Natural Language Generation (NLG)

Sumit Dugar
Technical University of Munich
sumit.dugar@tum.de

Abstract

Commercially the interest in NLG systems is growing and much of this interest focuses on data-to-text systems. There are many different ways of designing such a system. We can create a modularized pipeline based architecture or an end to end deep learning based architecture. Most common architecture in present data applied NLG is the classical three stage architecture. The goal of this text is to explore the classical three stage architecture along with it's several tasks in detail. We would also explore in detail a data driven news generation system for automated journalism. Additionally this work can also be considered as a tutorial on how to build an NLG system.

1 Introduction

Natural Language Generation is characterized as the – *"subfield of artificial intelligence and computational linguistics that is concerned with the construction of computer systems than can produce understandable texts in English or other human languages from some underlying non-linguistic representation of information."* – Reiter and Dale (1997)

There is a clear consensus on what the output of an NLG system should be – text. However the input can vary substantially for different systems. According to Reiter and Dale (1997); Gatt and Krahmer (2017) based on the type of input an NLG system can be mainly classified in the following categories. Other input/output types such as audio can simply be converted to text. Since we are just converting and not generating anything new in terms of natural language so this step is generally not considered a task of an NLG system.

Data-to-Text – This system takes input in the form of flat semantic representations, numerical data or structured knowledge bases. Some useful applications include soccer reports generation, news report generation on current affairs, weather and financial report generation, generating summaries of patient information in clinical context.

Text-to-Text – This system takes input in the form of natural language text, for example in case of language translation, text summarization, automatic generation of peer reviews for scientific papers, simplification of complex texts.

Visual-to-Text – This system takes input in the form of images or videos for automatic caption or description generation.

Recent advancements in machine learning in general have sparked a lot of research interest in the application of deep learning for Natural Language Generation(NLG) tasks. In Spite of the growing research interest, most of the industries where NLG is presently used, still heavily rely on the classical pipeline based architectures. Reiter (2016) in his blog posts very aptly explains some of the reasons for the slow adaptation of machine learning techniques in NLG. The focus of this text will be on a classical pipeline based architecture as this is currently dominant in the industry specially for data-to-text systems.

We also will discuss in detail about different tasks that needs to be performed by NLG systems. Finally, with help of Leppänen et al. (2017) we will apply all these concepts on a data driven news generation application.

1.1 Organization

Section 2 introduces the classical three stage architecture along with the detailed explanation about what each stage does and how the output of each stage looks like. Section 3 discusses about the six tasks that almost every NLG system needs to perform for generating natural language text. Section 4 applies the concepts learned in the above sections in building an NLG system that can automatically generate news articles (Leppänen et al., 2017). Finally, the last section shows a short glimpse of deep learning in NLG along with the concluding remarks.

2 Classical three-stage NLG Architecture

The goal of a natural language generating system is to produce some output text in human understandable natural language from input data. There are many different ways of designing such a system. We can create a modularized pipeline based architecture as shown in figure 1, where each subproblem is well defined and cab be solved using specific algorithms. Another way of tackling this problem is by creating an end-to-end system, without having any well defined boundaries for sub-tasks. Irrespective of the architecture used, every NLG system performs certain sub-tasks (explained in Section 3) for achieving the end goal. From a practical perspective the most common architecture in the present day applied NLG systems is the first one. It is a classical NLG architecture with three stages that define the complete pipeline. Each stage can have one or more tasks.

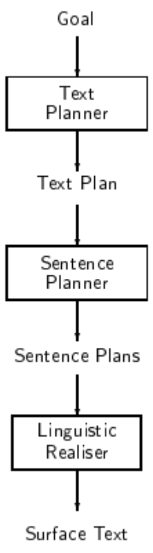


Figure 1: A block diagram of classical three-stage NLG Architecture from Reiter and Dale (1997). Here stage 1 is text planning, it's aim is to find *what to say*. Stage 2 is Sentence planning, it's aim is to decide *how to say*. Stage 3 is Linguistic Realisation, it's goal is to *actually say it*.

