SAARLAND UNIVERSITY

Faculty of Mathematics and Computer Science
Department of Computer Science
MASTER THESIS

Natural Language Generation from Drone Sensory Data to Warning Messages

submitted by
Jiajing Fang
Saarbrücken
October 2022

Advisor:
Ernie Chang
Department of Language Science and Technology
Saarland Informatics Campus
Saarbrücken, Germany

Reviewer 1:
Prof. Dr. Vera Demberg
Department of Language Science and Technology
Saarland Informatics Campus
Saarbrücken, Germany

Reviewer 2:
Prof. Dr. Antonio Krüger
German Research Center for Artificial Intelligence
Saarland Informatics Campus
Saarbrücken, Germany

Saarland University
Faculty MI – Mathematics and Computer Science
Department of Computer Science
Campus - Building E1.1
66123 Saarbrücken
Germany

Erklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Statement

I hereby confirm that I have written this thesis on my own and that I have not used any other media or materials than the ones referred to in this thesis

Einverständniserklärung

Ich bin damit einverstanden, dass meine (bestandene) Arbeit in beiden Versionen in die Bibliothek der Informatik aufgenommen und damit veröffentlicht wird.

Declaration of Consent

I agree to make both versions of my thesis (with a passing grade) accessible to the public by having them added to the library of the Computer Science Department.

Saarbrücken, Oct 25 2022
(Datum/Date)

Jinjing Fang
(Unterschrift/Signature)

Acknowledgements

I want to thank all people that helped me for making this thesis possible.

I sincerely thank Prof. Dr. Vera Demberg for offering me the opportunity to write this thesis under her supervision and I want to thank Prof. Dr. Antonio Krüger for having an interest in this thesis and being my second reviewer. Furthermore, I want to thank my advisor Ernie Chang for giving me valuable advice and spending many hours for discussion with me.

In addition, I would like to thank Hui-Syuan Yeh, Kathryn Chapman, and Alisa Kovtunova for contributing to our original data annotation. I would also want to thank anonymous contributors in Wikipedia and Github community for providing learning materials on the background research of related work and implementation for this thesis.

Lastly, I am grateful to my family and friends who have supported me throughout the entire process.

Abstract

The rapid development of drone technology has empowered non-professionals to perform a variety of applications. However, drone crashes occur frequently and drone safety hazards gradually emerge. How to leverage fruitful sensory data on drones and assist users in effectively performing dedicated tasks has become a central focus in the community. Therefore, a high-performance Natural Language Generation (NLG) system in the drone assistant domain, which can generate fluent and native warning messages from sensory data, is in high demand. This thesis proposes a pipeline for the data-to-text NLG task in the drone warning messages domain. Because research on sensory-data-to-text corpus is limited currently, an initial corpus, consisting of input sensory data and corresponding warning utterances, is designed and annotated first. Then, a neural NLG system is implemented using different scheme methods for initial corpus and training based on a pre-trained language model. Automatic metrics and human evaluation show that our NLG system significantly outperforms the baselines in all three scheme methods. In a further investigation for domain generalization, few-shot learning capability, and robustness to input noise, we found that the performance of different scheme methods varies greatly.

Contents

Chapter 1
Introduction

1.1 Motivation

Drone technology has developed so rapidly recently that drones with various features and functions are becoming more and more common (Fuhrman et al., 2019). Drones have already been applied to the military successfully, for example, in aerial surveys, mapping, and inspections. Due to the development of advanced control algorithms and sensors, flying drones is more and more simple for beginners, so its application extends to the individual level, such as selfie drones. During the Covid-19 epidemic, drones are also playing an essential role in logistics, epidemic prevention, and rescue. Furthermore, due to the increasing demand for contactless delivery, drone delivery services are undoubtedly a massive boon for the catering and logistics industry. It can effectively reduce labor, improve distribution efficiency and solve traffic congestion problems. In recent years, especially since the global outbreak of Covid-19, drones have been playing important role in express, remote-area and non-contact deliveries. Since instant and same-day delivery has become the trend in the next ten years, companies such as Google, DHL, UPS, and Amazon have put effort into drone delivery research. Since 2005 drone delivery services have been tested by some tech companies. They becomes more and more mature that plans for launching drone delivery services can be expected in the near future. In the global drone market research report by BIS (BIS, 2021), it reported that the worldwide drone delivery market is expected to have significant growth in the next decade. As a result, drones will be used for an increasingly wide range of tasks and manipulated by many other non-professionals, so interacting with drones becomes more critical.

On the one hand, existing drones are manipulated mainly by experts due to their successful application for professional uses. To enable non-professionals to interact with drones successfully, building a natural language generation (NLG) system that can generate fluent and native utterances from primary sensory data is essential. These utterances can warn and remind users in interaction as handover warning messages and can be applied not only in the specific situation for drones (see Figure 1.1), which is focused on in this

thesis, but also in situations for self-driving cars and other interaction between human and autonomous machines.

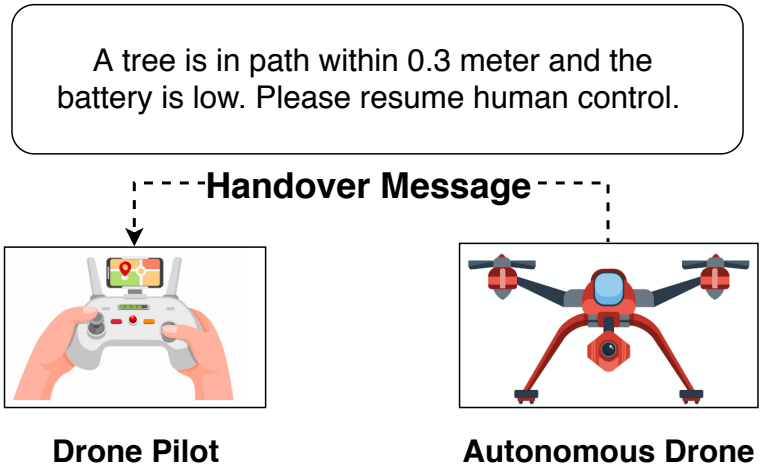


Figure 1.1: A scenario of our NLG task (Chang et al., 2022).

On the other hand, recent natural language generation research focuses on various NLG tasks e.g. few-shot, table-to-text, structured-data-to-text and so on (Liu et al., 2018; Freitag and Roy, 2018; Chen et al., 2020). Most NLG studies focus on fluency and integrity of output descriptions from simple record sequences, for example, E2E and WikiBio which are both in specific preset domains (Novikova et al., 2017; Lebret et al., 2016). In contrast, the scenario of our NLG task is more complex. A large variety of data records from different sensors are available in the drone setup environment, while not all available information must be mentioned in the warning messages. In an ideal drone handover assistant, only the critical information should be conveyed to the pilot concisely. In other words, our corpus is new and different from existing ones, and the implementation and evaluation of the NLG setup are also unique to meet our purpose and situation. To this end, we believe there is some value in research on this topic.

1.2 Research Goals

The main research goal for this thesis is to design and evaluate the neural NLG system for generating accurate and fluent warning messages from sensory data records as a drone handover assistant. There are three main challenges regarding our situation of diverse data input from different sensors in drones initially to meet the goal.

Firstly, it is how to design rules and logic to annotate the corpus from original video clips. With the help of others, we annotate a tabular dataset **DroneParrot** with corresponding native English utterances, and Figure 1.2 shows a depiction of one data sample in the dataset.

Secondly, it is how to implement and perform fine-tuning for the neural NLG model. As our dataset is brand new, this part includes investigating suitable model architecture, potential data linearization method, and the fine-tuning phase to generate a well-performed NLG model.

