EMBEDDIA

Cross-Lingual Embeddings for Less-Represented Languages in European News Media

Research and Innovation Action
Call: H2020-ICT-2018-1
Call topic: ICT-29-2018 A multilingual Next generation Internet
Project start: 1 January 2019 Project duration: 36 months

D5.2: Initial news generation technology (T5.1)

Executive summary
This deliverable describes the work conducted within the first 18 months on Task T5.1 of Work Package WP5. We describe how the multilingual language generation technology developed in Task T2.3 (see Deliverable D2.4) can be used to produce news articles in multiple language and in multiple domains. We demonstrate the flexibility of our approach by describing two case studies focusing on different types of news generation tasks. The first produces analytically light news text from a very wide variety of topics covered by the European statistical agency EuroStat, whereas the second provides a setting for very analysis-heavy textual output from a single topic, the very topical COVID-19 virus.

Partner in charge: UH

Project co-funded by the European Commission within Horizon 2020 Dissemination Level		
PU	Public	PU
PP	Restricted to other programme participants (including the Commission Services)	–
RE	Restricted to a group specified by the Consortium (including the Commission Services)	–
CO	Confidential, only for members of the Consortium (including the Commission Services)	–

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825153

Deliverable Information

Document administrative information	
Project acronym:	**EMBEDDIA**
Project number:	**825153**
Deliverable number:	**D5.2**
Deliverable full title:	**Initial news generation technology**
Deliverable short title:	**Initial news generation technology**
Document identifier:	**EMBEDDIA-D52-InitialNewsGenerationTechnology-T51-submitted**
Lead partner short name:	**UH**
Report version:	**submitted**
Report submission date:	**30/06/2020**
Dissemination level:	**PU**
Nature:	**R = Report**
Lead author(s):	**Leo Leppänen (UH)**
Co-author(s):	**Hannu Toivonen (UH)**
Status:	**__ draft, __ final, _x_ submitted**

The EMBEDDIA Consortium partner responsible for this deliverable has addressed all comments received. Changes to this document are detailed in the change log table below.

Change log

Date	Version number	Author/Editor	Summary of changes made
10/05/2020	v0.1	Leo Leppänen (UH)	First draft.
31/05/2020	v1.0	Leo Leppänen (UH)	Ready for internal review.
05/06/2020	v1.1	Salla Salmela (STT), Matej Martinc (JSI)	Internal Review.
11/06/2020	v1.2	Leo Leppänen (UH)	Addressed comments from internal review.
12/06/2020	v1.3	Hannu Toivonen (UH)	WP leader's check.
14/06/2020	v1.5	Leo Leppänen (UH)	Modifications based on WP leader's comments.
14/06/2020	v2.0	Leo Leppänen (UH)	Ready for quality control.
17/06/2020	v2.1	Nada Lavrač (JSI)	Quality control.
26/06/2020	v2.2	Leo Leppänen (UH)	Addressed comments from quality control.
28/06/2020	final	Leo Leppänen (UH)	Ready for submission.
30/06/2020	submitted	Tina Anžič (JSI)	Report submitted.

Table of Contents

1 Introduction

The main objective of the EMBEDDIA project is to develop methods and tools for effective exploration, generation and exploitation of online content across languages thereby building the foundations for the multilingual next generation internet, for the benefit of European citizens and industry using less-represented European languages. One facet of this effort is EMBEDDIA work package 5 (WP5), which is concerned with Natural Language Generation (NLG). In order to support journalists and media companies in efficiently reaching as many demographics as possible, the objective of WP5 is to design and develop news automation systems that are transferable across languages, transferable across domains, and are transparent in the NLG process. In particular, with the output of NLG that is dynamic, has narrative structures, and uses figurative and colourful language.

More specifically, WP5 aims to develop a flexible, accurate, and transparent NLG system architecture that can be transferred to new domains and languages with minimal human effort; develop tools for creation of dynamically evolving content, incorporating narrative structure and user knowledge; and develop tools for creation of figurative language and headlines. The work package consists of three primary tasks:

- Task T5.1, Multilingual text generation from structured data, will adapt NLG technology for the requirements of news generation. The task will develop mechanisms for (i) determining what is interesting or important in the given data and deciding what to report, and for (ii) rendering that information in an accurate manner (iii) in multiple languages.
- Task T5.2, Multilingual storytelling and dynamic content generation, will develop a novel method for automatically organising news articles based on the domain of the article.
- Task T5.3, Creative language use for multilingual news and headline generation, will make the generated texts more varied and colourful by generating creative expressions, especially in headlines. We will find similar terms and metaphors by finding analogous terms in different contexts using context-dependent embeddings. A special focus will be on cross-cultural metaphors.

In this deliverable, we report on the development relating to task T5.1 within the first 18 months of the project, with the chief focus being on how the abstract architecture developed in Task 2.3 and described in Deliverable D2.4 – “Multilingual language generation technology” can be applied to news automation.

In the following sections of this deliverable, we will first define and introduce automated journalism, the general process of automatically generating news text (Section 2). We then briefly describe the state of the art in Natural Language Generation, the technical process by which said news texts are generated (Section 3). Based on this background, we then conduct an analysis of how various approaches to NLG relate to the requirements imposed by the news domain (Section 4). Following this background, we describe very briefly the general natural language generation architecture employed within the EMBEDDIA project (Section 5). This is followed by the main contribution of this report: the descriptions of two instantiations of said architecture in the form of two case studies, highlighting how the architecture can be applied to different types of news generation contexts (Sections 6 and 7). In Section 8 we describe our plans for evaluating the success of our approach. Finally, we conclude this work by a brief analysis of the most important results and an enumeration of future research directions in Section 10.

The description of Automated Journalism (Section 2) borrows heavily from Leppänen, Tuulonen, and Sirén-Heikel (in press). The section on Natural Language Generation (Section 3) is abridged from a more indepth overview present in Deliverable D2.4. We repeat only the most points most salient for the following requirement analysis (Section 4) for the benefit of those readers not familiar with D2.4. Readers already familiar with Deliverable D2.4 should be able to largely skip Section 3.

2 Automated journalism

The use of automation to produce news text is increasingly employed and of interest to news organizations around the world (Sirén-Heikel, Leppänen, Lindén, & Bäck, 2019). This automation is often referred to, in the literature, as "automated journalism" (Caswell & Dörr, 2018; Dörr, 2016; Graefe, 2016) but terms such as "news automation" (Sirén-Heikel et al., 2019), "robot journalism" (Montal & Reich, 2017), "machine journalism" (Lindén et al., 2019) etc. are also used. Despite a wide agreement that automated text generation methods are both useful and being used in the real world, the literature shows a relatively wide spread of variation in how the concept is precisely defined.

This lack of a shared definition indicates that the term "automated journalism" shares a definitional difficulty like that observed with the term "Artificial Intelligence." As with AI, it is trivial to point out things that are *not* automated journalism (e.g. spelling correction in a text editor), but difficult to define precise boundaries or a positive definition. Trying to distinguish "automated journalism" from concepts such as "computer-assisted reporting" and "data journalism" is sufficiently non-trivial to have warranted research to identify ways the uses of the concepts differ and overlap (Coddington, 2015).

A commonly used definition is provided by Graefe (2016, p. 14), who defines automated journalism as "the process of using software or algorithms to automatically generate news stories without human intervention." While we agree that the systems falling within this definition indeed are examples of automated journalism, we believe it to be needlessly restrictive. In a way, Graefe defines automated journalism by virtue of what it is replacing: it is automated journalism if it is journalism and it is replacing a human. This definition ignores the possibility of using automation to *enhance* and *aid* the human, rather than simply replacing them.

Others, such as Dörr (2016) and Caswell and Dörr (2018), approach automated journalism through the technology employed. In their view automated journalism is about the employment of Natural Language Generation methods for producing news text. Natural language generation is a "subfield of artificial intelligence and computational linguistics that is concerned with the construction of computer systems that can produce understandable text in English or other human languages from some underlying non-linguistic representation of information" (originally Reiter and Dale, 1997, p. 57; also used by, e.g., Gatt and Krahmer, 2018, p. 68). As such, Caswell and Dörr's definition explicitly excludes, for example, systems that produce summaries of news content written by other humans.

