# An analysis of templates for generating text for use in comparing with data-driven models

Matthew Poulter
University of Cape Town
Rondebosch, Cape Town
pltmat001@myuct.ac.za

## ABSTRACT

Data-driven Natural Language Generation (NLG) systems have seen significant interest growth over the past decade, however, there is little to suggest that the benefits of using this data-driven model out way its costs when compared to a template-based system. This paper aims to provide an overview of the current state of template-based NLG systems in preparation for an investigation and comparison of their effectiveness and quality against data-driven models. This paper first looks at NLG and the overall design of such a system. It then provides an overview of the two approaches in question, namely template-based and data-driven models. This is followed by an analysis of three existing template-based NLG systems as well as discussing methods for evaluating and comparing these systems against potential data-driven ones. Finally, this paper looks at the practical application of developing a template-based NLG system.

## Keywords

natural language generation; data-to-text generation; template-based models; text analysis

## 1. INTRODUCTION

With the rise in popularity of data-to-text generation in recent years, so too has this field of study grown and developed. Data-to-text techniques have also evolved, and new techniques have been introduced. These techniques of data-to-text generation are part of a computing task known as Natural Language Generation (NLG), which this paper defines further in the next section.

NLG, specifically in the context of a data-to-text system, has been around for many years and has been researched across several fields, including weather forecast generation [5], journalism [17], medical reports [3], and sports commentary [9].

Most recently data-driven NLG models that use deep learning methods have seen significant interest growth. However, there is much debate as to whether these data-driven models are as effective as traditional models, such as template-based models. In fact, the E2E NLG Challenge[1] was recently formed to quantify the differences between the various models and perhaps resolve this debate.

After competing in the challenge, a pair of participants concluded that, "sometimes the costs of developing complex data-driven models are not justified and one is better off approaching the problem with simpler techniques [such as templates]" [20]. This, however, is not conclusive whether this will hold true for other NLG tasks, or outside of the specific context of the E2E NLG Challenge.

This paper therefore analyses the current state of template-based NLG systems, in preparation for an investigation and comparison of their effectiveness and quality against data-driven models.

## 2. NATURAL LANGUAGE GENERATION (NLG)

### 2.1 Definition

A widely cited definition of Natural Language Generation (NLG) is that NLG is "the subfield of artificial intelligence and computational linguistics that is concerned with the construction of computer systems that can produce understandable texts in English or other human languages from some underlying non-linguistic representation of information." [10]

However, there is some concern that this definition may encompass too great or too small a field. In this regard Gatt and Kramer [2] discuss at length the various concerns, but instead of debating this further here, this paper, similarly to Gatt and Kramer's survey, denotes NLG as having an input that is not linguistic.

Thus, for this purposes of this paper, Natural Language Generation, and by extension NLG, will refer to the task of generating human-readable text from non-linguistic [10] and structured data [20].

### 2.2 Architecture

Another widely debated area of Natural Language Generation is that of architecture. This is understandable considering the complexity of creating an NLG system. Many have argued that it is easier to build such a system when it is split up into distinct modules [11], however, some have said that while this modular approach may work conceptually, the vast array of programming languages and data representations makes it difficult for these modules to be interchangeable between systems [7], thus reducing the effectiveness of using a modular approach.

In terms of architecture specifications, this paper looks briefly at two of the most widely used and providing a comparison between them.

#### *2.2.1 Reiter/Dale architecture*

Originally published in 1997 [10], Reiter and Dale proposed a modular pipeline approach to architecture, which they revised and published again in their textbook in 2000 [11]. This pipeline

[1] http://www.macs.hw.ac.uk/InteractionLab/E2E/

consisted of three separate modules (see Figure 1) where the output of the previous module was used as the input for the next.

The first module, namely the Document Planner, which is responsible for "conceptual lexicalisation, content determination and document structuring" [11]. This produces a tree of packaged domain information known as the Document Plan which contains a breakdown of the input data as well as the overall structure of the text. This Document Plan is then passed to the Microplanner.