Thirdly, it is how to evaluate our NLG system in different aspects. Our NLG task is unique

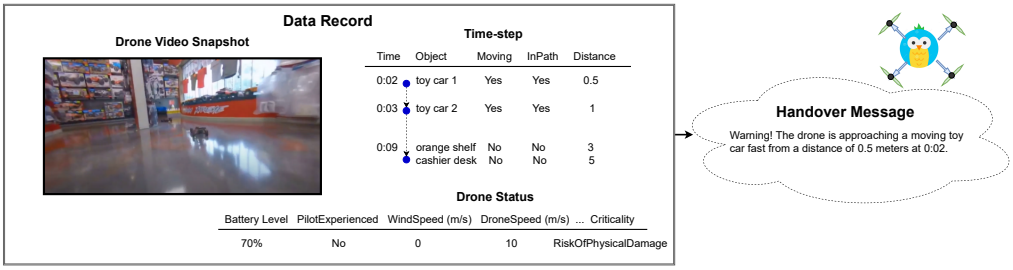


Figure 1.2: Description of one data sample from the dataset and its corresponding warning message. For full drone status and time step records refer to Table 3.3 and Table 3.4 (Chang et al., 2022).

compared to mainstream NLG tasks such as document summarization (Liang et al., 2009), machine translation (Hajic et al., 2002), conversational dialogue (Kale and Rastogi, 2020), etc. Also, our **DroneParrot** dataset has its own characteristics and application scenarios. Therefore, potential evaluation methods from the above aspects are investigated and applied to our evaluation part.

1.3 Outline

The structure of the thesis is outlined as follows. The chapter Introduction introduces the current state of drone development and gives an overview of the background and motivation of the NLG task for this thesis, as well as lists the research goals. The Related Work chapter discusses basic concepts of Natural Language Generation, a variety of related examples of NLG tasks from structured data or knowledge, the state-of-the-art data-driven datasets in NLG and their scenarios, state-of-the-art neural NLG model architecture as well as differences between mainstream NLG tasks and our NLG task. The logic and rules in establishing our dataset and the annotation process are described in the chapter Dataset Setup and Annotation. Details of our neural NLG system implementation and fine-tuning can be found in the chapter Implementation. It is divided into implementation linearizer using three different scheme methods, implementation of rewriter which is trained and fine-tuned from pre-trained T5 model on our dataset. In the chapter Experiments and Results, experiments are carried out to evaluate the neural NLG models in different aspects based on the characteristics of our NLG task, and further discussion is presented. Lastly, the chapter Conclusion summarizes the findings of the thesis, presents overall achievements from our research goal, and proposes approaches and recommendations for future work.

Chapter 2
Related Work

In the following section, related work about various methods and example tasks in NLG area is presented. The main purpose of this section is to summarize current research outputs related to our topic such that some insights can be given for further implementation. Therefore, it is essential to build a fundamental understanding of NLG at the beginning. Then, existing examples for NLG tasks in different scenarios and their performance have been investigated to show the values of our topic. What's more, state-of-the-art data-driven NLG datasets and neural NLG model architecture have been discussed in detail to give some insights for designing our implementation approach.

2.1 Concept of Natural Language Generation

Natural Language Generation (NLG) is to produce natural language output using a software process. It is characterized as the subfield of artificial intelligence and computational linguistics. The development of NLG can be traced back to the mid-1960s, while NLG methods were used commercially starting from the 90s. These methods and techniques can be various from simple template-based systems to more complicated systems utilizing various methods to improve the semantics and syntax correctness of the output. Nowadays, with the rapid development of machine learning, NLG tasks can also be implemented by training a statistical model which has promising performance (Reiter and Dale, 1997). Some common applications of NLG are automatic report generation, image captioning, and the dialogue system. To have a deeper and more comprehensive understanding of how NLG works in practice, the concept of the dialogue system, which is similar to the NLG task in this thesis, is described below.

A dialogue system is a computer system designed to talk to humans (Jurafsky and Martin, 2008). Since the early 1960, the dialogue system has developed rapidly. Compared to the written-text-only dialogue system, in the beginning, it utilized one or more communication modes, e.g. speech, plain text, and so on. Nowadays, dialogue systems are widely used for various purposes including information gathering, customer service, and so on. They can be classified into various usage categories. While some of them simply retrieve keywords from input and generate short and grammar-free responses

using common phrases obtained from the associated database, some might use extended word-classification methods, and other artificial intelligence method.

In a dialogue system, there are sets of components each of which has its responsibilities and might differ from system to system. In general, a typical activity cycle (see Figure 2.1) in a dialogue system consists of the following parts (Jurafsky and Martin, 2008):

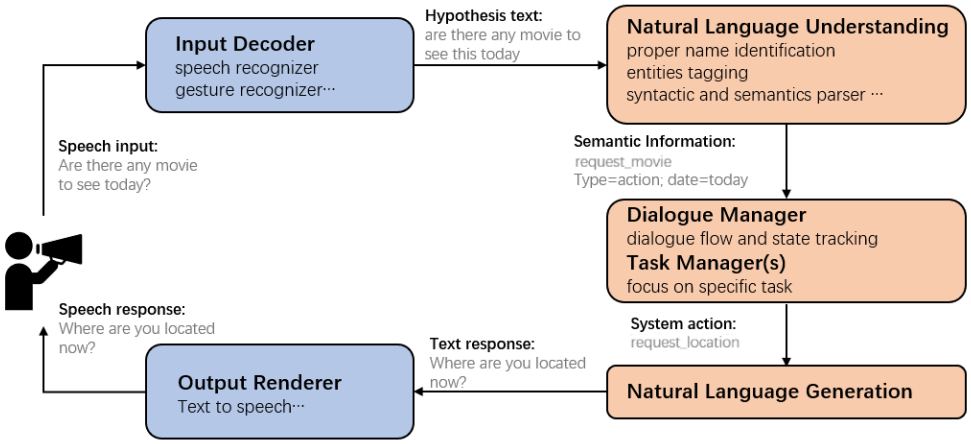


Figure 2.1: Activity cycle of a dialogue system (Young, 2000).

1. *Input decoder*: If a dialogue system is not only text-based, this phase should be included so that what the user delivers to the system as input is converted to plain text. As these inputs might be text or one and more of speech, gesture, haptics, etc. from system to system. So most of the time the decoder has other recognizer as components, like speech-to-text recognizer, gesture-to-text recognizer and so on.

2. *Natural language understanding*: The plain text from the input decoder is analyzed by the NLU unit to be understood by the system per se. In this part, methods for identifying proper noun and tagging entity as well as parsering syntactic and semantics meaning are needed.

3. *Dialog management*: It does the analyze for obtained information in previous step. Its main purpose is to keep tracking the conversation.

4. *Task management*: In dialogue management, there will be one or more task management that focuses on their specific task.

5. *Output generator*: The outputs from dialog management are produced by the output generator. As the plain texts are not the final output most of the case. Methods for generating natural language, speech or even gesture are required.

6. *Output renderer*: If the output of the dialogue system is not only text, this phase is the one that renders the output into the proper modality. It may be a text-to-speech engine, robot or avatar, etc.

In order to build a mature NLG system, interdisciplinary knowledge is needed. Furthermore, NLG is also a big topic with a close relationship with NLU and NLP.

2.2 Related NLG Tasks and Applications

Here we introduce several NLG tasks and their datasets. We can get a better understanding of the research development and pipeline of building up the NLG system in a specific task and get insights into the differences and similarities between our NLG task and other mainstream existing NLG tasks.

2.2.1 Weather Forecasts with Probabilistic Generative Model

This weather forecasts NLG task focuses on the method of learning the correspondences between an fruitful information in non-text structured format and a paragraph of text regarding to that information (Liang et al., 2009) (see Figure 2.2). The main obstacle is different ambiguities present in weather forecasts, including the utterance segmentation, the facts identification and the alignment inbetween utterances and facts. In the weather forecasts domain, the output text is the weather report and the input are various information about the weather features, like temperature, humidity, wind direction, raining possibility, in a structured format in chronological order.

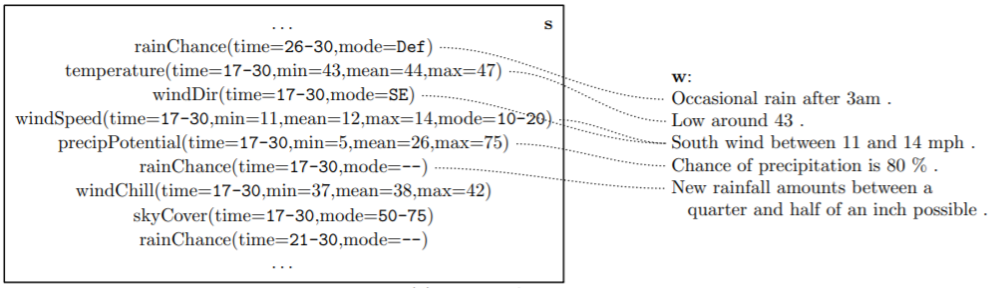


Figure 2.2: An NLG example in the weather forecasts domain includes a candidate set S on the left and a text W (Liang et al., 2009).