The decision of Caswell and Dörr to limit automated journalism to systems that function from non-linguistic inputs, and thus exclude e.g. automated translation, summarization and associated technologies, provides a clear delineation between news automation and the general use of technology in the newsroom. At the same time, it might also be overly restrictive: we believe that there can be automated systems working from linguistic inputs that would fall within an intuitive definition of "automated journalism." It is notable the definition used by Graefe (2016) in turn *allows* for these kinds of technologies to be included within the umbrella of the term.

In this deliverable, we use the term automated journalism along the lines of Caswell and Dörr. In our view, automated journalism is the act of automatically producing a complete or near-complete news text from some underlying (structural, non-linguistic) data. We include the qualifier "near-complete" as a conscious acknowledgement of the view that a human can – and perhaps should – be included in the journalistic process of publishing. In practice, this means that our definition includes systems that produce story "blanks", which already contain the main beats of the story but need further human editing before they are ready for audiences. Such a definition is supported by our discussions with the media partners taking part in EMBEDDIA, who agree that the type of automation described above (i.e. producing text for humans to refine) would be of interest to them.

As noted above, the general task conducted in automated journalism is known in the technical literature as natural language generation, or NLG for short. More specifically, the systems being developed in WP5 are performing 'data-to-text NLG', where 'data' refers to structured data. That is, the systems are not designed to ingest unstructured data, such as raw text. In the next section we first give an overview of (data-to-text) NLG in general as well as how it can be viewed as a series of subtasks, followed by a description of how the subtasks common to data-to-text NLG can be completed.

3 Natural language generation

As the relevant background on Natural Language Generation is also detailed in Deliverable D2.4, due concurrently with this deliverable, we only provide here an abridged version which provides a brief recap of the most salient points. Please see Deliverable D2.4 for a further details.

The academic literature on NLG explores a wide variety of approaches to NLG, applied to a variety of slightly different NLG problems. As noted above, our workdescribed in this deliverable focuses on a variant of NLG called data-to-text generation. Even within this one area, a wide range of NLG approaches have been suggested. While no consensus has been reached on 'the right way' has been reached, there is relatively broad agreement that the larger NLG task can be divided into subtasks. It is notable that this divisions is often only conceptual: many neural NLG systems train a single, unified neural network to conduct all the subtasks in a single, global, architecture (Seminally e.g. Wen et al., 2015, presenting the neural encoder-decoder approach to NLG). At the same time, later neural systems have also shown that reintroducing division into subprocessess can be beneficial (Puduppully, Dong, & Lapata, 2019; Ferreira, van der Lee, van Miltenburg, & Krahmer, 2019). Despite this, no consensus has been reached on what, precisely, the subtasks are. On a high level, there is some agreement that NLG involves three large subtasks: content determination, document and sentence planning, surface realization (Reiter & Dale, 2000). At the same time, later reviews have identified more fine-grained divisions, with e.g. Gatt and Krahmer (2018) using a six-way split and Reiter (2007) describing a four-way split.

The recent review of the NLG field by Gatt and Krahmer (2018) identified that, in addition to whether NLG is achieved in a modular fashion – with subcomponents each dedicated to some variant of the subtasks described above – or in a unified manner – for example with a neural encoder-decoder architecture –, the various systems can be characterized in terms of whether they employ manually programmed rules or approaches based on machine learning.[1] Here, it is important to highlight that these two questions of architecture and method are considered orthogonal: rule-based systems can be global and unified, and neural approaches can be modular.

Rule-based approaches use handcrafted rules, often derived from either corpus analysis and expert consultations (Gkatzia, 2016) to achieve the NLG task. As a consequence of their robustness, they provide a relatively high *quality floor* and allow for transparent and explainable processing. Furthermore, they allow for manual correction of any mistakes in the processing. It is likely due to these properties that especially newsrooms seem to prefer rule-based systems (See Sirén-Heikel et al., 2019, where all interviewed newsrooms used template-based NLG, a subcategory of rule-based systems). While commercial NLG providers are notoriously secretive of their systems' internals, the few available public source code repositories (E.g. Yleisradio, 2018), private conversations with stakeholders and the lack of any explicit advertisement of neural methods within the industry indicates that rule-based methods are, indeed, dominant outside of academia. At the same time, rule-based systems are costly to produce and require co-operation between domain experts and NLG/NLP experts to establish the system. It is especially difficult and costly to add variation into the generated texts. This is unfortunate, given the observation that the reusability of the commercial rule-based systems seems to also be very low, at least insofar as it is seen by the customers of NLG providers (Linden, 2017).

Academic work on the other type of NLG systems has in the recent half-decade been extremely focused on using neural networks. Compared to the rule-based NLG approaches, the neural approaches have various upsides and downsides. On the positive side, they seem to have a much higher *quality ceiling*. In other words, especially in complex domains, they can reach very good results and produce highly fluent

[1] While Gatt and Krahmer (2018) also identify a third category of 'planning-based approaches,' which we skip here in the interest of keeping this survey of the NLG background suitable concise. We do not believe the planning-based approaches (that to our understanding are rarer than the others) affects our analysis in a meaningful fashion.

text. They are also faster to build than the rule-based approaches, and the same model architecture can be often reused in another text domain, albeit with the models retrained. At the same time, on the negative side, the need for training data can be debilitating in some domains and languages (Gkatzia, 2016). The need for training data also effectively limits the automation to mimicking what humans have been doing, where as, for example in journalism, there is significant industry interest in applying NLG to produce texts that humans have traditionally been unable to produce. Even when the training data is technically available, the expected output text is often not aligned with the input data, and thus cannot be used directly for the development of an NLG system (Belz & Kow, 2010). Furthermore, at least in limited domains, even recent neural end-to-end approaches failed to conclusively outperform rule-based approaches (Dušek, Novikova, & Rieser, 2018).

Empirical evidence also suggests neural NLG – even the recent multi-stage variants (E.g. Puduppully et al., 2019) – suffer from a type of overfitting called '*hallucination*', where the system produces output that is not based on the underlying data (Reiter, 2018a; Nie, Yao, Wang, Pan, & Lin, 2019; Dušek, Howcroft, & Rieser, 2019). This is potentially fatal for the methods' applicability to real-world news generation. Finally, neural approaches are inherently opaque to inspection, which has significant consequences for trustworthiness and error correcting. As the systems are opaque, their *quality floors* are unknown, and must often be assumed to be relatively low. This is in stark contrast to the rule-based systems which have lower quality ceilings, but relatively high quality floors. With respect to error correction, neural systems do not allow for targeted system modifications to correct for a specific mistake the system is making. Rather, the system can only be trained further – or completely retrained – with more data. This, together with the unknown quality floor, means that it is very hard to know whether the general system performance has improved or decreased after some problem is 'fixed' by retraining. Especially this last problem is complicated by the observation that the most commonly used automated metrics for estimating the output quality of an NLG system correlate imperfectly with human judges (Reiter & Belz, 2009; Liu et al., 2016; Dušek et al., 2018; Gatt & Krahmer, 2018).

Our interpretation of the current state-of-the-art in NLG is that trainable end-to-end approaches are mainly ready for real-world use in situations where there is ample pre-existing training data of high quality and either the produced texts are very short (i.e. scenarios similar to the E2E Challenge described by Dušek et al. (2018)) or if even major mistakes in individual pieces of output are not problematic, but concurrently high linguistic variation in the output is needed.

4 Requirements analysis

In the case of news automation, several important requirements for the technology can be identified. Previously, we have identified the requirements for transparency; accuracy; modifiability and transferability; fluency; data availability; and topicality (Leppänen, Munezero, Granroth-Wilding, & Toivonen, 2017).

The requirement from *transparency* stems from a media-specific need for accountability (McBride & Rosenstiel, 2013; Stark & Diakopoulos, 2016) and strive for objectivity (Mindich, 2000), as well as the increased public scrutiny of the fairness of algorithmic decision making in general (e.g. Angwin, Larson, Mattu, & Kirchner, 2016). This need is also driven by a more concrete need to protect newsrooms from legal consequences. For example, in Finland the editor-in-chief of a newsroom is always accountable for everything that is published. It is very difficult – if not impossible – for the editor to ethically take responsibility of a complete black box without assigning a human to check the texts produced by the automation. Such a system of checks, however, distinctly diminishes the potential of automation.