The Microplanner concerns itself with "linguistic aggregation, expressive lexicalisation, and referring expression generation" [11]. In general, the purpose of the Microplanner is to refine the Document Plan to produce a complete Text Specification. This Text Specification still needs to be rendered to an output that is readable, which is where the Surface Realiser takes over.

This final module, the Surface Realiser, traverses through the Text Specification producing output text. Each node in the Text Specification informs the Surface Realiser what it needs to do. The Surface Realiser is therefore responsible for "linguistic realisation, along with any necessary post-processing such as rendering the text in some specific output format" [11].

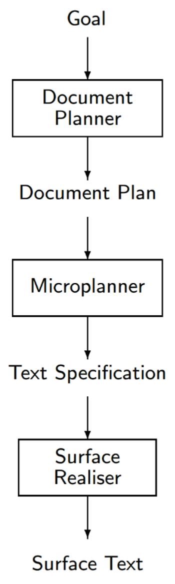

**Figure 1: Reiter and Dale's NLG system architecture [11].**

### *2.2.2 A Reference Architecture for Natural Language Generation Systems (RAGS)*

A 2004 article, written by Mellish et al. [8] and titled the same as this section, argues that the model put forth by Reiter and Dale is "more constraining than it may seem at first sight" [8]. Their argument surrounds the issue that many actual NLG systems were not able to follow the Reiter/Dale architecture at a granular level, and that the architecture was not necessarily practical.

Mellish et al. therefore proposed a framework titled "A Reference Architecture for Natural Language Generation Systems" (RAGS). The main components of this framework are:

- a high-level specification of the key data types used within an NLG system;
- a low-level specification of flexible data model able to handle the practicalities of an NLG system;
- an exact specification in XML for communicating between components of an NLG system;
- a generic overview of how the modules of an NLG system should fit together;
- sample implementations to provide demonstrations for how the framework can be successfully used.

The overall RAGS framework (Figure 2) is structured in such a way that the various modules can process data using native or potentially unknown methods, but that the communication between the modules is then controlled by the XML specification. This minimises the compatibility issues when using different programming languages for different modules, and therefore has the potential for these modules to be interchanged between NLG systems [8].

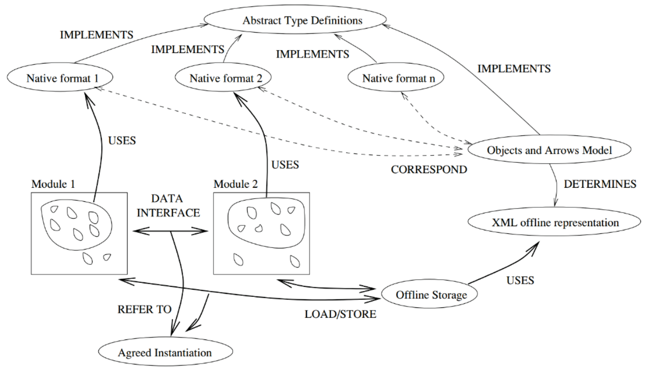

**Figure 2: Structure of the RAGS framework [8].**

### *2.2.3 Comparison of architectures*

While both RAGS and the Reiter/Dale architecture attempt to standardise the way NLG systems are designed, the core difference between these two architectures is that RAGS was designed to provide a more structured but also more abstract framework for NLG systems than the Reiter/Dale architecture offers [8].

This approach is certainly more freeing and allows for modules to be interchangeable between systems, however, it comes with the cost of additional layering and a learning curve in the software design which increases the development time.

The Reiter/Dale architecture is therefore more suited to individual or small team projects working on a tighter schedule, where modules are not necessarily needed to be swapped with modules from other systems.

## 3. NLG APPROACHES

While there are many different approaches to NLG, this paper, in its preparation for the investigation to follow, will focus on just two of them, namely the traditional template-based model and relatively new data-driven model. More so, this paper will pay particular attention to an analysis of current template-based systems.

## 3.1 Template-based model

Template-based text generation is not a new concept and is also certainly the wider used of these two models. In fact, one can even find a simple implementation of a template-based data-to-text model in the Mail Merge feature of the word processor, Microsoft Word [10].

In a more general setting, this template-based model can be applied when the domain of the application is small and the variation expected of their output is minimal, or at least expected to be so [19]. The following is an example of this.