And one more realistic ambiguity regarding weather forecasts is similar to our drone assistant case, that is, many facts in input data is not referenced at all in the output utterances. These ambiguities and noise challenge the learning process seriously. In the paper, a probabilistic generative model is demonstrated to solve with the ambiguities (Liang et al., 2009). This probabilistic generative model works similar to HMM model to deal with word alignment (Vogel et al., 1996) and can be concluded as follows (see Equation 2.1 and Figure 2.3):

$$p(r, f, c, w|s) = p(r|s)p(f|r)p(c, w|r, f, s) \tag{2.1}$$

First, information r are picked from the set of all information s, then fields f are picked for each information. Finally, words w and their segmentation c are picked for each field (Liang et al., 2009).

In the related paper, the performance of this generative model in the weather forecast domain is not presented statistically. But the output examples shows that it is difficult to solve the ambiguity of alignment without additional machine learning methods. Moreover, the evaluation is based only on text segmentation instead of complete utterances.

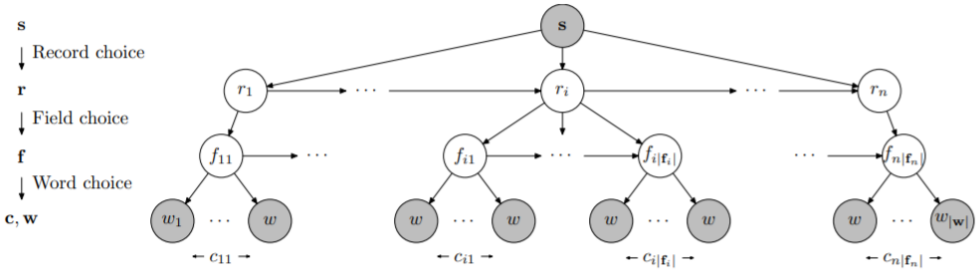


Figure 2.3: Graphic representation of the generative model (Liang et al., 2009).

2.2.2 Response Generation of Spoken Dialogue Systems

A high-performed NLG generator usually depends on various factors in different aspects: readability, fluency, variation, and adequacy (Stent et al., 2005). Before neural models are widely used in the NLG area, the most common approach is the templated-based approach (Mirkovic et al., 2011). However, the templated-based approach fails to generate outputs with variations, that is, unlike natural language which has different variations with the same content, the templated-based approach has identical output forms with the same content. Therefore, it's rather tedious and unnatural when this approach is applied to spoken dialogue systems. Furthermore, this approach doesn't easily scale to large open domain systems (Young et al., 2013; Gašic et al., 2014; Henderson et al., 2014).

To cope with these problems, deep learning methods such as the neural network model becomes popular to be applied to NLG tasks. In the former recurrent neural network (RNN) it is difficult to train with long-range dependencies because of the vanishing gradient problem. For improvement, the Long short-term memory (LSTM) neural network is proposed. By introducing memory block and multiplication gates in each block, LSTM can mitigate the disadvantage from RNN (Raffel et al., 2020). It is effective in various NLG tasks. The architecture of LSTM is illustrated in the upper part of Figure 2.4.

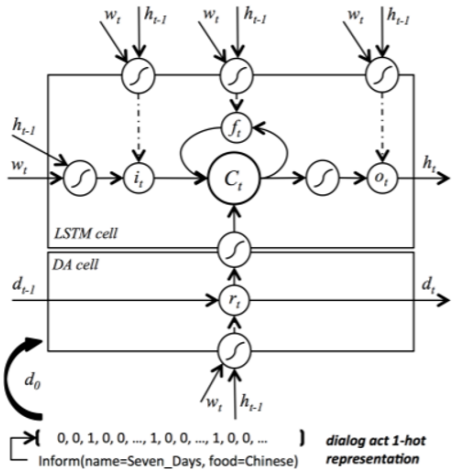


Figure 2.4: Visualization of a semantic controlled LSTM(SC-LSTM) cell (Wen et al., 2015).

In Wen's paper, a statistical NLG task based on a semantically controlled Long Short-term

Memory (SC-LSTM) recurrent network is presented for spoken dialogue systems (Wen et al., 2015) (see Figure 2.4). SC-LSTM is different from LSTM by adding a sigmoid control based dialogue act (DA) cell. The role of DA cell is to deal with the sentence planning problem, it acts on deciding what information should save for later steps and what information should discard. Consequently, jointly optimization from LSTM cell and DA cell can learn from unaligned data using cross-entropy training in both the sentence planning and the surface realization (Wen et al., 2015). The detail of LSTM will be discussed in Section 2.4.1.

As this research builds the NLG system in neural network architecture, the entire model is E2E trainable from data. As a result, the development life cycle from domain to domain is easy and quick. This system is evaluated in 2 different dialogue domain SF Restaurant and SF Hotel. With the training data of around 1k dialogues per domain and the average number of slots per DA for each domain being 2.25 and 1.95 respectively, this model achieved rather good performance on objective metrics BLEU (70%-80%) (Wen et al., 2015), as well as human evaluation. It seems promising, however, it should be pointed out the following two points: firstly, as a dialogue contains at least one turn of interaction between user and system, the number of 1k dialogues means that at least 2 thousand or even up to 3-5 thousand utterances serve as training data. Secondly, the average number of slots per DA in each domain is less than 3 (Wen et al., 2015), that is, the target references can be considered as short sentences conveying little information.

2.2.3 Sports Reports with End-to-end Neural Model

Compared to hand-engineered NLG systems, neural models have shown significant progress in generating short and descriptive texts regarding a small number of database records efficiently.

Traditionally, there are two respectively challenges in NLG, namely, the selection of the suitable subset of input data and the surface realization of output data (Reiter and Dale, 1997; Jurafsky and Martin, 2008). The wide application of neural networks to NLG which trained end-to-end blur distinction in between. In Wiseman's paper, a sports report NLG task is proposed and neural network models of the standard attention-based encoder-decoder model and its several extensions are applied to this data-to-document NLG task. The challenge of this sport report NLG task is considerable long text generation from structured data which stresses both content selection and surface realization (Wiseman et al., 2017). In the evaluation of this model (see Figure 2.5), it is shown that these neural encoder-decoder models perform well in generating fluent sentence outputs, but score quite poorly at the content selection and at capturing long-term structure. It indicates that an end-to-end neural model might be unsuitable for NLG tasks where content selection is in high demand. In our NLG task, it's obvious that handover warning messages should be short and convey vital information only. Therefore, content selection tends to be a challenge when applying a neural end-to-end neural method to it.

2.3 Recent Data-driven NLG Datasets and Related Methods

From the above-related NLG tasks, it shows that there is a tendency for NLG systems to be end-to-end and data-driven by applying a neural network model, which is showed having advantages and disadvantages. Compared to E2E seq2seq systems, hand-engineered

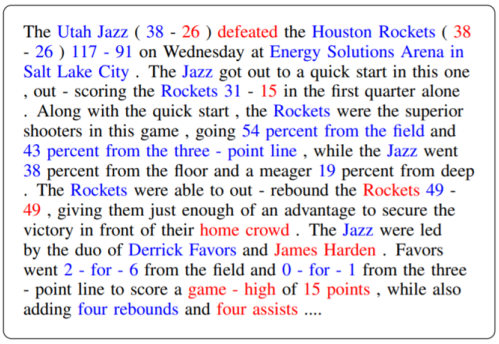


Figure 2.5: An example document generated by the sports reports NLG model. Correct texts are in blue and erroneous texts are in red (Wiseman et al., 2017).

systems perform better in generating more accurate and unified outputs. There is trade off between E2E data-driven and carefully hand-engineered NLG systems.