The requirement for *accuracy*, we hope, is self-evident. A system producing untruthful content both exposes the newsroom employing the system to legal liability, and also erodes the readership's trust in the news product. As such, the system be known to be accurate in its output. In fact, automation has been classically used in news domains that are prototypically objective and have the highest accuracy requirements, such as weather reports (Goldberg, Driedger, & Kittredge, 1994) and financial news coverage (2014).[2] The requirement for accuracy also interplays with the aforementioned requirement for transparency: establishing trust in the system's accuracy requires either very extensive testing or a transparent system.

The system must also be *modifiable and transferable*. NLG systems are costly to set up. Unless the same underlying technology can be reused in multiple domains, the newsrooms will have very few domains wherein the potential profits and savings offered by the use of automation can justify a from-scratch effort to produce an automated system. As noted by an anonymous interviewee of Linden (2017): "It is difficult to create generic solutions, we have to start from scratch for each new case, and relatively little is reusable."

In terms of *fluency*, the level required is dependent on how the system is intended to be used. In cases where the output is directed at human journalists who can polish the text and add additional analytical details, the requirement is significantly lower than in cases where the text is delivered directly to the news consumer. In both cases, however, the fluency must be high enough to ensure that the information content of the text is understood correctly by the readers.

The *availability of data* is less important from an academic perspective, but is crucial from a business perspective. Any developed systems must be able to produce enough content to cover the cost of their creation. As such, the system needs to produce content from datasets where multiple stories are available. It is notable, however, that this content needs not be produced in a single go. Rather, both a constant drip of news stories (for example, a constantly updating coverage of the present state of the COVID-19 situation) and an occasional bulk production (for example, generating a multitude of stories every time new data on the economy is released) are viable options. This last factors, however, also indicate a need for *topicality* in the data: however cheap, producing automatic summaries of decades old NHL ice hockey games is unlikely to be a sound business move.

Analysing these requirements further, we can note that while the two latter requirements apply more to the *data* of the system, the first four are more technological questions. As a summary, it is desirable to produce a transparent and accurate system that is modifiable, transferable and of some minimal fluency level. Of these four requirements, the first two speak highly in favour of rule-based or hybrid systems over fully neural approaches: as discussed above, especially end-to-end neural systems are effectively black boxes and also tend to suffer from hallucination in even the most restricted domains.

On its face, the third requirement for modifiability and transferability seems to favour systems based on machine learning, such as end-to-end neural NLG systems, over rule-based systems. It is, however, important in our view to distinguish here between the *theoretical* and *practical* transferability of the systems. In practice, neural end-to-end NLG system are only transferable at the cost of huge amounts of training data in the form of aligned input-output pairs of structured data and human-written text. It is our understanding that aligned training data is exceedingly rare in the real world outside of some specific domains such as sports, finance and weather. Furthermore, such data cannot, by definition, exist for new domains and text types where the costly human news production is not profitable, but where automation could be useful. While some systems have been presented for unsupervised learning of an NLG model (E.g. Schmitt, Sharifzadeh, Tresp, & Schütze, 2019), they make several very significant assumptions regarding the structure of the input data, effectively requiring a partially lexicalized document plan as input. For example, Schmitt et al. (2019) generate English language outputs using as input knowledge graphs defined using English language labels and relations, thus giving almost all the necessary lexical information 'for free.' This severely limits the *practical* transferability of neural approaches. These considerations apply whether the neural systems are global and unified or fully neural pipelines.

Simultaneously, the requirement for modifiability and transferability indicate that global rule-based systems are not optimal, as transferability is maximized when large parts of the system can be reused when transferring to a new domain. As such, we construe this requirement as pointing towards modular rule-based approaches or modular hybrid approaches that incorporates some neural components that are not dependent on aligned training data, but can for example be trained solely on textual corpora.

[2] This might be a consequence of most pre-existing automation approaches being unsuitable for more complex journalism, see Stray (2019).

Overall, our analysis of the requirements indicates that the needs of WP5 are best served by an NLG approach that is modular and at least partially rule-based, but also incorporates some neural processing, thus resulting in a hybrid approach. We believe that this interpretation is given more credibility by the fact that the little available evidence points to real-world newsrooms preferring rule-based systems over completely neural systems (Sirén-Heikel et al., 2019).

5 The EMBEDDIA news generation technology

The EMBEDDIA news generation approach (refer to Deliverable D2.4 for a detailed description of the approach) is based on a pipeline of components with dedicated responsibilities. This structure allows for the individual components to be modified and replaced without affecting the rest of the pipeline. As the domain and language specific aspects of the pipeline are largely segregated (with the parts specific to both domain and languages further delegated to specific subcomponents), the system at large can be transferred to new domains and languages much more easily than applications based on a non-modular approach to NLG.

The architecture consists of eight primary stages: message generation, document planning, template selection, lexicalization, aggregation, named entity resolution, morphological realization and surface realization. The modularity of the architecture allows us to employ both rule-based modules and neural (or otherwise machine learning based) modules in the same architecture. As the rule-based and machine learning approaches have complementing upsides and downsides, this hybrid approach allows us to always pick the option that fits best the requirements for any stage of the pipeline.

While the architecture itself is flexible, any concrete implementation will need to make various design decisions that affect which types of flexibility are prioritised over others. To demonstrate how the architecture is useful for various types of flexibility foci, we will next describe two concrete implementations. While both implementations are multilingual (i.e. produce textual output in more than one language), they differ in what types of domain flexibility they prioritise.

The system developed within the EuroStat case study (Section 6) is flexible with regard to addition and removal of data sources, but consequently is based on fundamental assumption about the format in which the data is provided and thus requires a degree of data preprocessing. Additionally, it is tailored towards a certain *type* of data and would not be suitable to, for example, reporting about a sports event.

The system developed within the COVID-19 case study (see Section 7) is very flexible in terms of incorporating new data analysis methods. In other words, the system can be enhanced easily to impute and derive more and more information about the underlying input data. As a consequence, however, the system is very tied to the underlying data and changing the domain would require potentially significant retooling.

Despite these different modularity foci, the two systems share the majority of their code base in the form of an *NLG core* module, which contains shared domain and language agnostic code (e.g. code for parsing the templating language described in Section 6.4 and code for passing information between the pipeline components), interface definitions and abstract classes that can be extended to concrete language and/or domain-dependent implementations (e.g. morphological realization described in Section 6.7).

6 Modularity of datasets: The EuroStat case study

The Eurostat case study is a system implementation that produced textual news content from various data tables provided by EuroStat. EuroStat, also known as the European Statistical Office, provides various data collected by and about the EU member countries – and some other non-EU countries – in a unified way through their online service portal. While the underlying data is collected by the member countries, EuroStat collates the data into single digital location, combining information from the various sources and ensuring that the data provided is comparable across the various sources.

As noted above, the EuroStat case study, an initial implementation of which is described herein, seeks to provide a system that is easily adaptible to any data table published by the EuroStat. The system identifies the most pertinent – newsworthy – information from the data tables it is provided with and reports them in natural language. As a consequence of this adaptability to the various data tables, the system does not provide for any significant imputation or derivation of additional information beyond what is provided directly by EuroStat. The goal is to provide a starting point – a story 'blank' of relatively raw text material – which the journalist can refine to a larger story or focus down to a more detailed report. While unlikely to be directly interesting to majority audiences, the raw textual outputs might also be useful for niche expert audiences outside of the newsroom whose work directly interfaces with said data.

An overview of the EuroStat system is shown in Figure 1. As in Figure **??**, the central column of Figure 1 contains the main components of the architecture. These components, in conducting their processing, refer to various resources described in the right-most column of boxes. The resources are either dependent on the domain alone (light blue), dependent on the output language alone (red), or dependent on both (hatched). The system is interfaced with via the *API and Control* element, which provides an HTTP API for communication and also initiates the generation pipeline.

We will next describe the components of the pipeline in the order they appear in the system.

6.1 Data preprocessing

As described above, the EuroStat case study assumes a uniform data input format. The *Data Preprocessor*, together with data table specific processing provided by the individual *Table Preprocessors* ensures this. Notably, this preprocessing is not conducted online – i.e. it is not triggered by a user requesting a text from the system – but is conducted once new data is provided to the system, and the results are stored and reused.