> `<player>` scored a goal for `<team>` in the `<minute>` minute.

If the variables, player, team, and minute, are filled with corresponding values, the output would look something like the following.

> Juan Mata scored a goal for Manchester United in the 11th minute.

As is demonstrated by Theune et al. [18], templates do not have to be restricted by just a single output and can instead be automated and change depending on the circumstances. This could be done by adding a syntactic system to the templates themselves. An example of this is the following, where the output changed from the previous output when Juan Mata scored his second goal.

> Juan Mata scored **his second** goal for Manchester United in the 28th minute.

An advantage of this template-based approach is that it allows for complete control of output, thus also allowing control over the quality. Adding syntactic information to the template categories, or even including complex rules, can also allow for a more varied and realistic output.

On the other hand, creating templates can be a timely process if done manually according to Gatt and Krahmer [2], however Puzikov and Gurevych [20] have argued that an NLG task "can also be approached with a template-based model developed in just a few hours." Further, templates do not offer much flexibility when it comes to variation expected of the application's output [2].

### 3.2 Data-driven model

In contrast to a template-based model, data-driven NLG systems do not have the same level of control over the output, and instead opt for machine learning techniques using sample data to train the system [13]. This relinquishing of control does, however, give data-driven NLG systems the ability to produce a vastly greater variation in output than template-based systems. This is because one no longer needs to craft all the combinations of semantic elements needed to produce output text [13].

Instead, data-driven NLG systems use techniques to learn both the structure of the text as well as the surface realisation patterns from an example dataset known as the corpus [20]. Generally, here are two different approaches to using this model. The first approach is to create the small base grammar manually and then use data-driven methods to filter the outputs. The second approach is to rely on data-driven methods to create the base generator as well [2].

Regardless of the approach, using a data-driven model significantly decreases the amount of time it takes to design and create an NLG system [2], when compared to the traditional template-based approach [20]. The opportunity cost of this speed improvement is that there is less control over the outputs produced by data-driven NLG systems, and the systems have the potential to produce output that is less readable or understandable than the outputs of a template-based system [2].

## 4. TEMPLATE-BASED NLG SYSTEMS

Focussing on template-based NLG systems, this paper presents three recently documented systems from various papers about Natural Language Generation. These systems vary in application but they by no means make up an exhaustive list of possible template-based systems or applications.

Table 1 briefly outlines the overall similarities and dissimilarities between each of these systems, paying attention to the designs and architectures used. Clarity refers to how understandable and clear the output text was, and fluency refers to how easy to read it was.

**Table 1: A comparison of three template-based NLG systems**

| | PASS | Model-T | DYD |
|---|---|---|---|
| Modular | Yes | No | Partially |
| Pipeline | Yes | Yes | No |
| Tailored Output/Bias | Yes | No | No |
| Syntactic System | No | No | Partially |
| Variance | Yes | Partially | Yes |
| Clarity | Positive response | - | - |
| Fluency | Positive response | - | - |
| Naturalness | - | Rated second with other data-driven systems | - |
| Quality | - | Rated second with other data-driven systems | - |

### 4.1 PASS

Van der Lee et al. [9] recently developed the PASS system, a "data-to-text system for soccer, targeted towards specific audiences" [9]. As the title suggests, the template-based system was designed in such a way that it produces output text that was identifiably tailored for an audience. In the context of the PASS system, which produces reports of soccer matches, one could easily identify which team the outputted report was biased towards.

With PASS, van der Lee et al. [9] followed three steps to convert their corpus material, a database of real match reports often containing an emotional tone, into suitable templates for use in the system.

Firstly, each sentence of the reports was categorised according to the event it described. The reports were also separated according the outcome of the match: a win, a draw, or a loss. Secondly, van der Lee et al. [9] employed a reductive step to remove categories for which they did not have available input data to reproduce.

Finally, each sentence was converted to a template, much like the examples used in Section 3.1.

In fact, van der Lee et al. [9] made use of changing templates depending on the circumstances, however unlike Theune et al. [18], they chose to store these templates as separate categories, thus creating a larger database than similar systems. This trade-off did however afford them more variation in the generated output.