However, it is obvious that E2E purely neural models suffer from content problems, that is, facts omission or hallucination (Dušek et al., 2018). Therefore, we investigated more recent neural data-driven NLG datasets and related methods which focus on content issues.

2.3.1 Recent Data-driven NLG Datasets

Data-driven NLG datasets are designed for E2E NLG system, so they contain data records which inputs are pairs of meaning representations (MRs) and targets are the corresponding texts. They are in massive size and based on crowd-sourcing in order to eliminate the need of semantic alignments in E2E pipeline (Dušek et al., 2020).

In recent research, a data-collection pipeline manages to gather a large-size corpus of human to human conversations in rich Linguistic Content (Budzianowski et al., 2018) (around 7k dialogues) based on crowd-souring. Through this pipeline, the dataset MultiWOZ can break the constraints in linguistic variability and lacking multi-domain use cases from the previous dataset of speech-oriented dialogue systems. MultiWOZ, which is labeled with dialogue states and dialogue actions, can not only offer valuable training data but also help push forward research in E2E dialogue modeling on a large scale.

Other recent NLG datasets benchmarks are mostly focused on surface-level generation. For example, in E2E dataset (see Figure 2.6), it is a restaurant-domain dataset with 50k of data instances for training E2E, data-driven NLG systems which is 10 times bigger than the existing and frequently used one. This dataset is based on crowdsourcing techniques and the content selection part is done by crowdsource. With the crowdsourcing annotation, the target texts are more rich in vocabulary and various in syntax which results in more natural output utterances can be learned from this dataset (Novikova et al., 2017).

The other dataset, WiKiBio (Lebret et al., 2016), scales to the large and very diverse problem of generating biographies based on Wikipedia info-boxes, with over 700k samples. As this dataset focuses on generating the first sentence of a biography, a fixed-content selection from the Wikipedia info-boxes is needed. A neural language model performs this fixed-content selection.

Flat MR	NL reference
name[Loch Fyne], eatType[restaurant], food[French], priceRange[less than £20], familyFriendly[yes]	Loch Fyne is a family-friendly restaurant providing wine and cheese at a low cost. Loch Fyne is a French family friendly restaurant catering to a budget of below £20. Loch Fyne is a French restaurant with a family setting and perfect on the wallet.

Figure 2.6: An example of an E2E data instance (Novikova et al., 2017).

Last but not least, ToTTo is a large table-to-text dataset with 120k samples for training. It proposes a NLG task to produce one-sentence description from a given Wikipedia table with some highlighted cells (Parikh et al., 2020) (see Figure 2.7 as an example). Human annotators do the dataset in high quality and large quantity. However, when the dataset is trained by state-of-the-art neural network models, it demonstrates that these neural models struggle to generate reliable results due to the content selection problem.

Table Title: Gabriele Becker
Section Title: International Competitions
Table Description: None

Year	Competition	Venue	Position	Event	Notes
Representing Germany					
1992	World Junior Championships	Seoul, South Korea	10th (semis)	100 m	11.83
1993	European Junior Championships	San Sebastián, Spain	7th	100 m	11.74
			3rd	4x100 m relay	44.60
1994	World Junior Championships	Lisbon, Portugal	12th (semis)	100 m	11.66 (wind: +1.3 m/s)
			2nd	4x100 m relay	44.78
1995	World Championships	Gothenburg, Sweden	7th (q-finals)	100 m	11.54
			3rd	4x100 m relay	43.01

Original Text: After winning the German under-23 100 m title, she was selected to run at the 1995 World Championships in Athletics both individually and in the relay.
Text after Deletion: she at the 1995 World Championships in both individually and in the relay.
Text After Decontextualization: Gabriele Becker competed at the 1995 World Championships in both individually and in the relay.
Final Text: Gabriele Becker competed at the 1995 World Championships both individually and in the relay.

Figure 2.7: An example data instance from ToTTo dataset (Parikh et al., 2020).

As shown above, the state-of-the-art datasets served as benchmarks for data-driven, end-to-end NLG area is considerably large, and the target utterances are only one sentence. Consequently, in order to achieve relatively good performance for our NLG task, either a more powerful neural model should be applied, or more extensive data samples of good quality for training should be annotated.

2.3.2 Related Method on Content Fidelity Issue

In Tian's work, a confidence score is designed to detect facts hallucination in the NLG task. The confidence score is evaluated by both attention score and a language model as follows equation,

$$C_t(y_t) := A_t + (1 - A_t)P_B(y_t|y_{<t}) \tag{2.2}$$

here $A_t \in [0, 1]$ is the attention score which shows how much the model generates based on the input. A_t equals to 1 means output words are directly from the input. $P_B(y_t|y_{<t})$ is the probability of the language model, and it is high for templatic words which can be generated without support from input but low for words conveying exact information from input. As a result, a low confidence score means that output word is neither supported from the input nor has high probability from the language model (Tian et al., 2019).

In Figure 2.8, there is an example. Templatic words(e.g., is) do not need support from the input, therefore a high confidence score is gained. On the other hand, words that convey input information (e.g., Eric) have a low probability from the language model. In this case, their confidence scores are still high with a high attention scores. Words, like author and radio, get low confidence scores. It is consistent with the fact that these words are not in the input and can be seen as the potential hallucination. Then, a variational Bayes

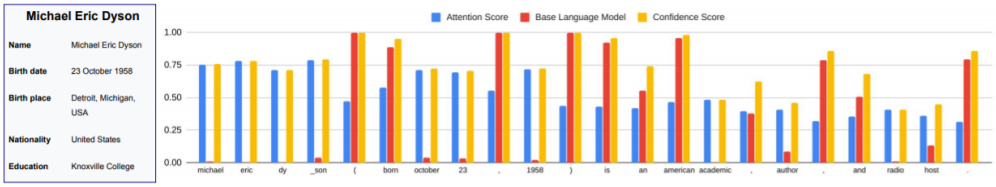


Figure 2.8: An example of the confidence score in Tian's work (Tian et al., 2019).

training framework is proposed to ensure generating output with high confidence in order to eliminate facts hallucination.

Experiments on the WiKiBio dataset demonstrate that this confidence score can reduce hallucination and get more faithful outputs.

Research on another new metric, PARENT (Dhingra et al., 2019) aims at evaluating content fidelity issue and it is shown that this metric performs well in WikiBio dataset based on a large-scale human evaluation.

2.4 Recent Neural Network Architecture in NLG

The current trend of applying neural network models for E2E NLG has been mentioned often. In this section, we will introduce the specific state-of-the-art neural network models for sequence data (e.g., text, music). Although more and more powerful and complicated neural network models have been developed in the past few decades, the first one introduced below can be regarded as the basic concept and fundamental for the other neural network models based on sequence data. This section will go through the evolution of sequence-data-based neural network models and the state-of-the-art neural network architecture planned to apply to this drone assistant dataset.

2.4.1 Basic Neural Network Models for Sequence Data

Recurrent Neural Network

Recurrent Neural Network(RNN) is the initial concept for neural network model architecture based on sequence data. Its concept has already been proposed in the late 20 century. However, because RNN has difficulty via training, researchers have proposed other adaptation neural network models, which is the modification based on RNN. In the following, the basic concept of RNN will be shown, and its disadvantage, which stops it from being widely used, will be discussed.

RNN can be seen as the adaptation of the feed-forward neural network in the sense of allowing it to model sequential data. In a basic feed-forward neural network, there are input, hidden and output layers. When training is performed, the data moves from layer to layer straightly without touching a node twice (Zeshan, 2019). As a result, the basic feed-forward neural networks can not remember the past prediction they made. Moreover, they are bad at predicting what is coming next. Without the concept of order in time, feed-forward neural networks can only consider the current input. Compared to only one input for the feed-forward neural network each time, RNN considers the current input and what it has learned from the previous inputs (see Figure 2.9).