First, the data provided by EuroStat is flattened into a two-dimensional table of data by a per-data table *Table Preprocessor*. The flattened table always contains a selection of metadata columns labeled `where`, which defines a location such as 'Finland'; `where_type`, which defines what type of a location the value in `where` is, for example 'country' in case of Finland; `timestamp`, which tells what time the data is related to, `timestamp_type`, which provides an interpretation for the `timestamp` column. In terms of database design, the values in these columns together form a unique key for each row of data. The rest of the columns contain the various datapoints pertaining to each location and timestamp. The only restriction imposed on these columns is that the names of the columns are defined as colon-separated sequences of labels, so that the column names form an implicit hierarchy.

For example, a column might be labeled `cphi:hicp2015:cp-hi02:rt01`, where the individual labels are `cphi` (the name of the underlying data table), `hicp2015` (stating the values are about the harmonized consumer price index, which uses the year 2015 as a starting point), `cp-hi02` (values are about the 'Alcoholic beverages and tobacco' category) and `rt01` (value is the growth rate on previous month ($\frac{t}{t-1}$)). A column labeled `cphi:hicp2015:cp-hi02:rt12` would provide the yearly growth rate of the same variable. This use of hierarchical column names is important later, during the document planning process as it allows for automated determination of how related various variables are.

Second, the shared *Data Preprocessor* attempts to impute some limited additional data by comparing the values of the individual countries to the relevant values for the EU as a whole, as well as the US, provided that data is available. Similarly, columns are added to describe the *ranks* of the various values in relation to other nations at the same instance in time.

Third, the shared Data Preprocessor computes an estimate of how statistically unexpected each value is and stores the information in the data table. This computation is at the present based on the interquartile

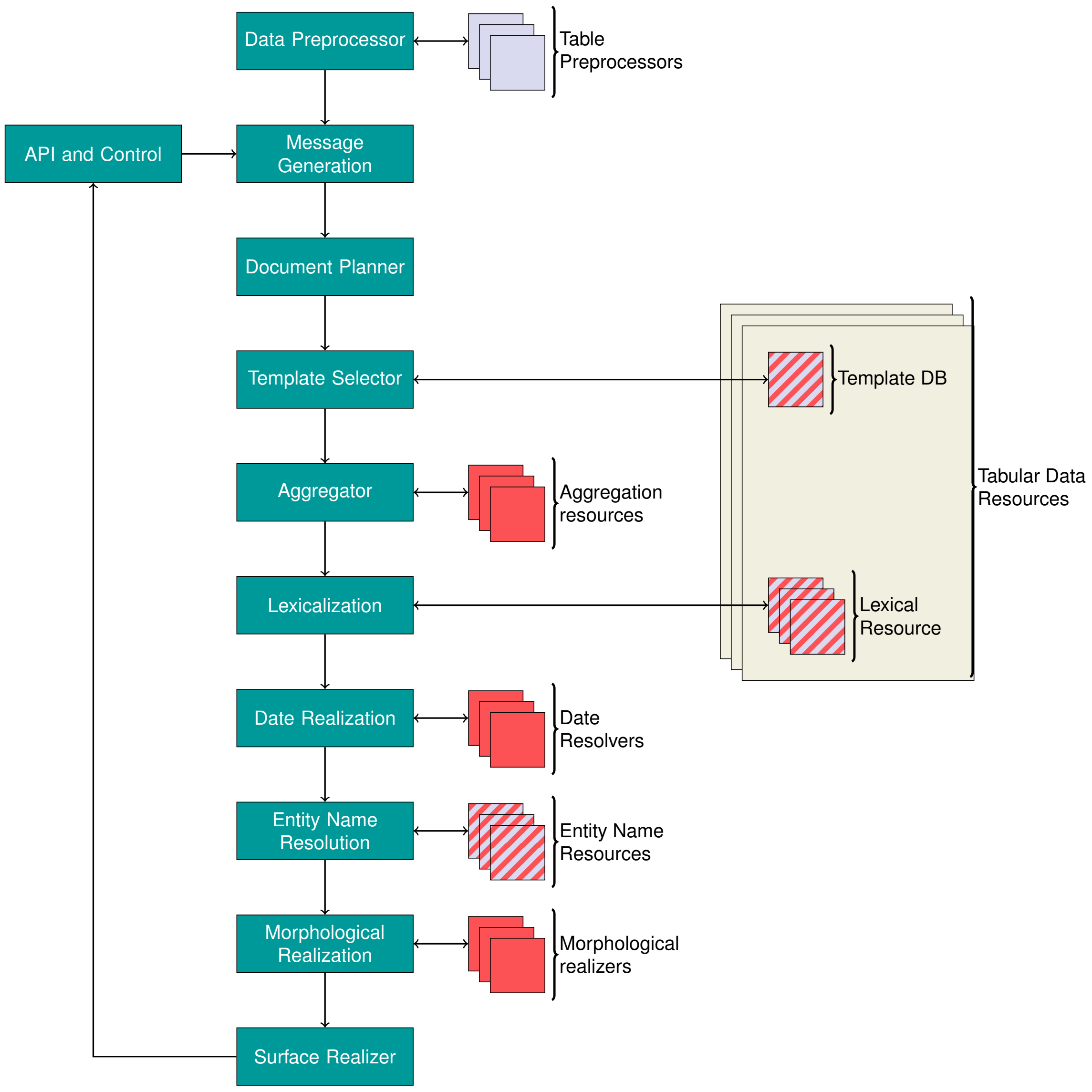

Figure 1: A high-level view of the EuroStat case study. On the two right-most columns, red boxes indicate modular components that are specific to a language, where as light-blue boxes indicate modular components that are specific to a domain. Hatched boxes are specific to both. Template databases and lexical resoures are grouped into a set of Tabular Data Resources, each specific to a certain data table and language.

Table 1: The fields of a Fact data structure in the EuroStat case study. The hypothetical Fact states that in the fifth month of 2020, in Finland, the consumer price index, using the year 2015 as the start of the index, of alcoholic beverages and tobacco changed by 0.01 with respect to the value of the index during the previous month.

Field	Description	Example value
`where`	What location the fact relates to	Finland
`where_type`	What the type of the location is	country
`timestamp`	The time (or time range) the fact relates to	2020M05
`timestamp_type`	The type of the timestamp	month
`value`	A (usually) numeric value	0.01
`value_type`	A descriptor defining how the numeric value should be intepreted	cphi:hicp2015:cp-hi02:rt01
`outlierness`	A statistical estimate of the outlierness of the Fact's value field's contents, as obtained using the IQR method.	1

ranges of the data, as described by us previously (Leppänen, Munezero, Sirén-Heikel, Granroth-Wilding, & Toivonen, 2017). This statistical estimate of unexpectedness forms the bases for estimating how *newsworthy* each piece of information in the data table is.

6.2 Message generation

When a user requests a new text be generated, the system initiates the main NLG pipeline. The user provides the system with some input parameters, most notably the language they want the system to output as well as the names of the data tables the system should process.

The first step taken is *Message Generation*, where the user-defined data tables (preprocessed as described above) are read and transformed into immutable atomic units of information called *Facts*. The format of a Fact data structure, as used in the EuroStat case study, are shown in Table 1. Note how the fields of the Fact relate to the columns of the data tables, described above: each Fact describes one row's one non-metadata column's value, with the value of the non-metadata column in the `value` field and the name of the column in the `value_type` field. The rest of the fields are filled from the metadata columns. This provides for a simple way to express arbitrary columns as Facts.

As noted, these Fact data structures are immutable and can not be modified after they have been created. This ensures that the data underlying the story is not accidentally modified during the generation process. At the same time, the ability to attach mutable information to these Facts is useful, if not necessary. As such, each Fact is encapsulated within a mutable *Message* data structure. In the case of the present implementation of the system, the Messages also have fields for a Template (discussed later), as well as a computational estimate of *newsworthiness*.

At the present, the newsworthiness of a message is the outlierness value of the underlying fact weighted to account for the *recency* of the information so that newer information is more newsworthy. In the future, this value could also be influenced by user-specific weights that describe the degrees to which the individual users believe different factors are newsworthy (See Leppänen, Munezero, Sirén-Heikel, et al., 2017, discussing automated newsworthiness detection in context of election results).

The output of the message generation step is an unordered set of Facts – each embedded in a Message – with estimates of the facts' newsworthiness. Together, these Facts and Messages represent the sum totality of the information that *could* be included in the final text document.