Another difference between the PASS system and the system designed by Theune et al. [18] is that PASS was designed to be modular. This decision was made to allow for replacing certain modules in the future if desired.

The basic outline of the modular system is that it uses a "governing module" to traverse each topic, sending them one by one into the pipeline. First, a collection of templates is selected based on the determined category of the current topic. The next module finds all the appropriate templates creating a list of options to be passed to the following module, which in turn selects a template in a weighted manner. [18]

At this stage, the template can be filled with the relevant information from the input. The following module assigns the topic an ordering based on where the topic happened within the soccer game. Once all of this has been complete, the output text is collected together in the correct order. Finally, the output is passed through three further modules to increase variety in the output text. [18]

Van der Lee et al. [18] proceeded to evaluate their system and their results showed that the system could accurately tailor 91% of its output reports. Further, both the clarity and fluency of the reports were rated well above the baseline score in their tests.

## 4.2 Model-T

For their submission to the E2E NLG Challenge, Puzikov and Gurevych [20] developed a template-based NLG system which they called Model-T. In comparison to PASS, the system is far more primitive, favouring efficiency over naturalness. This was because Puzikov and Gurevych [20] were attempting to show that this template-based system would hold its own against complex data-driven systems, while remaining less costly, particularly in time, than the other systems.

Like the examples used in Section 3.1, the Model-T system uses templates where values can be entered in based on the input data. The system is designed to output restaurant descriptions, and the following is an example of the type of template that is used, where each `<tag>` represents a different type of input datum.

> `<name>` is a `<type>` which serves `<food>` food in the `<price>` price range.

Some variations are included in the templates to allow for more fluent and natural language depending on the actual input. Further, the system employs a set of rules which split the templates into separate components and allow for components to be activated and deactivated depending on the availability of the input data [20]. Modifying the previous example, if the input did not include price data, the system would use the following template instead.

> `<name>` is a `<type>` which serves `<food>` food.

The Model-T system also included a post-processing step which added and corrected the punctuation and grammar of the output [20].

Puzikov and Gurevych [20] first performed five metric evaluations on the Model-T system and compared them to the data-driven system they had developed too, as well as another data-driven system as a baseline. Model-T scored below the other two systems in all the tests, however, Puzikov and Gurevych [20] surmised that this might be due to it not being data-driven and hence generating different outputs to the other two systems.

Upon submitting the Model-T system, the official evaluation results showed similar findings when it came to the metric scores, however in both human evaluation markers, quality and naturalness, the Model-T system performed on par with data-driven systems ranked second in the challenge.

Puzikov and Gurevych [20] therefore concluded from this that "sometimes the costs of developing complex data-driven models are not justified and one is better off approaching the problem with simpler techniques."

## 4.3 DYD

Created by van Deemter and Odijk [15], Dial-Your-Disc (DYD) is in fact a data-to-speech NLG system, however the system is conveniently split into two models, namely a language generation module and a speech generation module [14]. This paper exclusively refers to DYD in the context of its language generation module.

Within this module, the DYD system contains four submodules, however these submodules serve to augment the language generation module in its task, and much of the processing of the system is done in the main module itself [15]. This makes the DYD system only partially modular for the purposes of this paper.

From an overall design point of view, and unlike many other NLG systems which use sentence and paragraph planning to create suitable output text, the DYD system makes use of a "generate-and-test" strategy to build up its output. Adapting the example van Deemter and Odijk [15] give for this, if the system were to generate a sentence which contained a grammatical error, the testing stage of the DYD approach would simply discard the sentence and the system would attempt another sentence.

To generate sentences, the DYD system uses the notion of syntactic trees which contain variable parts which can themselves be filled with other syntactic trees recursively. This ultimately creates a template to be used for the sentence [15]. This template follows the same pattern as the previous two systems as well as the examples in Section 3.1.