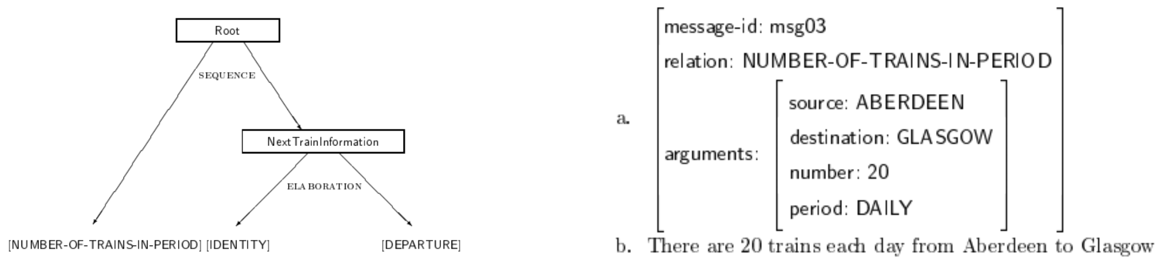
2.1 A sample excerpt from Rail Travel Information System

In order to understand the three stages and tasks better let's take a sample excerpt from Rail Travel Information System as an example – *"There are 20 trains each day from Aberdeen to Glasgow. The next train is the Caledonian Express. It leaves Aberdeen at 10 am."*

2.2 Stage 1 : Text Planning

This stage combines the content determination and text structuring tasks as shown in figure 4. It's mainly concerned with deciding – *what to say*.

The output of this stage is a Text Plan that goes into the Sentence Planner. Text plans are usually represented as trees. An example Text Plan is shown in first image of figure 2. Leaf nodes depict individual messages. Messages can be represented via templates or by an attribute value matrix as shown in second image of figure 2. Internal nodes show how messages are conceptually grouped together. This grouping can help in planning paragraph boundaries.



(a) (b)

Figure 2: First image represents a Text Plan of a sample excerpt from Rail Travel Information System. The three sentences of the excerpt as shown in section 2.1 are generated using the messages from the three leaf nodes respectively. The second image in this figure represents the first message using an attribute value matrix. A value can be a string or can optionally be represented by using another attribute value matrix as done for the *arguments* attribute. These images are taken from Reiter and Dale (1997)

2.3 Stage 2 : Sentence Planning

This stage combines sentence aggregation, lexicalization, and referring expression generation tasks as shown in figure 4. Sentence planning can be understood as deciding – *how to say it.*

Sentence Plans can be represented using templates in the form of boilerplate text and parameters. The actual values of these parameters are inserted using messages. The other common way of representing Sentence Plans is to use an abstract representation language such as SPL (Sentence Planning Language). It specifies the content words such as nouns, verbs, adjectives and adverbs of a sentence, and also the relationship between them. An SPL representation for a sentence is shown in figure 3.

```
(S1/exist
 :object (O1/train
         :cardinality 20
         :relations ((R1/period
                      :value daily)
                     (R2/source
                      :value Aberdeen)
                     (R3/destination
                      :value Glasgow))))
```

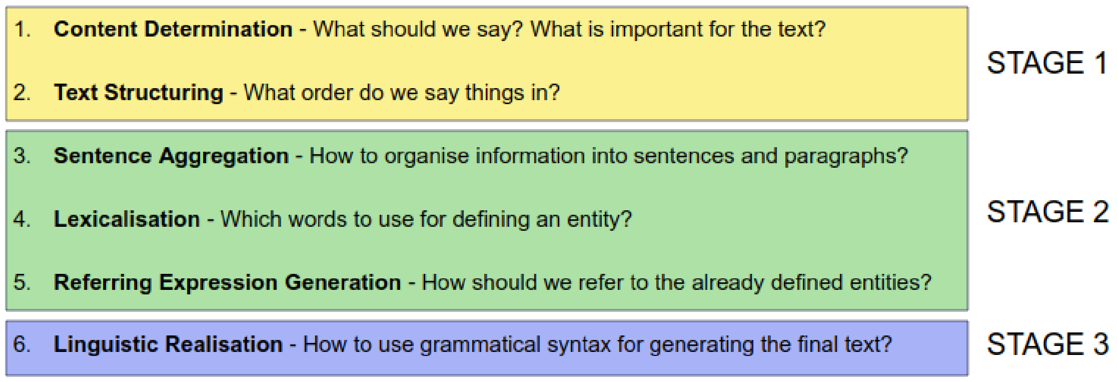
Figure 3: Sentence Plan from Reiter and Dale (1997) for the sentence - *"There are 20 trains each day from Aberdeen to Glasgow."*

2.4 Stage 3 : Linguistic Realisation

The goal of this stage is to generate sentences in a grammatically correct way, by applying syntactic and morphological rules. It can be understood as – *actually saying it.* The output of this stage is the final text that needs to be communicated. For example the sample excerpt shown in section 2.1.

