23					**73.93**	39.75			1.39
pt_bosque		39.53			39.82		**77.75**	52.69			51.18
pt_gsd		30.39			27.16		**75.93**	33.45			40.48
ru_gsd		54.58			32.04		**71.23**	55.09			6.84
ru_syntagrus		50.91					**76.95**	59.99			30.51
zh_gsd		58.72		68.54	59.64	32.87	**83.85**	48.21			53
en_pud	84.07	60.42	12.36	80.35		45.61	**86.61**	61.43	46.84	67.91	63.29
ja_pud		53.65		66.52			**86.64**	41.72			44.37
ru_pud		10.15					**58.38**	52.37			16.35
en_ewt$_{HIT}$	77.21	58.07	21.21	76.6		43.23	**81.8**	58.5	39.77	60.58	59.08
en_pud$_{LAT}$	80.66	53.46	12.89	76.22		44.06	**82.6**	55.4	41.5	66.18	57.92
es_ancora$_{HIT}$		61.26		77.28			**83.31**	43.2			59.58
hi_hdtb$_{HIT}$		64.27					**80.19**	26.99			61.54
ko_kaist$_{HIT}$		46.72					**74.27**	41.83			1.73
pt_bosque$_{STA}$		40.42					**78.97**	53.64			52.79

Table 16: SR'19 Deep Track scores of the participating teams



–T2–	BLEU			NIST			DIST		
	IMS	RAL	Sur	IMS	RAL	Sur	IMS	RAL	Sur
en_ewt	**54.75**	26.28	23.35	**11.79**	9.42	7.29	**76.3**	55.08	56.88
en_gum	**52.45**	26.17	17.97	**11.04**	9.14	5.88	**73.07**	51.64	49.45
en_lines	**47.29**	24.94	20.96	**10.63**	8.79	6.35	**71.93**	51.2	52.49
en_partut	**45.89**	23.82	17.19	**9.03**	7.67	4.66	**67.45**	48.88	47.2
es_ancora	**53.13**		18.59	**12.38**		5.66	**68.58**		47.19
es_gsd	**51.17**		18.69	**10.82**		5.53	**68.85**		48.06
fr_gsd	**53.62**		15.83	**10.79**		4.53	**68.82**		47.93
fr_partut	**46.95**		14.06	**8.27**		3.61	**68.99**		46.55
fr_sequoia	**57.41**		18.52	**11**		4.8	**72.06**		50.94
en_pud	**51.01**	26.39	18.11	**11.45**	9.63	6.18	**72.31**	49.91	49.88
en_ewt$_{HIT}$	**53.54**	24.54	22.42	**11.55**	9.19	6.9	**74.99**	52.54	54.86
en_pud$_{LAT}$	**47.6**	24.18	17.3	**11.08**	9.21	6.16	**71.65**	50.14	50.17
es_ancora$_{HIT}$	**53.54**		21.1	**12.36**		5.98	**70.02**		48.57

Table 17: SR'19 human evaluations: Meaning Similarity (left) and Readability (right)

		English			
Rank	Ave.	Ave. z	n	N	System
1	86.6	0.507	695	810	ADAPT-T1
	85.6	0.503	672	768	IMS-T1
3	82.5	0.407	702	812	CMU-T1
4	80.6	0.324	718	826	IMS-T2
	79.7	0.289	711	816	TILBURG-T1
	79.3	0.276	753	859	DEPDIST-T1
	78.4	0.255	720	836	OSU-FB-T1
	77.0	0.222	702	816	LORIA-T1
	73.5	0.164	695	796	BME-UW-T1
10	72.9	0.110	680	795	RALI-T1
	69.5	−0.006	700	811	DIPINFOUNITO-T1
	67.0	−0.040	692	789	SURFERS-T2
	68.3	−0.052	707	808	RALI-T2
14	60.9	−0.216	752	885	CLAC-T1
15	55.3	−0.390	674	775	LB-BASELINE-T1
	53.0	−0.422	733	853	LB-BASELINE-T2