Once these sentence templates have been generated, the system then attempts to create a coherent paragraph using them. It does this by determining what data is to be presented by the user, as well as keeping track of what has already been presented. Using the same "generate-and-test" strategy, the DYD system does not contain its own explicit text grammar. Instead it continually tests whether the sentence to be added makes sense within the current context, and only adds it to the text if this is the case. [15]

Further, the template sentence itself must satisfy four conditions before it is considered applicable for the context. These conditions are that the template must:

1. present the required information;
2. present only new information;

3. express the correct topic; and
4. be stylistically appropriate. [15]

Once these conditions have been met, the template can then be used to generate an actual sentence and be added to the output text paragraph being built. The paragraph is considered complete once all the input data needed to be conveyed has been compiled into sentences. [15]

## 5. EVALUATION OF TEMPLATE-BASED NLG SYSTEMS

This section looks at possible ways to evaluate template-based NLG systems with the view to being able to compare real systems based on the output they produce. Further, discussions are made on the suitability and appropriateness of using each of these methods to evaluate and compare data-driven NLG systems too and provide a useful comparison between these two types of NLG systems.

The evaluation methods have been categorised into three sections, namely Metric Evaluation, Error Analysis, and Human Indicators. The first two categories both involve automated testing of the systems, while the final category, somewhat obviously, requires human assessment.

### 5.1 Metric evaluation

The E2E NLG Challenge currently uses five metrics to evaluate the submitted NLG systems with [20]. This paper looks at two of these, namely Bilingual Evaluation Understudy (BLEU) and Recall Oriented Understudy for Gisting Evaluation (ROUGE).

#### *5.1.1 Bilingual Evaluation Understudy (BLEU)*

Bilingual Evaluation Understudy (BLEU) [16] is one of the most popular metrics when it comes to evaluating NLG systems. BLEU uses two tools to create its final score.

The first is called “modified n-gram precision” [16] and judges the output text of the system based on how similar it is to the reference texts which are inputted. It does this by first calculating the fraction of sequential words found when comparing to the reference texts. Secondly, it “modifies” this score by only matching a sequence as many times as it is found in the reference texts, to avoid unnecessarily long outputs. These scores are then summarised using a geometric mean to determine the final precision value. [16]

The second tool the BLEU metric uses is called the “sentence brevity penalty” [16] and, as the name suggests, simply penalises output text which is too concise. This value and the final precision value are then used to calculate the system’s BLEU score.

#### *5.1.2 Recall Oriented Understudy for Gisting Evaluation (ROUGE)*

There are four types of ROUGE metrics, however, the E2E NLG Challenge uses the ROUGE-L type for their comparisons [20]. This paper will therefore only look at the ROUGE-L metric, where “L” stands for “Longest Common Subsequence” [6].

The name explains much of how the metric works by finding the longest subsequence in common between the system output and the reference texts. It then calculates an F-score based on the and weightings assigned to each text by their lengths [6].

#### *5.1.3 Suitability for use with template-based systems*

Reiter [12] concluded that, at least in terms of BLEU, using the metric to evaluate non-data-driven systems is not supported. Further, he concluded that the metric should not be used to evaluate individual texts as opposed to systems.

As mentioned previously, Puzikov and Gurevych [20] came to the same conclusion when testing their Model-T system against data-driven systems. They state that this is expected since the output text a template-based system produces might be different to the reference texts. They further concluded that this was an issue with all the metrics used by the E2E NLG Challenge [20].

This is a notable issue considering Gatt and Krahmer’s [2] observation that template-based NLG systems do not produce a substantial variance in output. If the templates of the NLG system are similar to the reference texts, the metric will return a very high result, but if they are not, it will consistently return a very low result [20].

### 5.2 Error analysis

Puzikov and Gurevych [20] used a second test to compare their two systems. This test was based on the number of common errors made by each of the systems. The error types that were assessed can be seen in Table 2, which has been modified from Puzikov and Gurevych’s paper [20].

**Table 2: The types of errors detected by Puzikov and Gurevych’s paper [20] in their comparison of their NLG systems**

| Error type | Description |
|---|---|
| Bad grammar | Failure if the output text contains any bad grammar. |
| Modified contents | Failure if the output text alters a value provided in the input dataset. |
| Dropped contents | Failure if the output text leaves out a value provided in the input dataset. |
| Punctuation errors | Failure if the output text has punctuation errors. |

Puzikov and Gurevych [20] observed that the template-based system returned scores of zero for all the error types, while the data-driven system was predominantly altering or dropping contents from the input dataset. This was the first sign in their research that the template-based system may outperform the data-driven one.

