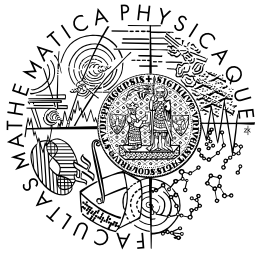

FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

BACHELOR THESIS

Dan Raffl

Building data-to-text Natural Language Generation system to create football articles (TODO-A-doplnit ze SISu a do názvu zapojit NLG (zde je zhruba můj návrh))

Institute of Formal and Applied Linguistics

Supervisor of the bachelor thesis: RNDr. Jiří Hana, Ph.D.

Study programme: Computer Science

Study branch: General Computer Science

Prague 2022

I declare that I carried out this bachelor thesis independently, and only with the cited sources, literature and other professional sources. It has not been used to obtain another or the same degree.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ....

Author's signature

Firstly, I would like to thank my supervisor RNDr. Jiří Hana, Ph.D.for his valuable advice as well as his support. Secondly, I am grateful to every teacher I had the chance to meet at Faculty of Mathematics and Physics for their attitude and inspirational work. Last but not least, I would like to thank my family for supporting me throughout my studies.

Title: Building data-to-text Natural Language Generation system to create football articles (TODO-A-doplnit ze SISu a do názvu zapojit NLG (zde je zhruba můj návrh))

Author: Dan Raffl

Institute: Institute of Formal and Applied Linguistics

Supervisor: RNDr. Jiří Hana, Ph.D., Institute of Formal and Applied Linguistics

Abstract: Journalism could become a tedious job as its main concern is to create as many articles as possible, usually prioritising quantity over quality. Some articles are quite routine and they need to exist just because most of the population is able to interpret data only in a very convenient text representation. The idea is to ease this job and generate articles, particularly about football in Czech language, automatically from non-linguistic data.

This paper is concerned with analysing implementation of such a linguistic software and moreover offers a brief overview of a Natural Language Generation (NLG) process. The major focus of this overview is on benefits and drawbacks of different approaches to NLG as well as describing NLG tasks and its challenges you need to overcome in order to produce a similar human language (not only Czech) producing program.

Keywords: linguistics, football, NLG, natural language generation, article

Contents

B - Introduction

Charles Babbage, the father of the computer, had the first impulse for the invention of a mechanically calculating system at college, when he was tired of mistakes in a table of logarithms. He suggested constructing a machine powered by steam, which could process a larger number of computations than humans while avoiding making mistakes. The idea of using computers to our advantage is carried even now in a highly-paced competitive technology-driven world. Even in the field of journalism, computer science has advanced and the results are getting quite stunning. In 2020 well-known British newspaper The Guardian published an article titled "A robot wrote this entire article. Are you scared yet, human?" GPT-3, in which AI language generator GPT-3 explains why its existence is not a threat to the existence of mankind. Ignoring the main point of the article, it is well written and I doubt that anyone would recognize that humans did not write the text. Furthermore, in 2022 GPT-3's abilities to write fluent prosaic text were described equivalent to that of a human by The New York Times Johnson and Iziev. This proves the results of AI in the field of language generating.

In this article we discuss numerous challenges of generating text in general and how to conceptually approach developing a language-producing software. This process of generating text (or some linguistic output as discussed below) is in the field of computational linguistics referred to as Natural Language Generation or NLG. These challenges are explained using various specific examples to illustrate exactly what may cause problems and how to prevent them. Also the aim of this paper is to introduce different techniques on how a NLG system can be organised and approached. Everyone who reads the paper should then be able to construct a solution for a NLG problem and especially be aware of the strengths or weaknesses of the solution.

Furthermore, one specific implementation of NLG is presented along with its analysis. This analysis includes description of overall structure, specification of how individual tasks are approached and finally a discussion of the solution. The task fully specified in Chapter (TODO - A - doplnit číslo kapitoly) was to generate a brief article that summarises what happened in a football match. To avoid ambiguity, throughout the article the word "football" refers to a sport, which American English speakers call soccer. This is a first neat example of detailed aspects you need to be aware of when developing a NLG system - meaning of a word can change with different locations. Therefore, it is very important to know the target audience of the generated language.

1. B - Natural Language Generation

1.1 B- What is NLG?

The intuitive meaning of the Natural Language Generation (NLG) is rather obvious, unlike the definitions that usually vary. I will now present definitions of NLG by two different authors:

1. "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."Reiter and Dale [1997]

2. "Natural Language Generation (NLG) is the process by which thought is rendered into language."McDonald [2010]

The definition (2) is much broader and less reliant on specification of what is output and especially input of such a task, which is here defined as "thought". The problem of identifying the source has been discussed even earlier in McDonald [1993]. He compared the situation to human conversation, namely when the speaker starts deciding what to say. Then the source can be state of mind, current situation, speaker's intentions etc. These inputs are bordering on the impossible to classify and represent in a computer. The output is defined simply as language, without further specifying its representation.

On the other hand, the definition (1) defines output as understandable text, which implies the form of the result is written and additionally restricts the input to be non-linguistic data. However, in the article authors also mention that their survey is focused on written texts, but the principles could be also applied to generating spoken language, which implies the definition could be extended. Examples shown above, and especially the contrast of specifying what output or input can or can not be, explain why it is extremely hard to define the term NLG precisely.

Let us consider the problem of summarising a book into a brief description. Given this problem and definitions above, we can see that it satisfies only the second vague definition, since the input is purely linguistic. This kind of problem is referred to as text-to-text generation. An example of such a generation is extracting summary from a dialogue of a customer and customer service department described by Liu et al. [2019] or pun generation Ritchie [2005].