		English			
Rank	Ave.	Ave. z	n	N	System
–	71.1	0.585	824	1,281	HUMAN
1	67.9	0.507	477	564	IMS-T1
	68.2	0.502	482	573	ADAPT-T1
3	61.9	0.313	512	582	IMS-T2
	62.5	0.285	500	575	LORIA-T1
	62.4	0.260	506	589	CMU-T1
	60.8	0.257	497	572	SURFERS-T2
	60.5	0.211	516	591	DEPDIST-T1
	59.2	0.160	516	594	TILBURG-T1
	58.3	0.156	488	554	BME-UW-T1
	57.4	0.121	507	583	OSU-FB-T1
	57.5	0.096	497	569	RALI-T1
12	50.3	−0.117	494	549	RALI-T2
	49.6	−0.195	515	598	DIPINFOUNITO-T1
	48.1	−0.202	524	610	CLAC-T1
15	37.8	−0.594	492	569	LB-Baseline-T2
	36.5	−0.677	468	534	LB-Baseline-T1

		UD Spanish			
Rank	Ave.	Ave. z	n	N	System
1	81.1	0.378	620	716	IMS
2	75.8	0.168	655	753	CMU
3	72.2	0.006	614	708	TILBURG
4	70.6	−0.080	617	704	DEPDIST
	69.1	−0.111	623	705	BME-UW
6	63.2	−0.302	625	706	LORIA

		UD Spanish			
Rank	Ave.	Ave. z	n	N	System
–	89.0	0.582	389	438	HUMAN
1	86.5	0.517	511	584	IMS
2	78.9	0.236	523	601	CMU
3	72.1	−0.009	513	596	BME-UW
	71.5	−0.037	498	562	TILBURG
5	67.7	−0.181	498	562	DEPDIST
6	60.6	−0.458	506	577	LORIA

		Pred. Spanish			
Rank	Ave.	Ave. z	n	N	System
1	82.7	0.394	686	799	IMS
2	78.4	0.272	683	804	CMU
3	70.3	−0.042	688	803	TILBURG
	67.8	−0.105	675	789	BME-UW
5	59.2	−0.422	652	754	LORIA

		Pred. Spanish			
Rank	Ave.	Ave. z	n	N	System
–	89.2	0.736	405	442	HUMAN
1	82.8	0.519	613	713	IMS
2	74.7	0.147	609	686	CMU
3	66.0	−0.103	642	737	TILBURG
	64.7	−0.169	640	734	BME-UW
5	53.8	−0.531	594	670	LORIA

7 Conclusions and Summary

This deliverable reports on the final version of the MRG component, which is responsible for producing multilingual reports in the final prototype of the beAWARE platform. The main improvements performed during the second half of the project are: (i) the significant increase of the coverage of the language-specific generation grammars, in particular for Spanish, (ii) the increase of the coverage and quality of the language-independent grammars, in particular of the semantic aggregation module to package the beAWARE summaries coherently into small texts, (iii) the development of a new state-of-the art multilingual neural linearizer trained on a UD-based dataset developed for this purpose.

The final beAWARE MRG makes use of the FORGe generation system. The advanced linearisation module can be used in case the rule-based system shows limitations in the coverage of its word ordering grammars. The FORGe generator has been largely improved during the course of the project and now achieves a relatively large coverage, in particular for inputs proceeding from structured data repositories such as DBpedia or the beAWARE KBS, in particular in English and Spanish (for Spanish, it is the first known DBpedia generator developed). The coverage of this generator is use case-specific for Italian and Greek, although many rules developed for the other languages also apply to these. Thanks to its flexibility and adaptability, FORGe got the best overall scores at the WebNLG international shared task.

In terms of dissemination, during the course of the project, UPF produced 12 scientific papers and 2 technical reports (shared task system descriptions) related to the datasets and generation techniques. UPF also organised the first two shared tasks on multilingual surface realisation (and two corresponding workshops at ACL and EMNLP) to promote the datasets, with up to 14 submissions for the English data (the second highest participation to this day on a text generation shared task).

8 References

Aparicio, Juan, Mariona Taulé, and Maria Antònia Martí. "AnCora-Verb: A Lexical Resource for the Semantic Annotation of Corpora." In Proceedings of LREC. 2008..

Bahdanau, D., Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473, 2014.

Ballesteros, M., Bernd Bohnet, Simon Mille, and Leo Wanner. 2015. Data-driven deep-syntactic dependency parsing. Natural Language Engineering, pages 1–36.

Banerjee, Satanjeev, and Alon Lavie. "METEOR: An automatic metric for MT evaluation with improved correlation with human judgments." In Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization, pp. 65-72. 2005.

Belz, A., Mike White, Dominic Espinosa, Eric Kow, Deirdre Hogan, and Amanda Stent. The first surface realisation shared task: Overview and evaluation results. In Proceedings of the 13th European Workshop on Natural Language Generation, pages 217–226, Nancy, France, September 2011. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W11-2832.