6.3 Document planning

Working from this set of atomic pieces of information, the next step in the processing pipeline is to determine which pieces of information to actually include in the document. This decision is done jointly with the planning of the text *structure*: the naive approach of simply greedily selecting the most newsworthy pieces of information fails to account for situations where some piece of information that is of medium newsworthiness becomes highly relevant after some more newsworthy piece of information is selected for inclusion.

The details of the processing conducted in this stage are, however, skipped here for the reason that the processing is explained in great detail in Deliverable D5.3 - 'Initial dynamic new generation technology'.

The output of this process is a tree-structure called '*Document Plan*', which details what pieces of information (Messages) are to be included in the document and in which order they are to be presented. The Document Plan also contains structures that approximate the eventual division of the text into paragraphs.

6.4 Template selection

Following the production of document planning, the first language-specific step of the process is undertaken. The system inspects the various Messages in the Document Plan and associates each with a phrase skeleton that provides basic information as to how the Fact contained within the Message is to be realized into the target natural language.

These skeletal phrases take the form of *Templates*, which are defined in databases that are specific to both the language being generated and the dataset being processed. They are defined in a custom templating language and consist of canned text and short phrases interspersed by *Slots* that refer to fields of the Facts each Template is to be associated with. An example of the templating language is provided in Figure 2.

```
en: {location} had the {value, ord} highest {value_type} [in {time}]
| value_type = cphi:.*:rank.*
```

Figure 2: Example of an English language template used in the EuroStat case study. The template might be eventually realized as, for example, "*Finland had the 4th highest monthly growth rate of the harmonized consumer price index for the category 'food and non-alcoholic beverages' in June 2019*."

The Templates consist of two types of content. The first are the template strings, identifiable by the language code ('`en:`' in Figure 2). The Slots are defined using brackets { and } and can also contain additional metadata. In Figure 2, the slot '`{value, ord}`' will eventually be replaced by a number from the underlying Fact's `value` field, displayed as an ordinal. Content marked by square brackets [and] is optional, and the single template string is realized to Template variants both with and without the marked content.

In addition, each Template is associated by *rules*, which define where the template can be applied. In the case of Figure 2, the template is applicable to any Fact where the `value_type` field matches the regular expressions `cphi:.*:rank.*`. In other words, the example is applicable to any fact derived from a table called '`cphi`' and discussing a rank of a value. Notably, multiple template lines can be associated with a single group of rules, thus allowing for addition of variation without repetition of the associated rules. The templating language also allows for multiple languages' templates to be defined together with shared rules. In this instantiation of the architecture, however, we have elected to separate the templates by language.

Template candidates are selected from the database first by filtering for the correct language and then by matching the rules. This usually results in several possible templates that contain references to varying optional fields. For example in Figure 2, we would obtain both a variant that includes a reference to time and a variant that does not refer to time.

To obey the Gricean maxim of quantity, the template selector moves through the Document Plan, keeping track of the *present context*. In cases where the Message to which a template needs to be assigned shares values of contextual fields, for example `timestamp`, with the previous Message, the selector attempts to identify a template without reference to said field. Only if no such template is available, does it select a template that conveys 'redundant' information. In other words, the template selector attempts to avoid repetition such as two subsequent sentences both ending in '*in June 2019*.'

During the template selection process, each Message in the Document Plan is associated with a Template, resulting in a modified Document Plan.

6.5 Aggregation

The next step, aggregation, inspects the document plan and attempts to identify scenarios wherein two subsequent templates can be condensed. For example, the sentences 'Unemployment grew by 5 percentage points in Finland' and 'Unemployment grew by 6 percentage points in Sweden', contains a shared prefix 'Unemployment grew by', and can thus be condensed to the sentences 'Unemployment grew by 5 percentage points in Finland and 6 percentage points in Sweden.'

As demonstrated above, this task is presently done simply by considering prefixes of sentences and reducing those where possible, with the added limitation that aggregation cannot remove the actual values. That is, the system is not allowed to aggregate the sentence pair 'The number of unemployed people grew by 125 in Finland. The number of unemployed people grew by 125 in Sweden.' to 'The number of unemployed people grew by 125 in Finland and Sweden'. The example demonstrates the danger of such aggregations, where it's no longer clear from the aggregated sentence whether there are a total of 125 more unemployed in Finland and Sweden *combined*, or whether both countries saw an increase of 125, meaning the total increase is 250 people.

A consequence of this relatively simple aggregation approach is that only a trivial amount of language-specific resources are needed. At the present, the only required information is what is each language's corresponding expression to the English language 'and.'

To accommodate these aggregated sentences, the Facts of the two combined sentences are both associated with a single Message that also associates with the combined Template. As such, beyond this stage the Fact-Message relation is no longer one-to-one, as it has been thus far, but potentially many-to-one. At the present, the system is only allowed to aggregate at most two Facts into a single Message to limit the creation of extremely long and complex sentences. This is because we have, so far, been unable to identify a suitable heuristic to distinguish whether further aggregation improves or reduces the overall text fluency. In the future, we intend to investigate how text readability metrics such as those described in Deliverable D2.4 could be used to intelligently decide on a suitable degree of aggregation.

6.6 Lexicalization, date realization and entity name resolution

Following aggregation, several steps are taken to realize the slots in the templates as natural language expressions. The first stage, *lexicalization*, inspects the slots and matches the content referenced by them (i.e. the contents of the Facts) with language and domain-specific rules called *Lexical resources*.

These resource contain information that tells the Lexicalization module that, for example, column names of the form `cphi:X:Y`, when included in the text via Slots referring to the contents of the `value_type` fields, are to be realized in English as 'the X for the category Y', where X and Y are parts of the column name. Further resources then state that values for X, such as '`hicp2015`' ought to be realized in English as the phrase ‘harmonized consumer price index’. Together, these rules can be thought of as forming Context-sensitive Grammar that is applied to the various templates until the template stabilizes. In other words, the rules can realize an individual token into set of tokens where some of the new tokens are then matched and processed by further rules. For instance, the token ‘`cphi:hicp2015:cp-hi06`’ would first be realized into six tokens as ‘*the* `hicp2015` *for the category* `cp-hi06`’ after which further rules would realize `hicp2015` as ‘*harmonized consumer price index*’ and `cp-hi06` as ‘*‘health’*’, resulting the stabilized realization ‘*the harmonized consumer price index for the category ‘health’*’.

This stage also includes some domain-independent processing. For example, the various numbers referred to by the document plan are rounded to some suitable amount of decimal places to avoid numbers with an excessive number of decimal places.

Next, a separate component called *Date Realizer* realizes any references to the `timestamp` fields of the Facts, realizing them to some suitable language-specific expression. Language-specific information is retrieved from separate *Date resolvers* which inform the process, for example, about the fact that the first month of the year is known as ‘January’ in English but ‘Tammikuu’ in Finnish. In the future, this stage will be incorporated into the next stage, Entity Name Resolution and is at the present separate only as an artefact of the system’s evolution over time.

Finally, the *Entity Name Resolution* stage realizes references to domain entities, such as countries. This step, too, consults language specific information provided in *Entity Name Resources*. These contain information such as that Austria is called ‘Itävalta’ in Finnish. While Figure 1 denotes these resources as dependent on both the language and the domain, we note that the dependence of these resources on individual domains is very light and the same resources (such as country names) are needed almost always irrespective of the individual data tables being realized. At the same time, it is possible that future domains might discuss more domain-specific entities and as such would need more unique resources.

A notable property of the entity name resolution process is that in addition to realizing the various references to domain entities, the process also considers the *form* of the references. For example, in a situation where the previous sentence already referred to Finland by name, the follow-up sentence uses a pronoun-like referential construct such as ‘the country’ if a reference is needed. This processing becomes increasingly important if the system at any point in the future includes people as the domain entities, as it would then need to decide e.g. whether to refer to a person using their full name, a surname only, some position they hold (‘the prime minister’) or a pronoun.

6.7 Morphological realization

After all the lexicalization-related stages, the templates in the document plan are all expressed solely as linguistic constructs. These constructs, however, are not necessarily yet in their correct grammatical forms. For example, in the case of Finnish, a previous stage might simply decide that a slot ought to be realized as the lemma ‘Suomi’ (engl. ‘Finland’) in the inessive noun case (‘Suomessa’, corresponding roughly to the English preposition ‘in’). The actual realization is left to the following stage, *morphological realization*.