Similarly, the initial problem can be transformed into video-to-text generation by replacing a book with a movie. In this scenario the line blurs even more. A movie has two components - sound and the video itself. The video has implicitly some semantic meaning and is surely non-linguistic. However, the sound usually contains spoken language and therefore is linguistic. Take, for instance, a video showing a person named Mark pointing at an apple with the sound being "This is a pear.". The sound itself implies there is a pear, but the message could be summarised as Mark is lying or Mark is not able to recognize a fruit. Finding a correct

summarization would be impossible with one of the components missing, because they both affect the overall message, which makes deciding if video-to-text could be classified as NLG difficult. This logic of reasoning can be applied to other problems that vary in their initial inputs such as generating diagnostic reports from image (e.g. roentgen) by Zeng et al. [2020], which could be characterised as image-to-text generation.

In this article, NLG is perceived as described by Reiter and Dale [1997], meaning creating text from non-linguistic data. This task is often referred to as data-to-text generation. Examples given throughout the article will fit this definition and the idea in general. Methods and approaches mentioned in this article may presumably be applied to any problem concerned with computational linguistics if it is suitable without a need to classify the problem as NLG.

1.2 B - Usage of NLG

The aim of the NLG is to generate documents, articles, reports, messages, emails, descriptions and other forms of texts in order to either reduce workload or to offer a reader a user-friendly interpretation of data in a given language. Various sources papers offer data-to-text implementations operating with different domains, input data and overall aims of the language:

- summarising data and creating reports
  - summarising statistics from a baseball match Puduppully et al. [2022]
  - summarising geo-referenced data such as map Thomas and Sripada [2007]
  - creating textual weather forecasts Sripada et al. [2014]
  - creating report of student's academic performance
- creating poetry (e.g. in Finnish by Hämäläinen et al. [2018])
- producing text to persuade reader something is good or bad Carenini and Moore [2006]

The most common usage of NLG is to summarise less readable data to more convenient textual form regardless of the domain (sports, weather, geography, etc.). Even though the output is the same, the reason to apply the NLG system is different. Compare textual weather forecasts to reports of a student's academic performance. Weather reports are produced in order to enable the general public to find out information about weather since their ability to interpret meteorological raw data is probably lacking. Same reason appears when summarising baseball statistics and maps. On the other hand, there is no doubt that teachers and professors can correctly interpret grades from student's studies, but the problem is to do so quickly as well as obtaining the data in a well-readable layout. Generating textual summary automatically resolves both issues effectively at the same time.

The summaries and reports do vary in one more important aspect - the amount of the information and level of terminology. For instance, when creating a weather report for everybody to read some information such as type of the rain or concrete

amount of precipitation (mm) will not be mentioned in the output text and the information will be abbreviated to "heavy" rain since this is the information the reader is interested in. Contrastingly, when creating a medical report (e.g. from surgical procedure) the terms should be precise and technical (possibly in Latin) and the amount of information left out is close to minimum. To conclude, the target audience and their knowledge of the domain will alter the produced text substantially.

The overall aims of the language diverge significantly. In producing poetry the rhymed text filled with phrases calling forth emotions is used to achieve experience as powerful and captivating as possible when reading it. In summaries the aim is to convey the factual information to the reader in an easy and understandable way. Notice that this should be a significant aspect of choosing an approach for the problem. If the goal of the text is to inform, then text is highly recommended to be as simple as possible. On the other hand, if the aim is to captivate, as it is in newspaper articles, then creating simple dull sentences is not a suitable option.

This was the first glance into the aspects that are crucial when deciding how to approach a language generating problem. These aspects are mentioned many times across the text since the goal of this article is to present to the reader how to approach the NLG and the correct understanding is one of the key parts to develop a solid NLG system.

Is the NLG system good to use every time? Debate whether the software is worth creating is indeed viable and maybe a little underestimated since the articles usually do not analyse this feature elaborately. In the real world this is a question of economic resources. Wages and time saved on the human work that has been replaced with software must outweigh the cost of the creation and maintenance of the software. Easy examples of evidently beneficial usage of the NLG system is customer service emails generator. Even a simple inserting name to the start of the email, then stating the product, order number and a little survey of satisfaction with the product is quite a trivial email to generate, does the job and saves tons of company time. Possible example of a non-optimal usage is when text is not the most convenient form of data for the user to comprehend. Charts, tables, schemata, maps or pictures could be considerably easier to transfer the intentional message to the end user as these structures are usually attractive and intuitive[1]. To give you an example, imagine coloured map of a city and using a red line to highlight the path to the nearest tourist information centre in comparison with text composed of verbal instructions where to go. Obviously, unless the route is fairly trivial, a map is a better solution due to its simplicity and visuality.

[1]Combination of visual representation of data and text could also be the best alternative in a certain scenario.

2. A- NLG Tasks

Transforming non-linguistic data into grammatically correct sentences in a given language seems like a rather complicated problem. Therefore it is intuitive to divide the initial problem to smaller tasks, which are easier to solve. This task structure is described by Reiter and Dale [1997] and it is widely-used to cover every challenge a fundamental NLG problem should deal with:

- Content determination
- Discourse planning
- Sentence aggregation
- Lexicalization
- Referring expression generation
- Linguistic realisation

In this section of the article we will discuss every task mentioned above. Note that approaches, which will be described later, may change this structure - there might be just a few changes by combining multiple steps and processing them simultaneously or the structure may be completely decomposed. Moreover, using data-driven methods is dominant approach and can be utilized across every task of the NLG. Their usage will be omitted in this section and further closely discussed in chapter 4 taking the uniqueness and dissimilarity of this approach into consideration.

The reason we describe every task individually is to highlight challenges that will arise along the way of creating the NLG system. Understanding these tasks is a crucial aspect to produce a well-built software regardless of the choice of the approach. To illustrate the problems we state numerous simple examples that should ease the process of fully recognizing the extent of issues related to each task.

2.1 B - Content determination

The goal of this task is to decide what information from input data should be included in the text. Usually the range of the input data is significantly larger than the amount of information we would actually transfer to the user. Naturally, this task is heavily influenced by the specifications of the assignment, namely domain, intention of the text and target audience. Consider the problem of creating a medical report from a complete blood count for a patient to read. This domain requires data preprocessing to recognize negative indicators from values contained in test results. In addition, a doctor or expert is needed to assist in order to interpret values correctly and set rules on how to identify diagnosis. The goal of the text is to describe the diagnosis in an understandable way and therefore using Latin or overly technical terminology should be avoided. However, if we take the doctor as a target audience, we now want to use the as precise expert

language as possible and probably change the content to present segments of the test results and not just the overall diagnosis to enable the doctor to better interpret marginal symptoms or flag values.