### 5.3 Human indicators

Both van der Lee et al. [9] and the E2E NLG Challenge used human indicators to evaluate their systems. These indicators have been summarised in Table 3, where the first two correspond to those used by van der Lee et al. [9] and the last two were used by the E2E NLG Challenge [20].

Their findings in this regard have already been discussed in this paper, so won’t be repeated here, however, an important observation by Puzikov and Gurevych [20] should be noted here. Specifically, while their Model-T system performed relatively poorly in actual the actual score of the two indicators used, it was ranked in the second top cluster of systems, where systems within the same cluster are tied.

Further, both Lee et al. [9] and Puzikov and Gurevych [20] considered these indicators to be sufficient in comparing systems and evaluating their effectiveness and quality.

**Table 3: A summary of human indicators used in the works analysed in this paper**

| Indicator | Description |
|---|---|
| Clarity | A rating based on ow understandable and clear the output text is. |
| Fluency | A rating based on how easy to read the output text is. |
| Quality | A rating based on the overall quality of the text, when looking at grammar and format. |
| Naturalness | A rating based on how natural the output text felt to the reader. |

## 6. APPLICATION OF TEMPLATES

This paper has analysed many template-based systems in terms of both architecture and design choices. However, it has not yet discussed in depth the practicalities of developing a new template-based NLG system, and how this could be done at the present. In this section, this paper looks at Gatt and Reiter's [1] SimpleNLG, a "realisation engine" for Natural Language Generation. It further compares it to the methods employed by the van der Lee et al. [9] PASS system.

SimpleNLG is a Java library designed to perform tasks relating to the construction of grammatically correct and fluent sentences. It performs these tasks by receiving input in the form of English words and phrases including the desired role they should have within the sentence. [1]

In terms of Reiter and Dale's NLG pipeline [11], SimpleNLG would take the place of the Surface Realiser. The modular PASS system also contains a set of modules dedicated to constructing sentences and ensuring correctness of grammar. Once could therefore use the template approach employed by van der Lee et al. [9] to generate textual data to be inputted into the SimpleNLG library. This would be a very practical way to create a template-based NLG system.

## 7. SUMMARY

This paper has provided an overview of Natural Language Generation, showing the various architectures available when designing an NLG system. It has further given breakdowns of two models of NLG systems, namely template-based systems and data-driven systems. The findings of this comparison were that while template-based systems can produce more rigid output than data-driven systems, templates offer the reliability of complete control over the output.

An analysis of three existing template-based NLG systems showed that there is a wide spectrum of design decisions which can be made when developing these systems. Further, the analysis showed that variability of output was possible to a certain extent, and that these systems do not need to be grossly large and time-consuming to produce this variability and naturalness.

This paper then provided a summary of current evaluation methods used to evaluate NLG systems, as well as presenting the various arguments for and against each of them with regards to their use with template-based systems. Finally, a brief description of the practical aspects of designing such a template-based NLG system was presented.

## 8. REFERENCES

[1] Albert Gatt and Ehud Reiter. 2009. SimpleNLG: a realisation engine for practical applications. In *Proceedings of the 12th European Workshop on Natural Language Generation (ENLG '09).* Association for Computational Linguistics, Stroudsburg, PA, USA, 90-93.

[2] Albert Gatt and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: core tasks, applications and evaluation. *J. Artif. Int. Res.* 61, 1 (January 2018), 65-170.

[3] Albert Gatt, Ielka van der Sluis, and Kees van Deemter. 2007. Evaluating algorithms for the generation of referring expressions using a balanced corpus. In *Proceedings of the Eleventh European Workshop on Natural Language Generation (ENLG '07).* Association for Computational Linguistics, Stroudsburg, PA, USA, 49-56.

[4] Anja Belz and Eric Kow. 2010. Comparing rating scales and preference judgements in language evaluation. In *Proceedings of the 6th International Natural Language Generation Conference (INLG '10).* Association for Computational Linguistics, Stroudsburg, PA, USA, 7-15.