3 NLG Tasks

As per Gatt and Krahmer (2017); Reiter and Dale (1997) there is a general consensus in the natural language community that there are six basic tasks that needs to be carried out for transforming input data into output text in some natural language. A stage wise separation of these tasks for the classical three stage architecture can be seen in figure 4.



1. **Content Determination** - What should we say? What is important for the text?
2. **Text Structuring** - What order do we say things in?
3. **Sentence Aggregation** - How to organise information into sentences and paragraphs?
4. **Lexicalisation** - Which words to use for defining an entity?
5. **Referring Expression Generation** - How should we refer to the already defined entities?
6. **Linguistic Realisation** - How to use grammatical syntax for generating the final text?

Figure 4: A stage wise separation of six NLG tasks for the classical three stage architecture.

3.1 Content Determination

The aim of this task is to determine which information to share with the end user in the final text. For this input content is filtered, abstracted and represented in some formal representation language e.g attribute value matrices, database structures etc.

Input content can be filtered using an importance score based on domain specific rules. These rules are either manually acquired by domain experts or are learned from a data corpus. Alternatively, we can also filter content based on user's query. For example in the train example that we have been following, content for the output response can be determined by filtering messages based on a particular source/destination that is been asked in the query.

3.2 Text Structuring

The aim of this task is to structure the messages produced by the content determination process into a coherent text. It is also known as discourse/document planning. The structuring is generally application domain dependent in the sense that, for example in case of a soccer report, it is reasonable to start with some general information like where and between whom the game was played and then move on to details about goals in temporal order. Whereas other applications may choose to impose structure based on relations such as – contrasts, elaborations, relatedness etc.

For development of practical systems generally domain specific schema based approaches are used. Schema is a pattern that species how a particular text plan should be constructed using smaller schemas or atomic messages. For example, a text plan that provides a response to the query *When is the next train to x?* might be constructed using the schemas shown in figure 5.

```
a.  Inform-Next-Train-Schema ⟶
            Sequence(Message:NUMBER-OF-TRAINS-IN-PERIOD,
                     Next-Train-Information-Schema)
b.  Next-Train-Information-Schema ⟶
            Elaboration(Message:IDENTITY, Message:DEPARTURE)
```

Figure 5: An example schema based representation for defining message ordering structure that can be used to generate response to the query *When is the next train to x?* In this example by Reiter and Dale (1997), for instance, the NLG system might start by executing the first schema *Inform-Next-Train-Schema*, which would in turn request the construction of a *NUMBER-OF-TRAINS-IN-PERIOD* message. After this, a subsequent call to execute the *Next-Train-Information-Schema* will result in requests for an *IDENTITY* message and a *DEPARTURE* message. In this way, content determination is interleaved with discourse planning.

3.3 Sentence Aggregation

It is the process by which related messages are grouped together into coherent sentences. It can also be considered as a process of removing redundancy to make the final text more fluid in terms of readability. For example, a less fluid text will have sentences generated out of every message – *"There are 20 trains each day from Aberdeen to Glasgow. The next train is the Caledonian Express. The Caledonian Express leaves at 10am."*

A more readable text would combine these sentences in some coherent way removing the redundancy - *There are 20 trains each day from Aberdeen to Glasgow, and the next train is the Caledonian Express, which leaves at 10am.* Although aggregation does not change the information content of a text, it does contribute to readability and fluency. A straight forward way of aggregation is to use simple connectives such as and to produce a sentence plan which communicates more than one messages. Additionally, if two or more messages being aggregated have a common prefix, it may be possible to omit the repeated constituent. For example, messages which might be independently realised as *John went to the bank. John deposited $50.* can be aggregated to produce a sentence plan that is realised as *John went to the bank and deposited $50.* Alternatively, if the messages being aggregated together have some common constituents, then it may be possible to group these constituents into some kind of set. For example, three messages that would otherwise be realized independently as *John bought an apple. John bought a banana. John bought a pear* can be combined into a sentence as *John bought an apple, a banana and a pear.* One general constraint used by many aggregation modules is to only aggregate nodes/messages that are siblings in the text plan. A weaker version of this rule is to allow all descendants of an internal node to be aggregated.