The result of the content determination is usually outputted as a set of preverbal messages, carrying semantic meaning of the statement. To carry all the information an implementation that can describe abstract concepts such relations between statements, entities or conditions is needed. This sub-task of creating suitable representation is usually domain-dependent. Since the important semantic information is mapped into some formal language, there is no need for (human) language to be specified, and therefore this task is language-independent. Concrete examples of formal representation language used to store these semantic attributes are for instance logical language, attribute-value matrices or graphs.

Example of the result of possible content determination is shown below in figure 2.1, where we would like to choose the content to report one simple message - a goal being scored in a football domain. We have two related (3) sets of attribute-value pairs - (1) is about a player and (2) contains goal statistics. Obviously some information in tables (1) and (2) are too specific (e.g. height of the player) for the message to convey and therefore redundant. Bold attribute-value pairs are highlighted as the ones to be present in the final text, creating preverbal message (4) in pseudocode, which is an output of content determination task. After performing the remaining NLG tasks possible result could look like (5).

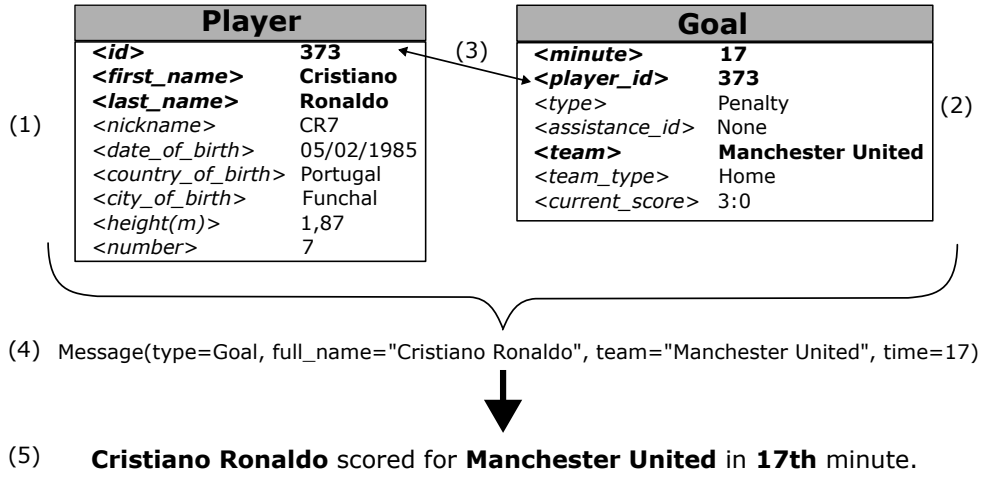

Figure 2.1: Process of content determination

2.2 B-Discourse planning

Previous part determines what messages will be transmitted to the reader and this part resolves the issue of the order in which the information is presented. This process is also referred to as text or document structuring. Selecting the right sequence of messages is crucial for text to accomplish its goal. Similarly we structure academic texts to logically ordered paragraphs, which present the topic in a way as understandable as possible for the reader to gain knowledge.

Similarly to content determination this task is highly domain-dependent as we have to know how to order messages. For instance, a medical report (as an example mentioned earlier) would likely display diagnoses and order decreasingly by how dangerous and life threatening they are. On the other hand, a report from a business meeting could start with a brief overview of achievements and goals and then with issues that were discussed ordered chronologically to allow the reader to follow the course of the meeting.

Human brain orders information to be conveyed in a speech intuitively, but the process as an algorithm itself is not quite trivial. Most of common method is to create rules based on the specific domain since the suitable structure heavily relies on the domain. Some researchers suggest using machine learning techniques for creating a uniform algorithm independent of the domain as seen in Dimitromanolaki and Androutsopoulos [2003].

Form of the output of discourse planning can differ. One possible option as described by Reiter and Dale [1997] is a tree structure. Leaves of the trees are messages and inner nodes describe specifics of their function in a sentence. This may seem like an unnecessary complicated solution when clustering messages to be said in one sentence can be just an array of messages. The benefit of the tree structure is the amount of information we can store along the messages including constraints under which the message can be said, relations between them and their overall structure.

2.3 B-Sentence Aggregation

The cardinality of the relation message and sentence is rarely one-to-one. Usually multiple messages are formed into one sentence. This process is called sentence aggregation and it is fundamental for generating text that is readable and flows well. To clarify we provide set of verbal messages:

1. *Peter bought an apple.*
2. *Peter bought a banana.*
3. *Anne did not buy anything.*

This set of sentences is clearly non-optimal[1] and can be aggregated in two steps as follows:

4. *Peter bought an apple and a banana. Anne did not buy anything.*
5. *Peter bought an apple and a banana, whereas Anne did not buy anything.*

We can notice two types of aggregation leading to the optimal sentence (5). Aggregation of:

- **constituents** - Constituents that have equal syntactic importance can be aggregated using suitable coordinating conjunctions expressing their relation. Take example sentences (1) and (2). *Apple* and *banana* are both

[1]Naturally, no such concept as "optimal" sentence exists. The optimum in this case is to express the information in a sentence that would likely occur in a spoken human language and also would appear fluid and natural.

items *Peter* bought so their semantic meaning is identical. Therefore they can be aggregated via cumulative conjunction *and* creating a new noun phrase in the result sentence (4) *an apple and a banana*. Another example of cumulative conjunctions is *both ... and* or *as well as*.

- **sentences** - Sentences can be aggregated as well using coordinators as seen in the result sentence, which was created by inserting an adversative conjunction in between sentences in example (4) to express opposition. This contrast can be expressed by other words like *but*, *yet*, *while*, etc. More relations can be expressed when aggregating sentences using another kinds of coordinating conjunctions - *alternative* (*or*, *either ...or*, *nor*) to express two or more alternatives and *illative* (*for*, *so*) to express interference or consequence.