In this stage, the system consults language-specific morphological realizers to correctly inflect the words. In the above example, the lemma ‘Suomi’ (‘Finland’) is realized as ‘Suomessa’ (‘in Finland’). This separation of morphological realization from the lexicalization is not strictly necessary in case of languages such as English, where morphological complexity is fairly low, as the amount of variants needed of each domain entity’s name is very low.

However, for languages such as Finnish with high morphological realization, it is not feasible to predefine all the inflected forms. This is demonstrated well by Karlsson (1996) who provides 2253 inflections of a single basic Finnish noun, ‘kauppa’ (eng. ‘shop’). While the listing contains word forms that are unlikely to occur in real use of the language, it is nevertheless infeasible to manually curate all the possible word forms in a dictionary. Similarly, the Finnish morphology does not allow for simply appending static

In Austria, the cost of rehabilitative care was 2594.06 million euro in 2017. It was 2474.92 million euro in 2016 and 2348.41 million euro in 2015. [..] The cost of inpatient rehabilitative care was 1577.18 million purchasing power standards (PPS) in 2017. The cost of inpatient rehabilitative care was 0.48 percent of the gross domestic product. It was 0.47 percent of the gross domestic product in 2016 and 0.46 percent of the gross domestic product in 2015.

Figure 3: Example of output from the EuroStat case study system, discussing the Austrian spending on health care. We emphasize that both the structure and contents of the story are decided in a completely dynamic manner, as described in Deliverable D5.3.

suffixes, as morphological processes necessitate some variance in the suffixes based on the lemma being inflected, and in some cases changes in the root (lemma) itself.

In case of Finnish, we employ the UralicNLP library by Hämäläinen (2019) which provides facilities for both analysis and generation of arbitrary morphological forms of Finnish language words. The library also supports English, and as such it is used for English language morphological realization as well.

6.8 Surface realization

Following morphological realization, the document plan is a complete description of a natural language text. What remains for the surface realizer to do is to flatten the tree-structure into a linear text and add orthographic details such as sentence-first capitalization and sentence-final periods. While these orthographic details are in reality language-specific – consider, for example, Spanish with its sentence-first inverted exclamation points and the various quotation methods used by different languages – it is presently handled in a language agnostic manner. If this needs to change in the future, the process will be modified to extract the language-specific decision making into separate components as with some of the preceding components.

It is also necessary to decide at this point the format in which the text is output from the system. For this last purpose, the surface realization module allows for inclusion of HTML markup elements to designate paragraphs, or the formatting of the text as a JSON formatted list of paragraphs. This processing is independent of both the language and the domain.

Finally, the readied text – in the user-specified format – is returned to the user and the generation process is complete. An abridged example of system output is provided in Figure 3. Note that the document structure is determined purely using computational means taking into account how statistically surprising the various mentioned facts are. Note also the aggregation in the second sentence, with 'the cost of rehabilitative care' being referred to as 'it'.

6.9 Modularity of datasets

As observed above, the system needs a limited amount of information that is specific to both the domain (dataset) and the language being generated. These are provided in the shapes of the template databases, the lexical resources and to some degree the entity name resources. In addition, domain-specific but language-independent resources are needed in the form of the table preprocessors.

To make the system modular with regard to various datasets, the resources specific to both a language and domain are grouped together into *tabular data resources*, shown as the tan boxes in Figure 1. For example, resources specific to the consumer price index dataset (`cphi`) and the English language are provided in resource named '`cphi_english_resource`'. These resources are defined as Python 3 files and contain all the relevant information needed information to produce text about the consumer price index dataset in English. The only additional dataset specific information needed is the language-independent table preprocessor for that specific input data table.

This modularity of datasets allows the system to be easily extended to support a new dataset, as only two modular components need to be produced. Once a dataset is supported in one language, addition of support for another language – already supported by the system – only requires the tabular data resource to be translated to the new file.

At the same time, addition of support for a new language the first time is still non-trivial, as resources need to be provided for aggregation, date and entity name realization as well as morphological realization.

7 Modularity of analysis: The COVID-19 case study

The COVID-19 case study system produces textual news content about the COVID-19 situation based on frequently updating online data. The intended use case of the system is as a provider of constantly updating stories, each tailored to some specific geographic area. The system uses data described in Dong, Du, and Gardner (2020), available for academic research purposes. As the COVID-19 case study is built on the same technology as the EuroStat case study above, the significant bulk of the processing is identical. As such, we will limit our discussion to the differences between the two case studies.

As already mentioned above, the most significant difference between the two systems is the goal they are seeking to achieve: whereas the EuroStat case study investigates how a single larger NLG system can easily accommodate multiple domains (with some assumptions of a shared format etc.), the COVID-19 system's focus is on allowing for extensive analysis of a single known data source.

In other words, the COVID-19 case study system's goal is to be modular with regard to *analysis*, enabling system operators to easily add and remove self-contained, modular, analysis components. As a consequence of focusing on a single domain and input data source, the COVID-19 system architecture (See Figure 4) lacks the preprocessing step of the EuroStat system, but instead incorporates a series of *Message Parsers*. These Message Parsers each tell the system how to extract a certain type of information from the input data. In their simplest, they might simply extract the values of certain fields in the input data, such as the number of identified patients in a certain country at a specific date, to their own messages. They can, however, also conduct arbitrarily complex processing and as such can produce Messages describing information that is not provided directly in the input data as a specific value, but is rather a result of some computation. For example, a specific Message Parser might compute the rate of change in the number of infected by looking at a combination of pre-existing values.

As with the EuroStat case study, the system also needs templates and other linguistic resources to express these messages. Whereas in the EuroStat case study we described these as 'domain specific,' in the case of the COVID-19 case study, it's more accurate to describe them as *analysis specific*, as each analysis needs their own templates and lexical resources. Naturally, these resources are also language specific. These analysis specific elements (message parsers, templates, lexical resources) are grouped into *analysis resources*, which are analogous to the tabular data resources used by the EuroStat case study. Adding a new analysis to the system thus only warrants the addition of a new analysis resource module. These modules can be added and removed from the system without affecting the rest of the system, thus providing significant modularity of analysis. Support for new languages is added by modifying each analysis resource's template database (as noted above in Section 6.4, multiple languages' templates can be defined using shared rules) and adding the required lexical resources. In addition, when adding support for a language the first time, resources also need to be provided for aggregation, entity name resolution and morphological realization, as shown in Figure 4.

As the COVID-19 case study system is focused on a single domain only, the principally domain-dependent entity name resolution process becomes static across the various analyses conducted and reported by the system. As the domain is static and the entity name resolution is not specific to the individual