Bohnet, B. A graph grammar approach to map between dependency trees and topological models. In International Conference on Natural Language Processing, pages 636–645. Springer, 2004.

Bohnet, B. and Wanner, L., (2010). “Open Source Graph Transducer Interpreter and Grammar Development Environment”. In Proceedings of LREC.

Bohnet, B., L. Wanner, S. Mille, and A. Burga. "Broad Coverage Multilingual Deep Sentence Generation with a Stochastic Multi-Level Realizer." Proceedings of the 23rd International Conference on Computational Linguistics. Beijing, 2010.

Bohnet, B., Simon Mille, Benoît Favre, and Leo Wanner. < stumaba>: from deep representation to surface. In Proceedings of the 13th European workshop on natural language generation, pages 232–235, 2011.

Bohnet, B., Anders Björkelund, Jonas Kuhn, Wolfgang Seeker, and Sina Zarrieß. Generating non-projective word order in statistical linearisation . In Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pages 928–939, 2012.

Castro Ferreira, T., Sander Wubben, and Emiel Krahmer. Surface realization shared task 2018 (sr18): The tilburg university approach. In Proceedings of the First Workshop on Multilingual Surface Realisation, pages 35–38, 2018.

Cho, K., Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078, 2014.

de Gispert, A., Marcus Tomalin, and Bill Byrne. Word ordering with phrase-based grammars. In Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, pages 259–268, 2014.

Gardent, Claire, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. "The webnlg challenge: Generating text from rdf data." In Proceedings of the 10th International Conference on Natural Language Generation, pp. 124-133. 2017.

Gerdes, K. and Sylvain Kahane. Word order in german: A formal dependency grammar using a topological hierarchy. In Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics, 2001.

Graves, A., Greg Wayne, and Ivo Danihelka. Neural turing machines. arXiv preprint arXiv:1410.5401, 2014.

Gu, J., Qi Liu, and Kyunghyun Cho. Insertion-based decoding with automatically inferred generation order. arXiv preprint arXiv:1902.01370, 2019.

Guo, Y., Deirdre Hogan, and Josef Van Genabith. Dcu at generation challenges 2011 surface realisation track. In Proceedings of the 13th European workshop on natural language generation, pages 227–229, 2011.

Hasler, E., Felix Stahlberg, Marcus Tomalin, Adri de Gispert, and Bill Byrne. A comparison of neural models for word ordering. arXiv preprint arXiv:1708.01809, 2017.

Johansson, R., and Nugues, P. (2007). “Extended constituent-to-dependency conversion for English”, Proceedings of 16th Nordic Conference of Computational Linguistics, p. 105-112.

Kathol, A. and Carl Pollard. Extraposition via complex domain formation. In 33rd Annual Meeting of the Association for Computational Linguistics, 1995.

King, D. and Michael White. The osu realizer for srst ‘18: Neural sequence-to-sequence inflection and incremental locality-based linearisation . In Proceedings of the First Workshop on Multilingual Surface Realisation, pages 39–48, 2018.

Kingsbury, P., and Palmer, M. (2002). “From Treebank to PropBank”, Proceedings of the Third International Conference on Language Resources and Evaluation, p. 1989-1993.

Koehn, P., Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open source toolkit for statistical machine translation. In Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions, ACL’07, pages 177–180, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1557769.1557821.

Langkilde, I. and Kevin Knight. Generation that exploits corpus-based statistical knowledge. In Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1, pages 704–710. Association for Computational Linguistics,1998.

Liu, Y., Yue Zhang, Wanxiang Che, and Bing Qin. Transition-based syntactic linearisation . In Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 113–122, 2015.

Luong, M.-T., Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. arXiv preprint arXiv:1508.04025, 2015.

Marcus, M., Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. 1993.

Mel'čuk, I. (1988). Dependency syntax: Theory and practice, State University of New York Press.

Meyers, A., Reeves, R., Macleod, C., Szekely, R., Zielinska, V., Young, B., and Grishman, R. (2004). "The NomBank Project: An Interim Report", Proceedings of the Workshop on Frontiers in Corpus Annotation, p. 24-31.

Mille, S., A.Burga, and L. Wanner (2013). AnCora-UPF: A Multi-Level Annotation of Spanish.In Proceedings of the 2nd International Conference on Dependency Linguistics (DepLing).

Mille, S., Wanner, L. (2015). "Towards large-coverage detailed lexical resources for data-to-text generation". In Proceedings of the first workshop on data to text generation, Edinburgh, Scotland, UK.