One more type of aggregation can occur based on explicit hand-crafted domain-based rules. Take these three preverbal messages from football domain reporting a goal, which are similar to the one as used in figure 2.1-(4) (Manchester United shortened to MU):

6. *(type=Goal, full_name="Cristiano Ronaldo", team = "MU", time = 4)*
7. *(type=Goal, full_name="Cristiano Ronaldo", team = "MU", time = 8)*
8. *(type=Goal, full_name="Cristiano Ronaldo", team = "MU", time = 14)*

Surely realising messages (6), (7) and (8) as three different sentences would not create fluid and natural results as sentences would vary only in time of the goal. Since three goals in football form a so called hat-trick we could aggregate messages in one sentence:

Cristiano Ronaldo completes hat-trick for MU in under 15 minutes.

This aggregation realises the messages (6, 7, 8) beautifully as the result is well-formed and natural. Notice that the aggregation happened also in expressing the time of the goals - instead of mentioning three different timestamps the time is summarised as "under 15 minutes" highlighting the fact that this rare figure was achieved in a short amount of time. As described in the previous section, domain knowledge is necessary to decide what is "long", "short" or "average" (and therefore not worth mentioning) amount of time for an event to happen.

Note that these aggregations are simple for humans, but to perform them in a NLG we need some semantic knowledge and relations of the sentences (or constituents). The easiest approach is to define domain-specific constraints when to perform aggregation. Defining complex domain-independent rules and universal representation of relations is rather a difficult task and nowadays often solved using data-driven methods, which are described later in chapter 4.

Furthermore, the idea that the more aggregations we perform the better the final text is wrong. Sometimes slowing down the flow of information by fracturing the message into smaller individual sentences is useful in order to produce more understandable text. Overloading sentences can often result in less fluency as the more information is conveyed in one sentence the harder it is for a reader to follow. Barzilay and Lapata [2006] are perceiving this as a linear programming problem

where similarity is classified for each pair of database entries. Using this similarity, transitivity and global constraints (e.g. maximum number of aggregation across the document) they find an optimal solution.

2.4 B-Lexicalization

After performing discourse planning and sentence aggregation the preverbal messages are in a correct order and they contain suitably aggregated information. Goal of this task is to create mapping from these messages to specific expressions in a given human language. This task is the first that is language-dependent. There are two main problems associated with lexicalization. Firstly, the amount of combinations of how to narrate a message is enormous, only restricted to those that fit into the given context. And secondly, transformation of concept into a word (or more words) is very abstract and interferes with many layers of the language (semantics, phonetics and pragmatics) and therefore choosing a suitable expression is rather difficult. This transformation is not even easy for humans. Imagine an essay contest in grammar school with a given topic of the essay. If the transformation was easy and had only one solution, the contest would not exist as essays would be identical. In fact, the perspective and overall understanding of the topic, style of describing one's point of view and finally even choosing words to present the idea is partly what distinguishes us as people.

Another factor is the target audience and the overall goal of the language. If the target audience is educated on the matter then using adequate technical terminology is reasonable. Contrastingly, for low-skilled readers all terminology must be explained in an easy way and the content of the text should be more about overall ideas rather than about specific concepts.

Trivial approach to this task is to hand-craft pairings of a word or a whole phrase and a concept in a message. This solution results in monotonic outputs as the aspect of choice is missing. Slight improvement would be to add more semantically similar options for each item. However, this can cause problems. First of them is how to decide, which possibility is the best. Second is the possible non-viable combinations of words together. One example, that may be not visible on the first glance, is generation of adjectives interpreting numbers. For example, take a person with height 185 cm. If it was a man, the height is "average", while a woman could be described as "tall". Therefore semantic background and suitable comparison need to be taken into account. What is more, combinations of chosen phrases may result in non-realisable or simply weird expressions.

Due to the vagueness and coherence of the process, NLG systems combine lexicalization, REG and linguistic realisation under one operation called surface realisation or tactical part of the process.

2.5 B-Referring expression generation

Referring expression generation (REG) is a process, when you choose words to express domain entities or other constituents of the message. Naturally, utilising one noun phrase for one specific entity, which is used more than once in a short amount of text, results in less readable and fluid text. On the contrary, there

is a limit to how many such expressions we can generate since a reader needs to identify the entity correctly. Ambiguity is a highly unwanted effect since the information that needs to be conveyed may differ from its actual language semantic meaning.

To fully understand the challenges and also possible solutions for REG here is an example of sentence where we would like to lexicalize its subject represented as an entity in pseudocode:

(entity=Country, name="Czech Republic") has a population of 10,3 million.

This particular country can be lexically expressed in this sentence for example as:

1. *Czech Republic*
2. *Czechia*
3. *Country nicknamed "The heart of Europe"*
4. *Bohemia, Moravia and Czech Silesia*
5. *Country in the central Europe*
6. *Country which borders Germany, Poland, Slovakia and Austria*
7. *Beautiful rustic country in the heart of Europe*
8. *It*
9. *This country*

Notice the linguistic techniques we used to express this subject:

- **Entity name** - Using the name of the entity is a trivial solution and works fine as seen example (1).
- **Synonyms** - Using a synonym or a different name having the identical semantic meaning for the entity as shown in example (2) and (3).
- **Descriptive transcription** - using the knowledge of physical appearance, characteristics or its location we can describe an entity without any need of using its initial name. Examples (5), (6) and (7) are all using the location in Europe. Although expression (6) identifies Czechia unambiguously, the description of the location may be too specific for a less-educated reader (or possibly for a reader living outside of Europe), who has no idea where Germany is. The information that the country is situated in Europe is then sufficient. Therefore the target reader, his knowledge about a topic and also the purpose of the text are important even in this task. This technique is also prone to ambiguity as seen in example (5) - reader should already know what country is described in the text to use this expression since more countries can be characterised as "central European".