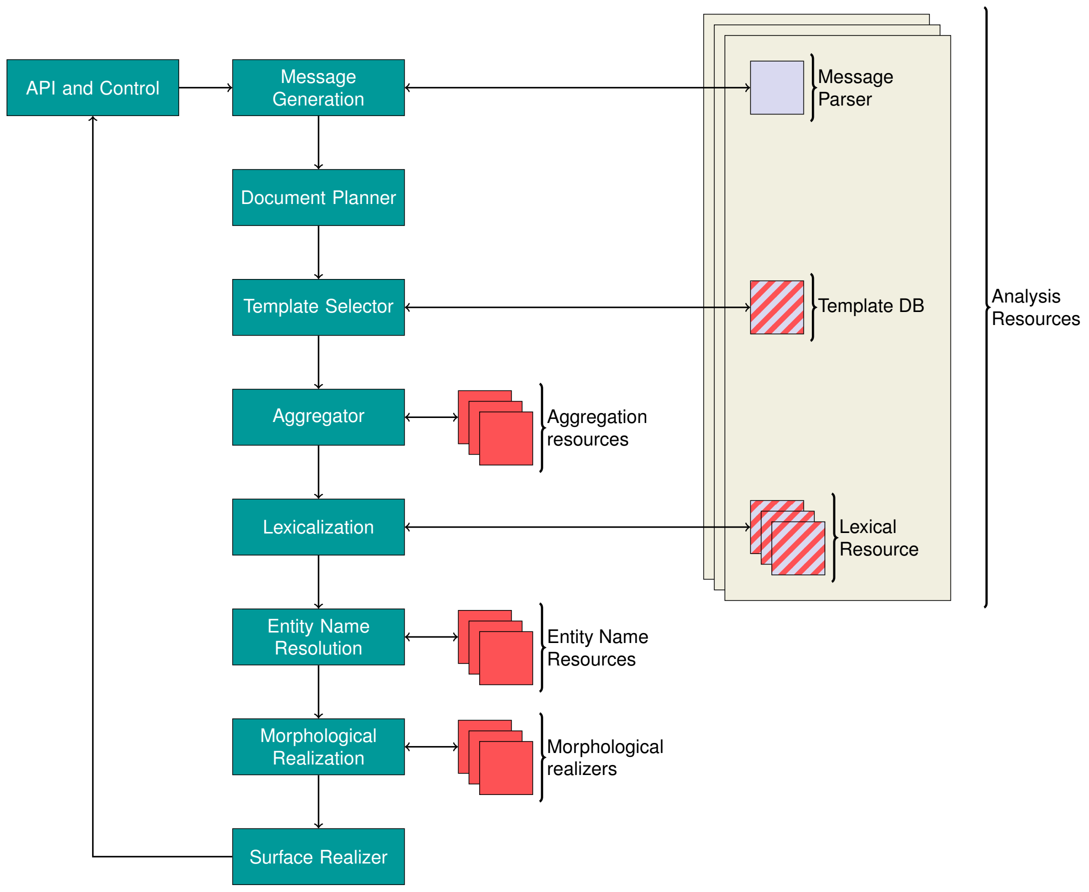

Figure 4: A high-level view of the COVID-19 case study. On the two right-most columns, red boxes indicate modular components that are specific to a language, whereas light-blue boxes indicate modular components that are specific to an analysis. Hatched boxes are specific to both. A message parser, a template DB (containing templates for multiple languages) and lexical resources are defined in modular Analysis Resources, each containing the required resources to conduct a specific type of analysis and to realize the results in any relevant language.

analyses, it is effectively only specific to language. This is reflected in the architecture shown in Figure 4.

To highlight the analytical modularity, the COVID-19 input data is very limited, forcing the system to conduct data enrichment for even very simple messages. In fact, the only messages immediately available from the input data are the per-date running totals of identified COVID-19 cases, deaths and recoveries per country. All other data is computed from these raw values. An example of system output is provided in Figure 5.

6826 total COVID-19 cases in Finland by yesterday

In Finland, there have been 6826 confirmed cases by yesterday. The number of confirmed cases increased by 0.7 percentage and 50 cases between the day before yesterday and yesterday. The number of confirmed cases increased by 3.9 percentage and 258 cases between the day before yesterday last week and yesterday.

There have been 316 confirmed deaths by yesterday. The number of deaths increased by 0.6 percentage and 2 cases between the day before yesterday and yesterday. The number of deaths increased by 3.3 percentage and 10 between the day before yesterday last week and yesterday.

Figure 5: Example of output from the COVID-19 case study system.

8 Evaluation methods

As work on both systems described above still continues, we have not yet conducted a formal evaluation of their performance. At the same time, even our initial successess in implementing the systems and producing meaningful textual outputs points towards at least a reasonably successful NLG approach. Our plans for evaluating the NLG systems are detailed in Deliverable D5.1 – “Datasets, benchmarks and evaluation metrics for multilingual text generation” and as such we only provide a brief outline of our plans.

In Deliverable D5.1, we identified that automated evaluation based on gold standard textual outputs is not feasible for our scenario. For a meaningful automated evaluation, we would need a corpus consisting of aligned pairs of input data and the expected system output. Such datasets are not available and producing them ourselves would be prohibitively costly, given the complexity of the domain. In addition, we would need to produce these corpora in multiple languages for the evaluation to meaningful. In addition, recent works have highlighted severe problems with various *de facto* standard evaluation metrics used in NLG, such as BLEU (Papineni, Roukos, Ward, & Zhu, 2002). For example, Reiter (2018b) states that ‘evidence [..] does not support using BLEU [..] for scientific hypothesis testing.’ Other metrics, too, have been criticized as ‘uninterpretable’ and ‘[uncorrelated] with human judgements’ (van der Lee, Gatt, van Miltenburg, Wubben, & Krahmer, 2019).

Due to these concerns regarding the validity of the various available evaluation metrics, we do not view our lack of suitable gold standard outputs as a significant problem: even if such a corpus existed, human evaluations would be preferable over automated metrics. As such, we intend to evaluate the systems by virtue of an intrinsic human evaluation, where online judges are shown documents produced by the system (rather, various versions of the system) and are asked to evaluate them along properties such as whether they are pleasant to read, contain meaningful information, etc. Such evaluations are easy to conduct for large languages, such as English, but becomes increasingly difficult to conduct in reasonable time for smaller languages such as Finnish, Estonian and Croatian. For cases where insufficient online judges are available on various online platforms, we hope to leverage qualitative results obtained from domain experts employed by project media partners.

The overall success of the NLG approach is then to be judged as a combination of the quantitative results obtained from online judges for larger languages, the qualitative results obtained from domain experts in smaller languages as well as qualitative analysis of how the described software and architecture fit to the requirements identified in Section 4. For this last component, we can already provide some initial, tentative analysis.

The systems are built using a transparent approach which enables inspection of the decisions and reasoning made by the system through extensive logging. The present implementations are also accurate, in that the processing does not at the present allow the system to hallucinate any content. At the same time, the nature of the implementation ensures that if any problems were detected, the errors can be corrected with targeted modifications. The two case studies together also demonstrate that the system is transferable: in fact, the COVID-19 system was built by taking the EuroStat system and transferring it to fit the new domain. The high degree of code reuse between these two systems further reinforces our analysis that even these initial versions of the produced systems are very transferable. Similarly, the two case studies together show the modifiability of the systems, with the COVID-19 system taking the EuroStat system and modifying it so that the system is structured around modules defined in terms of individual analyses rather than datasets. Clearly the same basic approach is easily modifiable to suit various different processing needs.

While especially the COVID-19 case study produces relatively fluent language (see Figure 5), the fluency of the output has significant room for improvement. At the same time, even these initial versions' output should be suitable for highlighting potentially interesting aspects of possible very large datasets. Together with the above analysis on the other requirements, we believe the systems together demonstrate our news generation approach's suitability for producing news in various languages and in various domains using a reusable and transferable core system. This result also serves as validation of the architectural work conducted in Deliverable D2.4.

9 Associated outputs

The work described in this deliverable has resulted in the following resources:

Description	URL	Availability
EuroStat news generation system (source code)	`https://github.com/EMBEDDIA/eurostat-nlg`	To become public
COVID-19 news generation system (source code)	`https://github.com/EMBEDDIA/covid-nlg`	To become public

We note that the source code repositories listed above are under active development, as work on both case studies is expected to continue to the end of the project. In addition, we are preparing scientific publications on these systems. As such, the source code repositories listed above are not yet public. The source code will be made public later with a suitable open source license, as the code bases stabilizes and the scientific publications being prepared become public.

10 Conclusions and further work

In this Deliverable, we have described two case studies exemplifying our approach for adapting the multilingual language generation technology developed in Task T2.3 to automated journalism. Together, the case studies highlight the adaptability of the underlying technology by providing system modularity with regard to different datasets (EuroStat case study, Section 6) and analyses within a dataset (COVID-19 case study, Section 7). The two case studies share a significant bulk of their code bases, both being built on a shared *NLG core* that defines common interfaces and contains domain and language agnostic code and default implementations. In addition, the two systems share major subsystem *implementations*, for example the document planners (see Deliverable D5.3) and morphological realizers. Our initial analysis of the systems' properties indicates that the approach selected is very promising.