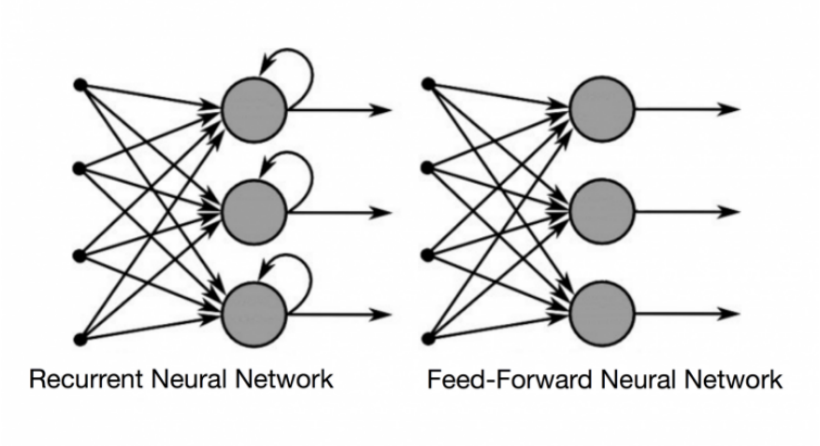


Figure 2.9: Comparison between RNN and feed-forward neural network (Zeshan, 2019).

Since sequential data can be regarded as ordered data, for example, DNA or financial data. Furthermore, the most popular type of sequential data is time-series data. In order to learn this kind of data structure, the concept of timestep is introduced in RNN. At each timestep t, the RNN gets x_t the input from that timestep, renews h_t its hidden state, and predicts o_t. The standard RNN can be formalized by iterating the following equations in time order (Sutskever et al., 2011):

for t = 1 to T

$$h_t = \tanh(W_{hx}x_t + W_{hh}h_{t-1} + b_h) \tag{2.3}$$

$$o_t = W_{oh}h_t + b_o \tag{2.4}$$

Here W_hx, W_{hh} and W_{oh} are the weight matrix from input to hidden layer, from the hidden layer to the hidden layer and from the hidden layer to output respectively. Vectors

b_h and b_o are the corresponding biases. A specific bias vector h_{init} is set at t = 1 when the expression $W_{hh}h_{t-1}$ is undefined.

By applying the timestep part, the output o_t can be regarded as going through the same hidden layers t-1 times which makes RNN have a high dimensional hidden state. Plus, with the nonlinear evolution by tanh function, RNN has great expressive power and rich dynamics over information integrated with many timesteps. It can be visualized as Figure 2.10.

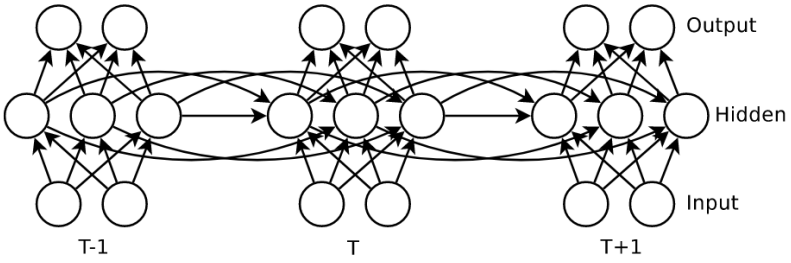


Figure 2.10: Visualization of RNN whose weights are shared across the time (Bengio et al., 1994).

Consequently, compared to a feed-forward neural network which has only one-to-one mapping from input to output, there are various mapping existing (see Figure 2.11) in RNN, including one to many, many to many (e.g. translation), and many to one (e.g. classifying a voice) (Karpathy, 2015).

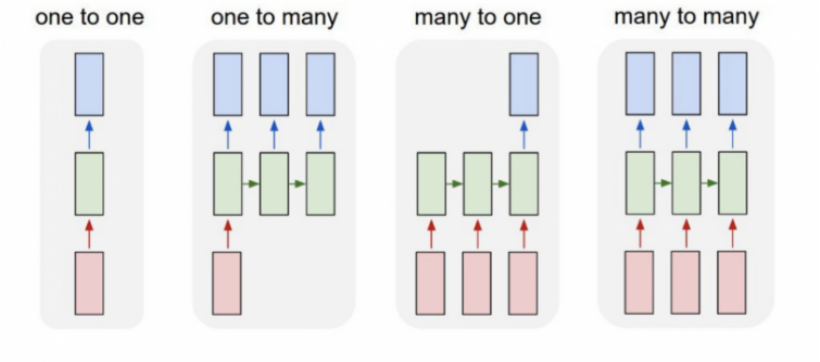


Figure 2.11: Mapping relationships for RNN (Karpathy, 2015).

As shown in the equations, the gradients of RNN can be easily computed via backpropagation through time. However, it does not make RNNs easy to train with gradient descent. In Bengio's (Bengio et al., 1994) research, they proved that the gradient decays exponentially as it is backpropagated through time, which is used to make the argument that RNNs can not learn long-term temporal dependencies. Moreover, the fact that backpropagated gradients exponentially blow-up dramatically increases the variance of gradients and instability of the learning process. As we all know, gradient descent has been used as the main algorithm for training neural networks till now. This disadvantage led the research of RNN slow down, and researchers came up with other adaptation and replacement models which share a similar concept to RNN.

Long-Short Term Memory

Long-Short Term Memory (LSTM) Neural Network is one of the successful adaptations of RNN to solve the gradient descent problem. The method to assign "weights" into the neural network model of LSTM makes it possible to either let new information in, forget information, or convey its importance to further generate the output. By this means, LSTM can learn long-term dependencies.

LSTMs were introduced by Hochreiter (Hochreiter and Schmidhuber, 1997) and were refined and popularized by other researchers. They are proven to work well on various tasks. The concept of cell state, which means the state of a repeating module in the model, is introduced to LSTMs. Instead of using only one tanh layer to control weights from previous input, LSTMs have three different gates to manipulate the cell state. It can be summarized as following equations:

$$f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f) \tag{2.5}$$
$$i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i) \tag{2.6}$$
$$o_t = \sigma(W_o * [h_{t-1}, x_t] + b_o) \tag{2.7}$$
$$\widetilde{(C)}_t = \tanh(W_C * [h_{t-1}, x_t] + b_C) \tag{2.8}$$
$$C_t = f_t * C_{t-1} + i_t * \widetilde{(C)}_t \tag{2.9}$$
$$h_t = o_t * \tanh(C_t) \tag{2.10}$$

Firstly, the "forget gate layer" f_t is defined taking h_{t-1} and x_t into account. f_t is then output as value between 0 and 1 and used in the second last equation for C_{t-1}. Obviously, 1 means "completely keep C_{t-1}" while 0 means "totally forget this" (Olah, 2015). Similar to that, the "input gate layer" and "output gate layer" can be computed as i_t and o_t. Then, the update value to the state, $\widetilde{(C)}_t$, can also be computed (Olah, 2015). With this update value and the other gate values, the new cell state value C_t can be further computed in the second last equation. Lastly, output value, h_t, is generated based on the previous new cell state value C_t and output gate value o_t.

The steps presented above are the repeating module in LSTMs. It can be visualized as the following Figure 2.12. By applying this complicated repeating module, LSTMs manage to consider the long-term dependency.

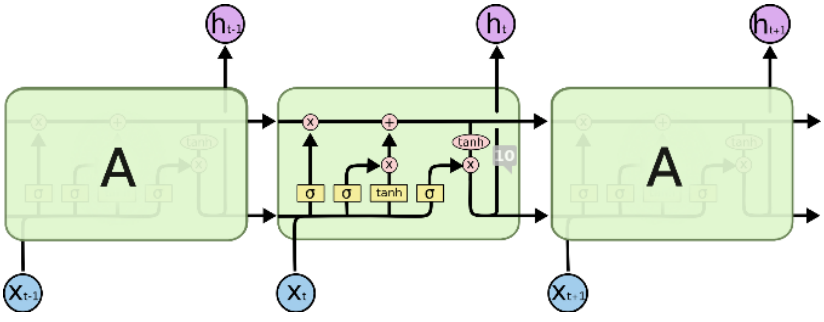


Figure 2.12: Visualization of the repeating module in LSTM (Olah, 2015).

2.4.2 T5 with Self-attention Transformer

The basic neural network architectures for sequence data are introduced in the section above. However, they are far from reaching a state-of-the-art neural network model to perform NLG tasks. The key to a practical NLG and NLP neural network model is to "understand" text. Namely, the model should be developed for general-purpose knowledge. This knowledge can range from the spelling or meaning of words to understand the whole sentence. The common approach uses word vectors to map word identities to a spatial representation. The mapping methodology is generalized so that words with similar meaning can map to nearby vectors in the vector space.