- **Definite descriptions** - The expression can be enriched by adding valid adjectives, adverbs or other linguistic structures to further specify the object as seen in example (7) where adjectives rustic and beautiful describe the country even a little bit more creating enriched and complex noun phrase.
- **Pronouns** - In a human language pronouns are often used to represent entities (e.g. "I have seen him." or examples (8), (9)). Using pronouns correctly can help to improve readability of the text and also minimise the obvious flags of computer-generated text. The main obstacle to overcome is when to use pronouns. Sometimes usage of a pronoun can arise from context, sometimes if there is absolutely certainty that everyone knows what the pronoun is referring to - those examples are hard to deal with and usually handled explicitly. Usual approach is to use pronoun if the entity was mentioned in a previous sentence under the condition the entity was the only constituent the pronoun could refer to.

How to approach REG depends also on repetition of the entities in the text and the final text variability. In a domain where identifying the entity unambiguously is primary (e.g. city in air travel) the usage of REG is even harmful. For instance, expressing "New York, USA" as "The city that never sleeps" in the air travel domain. To the contrary, expressing an entity identically multiple times in a short span of prosaic text eventuates in dull, plain and stereotypical language.

2.6 A-Linguistic realisation

This task transforms every constituent to form a grammatically and syntactically well-built sentence. Example of a struggle when building even a simple noun phrase is:

1. *(entity=Animal, name ="dog", count=1)* → *one dog*
2. *(entity=Animal, name="dog", count=23)* → *23 dogs*
3. *(entity=Animal, name="mouse", count=2)* → *two mice*
4. *(entity=Animal, name="fish", count=1000)* → *thousand fish*

Trivial and also naive solution for this simple noun phrase building can be to append morpheme "s" to the name of the animal if the count is more than one. As you can see in example (3) and (4) this solution can work only for animals that have regular plural. In addition, the count itself is recommended to be expressed by a word and not by numeral if the number is either a small integer (1-10) as seen in the comparison of examples (1) and (2). Same rule applies as well for a well-known rounded number (hundred, thousand, billion, etc.) shown in (4).

One so far omitted important aspect is the principles of morphology and syntax of the language. Concept of appending morpheme "s" to express plural might not be so easily transferable into different language. In Slavic languages morphemes to express plural differ and also can be infixed, meaning they could be inserted into the word stem instead of using a suffix. In Czech, a word for dog is *pes*. Then realising their number would look like: *1 pes, 2 psi, 5 psů*. To

further complicate the situation, Slavic languages are fusional. That means a single morpheme carries multiple grammatical, syntactical or semantic meanings. In Czech, agreement between the grammatical case and the noun is resolved as well with a infix morpheme. Here are three examples in Czech using different grammatical cases along with translation from English:

- (case: nominative) *two* ***dogs*** → *dva* ***psi***
- (case: genitive) *without two* ***dogs*** → *bez dvou* ***psů***
- (case: instrumental) *with two* ***dogs*** → *s dvěma* ***psy***

Notice that not only the noun, but the word for number 2 (*dva*) as well are subject to the linguistic transformation as the infix morpheme carries both the agreement with case and the plural.

We have mentioned typology according to morphology (fusional language). Other type is analytic languages (e.g. Vietnamese), where every single word is exactly one morpheme. Lastly, the complete opposite is polysynthetic languages (e.g. Inuit languages), where one word is constructed by combining plenty of morphemes representing a whole sentence. In central Nunavut Inuktitut *Tusaatsiarunnanngittualuujunga* means *I cannot hear very well*. This section just illustrated the need of the knowledge and principles of the language during the linguistic realisation.

Realisation has quite an extensive magnitude caused by the non-trivial goal of correctly morphologically and syntactically (as well as semantically) express each of the lexical item in the sentence. This process requires adding auxiliary words (prepositions, verbs, etc.), handling agreements, ordering, inserting punctuation and other similar transformations all in order to present the language not only factually, but even grammatically right. Due to the complexity of the task multiple approaches exist. Here are two among one of the most used approaches for lexical realisation described further below:

- **Templates** - example of a template usage in football domain (same as already seen in figure 2.1):
  1. preverbal message →
     Message(type=Goal, full_name = "Cristiano Ronaldo", team = "Manchester United", time = 17)
  2. template →
     $full_name *scored for* **$team** *in* **$minute***th minute.*
  3. result →
     Cristiano Ronaldo *scored for* **Manchester United** *in* **17***th minute.*

- **Grammars**

Templates are hand-crafted using fixed lexical items and attributes substituted in the template. Preverbal message (1) is assigned to a template (2) and three variables are then substituted with values creating the target sentence (3).

The advantage of this approach is simplicity and prevention of grammatical errors considering that we have full control of what the fixed segments are and any unwanted error is highly improbable. The disadvantages prevail. First of all, applying templates could be only feasible in a well-defined low-volume domains as entities must have easy to-text-interpretation. Another reason is creating templates is time-consuming. And most importantly, the variation of the output is low as the immutable parts of the text generate very limited output. In addition, template approach for more complicated languages tends to struggle, because the constituents usually depend on each other (e.g. agreement, auxiliary words, etc.) creating requirements that hard to fulfil.

However, templates are extremely practical when the expected output is expected to be simple and rarely changing. Great example is generating spoken announcements transportation domain e.g. departures of flights on the airport, where the template could look like: *The departure of flight number $number from $destination_from to $destiantion_to will be slightly delayed.* → *The departure of flight number FD-2018 from Rome to Paris will be slightly delayed.* Results are admittedly blunt in terms of language richness, but they are factually correct, clear and easy to comprehend, which was the initial purpose.

Grammars are the approach ...

(TODO -A - GRAMATIKÁCH v rámci realizace, pak to projet google doc překladačem)

(TODO -A - pak se vrhnout na přepsání corpusu ve smyslu získávání dat i automaticky, nejen hand-crafting)

(TODO -A - pak přidat schéma modulárního approache)

(TODO -A - pak přidat evaluaci)

(TODO -A - pak zadefinovat a vhodně propojit se zbytkem textu NLG process - analýza jazyka, cíle, dat, evaluace, cíle, prostředků atd.)