[5] Anja Belz. 2008. Automatic generation of weather forecast texts using comprehensive probabilistic generation-space models. *Nat. Lang. Eng.* 14, 4 (October 2008), 431-455. DOI: http://dx.doi.org/10.1017/S1351324907004664

[6] Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Proceedings of the Workshop on Text Summarization Branches Out (ACL-2004).* Association for Computational Linguistics, Barcelona, Spain, 74-81.

[7] Chris Mellish and Roger Evans. 2004. Implementation architectures for natural language generation. *Nat. Lang. Eng.* 10, 3-4 (September 2004), 261-282. DOI: http://dx.doi.org/10.1017/S1351324904003511

[8] Chris Mellish, Donia Scott, Lynne Cahill, Daniel Paiva, Roger Evans, and Mike Reape. 2006. A Reference Architecture for Natural Language Generation Systems. *Nat. Lang. Eng.* 12, 1 (March 2006), 1-34. DOI: http://dx.doi.org/10.1017/S1351324906004104

[9] Chris van der Lee, Emiel Krahmer and Sander Wubben. PASS: A Dutch data-to-text system for soccer, targeted towards specific audiences. In *Proceedings of the 10th International Conference On Natural Language Generation (INLG17).* Santiago de Compostela, Spain, 95-104. DOI: https://doi.org/10.18653/v1/w17-3513

[10] Ehud Reiter and Robert Dale. 1997. Building applied natural language generation systems. *Nat. Lang. Eng.* 3, 1 (March 1997), 57-87. DOI: http://dx.doi.org/10.1017/S1351324997001502

[11] Ehud Reiter and Robert Dale. 2000. *Building Natural Language Generation Systems.* Cambridge University Press, New York, NY, USA.

[12] Ehud Reiter. 2018. *A structured review of the validity of BLEU.* Comput. Linguist. 44, 3 (September 2018), 393-401. DOI: https://doi.org/10.1162/coli_a_00322

[13] Elena Manishina. 2016. *Data-driven natural language generation using statistical machine translation and discriminative learning (Computation and Language).* Ph.D. Dissertation. Université d'Avignon, Avignon, France.

[14] Kees Van Deemter, Emiel Krahmer, and Mariët Theune. 2005. Real versus Template-Based Natural Language Generation: A False Opposition?. *Comput. Linguist.* 31, 1 (March 2005), 15-24. DOI: http://dx.doi.org/10.1162/0891201053630291

[15] Kees van Deemter. 1997. Context modeling for language and speech generation. In *Interactive Spoken Dialog Systems on Bringing Speech and NLP Together in Real Applications (ISDS '97).* Association for Computational Linguistics, Stroudsburg, PA, USA, 48-52.

[16] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (ACL '02).* Association for Computational Linguistics, Stroudsburg, PA, USA, 311-318. DOI: https://doi.org/10.3115/1073083.1073135

[17] Leo Leppanen, Myriam Munezero, Mark Granroth-Wilding and Hannu Toivonen. Data-Driven News Generation for Automated Journalism. In *Proceedings of the 10th International Conference On Natural Language Generation (INLG17).* Santiago de Compostela, Spain, 188-197. DOI: http://dx.doi.org/10.18653/v1/W17-3528

[18] Mariet Theune, Esther Klabbers, J. R. De Pijper, Emiel Krahmer, and Jan Odijk. 2001. From data to speech: a general approach. *Nat. Lang. Eng.* 7, 1 (March 2001), 47-86.

[19] Susan W. Mcroy, Songsak Channarukul, and Syed S. Ali. 2003. An augmented template-based approach to text realization. *Nat. Lang. Eng.* 9, 4 (December 2003), 381-420. DOI: http://dx.doi.org/10.1017/S1351324903003188

[20] Yevgeniy Puzikov and Iryna Gurevych. 2018. E2E NLG Challenge: Neural Models vs. Templates. *In Proceedings of The 11th International Natural Language Generation Conference (INLG18).* Tilburg, The Netherlands, 463-471.