Based on what is mentioned above, it becomes more and more common to pre-train the neural model on various data-rich tasks. By this means, general-purpose abilities can be obtained by model itself which can then be transferred to other downstream tasks, that is, transfer learning. Unsupervised pre-training for natural language areas is promising because unlabeled text data is readily available (Raffel et al., 2020). As a result, more and more recent natural language work using transfer learning methodology has been proposed in different aspects like unlabeled data sets, benchmarks, fine-tuning methods, and more. As this techniques have developed rapidly, it is not easy to compare different algorithms in different aspects. Therefore, T5 is proposed. T5 takes the NLG task as a "text-to-text" problem, that is, using the text as input and producing new text as output (Raffel et al., 2020).

The text-to-text framework (see Figure 2.13) allows people to directly apply the same model, decoding step, objective, and training process to every different task directly (Raffel et al., 2020). With this unified approach, it is evident and practical to implement the NLG task in our case and compare different factors in the implementation pipeline.

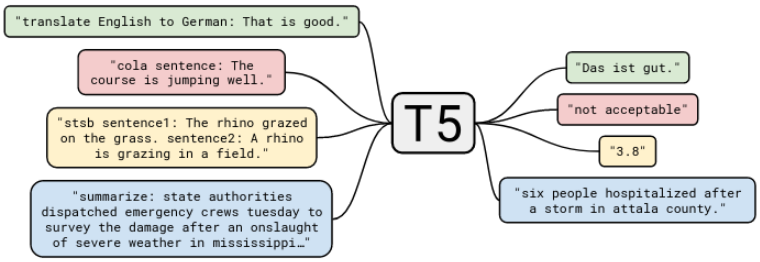


Figure 2.13: The text-to-text framework (Raffel et al., 2020).

As we know, T5 is a framework using transfer learning. However, transfer learning can be applied in different neural network models, from the primary, RNNs to the state-of-the-art commonly used models based on the "Transformer" architectures (see Figure 2.14) nowadays. The Transformer was initially shown to be effective in natural language area and has been used in various NLG tasks subsequently. Due to its effectiveness and ubiquity, T5 applies the Transformer architecture similar to what it is originally proposed (Raffel et al., 2020).

As is shown in the previous figure about the Transformer architecture, the main difference between other models and the Transformer is the self-attention block. Self-attention can be seen as a variant of attention. In the following, an example of how does self-attention compute is shown for better understanding.

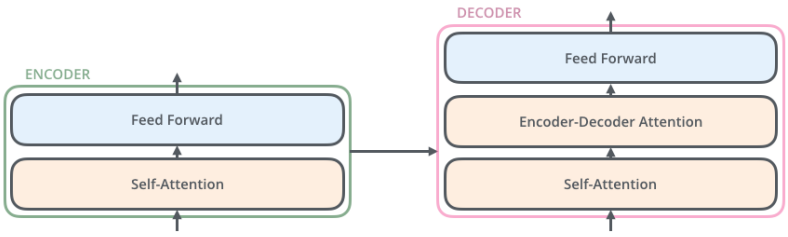


Figure 2.14: The transformer architecture (Raffel et al., 2020).

1. **Initialize Q, K, V**: The first step is to create Query, Key and Value vectors for each input words' embedding vector. It works as the following equations

$$Q = W_q * X \tag{2.11}$$
$$K = W_k * X \tag{2.12}$$
$$V = W_v * X \tag{2.13}$$

2. **Calculate self-attention score**: This step computes Self-attention score for each word. For example, if the computation is performed for the first word. Based on the first word, each word in the input needs to be scored. Thus, if it is to deal with the self-attention of the words in the first position and the input has two words, the computation can be seen as the "Score" in Figure 2.15.

3. **Scale and normalize the self-attention score**: In order to obtain the softmax score, scaling the scores computed in the last step and normalizing the result by softmax is performed in this step.

4. **Multiply and sum**: Each value vector is multiplied by the softmax score to obtain the weighted $v1$ and $v2$, and the weighted $v1$ and $v2$ are summed to obtain $z1$. Thus, the Attention Value for the first word is computed. The other words are computed similarly (see Figure 2.15).

In this way, Self-attention can capture some syntactic features between words in the same sequence, such as the phrase structure with a certain distance shown in Figure 2.16. It also captures some semantic features between words in the same sentence (e.g., "Law" is the referent object of "its" shown in Figure 2.17).

The introduction of Self-attention shows its ability to capture long-distance interdependent features in sentences, which RNN or LSTM can not. In other words, "Transformer" with the Self-attention mechanism can better "understand" the text than the previously mentioned models.

Overall, T5 with the encoder-decoder Transformer is powerful neural network architecture for NLG area.

2.5 Comparison to our NLG Task

In the following, we look back at the presented related work to find the advantages and disadvantages of recent NLG tasks, methods, and datasets and compare them to our

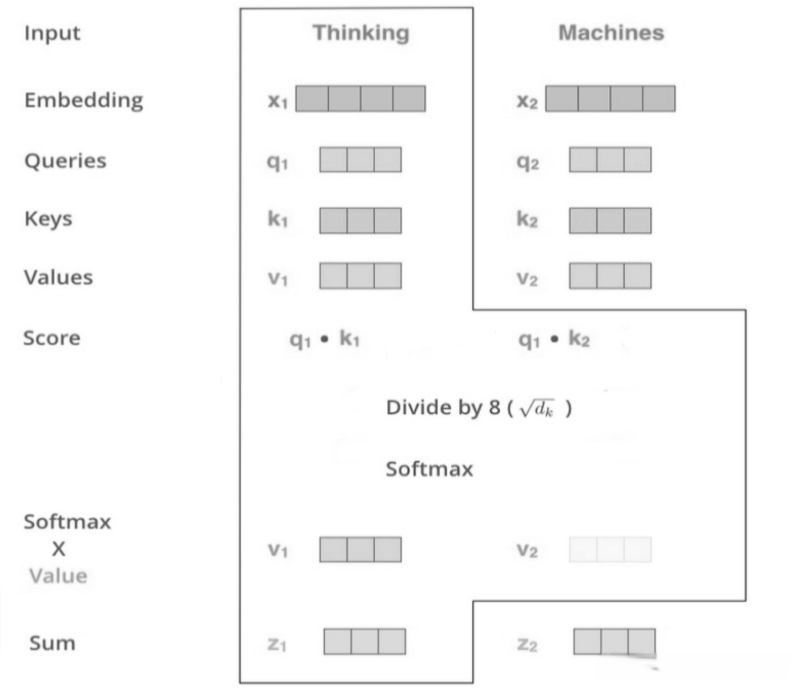


Figure 2.15: The self-attention computation example (Alammar, 2018).

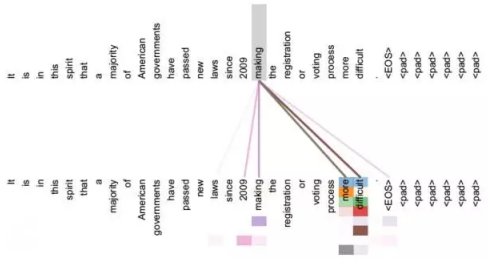


Figure 2.16: The self-attention example results in syntactic features (Alammar, 2018).

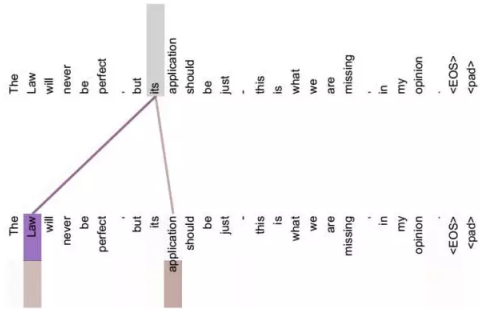


Figure 2.17: The second self-attention example result (Alammar, 2018).

NLG task to get some insights such that challenging points in our task can be addressed.