3. A- Requirements analysis

When developing a text-producing software, we need to analyse four fundamental aspects:

- Input data
- Expected output
- Goal of the language
- Target audience

As we mentioned above using various examples these factors can greatly influence methods and our overall approach to the problem. For instance, producing a routine text with formal requirements will be approached differently than generating an eye-catching book teaser. Communicating these specifics can be tricky when only one side has an insight into computational linguistics. The most suggested method[1] is to create a corpus, which is a collection of inputs and corresponding text outputs. This process ensures clear definition of expectations of what the output will look like and prevent proceeding missunderstanding.

3.1 A-Corpora building

Building a corpus should be supervised and consulted with a domain expert (e.g. for creating medical reports the doctor should participate in creating example texts) in order to achieve the best result possible. For corresponding input an example output should be written by a domain expert. Some NLG problems can find better results when extracting output texts from already existing "approved" texts. For example, when generating a short weather forecast it would be ineffective to create newly-written texts. Much better approach would be to extract forecasts from the most popular weather websites and take those as expected outputs. Benefits of this approach are reduced time spent acquiring example outputs and also ensured quality of the text since using popular websites. Some downsides could arise when applying this approach - acquiring this data can contradict with copyright law and also the output text of our new implemented NLG system (when done optimally) will produce the same texts as those popular websites and therefore there is no reason why our forecast should become more read and popular.

What should a good corpus look like? Corpus should be comprehensive and offer a wide range of pairs input-output. Edge cases, exceptions or less unusual texts should be incorporated in the corpus as well as average text to produce. Corpus should be exhaustive in terms of expectations - once the corpus is agreed upon and finished, adding functionality explicitly is very complicated and is likely to change the overall structure (depending on its complexity). The result of the process above is called initial corpus.

[1] This method is useful even when developing software independently without any specific party, that would require the specifics, meaning we can create a corpus to our liking.

The developer should now make a revision of the initial corpus to guarantee that the NLG system can deliver the expected text, because not every input and corresponding output must necessarily be correct. Firstly, the output text can be improved (e.g. when acquiring texts from existing one). Secondly, the information that is contained in the sentence may not be present nor computable from the given input data[2]. This is a critical part of the development as there is no way to resolve this problem every time and it is highly application-dependent. Common solutions are to extend the input data or remove unavailable information from the text and create a new version avoiding the information. Similarly, a compromise of finding hand-formed rules when to convey the information may be the solution. After all the necessary changes to the initial corpus were made, the corpus is now composed strictly of well-built and agreed upon example texts, where every information needed to its generation is contained in a data directly or it can be computed from given data. This finalised corpus is called target text corpus.

To summarise, building a corpus is not obligatory, but highly recommended practice to ease the process of developing the NLG system, especially finding requirements. Precise target primarily precedes the problem that the quality of the developed system is insufficient as the user knows exactly what the output will look like and match his expectations perfectly. In addition, the corpus is a fine tool even for the developer himself. Analysis of each record of the corpus results in an outline of the challenges to overcome and gives a basic idea how the particular NLG problem should be approached.

[2]The information can also be somehow contained, but building tools for its extraction would be insanely time consuming or the time complexity of the extraction would be high and therefore not possible - this is up to the developer to analyse and determine data transformations within his reach.

4. NLG approaches

So far we have covered what individual tasks we need to cover in our solution without suggesting a proper structure, which organises these tasks together and creates a compact implementation. There are two main approaches:

- Modular architecture - The basic idea is to split the program into parts that handle the NLG problem task-by-task. The division of the tasks is not always clear and therefore the idea of this approach is joining closely related tasks into an individual unit - module. These modules are then linked with a pipeline offering systematic and well-structured architecture.
- Global approach - The tasks are fully deconstructed and the problem is solved globally in order to get rid of the limitations associated with modular architecture. Such NLG systems then process smaller segments of NLG tasks alternately in order to make the optimal choice each time with every context state and next step available. Utilising this freedom is the core aspect for success of this approach. Two methods realising decomposed architecture will be discussed - Planning and data-driven methods.

This division is definitely not strict as NLG systems offer a wide range of architectures and their combinations. Data-driven (or statistical, stochastic) methods are a widely discussed topic among the NLG community as their popularity grow and their results are recently getting better, often outperforming more traditional procedures. Note that statistical methods can be incorporated inside a modular architecture to resolve one or more tasks. Similarly, you can take the global approach and avoid data-driven methods completely. Also grouping many arbitrary tasks together and approaching them differently is a possibility. This section is about describing such approaches in detail.

4.1 Modular architecture

This approach was described by Reiter and Dale [1997] and became almost a standard for a long time. However, nowadays stochastic and data-driven methods are prioritised over this approach (further discussed later in Chapter). The idea is to construct a module for each NLG task and link those modules via a one-way pipeline. Hence, output of content determination is an input for discourse planning and so forth. But some of the tasks are closely related and therefore it is efficient to assign those tasks to one module. Accustomed layout by Reiter and Dale [1997] of modules is: (TODO - A - obrázek schématu)

- Text (document) planner → content determination + discourse planning
  = *"what to say"*
- Sentence planner → sentence aggregation + lexicalization + REG
  = *"how to say it"*
- Linguistic realiser
  = *"saying it correctly"*

Text planner determines the content of the text as well as the order, in which we present the content to the reader. This part of the NLG process is also referred to as strategic generation. Sentence planner transforms messages from text planner to a lexicalized expression. And finally linguistic realiser combines these expressions with regard to syntactic and grammatical rules of given language. Choices made during sentence planning and linguistic realisation are often together called, on the other hand, tactical. Clear distinction between tactical and strategic will be more important later when discussing data-driven methods. The main advantage of this 3-module is the structure itself, which is easy to follow and understand. In addition, offering clear division of what is each module's obligations as well as what issue they don't resolve. Simplicity results in accessible and well-structured code. Moreover, it is easier to change a minor functionality of the program since classifying where to change the code is intuitive. The same logic applies for upgrades - one or more tasks can be approached completely differently than the rest of the code using other methods (e.g. statistical) by keeping the input and output structure.