3.4 Lexicalisation

The aim of this step is to find words or phrases from natural language to represent concepts and relations of the selected messages. There is no straightforward way to model lexicalisation as a unique concept-to-words mapping. For example, both *animal* or *dog* can be used to represent a specific concept. Another example could be representation of the timestamp *00:00*. It can be expressed as *midnight* or *late evening* or simply *night*. Similarly, we can either use *departs* or *leaves* for communicating the relation *DEPARTURE* from the train example. Things like these, where a selection from semantically similar and near synonymous words has to be made increases the complexity of this task.

3.5 Referring Expression Generation

It is the task of selecting words and phrases to identify domain entities. It is different from lexicalisation in the sense that It's a discrimination task where the system needs to communicate sufficient information to distinguish between domain entities. The entities can be referred to using a pronoun, a proper name, or a definite description. Which form to choose depends on the context. For example, consider this text – *"The next train is the Caledonian Express. It leaves at 10am. Many tourist guidebooks highly recommend this train."*

Here, the entity *CALEDONIAN-EXPRESS* is initially referred to by name *the Caledonian Express*, which is a standard way of introducing a named object into a discourse. The second reference to *CALEDONIAN-EXPRESS* uses the pronoun *it*, this is again a standard way of referring to an object which has been mentioned recently. The final reference is a definite description *this train*, which is a standard way of referring to an entity when it has already been introduced, but where the context rules out the use of a pronoun.

From a practical perspective, a simple algorithm that works surprisingly well in many cases is to use a pronoun to refer to an entity if the entity was mentioned in the previous clause, and there is no other entity in the previous clause that the pronoun could possibly refer to. This is a fairly conservative algorithm, in the sense that it will not generate a pronoun in many cases where one could be used; but it has the advantage that it does not often inappropriately insert a pronoun.

An algorithm for definite descriptions, is to begin by including in the description a base noun describing the object for example, *train*, and then if necessary add adjectives or other modifiers to distinguish the target object from all other objects mentioned in the discourse. For example, if the discourse has just discussed the *Caledonian Express* and no other trains, then the train can be used to refer to the *Caledonian Express*. However, if the discourse has also mentioned the 10:15 am train from Aberdeen to Edinburgh, then a definite description for *CALEDONIAN-EXPRESS* should include extra information to distinguish this from the other train. We might build the description *the Glasgow train*.

3.6 Linguistic Realisation

It is the process of applying grammar rules to generate syntactically and morphologically correct sentences. This involves generating verb conjugations, inserting functional words such as verbs and prepositions, and applying punctuation marks.

If the application domain requires minimum variations a good approach for linguistic realisation is by using human crafted templates. Another advantage of templates is that they allow for control over the quality of the output and avoid generation of ungrammatical structures. The disadvantage of templates is that they are labour-intensive if constructed by hand. Alternatively, templates can be learned automatically from corpus data. Another commonly used approach for linguistic realisation is by using grammar based systems. These system make all their choices based on some written grammar rules. These rules can be handwritten in some realizers e.g MUMBLE, KPML. Recently statistical approaches are also becoming popular where a base generator is handcrafted and statistical information is used for filtering out the options. In one of the approaches first multiple alternatives are generated, and then a stochastic re-ranker selects the best realisation and another method uses statistical information directly for generating realisations rather than for filtering them.

4 Case Study : Data-Driven News Generation for Automated Journalism

In this case study we will investigate design and development of a NLG system that is capable of automatically producing natural looking news reports. The system developed by Leppänen et al. (2017) has been used during 2017 Finnish municipal elections for generating news in three different languages.