Firstly, we introduce the fundamental concept of NLG and dialogue systems in order to give insights into what is a NLG task and a specific scenario of NLG. And a typical activity cycle of the dialogue system is presented. In the NLG task, we focus on the output generator, using the NLG method to output utterances from structured table input data.

Secondly, we investigate some recent NLG tasks from structured data and its development. In the NLG task weather forecasts with probabilistic generative model example (Liang et al., 2009), the NLG method is a combination of learning the correspondences between a rich structured data and a stream of referred text as well as a probabilistic generative model. Due to different ambiguities from the facts to the text and the content selection problem of this weather forecast scenario, the performance of this method is only reasonable. Through this, we can get the idea that the previous method of probabilistic model plus supervised learning has limitation for general NLG problems. Nevertheless, it indicates the universal problem in NLG, that is, content fidelity. In other recent research on spoken dialogue system NLG task, neural models started to be applied as NLG method. It shows that by using the LSTM neural network in a unified E2E architecture, this NLG model can perform considerably well in variations and fluency and is easy to scale up and develop across domains. However, although the E2E neural model has many advantages, there is a trade-off between the traditional hand-engineered system and the E2E neural system. One more example is present in the sports report, which uses the standard attention-based encoder-decoder model when content selection is needed in this application. The performance demonstrates that the E2E neural model struggles in preserving content fidelity. With the above examples, we know that although the E2E neural model is a new trend for NLG because of its efficiency, a simple E2E neural model might not be enough in our NLG task, which aims to generate warning messages from sensory data in the drone domain. That's because there is logic from all sensory data to what should be output in the warning messages. If the neural model can not learn this logic well, another method should be added to our implementation to improve the performance.

Thirdly, we investigate more recent NLG methods which stress the above content fidelity problem and other existing data-driven NLG datasets. There is some research on neural models dealing with the content fidelity (omission or hallucination of facts). It uses a confidence score with a variational Bayes training framework. This model outperforms to generate much more faithful outputs than other purely neural models when applied to WiKiBio. It remains a problem to apply to our dataset. Facing the disadvantage of the E2E neural model and uncertainty of the confident score method towards our dataset, we propose that it will ease the difficulty of performing the content selection before the E2E neural model step as there is a certain logic and rules when annotating the utterance in our dataset. Moreover, we introduce several well-known NLG datasets used for research benchmarking. The E2E restaurant domain dataset focuses more on lexical richness ad syntactic variation with large size. For WiKiBio, its output aim is a simple and fixed-length first sentence of the biography. Namely, its content selection problem remains easy to solve. For ToTTo, it's a controlled table-to-text generation dataset in which content selection is random. These NLG datasets mentioned above are all considerably large, with at least 50k data samples, which suggests that either a large dataset should be annotated or a powerful neural architecture with a few-shot learning method should be applied in our NLG task.

Lastly, we outline the architecture of neural network models for sequence data from the basic model RNN, LSTM to state-of-the-art "Transformer" architecture which uses transfer learning and a self-attention mechanism so that the model can have a better

understanding of some syntactic and semantic features out of the sentences.

In contrast to other E2E data-driven NLG tasks and datasets, our drone corpus's content selection will be based on well-designed logic and rules from structured data records. And the T5 framework, along with three different methods for few-shot learning, is implemented in building up the neural model for our NLG task. To this end, we believe that our NLG task and the dataset are unique and different from the current NLG tasks and datasets. With our brand-new **DroneParrot** dataset, the implementation and evaluation will also be customized based on the characteristics of our dataset.

Chapter 3
Dataset Setup and Annotation

In this chapter, we will provide an overview of the dataset setup underlying our drone hand-over assistant corpus, as well as some details on implementation. Our empirical investigation consisted of three phases. First, we addressed our data record scheme from original video-based data. Second, we designed and described the scenarios and criticality types for the drone assistant warning message corpus. In the last phase, we described how we systematically annotate the warning utterances.

We describe how to perform the original data collection of our corpus in Section 3.1. In Section 3.2, we describe the logic and scenarios which are designed for annotating our original data. We present the annotation process applied to our dataset in Section 3.3.

3.1 Original Data Collection

As we mentioned before that our ultimate NLG task is to build a neural NLG model for generating warning messages in the drone hand-over assistant domain. The first step is to obtain the original data from scratch to meet the ultimate goal. The criteria for original data is that it should be in a real and hand-over needed situation in the drone domain. Consequently, we selected various clips from the drone flying recordings when there is a high probability of triggering a hand-over situation. In order to have a better understanding of which features should be taken into account in our data, we referred to a real drone's user interface. Figure 3.1[1] shows the user interface in DJI GO, which gives us insight into which features are essential in the drone domain, namely, which features should be annotated. After investigation, features captured frequently during drone flying are listed below.

- **WindSpeed(m/s)**: the speed of the wind in flying situation
- **DroneSpeed(m/s)**: the speed of the flying drone
- **PilotExperienced**: whether the pilot(user) is experienced or not

[1] `https://forum.dji.com/forum.php?mod=viewthread&tid=76304`

Figure 3.1: A snapshot of drone controller interface from DJI GO.

- **Altitude(m)**: the altitude of the flying drone
- **Temperature(Celcius)**: the temperature in flying situation
- **Distance from remote control(m)**: the distance between the flying drone and the controller
- **Battery level**: how many percentages of battery left
- **Low visibility**: whether the flying situation is in low visibility or not
- **Normal frame**: whether the frame of the drone is in normal status or not
- **Weather**: the weather status of the flying situation
- **Upside down**: the drone is flying upside down or not
- **Good motor condition**: whether the flying drone is in good motor condition or not
- **Going backward**: whether the drone is flying backward or not
- **Indoor**: whether the drone is flying indoors or not
- **Waterproof drone**: whether the flying drone is waterproof or not
- **Flying over**: where the drone is flying over

Besides, since the rapid development of drone tracking and detecting system, current drones can also capture obstacle information by sensors and algorithms. It is obvious that the obstacle is highly likely to trigger a hand-over situation. Consequently, in addition to the above flying status features, we also add obstacle features:

- **Name**: how the obstacle is entitled
- **Type**: what type does the obstacle belong to (human, building, tree)

- **Moving**: whether the obstacle is moving or not
- **InPath**: whether the obstacle is in the flying path or not
- **Distance(m)**: the distance between the flying drone and the obstacle

What is more, in reality, hand-over assistant for drones should be flexible from the environment to environment. For example, a drone flying against high wind speed consumes battery dramatically faster than one against average wind speed. Moreover, a drone flying outdoors means a higher probability of signal interference. In these cases, a battery capacity higher than usual will trigger a hand-over situation. Regarding the above, we picked eight common flying environments and grouped the flying recording clips by the environment. These drone clips are recorded from the perspective of drones, and they are genuinely from either real drone maneuvers or drone simulators. Figure 3.2 shows all the environments taken into consideration. Table 3.1 shows the details for our original data.