The core drawback of the approach arises from the same aspect which was mentioned above as an advantage - well defined clear systematic structure. This brings certain limitations. Once we decide sentence order and content we have no chance of changing it later and so the choices in the early stages may later result in an unsolvable issue. Imagine generating a sentence with a limited maximum number of characters. In a text planner we have chosen the content of the sentence, but even when we do every possible combination of lexicalization of the sentence the number of characters still exceeds the upper bound. So what now?! The solution would be to retroactively change the content of the sentence and drop part of the initial information, but this is not possible since the pipeline is one sided. Of course this problem could be bypassed by making the pipeline go backwards, but this would completely break the point of modular approach. The clear line of division among modules would disappear and modules' functionality and objectives would suddenly overlap. To put it simply, assembling the structure of the text before knowing linguistic resources may result in incorrect, ambiguous or bizzare expressions. Therefore other approaches are usually based on breaking the structure and skipping to different tasks depending on the state of the development and the constraints that arise along the process.

4.2 Planning approach

In order to produce a text in a decent quality many decisions must be made resulting in various alternatives. Similarly as described in Fikes and Nilsson [1971] where the idea is to find a universal robot solver for the world model represented as first-order predicate formulas. The broadness and vagueness of the process of starting with various preconditions and getting to the desired result lead to a another approach, which highly differs from previous two mentioned approaches - planning.

Planning problem (as well as planning approach in general) is described by Gatt and Krahmer [2018] as "the process of identifying a sequence of one or more actions to satisfy a particular goal". Actions are then described as preconditions and their effect after applying them. In the terms of NLG, the goal is to convey

the message along with its aim (persuade, inform, captive, ...) to the target person. The actions then have constraints under which they can be performed and the effect is the change to the current context all leading to the desired goal of the language. The main idea is to create formalism that does not rely on a strict structure of the NLG system and available solutions to alternate between different NLG tasks with the recognition of both current state and effects of the chosen actions all in order to create the best possible language and broaden the limits of pure modular approach.

4.3 Data-driven approach

Unlike both mentioned approaches above, data-driven methods do not define the architecture of the NLG process. Modular approach states three-moduled architecture connected via one-way pipeline, whereas the main idea behind planning is not having predetermined architecture at all and approaching the process globally with a well-build formalism enabling freedom for combinating actions in. Data-driven methods can be applied regardless of the choice of the architecture. Meaning that stochastic methods can be used in a global approach, single NLG task or certain bigger subsections like strategic or tactical parts, for instance.

As the name implies, data-driven methods crucially rely on data, which consist of inputs and corresponding outputs. Using statistical or probabilistic principles of comparing our current state of the NLG process to a similar state in the data ensures making choices similar to those in the data. Note this definition data can be grasped as a corpus. However, corpus is restricted to only using the endpoints of the NLG process - the initial data and the final result output text. Since these methods can be applied on even smaller segments of the generation data can contain processed input or output data in various internal representations, which appear during the process and not just the initial and final stage. For example, when computing sentence planner inputs of our testing data are preverbal messages and outputs are lexicalized texts.

The very first obstacle that arises when performing these methods is the acquisition of the input-output data, because the data must fulfil some requirement. The amount of the dataset should be big enough to ensure the validity and overall principal of the statistical and probabilistic approach. The dataset must not only satisfy the requirement on the quantity, but certain variety must be ensured as monotonic data tend to give one-sided and misleading results.

Acquiring data itself can be tricky. Easiest scenario when an already established corpus for a specific domain (e.g. weather forecasting, hotel and restaurant recommendation, sport reports and more) exists. These corpora are well-built, but on the other hand useful only when working with the same domain. The reason for that these corpora may be available is firstly because their usage is common and secondly for their input data simplicity. Working with a less-common domain with larger range, types or complexity of data will result probably in the absence of a viable corpus and therefore building the corpus is another problem appended to the NLG process. To overcome this problem we can either build a new corpus from scratch or exploit more stochastic methods that automatically align input with outputs. Then again one disadvantage is that the data are heavily domain-specific. The alignment of the input to the segments of out-

put is crucial for most of the methods except deep neural networks and other machine-learning methods. These methods are recently becoming dominant in certain subfields of NLG such as image-to-text generation.

After assuming we have acquired data for the full-ranged NLG process, we can classify a stochastic approach based on the overall architecture. One group approaches the problem globally and completely decomposes the modular approach in order to both avoid error propagation and allow the software to make decisions freely across multiple tasks and different stages of generation. In opposition, the second group uphold at least the division between tactical and strategic choices.

As mentioned in the beginning of this section, the data-driven methods can be applied only to process on of the NLG task and not a whole generation in general. So far we have described how these data-driven methods work in general without specifying these methods in detail. We state some examples to further illustrate usage of data-driven approach with different specific methods and range of NLG process they cover:

1. **stochastic process** - Work of Ratnaparkhi [2000] is a nice first example to introduce data-driven methods, because he described three systems (NLG1-3) along with their comparison. System are used for tactical generation (semantic context is provided in a corpus given attribute-value pairs aligned with textual outputs) in a air travel domain. First system (NLG1), for given attributes simply chooses a template with the highest number of occurrences in the training data. Second system uses maximum entropy probabilistic model to predict the best proceeding word taking the already generated and also the attributes yet to be generated into account. However, the dependency of the words in language may not come from their order. Therefore NLG3 predicts the best words based on their syntactic relations represented by a tree. Along with the systems description Ratnaparkhi [2000] offers their results in terms of correctness. Both NLG2 and NLG3 heavily outperformed NLG1. Furthermore, NLG3 was performing slightly more accurate than NLG2. Ratnaparkhi [2000] states that both NLG2 and NLG3 can be used in other domains as well, but the complexity of the domain must be somewhat similar to the air travel (meaning quite low). Implicitly domain annotated data must be provided to further exploit NLG2/3 systems.