4.1 Requirements for Journalistic Natural Language Generation

According to Leppänen et al. (2017) there are some requirements that are generally important in journalism and these should be fulfilled by any journalistic NLG system. The NLG system should be *trustwor-*

thy, accountable and transparent. Traditional pipeline based architectures are more suited for providing transparency than the end-to-end deep learning based NLG systems. The produced text should be *accurate* and it should not provide any misleading statements. In order to be able to generate news articles over several domains with minimum efforts the NLG system should be *modifiable and transferable*. NLG systems are expected to produce natural sounding texts that are *coherent and fluent*. For good quality and regularity of news generation, it is necessary that good quality data is *easily available*. Highly *localized* news targeted to the user is desirable, since this kind of news mostly go unreported currently.

4.2 Data Source

Election data by Finnish Ministry of Justice (MoJ) was used. It was made available in a machine-readable format. It included the results for each party on the level of the whole country, each of the 13 electoral districts, 311 municipalities, and 2,012 polling stations. For each of the 33,316 candidates, the data included details of their success in their own municipality and all of the municipalitys polling stations.

4.3 Overview of the Architecture

A data driven modular architecture as shown in figure 6 was chosen by Leppänen et al. (2017). Based on the above requirements it would be a better fit than an end-to-end deep learning based architecture.

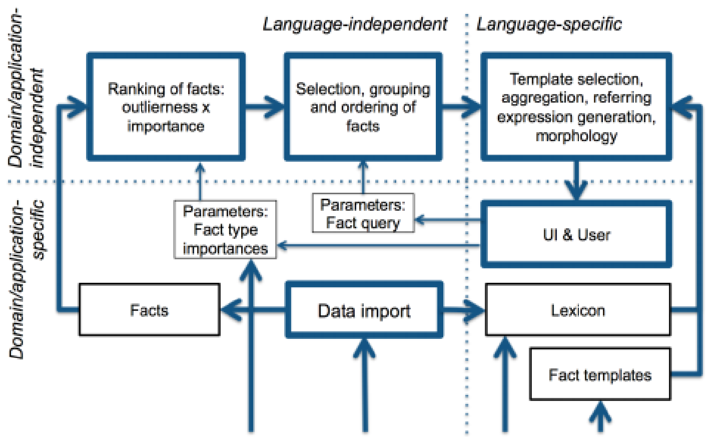

Language-independent
Language-specific
Domain/application-independent
Domain/application-specific
Ranking of facts: outlierness x importance
Selection, grouping and ordering of facts
Template selection, aggregation, referring expression generation, morphology
Parameters: Fact type importances
Parameters: Fact query
UI & User
Facts
Data import
Lexicon
Fact templates

Figure 6: Leppänen et al. (2017) chose a modularized data driven architecture. Thick boxes represent software components and thin boxes depict data structures. Software components corresponding to the top part of the figure are generic, and any information about the election/application domain is provided to the system as parameters or data during run-time. The bottom part of the figure is domain and application specific.

Fact Representation – Facts or messages were represented as a triplet of *(entity, location, value)*. Each field also needed to specify a category or type. For example, *entity* has two types candidate and party, *location* has four types the whole country, electoral district, municipality, and polling station. Value types are more varied. They capture numerical information such as percentage/count of votes received or number of seats won. Thus triplets became sextuplets *(entity type, entity, location type, location, value type, value)*. Slightly over 10 million facts were obtained. This flat structure of representing facts helped in achieving domain independence.

Lexicon – Lexicons are automatically extracted from the dataset for entities and locations. For values no lexicals were required as they are largely numbers or booleans.

Ranking of Facts – A measure of newsworthiness is computed for each fact/message as a product of outlierness and importance. Outlierness measures how exceptional a given value is when compared to other values of the same value type within the same location. Importance is defined by parameters which specify weights for different entity types, location types and value types. The value of these weights can also be adjusted at the run time according to a user's interest.

Fact Selection, Grouping and Ordering – The system allows users to make queries based on an entity and a location. The system then extracts only those facts that have the specified location and entity. The extracted facts are then ordered by their newsworthiness.

Fact Templates – A template consists of template expression and condition on which to apply this expression. A template expression consists of a sentence expressed in natural language with slots that can be filled with values from facts, such as *entity won value new seats in location*. Conditions are constraints on the facts that define when a template can be used e.g *value type = nr of seats, value ¿ 0.* Templates are generally written by domain experts.