The systems have been implemented as Dockerized (Merkel, 2014) standalone software and provide clearly documented restful Application Programming Interfaces. This has enables their easy integration with WP6 within the EMBEDDIA project, as well as easy future integration into other systems as necessary. This approach also enables the provision of the systems as public online services later on if such an action is deemed useful. The systems will be made initially accessible through the EMBEDDIA platform developed in WP6 for testing and evaluation by project media partners. Commercial use of the systems in their present state is partially limited by the licensing terms of the used data.[3]

In terms of future work, these work-in-progress systems provide us with a rule-based baseline to which further hybrid components can be added and in the context of which the hybrid approaches can be evaluated. Based on our experience in building these systems, we believe that the most potential for hybrid approaches is found in lexicalization (see Deliverable D2.4 for an initial trial of potential solution), document planning (see Deliverable D5.3 for description of ongoing work), newsworthiness determination and aggregation. In the future, we intend to continue our work by investigating how these components would best benefit from the use of technologies based on the contextual and cross-lingual word embeddings developed in Work Package WP1. We will also elicit feedback from domain experts in the project media partners' newsrooms to identify what aspects of the systems are the most important to focus on and conduct a more rigorous evaluation as described in Section 8.

References

Angwin, J., Larson, J., Mattu, S., & Kirchner, L. (2016). Machine bias. *ProPublica, May*, *23*, 2016.

Belz, A., & Kow, E. (2010). Extracting parallel fragments from comparable corpora for data-to-text generation. In *Proceedings of the 6th international natural language generation conference* (pp. 167–171).

Caswell, D., & Dörr, K. (2018). Automated journalism 2.0: Event-driven narratives: From simple descriptions to real stories. *Journalism practice*, *12*(4), 477–496.

Coddington, M. (2015). Clarifying journalism's quantitative turn: A typology for evaluating data journalism, computational journalism, and computer-assisted reporting. *Digital journalism*, *3*(3), 331–348.

Dong, E., Du, H., & Gardner, L. (2020). An interactive web-based dashboard to track COVID-19 in real time. *The Lancet infectious diseases*, *20*(5), 533–534.

Dörr, K. N. (2016). Mapping the field of algorithmic journalism. *Digital Journalism*, *4*(6), 700–722.

Dušek, O., Howcroft, D. M., & Rieser, V. (2019). Semantic noise matters for neural natural language generation. In *Proceedings of the 12th international conference on natural language generation* (pp. 421–426).

Dušek, O., Novikova, J., & Rieser, V. (2018). Findings of the E2E NLG challenge. *arXiv preprint arXiv:1810.01170*.

Ferreira, T. C., van der Lee, C., van Miltenburg, E., & Krahmer, E. (2019). Neural data-to-text generation: A comparison between pipeline and end-to-end architectures. *arXiv preprint arXiv:1908.09022*.

[3]It is our understanding that the EuroStat dataset is *mostly* eligible for commercial use with limited exceptions to data pertaining to non-EU and non-EFTA nations that are not official EU acceding and candidate countries. For example, the data pertaining to the US is not available for commercial use. See `https://ec.europa.eu/eurostat/about/policies/copyright`. To enable commercial use, such data needs to be removed from the dataset as part of the data preprocessing step. The COVID-19 dataset is not available for commercial use, but can be replaced by data provided by e.g. the European Centre for Disease Prevention and Control through `https://data.europa.eu/euodp/en/data/dataset/covid-19-coronavirus-data`.

Gatt, A., & Krahmer, E. (2018). Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, *61*, 65–170.

Gkatzia, D. (2016). Content selection in data-to-text systems: A survey. *arXiv preprint*. (Available at `https://arxiv.org/abs/1610.08375`)

Goldberg, E., Driedger, N., & Kittredge, R. I. (1994). Using natural-language processing to produce weather forecasts. *IEEE Expert*, *9*(2), 45–53.

Graefe, A. (2016). Guide to automated journalism. Tow Center for Digital Journalism, Columbia University.

Hämäläinen, M. (2019). UralicNLP: An NLP library for Uralic languages. *Journal of Open Source Software*, *4*(37), 1345. doi: 10.21105/joss.01345

Karlsson, F. (1996). *The word-forms of the finnish noun kauppa 'shop'.* (Available online: `http://www.ling.helsinki.fi/ fkarlsso/genkau2.html`)

Leppänen, L., Munezero, M., Granroth-Wilding, M., & Toivonen, H. (2017). Data-driven news generation for automated journalism. In *Proceedings of the 10th international conference on natural language generation* (pp. 188–197).

Leppänen, L., Munezero, M., Sirén-Heikel, S., Granroth-Wilding, M., & Toivonen, H. (2017). Finding and expressing news from structured data. In *Proceedings of the 21st international academic mindtrek conference* (pp. 174–183).

Leppänen, L., Tuulonen, H., & Sirén-Heikel, S. (in press). Automated journalism as a source of and a diagnostic device for bias in reporting. *Media and Communication*.

Linden, C.-G. (2017). Decades of automation in the newsroom: Why are there still so many jobs in journalism? *Digital Journalism*, *5*(2), 123–140.

Lindén, C.-G., Tuulonen, H., Bäck, A., Diakopoulos, N., Granroth-Wilding, M., Haapanen, L., ... others (2019). News automation: The rewards, risks and realities of 'machine journalism'.

Liu, C.-W., Lowe, R., Serban, I. V., Noseworthy, M., Charlin, L., & Pineau, J. (2016). How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. *arXiv preprint arXiv:1603.08023*.

McBride, K., & Rosenstiel, T. (2013). *The new ethics of journalism: Principles for the 21st century*. CQ Press.

Merkel, D. (2014). Docker: lightweight linux containers for consistent development and deployment. *Linux journal*, *2014*(239), 2.

Mindich, D. T. (2000). *Just the facts: How" objectivity" came to define american journalism*. NYU Press.

Montal, T., & Reich, Z. (2017). I, robot. you, journalist. who is the author? authorship, bylines and full disclosure in automated journalism. *Digital journalism*, *5*(7), 829–849.

Nie, F., Yao, J.-G., Wang, J., Pan, R., & Lin, C.-Y. (2019). A simple recipe towards reducing hallucination in neural surface realisation. In *Proceedings of the 57th annual meeting of the association for computational linguistics* (pp. 2673–2679).

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the association for computational linguistics* (pp. 311–318).

Puduppully, R., Dong, L., & Lapata, M. (2019). Data-to-text generation with content selection and planning. In *Proc. 33rd aaai conference on artificial intelligence.*

Reiter, E. (2007). An architecture for data-to-text systems. In *Proceedings of the eleventh european workshop on natural language generation* (pp. 97–104).

Reiter, E. (2018a). *Hallucination in neural NLG.* https://ehudreiter.com/2018/11/12/hallucination-in-neural-nlg/. (Accessed: 2020-03-02)

Reiter, E. (2018b). A structured review of the validity of BLEU. *Computational Linguistics*, *44*(3), 393–401.

Reiter, E., & Belz, A. (2009). An investigation into the validity of some metrics for automatically evaluating natural language generation systems. *Computational Linguistics*, *35*(4), 529–558.

Reiter, E., & Dale, R. (1997). Building applied natural language generation systems. *Natural Language Engineering*, *3*(1), 57–87.

Reiter, E., & Dale, R. (2000). *Building natural language generation systems*.

Schmitt, M., Sharifzadeh, S., Tresp, V., & Schütze, H. (2019). Unsupervised text generation from structured data. *arXiv preprint arXiv:1904.09447*.

Sirén-Heikel, S., Leppänen, L., Lindén, C.-G., & Bäck, A. (2019). Unboxing news automation: Exploring imagined affordances of automation in news journalism. *Nordic Journal of Media Studies*, *1*(1), 47–66.

Stark, J. A., & Diakopoulos, N. (2016). Towards editorial transparency in computational journalism. *Computation + Journalism Symposium*.

Stray, J. (2019). Making artificial intelligence work for investigative journalism. *Digital Journalism*, *7*(8), 1076–1097.

van der Lee, C., Gatt, A., van Miltenburg, E., Wubben, S., & Krahmer, E. (2019, October–November). Best practices for the human evaluation of automatically generated text. In *Proceedings of the 12th international conference on natural language generation* (pp. 355–368). Tokyo, Japan: Association for Computational Linguistics. Retrieved from https://www.aclweb.org/anthology/W19-8643 doi: 10.18653/v1/W19-8643

Wen, T.-H., Gasic, M., Mrkšić, N., Su, P.-H., Vandyke, D., & Young, S. (2015). Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 conference on empirical methods in natural language processing* (pp. 1711–1721).

Yleisradio. (2018). *Avoin voitto.* https://github.com/Yleisradio/avoin-voitto. GitHub.

Yu, R. (2014). How robots will write earnings stories for the ap. *USA Today*, *30*.