2. **classification** - Duboue and McKeown [2003] used classification process in their system for automatic content determination illustrated on biography generation problem. System is provided with initial data and target texts. Algorithm starts with clustering the semantic data (e.g. by age) and matching segments of the output to the pieces of input. This forms a solid base for the process of creating content selection rules using the binary classifier for each attribute of input data whether the attribute should be mentioned or not.

3. **optimisation** - The article of Marciniak and Strube [2005] researches the optimisation process in Natural Language Processing (NLP) approached as a integer linear programming problem. They use this approach even in a field of NLG when generating textual route directions. The global

approach of results in the elimination of error propagation and has better overall results as a consequence according to Marciniak and Strube [2005]. Summary of the specifics such as the metric that should be minimized during linear programming is further described, for instance, in Gatt and Krahmer [2018]).

4. **probabilistic context-free grammar (PCFG) and parsing** - The idea of mixing the NLG tasks together is further exploit by Konstas and Lapata [2013] as they call the resolvement of every single task separately "greedy". The need for domain-specific approach is here eliminated as this work is concerned with concept-to-text generation. They use input to model a PCFG and then using the stochastic methods to acquire the best word sentence satisfying the grammar. Such a process can be viewed as an opposite to semantic parsing. This system was tested on three different domains - sportscasting, weather forecasting and air travel query generation. Performance results were described as same or even better to the methods known at the time.

Note that these four examples not only show various specific statistical methods, but they illustrate other niances as well. Take the range of the process they cover for instance. Examples (4) and (3) cover end-to-end process, contrastingly example (2) covers only the tactical part and example (1) only covers content determination. Moreover (1) and (2) keep the strategic and tactical division unlike (4) and (3). Domains differ as well, especially in example (4), in which domain does not have to be specified.

These examples share two more similarities except the implicit statistical approach. Firstly, they rely heavily on the testing data and especially their alignment of input and output. Secondly, their results are somewhat superior to other hand-engineered systems. Often hand-crafting a NLG system is relies heavily on the domain and lacks portabiltity and certain variabilitly of the output (respectively achieve the variability by hand is the more tideous job the more variable the output should be). NLG systems grounded on statistics are robust and a task of data acquirement is added to the end-to-end solution. This is counterbalanced by overall better results and more portability as example (4) is domain-independent. These methods along with the solid linguistic foundation are nowadays dominant since the robustness is not big enough to be uncomputable with modern computers and the availability (or the opportunity to compute the) and amount of testing data is much better.

5. Implementation

5.1 architektura (zdůvodnění výběru) + přehled fungování

5.2 jak jsou řešeny a implementovány úlohy NLG (zdůvodnění jejich výběru)

5.3 diskuze řešení - alternativy, rozšíření, další podněty

Conclusion

Bibliography

Regina Barzilay and Mirella Lapata. Aggregation via set partitioning for natural language generation. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 359–366. Citeseer, 2006.

Giuseppe Carenini and Johanna D Moore. Generating and evaluating evaluative arguments. *Artificial Intelligence*, 170(11):925–952, 2006.

Aggeliki Dimitromanolaki and Ion Androutsopoulos. Learning to order facts for discourse planning in natural language generation. *arXiv preprint cs/0306062*, 2003.

Pablo A Duboue and Kathleen McKeown. Statistical acquisition of content selection rules for natural language generation. 2003.

Richard E Fikes and Nils J Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4):189–208, 1971.

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

GPT-3. A robot wrote this entire article. are you scared yet, human? *The Guardian*. URL `https://www.theguardian.com/commentisfree/2020/sep/08/robot-wrote-this-article-gpt-3`.

Mika Hämäläinen et al. Harnessing nlg to create finnish poetry automatically. In *Proceedings of the ninth international conference on computational creativity*. Association for Computational Creativity (ACC), 2018.

Steven Johnson and Nikita Iziev. A.i. is mastering language. should we trust what it says? *The New York Times*. URL `https://www.nytimes.com/2022/04/15/magazine/ai-language.html`.

Ioannis Konstas and Mirella Lapata. A global model for concept-to-text generation. *Journal of Artificial Intelligence Research*, 48:305–346, 2013.

Chunyi Liu, Peng Wang, Jiang Xu, Zang Li, and Jieping Ye. Automatic dialogue summary generation for customer service. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1957–1965, 2019.

Tomasz Marciniak and Michael Strube. Beyond the pipeline: Discrete optimization in nlp. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 136–143, 2005.

David D McDonald. Issues in the choice of a source for natural language generation. *Computational Linguistics*, 19(1):191–197, 1993.

David D McDonald. Natural language generation. *Handbook of Natural Language Processing*, 2:121–144, 2010.

Ratish Puduppully, Yao Fu, and Mirella Lapata. Data-to-text generation with variational sequential planning. *arXiv preprint arXiv:2202.13756*, 2022.

Adwait Ratnaparkhi. Trainable methods for surface natural language generation. *arXiv preprint cs/0006028*, 2000.

Ehud Reiter and Robert Dale. Building applied natural language generation systems. *Natural Language Engineering*, 3(1):57–87, 1997.

Graeme Ritchie. Computational mechanisms for pun generation. In *Proceedings of the Tenth European Workshop on Natural Language Generation (ENLG-05)*, 2005.

Somayajulu Sripada, Neil Burnett, Ross Turner, John Mastin, and Dave Evans. A case study: Nlg meeting weather industry demand for quality and quantity of textual weather forecasts. In *Proceedings of the 8th International Natural Language Generation Conference (INLG)*, pages 1–5, 2014.

Kavita Thomas and Gowri Somayajulu Sripada. Atlas. txt: Linking geo-referenced data to text for nlg. 2007.

Xianhua Zeng, Li Wen, Yang Xu, and Conghui Ji. Generating diagnostic report for medical image by high-middle-level visual information incorporation on double deep learning models. *Computer Methods and Programs in Biomedicine*, 197:105700, 2020.

List of Figures

List of Tables

List of Abbreviations

A. Attachments

A.1 First Attachment