Mille, S. and S. Dasiopoulou. (2017a). FORGe at WebNLG 2017. Technical Report 17-09, Dept. of Engineering and Information and Communication Technologies, Universitat Pompeu Fabra, Barcelona, Spain. Retrieved from **http://talc1.loria.fr/webnlg/stories/upf-forge_report.pdf**

Mille, S. and S. Dasiopoulou. (2017b). FORGe at E2E 2017. Technical Report 17-12, Dept. of Engineering and Information and Communication Technologies, Universitat Pompeu Fabra, Barcelona, Spain. Retrieved from **http://www.macs.hw.ac.uk/InteractionLab/E2E/final_papers/E2E-FORGe.pdf.**

Mille, S. and L. Wanner. (2017). A demo of FORGe: the Pompeu Fabra Open Rule-based Generator. In Proceedings of the 10th International Conference on Natural Language Generation (INLG), Santiago de Compostela, Spain.

Mille, S., R. Carlini, A. Burga and L. Wanner. (2017). "FORGe at SemEval-2017 Task 9: Deep sentence generation based on a sequence of graph transducers". In Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017), 920-923, Vancouver, Canada.

Mille, S., Anja Belz, Bernd Bohnet, Yvette Graham, Emily Pitler, and Leo Wanner. The first multilingual surface realisation shared task (SR'18): Overview and evaluation results. In Proceedings of the First Workshop on Multilingual Surface Realisation, pages 1–12, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-3601. URL https://www.aclweb.org/anthology/W18-3601.

Mille, S., A. Belz, B. Bohnet, Y. Graham, L. Wanner. (2019). The Second Multilingual Surface Realisation Shared Task (SR'19): Overview and Evaluation Results. In Proceedings of the 2st Workshop on Multilingual Surface Realisation (MSR), 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp.1-17, Hong Kong, China.

Papineni, K., Roukos, S., Ward, T., and Zhu, W. (2002). "Bleu: a Method for Automatic Evaluation of Machine Translation". Proceedings of the 2002 Conference of the Association of Computational Linguistics (ACL), p. 311-318.

Perez-Beltrachini, Laura, Rania Sayed, and Claire Gardent. "Building rdf content for data-to-text generation." In Proceedings of The 26th International Conference on Computational Linguistics (COLING 2016), Osaka, Japan. 2016.

Puduppully, R., Yue Zhang, and Manish Shrivastava. Transition-based syntactic linearisation with lookahead features. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 488–493, 2016.

Rambow, O. and Aravind Joshi. A formal look at dependency grammars and phrase-structure grammars, with special consideration of word-order phenomena. Recent trends in meaning-text theory, 39:167–190, 1997.

Shaw, P., Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. arXiv preprint arXiv:1803.02155, 2018.

Schmaltz, A., Alexander M Rush, and Stuart M Shieber. Word ordering without syntax. arXiv preprint arXiv:1604.08633, 2016.

Schuler, K. K. (2005). VerbNet: A broad-coverage, comprehensive verb lexicon, Ph.D. thesis, University of Pennsylvania.

Shvets, A., S. Mille, L. Wanner. (2018). Sentence Packaging in Text Generation from Semantic Graphs as a Community Detection Problem. In Proceedings of the 11th International Conference on Natural Language Generation, 350-359, Tilburg, The Netherlands.

Snover, Matthew G., Nitin Madnani, Bonnie Dorr, and Richard Schwartz. "TER-Plus: paraphrase, semantic, and alignment enhancements to Translation Edit Rate." Machine Translation 23, no. 2-3 (2009): 117-127.

Song, L., Yue Zhang, and Daniel Gildea. Neural transition-based syntactic linearisation . arXiv preprint arXiv:1810.09609, 2018.

Stent, A.. Att-0: submission to generation challenges 2011 surface realization: shared task. In Proceedings of the 13th European workshop on natural language generation, pages 230–231. Association for Computational Linguistics, 2011.

Sutskever, I., Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In Advances in neural information processing systems, pages 3104–3112, 2014.

Vaswani, A., Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Advances in neural information processing systems, pages 5998–6008, 2017.

Vinyals, O., Meire Fortunato, and Navdeep Jaitly. Pointer networks. In Advances in Neural Information Processing Systems, pages 2692–2700, 2015.

Weng, L. Attention? attention! lilianweng.github.io/lil-log, 2018. URL http://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html.