Template Selection – Each fact in the document plan is associated with a template. If a fact satisfies multiple template conditions then, one is picked at random.

Aggregation – A simple approach was used where the system checks whether two subsequent sentences contain a common prefix or not. If they do, and the first sentence is not already a product of aggregation, the two sentences are joined together by a conjunction (e.g., and), and common prefix is removed from the second sentence.

Referring Expression Generator – A simple approach was used in which when an entity is first encountered in the document plan, the full name (e.g. Kansallinen Kokoomus) was used. On subsequent encounters, the short name (e.g. Kokoomus) or the pronoun form was used depending on whether the previously mentioned named entity is a different entity or not.

Morphology – To produce morphologically correct forms of filled slots, a library called Omorfi by Pirinen et al. (2017) was used.

4.4 Observations and Results

It was launched as a web application at https://www.vaalibotti.fi. At the day of launch, 398 unique visitors accessed 573 unique news stories. A sample excerpt of the system's output is shown below. In this the user has requested an overview article on country-wide results without any specific candidate or party as focus.

Perussuomalaiset drop most seats across Finland

Perussuomalaiset dropped the most council seats throughout Finland and got 3.5 percentage points fewer votes than in the last municipal election. Perussuomalaiset decreased their voter support by the greatest margin and has 769 seats in the new council. The party secured 4th most council seats.
Throughout Finland, Vihrea Liitto secured the largest gain in council seats. The party increased their voter support by the greatest margin and got 3.9 percentage points more votes than in the last municipal election. 12.4% of the vote went to the party. The party got 319440 votes.
Suomen Keskusta lost 254 seats and the second most council seats. ***The party has 2823 seats in the new council and won the most council seats nationwide in the previous election. The party secured 17.5% of the vote and had 3077 seats in the previous council.***

Overall, the generated news is expressed in clear language and the texts are fluent. But it is not always as fluent as human-written texts. There seem to be three main reasons for this. Firstly, fact ordering is sometimes suboptimal. For example, the second last line in the excerpt shows this suboptimality. Secondly simple aggregation methods can sometimes combine sentences in unnatural or misleading ways. For example, the last line in the excerpt shows this problem. In this line 17.5% refers to the current election and not the previous ones. Thirdly, fact templates need to be carefully written to not misrepresent any situation. For example, the template *"{party} got the most votes in {location}"* can be technically true even when two distinct parties were tied for the most votes received, but the reader would be misled to think that there was one winner.

4.5 Deep Learning in NLG

Now a days a lot of research is focused on deep learning based data driven approaches. A lot of end-to-end architectures have been proposed for both data-to-text and text-to-text applications. There are some other architectures which use classification networks for subtasks and optimize on the combined loss of these subtasks. Although theres a lot of research interest in deep learning based architectures, but most of them still arent the dominant approaches used in the industry. Some of the possible reasons could be –Deep Learning is still a black box at large and its really difficult to understand the reasons for a networks output. This makes it really difficult to fix problems with deep learning based architectures. Deep learning usually requires a lot of training data and in most application domains of NLG its really hard to find a good annotated training dataset.

In Spite of the above reasons there are some application domains such as image-to-text, generating text with style, personality etc, where deep learning architectures have become dominant approaches.

4.6 Conclusion

One of the common trends that can be identified here is that the steady move from early, hand-crafted approaches based on rules, to the more recent stochastic approaches that rely on corpus data and machine learning. This trend is catching up really quickly in the research community but not so fast in the industry.

References

Albert Gatt and Emiel Krahmer. 2017. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *CoRR*, abs/1703.09902.

Leo Leppänen, Myriam Munezero, Mark Granroth-Wilding, and Hannu Toivonen. 2017. Data-driven news generation for automated journalism. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 188–197. Association for Computational Linguistics.

Tommi A Pirinen, Inari Listenmaa, Ryan Johnson, Francis M. Tyers, and Juha Kuokkala. 2017. Open morphology of finnish. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University.

Ehud Reiter. 2016. Natural language generation and machine learning.

Ehud Reiter and Robert Dale. 1997. Building applied natural language generation systems. *Nat. Lang. Eng.*, 3(1):57–87.