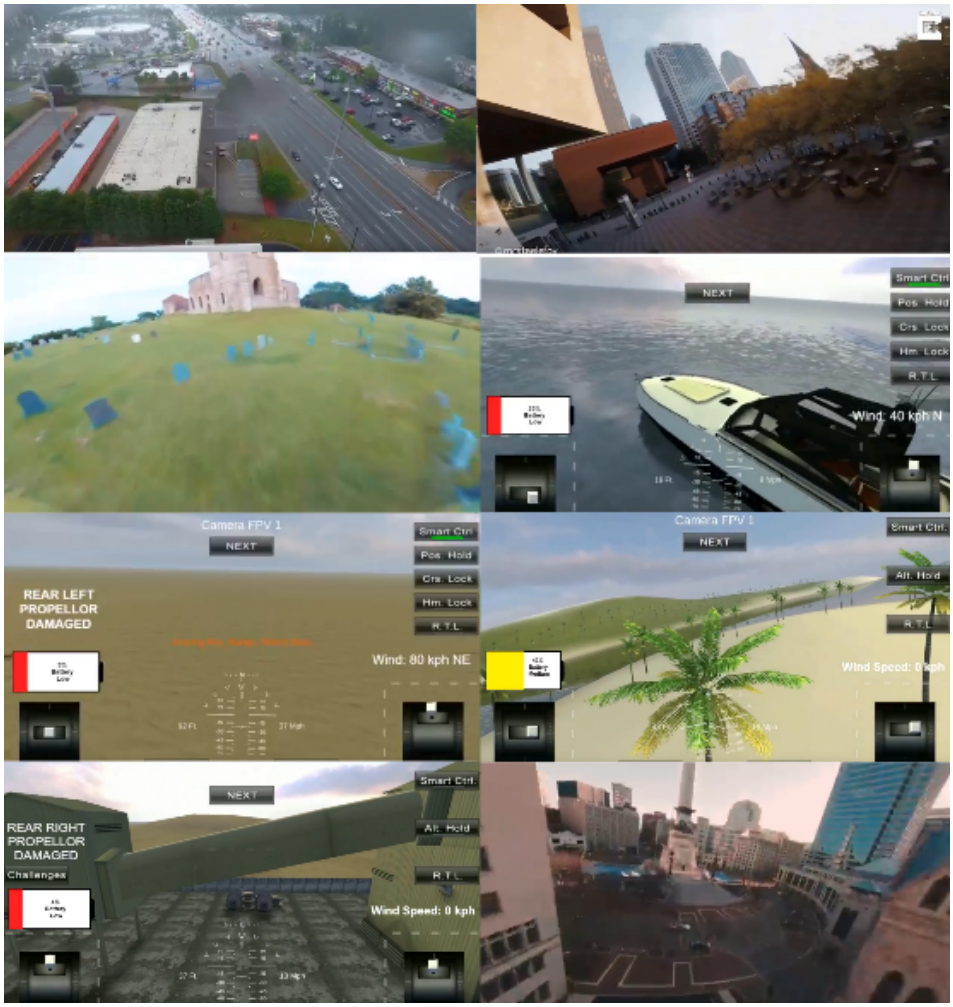


Figure 3.2: Snapshots from recording clips in Disturbance, Urban, Rural, Ocean, Desert, Island, Factory, and Miscellaneous environments, respectively, from left to right and top to bottom.

Environment	Number of clips
Disturbance	60
Urban	20
Rural	100
Ocean	24
Desert	22
Island	25
Factory	25
Miscellaneous	40
Total	316

Table 3.1: The number of video clips in each environment in the original data.

3.2 Logic of Category and Ranking for Annotation

In this section, we will discuss the design for the utterances category and the ranking logic applied in the further annotation.

The drone assistant is used in the hand-over situations during the drone is flying. Hand-over situations occur whenever there is an internal or external problem such that the drone cannot continue flying autonomously. Therefore, the drone assistant will send a warning message to inform and alert human pilots. This message will be shown in the user interface of the drone assistant or might be spoken up using other text-to-speech output renderers. Our corpus is a set of these warning messages in various flying situations and environments.

Due to the various sensory data collected by drones and various types of hand-over situations, designing the category and grading system to group the situations and rank them in different emergency levels is essential. Different types from the category that group the hand-over environments and correspond to the urgency level for control to be handed over to human drone pilots are designed and shown in Table 3.2. We believe this designed logic is crucial to the drone assistant in the following ways. Firstly, they groups and summarizes the situations into different types so that the users can understand the situation better and faster in the warning messages. Secondly, whenever many factors trigger a hand-over situation simultaneously, selecting which situations and factors should be mentioned is necessary. Moreover, output formats can be different from level to the level of criticality to arouse human pilots' attention.

Situation type	Emergency level	Description	Example Expression
RiskOfPhysicalDamage	1	Potential physical damage (e.g. crash) OR InPath: true Distance: 3 at 00:02	Altitude (m): 20 Battery_level: 30
RiskOfInternalDamage	2	Potential internal damage	weather: gloomy waterproof_drone: false
RiskOfHumanDamage	3	Risk of injuring nearby humans	indoor: true Distance: 0.5 Type: Human at 00:16
LostConnection	4	Drone connectivity/signal strength	Distance_from_remote_control (m): 162 Battery_level: 0

Table 3.2: Guideline for category and ranking (Chang et al., 2022).

For instance, when an object appears in the flying path (e.g., a building, a flock of birds), it needs to alert the human pilot of the imminent threat of the type **Risk of physical damage**. In contrast, When the battery is at a low level and the drone is far from the controller, the drone assistant also needs to inform the human pilot but less imminently with the type of **Lost connection**.

3.3 Annotation Process

In this section, we describe how we collect and annotate our corpus. As mentioned before, we collected 316 drone recording clips, around 10 seconds, respectively, from various hand-over situations and flying environments and split them into eight environments. At this stage, only video clips are available. In order to generate the corpus, a complete annotation procedure is designed and performed. It consists of 3 steps. First, it is the human annotation of tabular data as a simulation of sensory data. Second, Description Logic is performed to further select prioritized features from each data record in the first step. Lastly, the hand-over messages are labeled by native English speakers according to the prioritized features provided in the second step.

3.3.1 Annotation from Videos to Data

Based on the investigation in Section 3.1, the annotation from videos to tabular data is performed manually. This tabular data is annotated as the simulation of realistic and static sensory data records, consisting of **drone status** and **time step** records.

On the one hand, the **drone status** record indicates the device's internal information, such as whether it is a waterproof drone or not as well as other information of more permanence such as the weather and visibility when flying (see Table 3.3 as an example). As we know that internal drone status is not supposed to change dramatically over time, the **drone status** record is regarded as remaining the same for each 10-second video.

Wind_speed (m/s)	0
Drone_speed (m/s)	10
Pilot_experienced	FALSE
Altitude (m)	200
Temperature (Celcius)	5
Distance_from_remote_control (m)	16
Battery_level	70
Low_visibility	FALSE
Normal_frame	FALSE
weather	gloomy
upside_down	FALSE
good_motor_condition	TRUE
going_backwards	FALSE
indoor	TRUE
waterproof_drone	FALSE
flying_over	ground

Table 3.3: Sample of a drone status record that is part of a data record (Chang et al., 2022).

On the other hand, the **time step** record indicates the external object information, which is so close to the drone that it might trigger hand-over information. Because drones generally fly at high speed and low altitude, the object information changes instantly. Comparing to **drone status** record, the **time step** record are annotated at 1-second intervals (see Table 3.4 as an example).

Based on the videos and the scheme mentioned above, we simulate tabular drone data records as the supposed sensory data that a drone can capture. Figure 1.2 shows an example of the tabular data record. One data record consists of **several time step records** of nearby objects and **one separate drone status record**. While time step records report

name	castle
Type	Building
Moving	FALSE
InPath	True
Distance(m)	8

Table 3.4: Sample of a time step record that is part of a data record.

5 attributes of information about the surrounding objects and their detail at various time steps, the drone status record reporting 16 attributes is included in conveying other information of more permanence. Together, each data record, mapping to one video, consists of several time step records (up to 1-second interval) plus one drone status record. A snapshot data record is a unit of one drone status record and one time step record referring to the same time step as the snapshot.

3.3.2 Annotation with Description Logic

In a good hand-over assistant, the most crucial and essential situation should be conveyed to users cleanly and briefly. Besides the category and emergency level, which is designed in Table 3.2 above, a rigorous mechanism is set to define which combination of values of features will trigger a particular type of warning. In order to simulate more realistically, the mechanism is designed delicately.

On the one hand, it is difficult for annotators to annotate the utterance based on the original sensory data and the mechanism rules. On the other hand, as mentioned in the previous chapter, the content selection remains a challenge in neural E2E NLG tasks. A method called Description Logic is proposed in this step[2], in order to select the features which trigger the warning, so that annotator can annotate the utterances based on the selected features such that the content selection issue is solved somehow.

As mentioned in the setup of the previous scenario, four criticality types are designed, which are useful for classifying and ranking the situations to trigger the hand-over messages:

- **Risk Of Physical Damage**
- **Risk Of Internal Damage**
- **Risk Of Human Damage**
- **Lost Connection**

Combined with these four criticality types, the mechanism for defining hand-over situations is designed and shown in Appendix A.1

In description logic reasoning (Baader et al., 2007), the mechanism acts like an ontology containing background knowledge about drones and surrounding objects for its semantic interpretation. After the ontology and the data record are encoded as a description logic *Abox*, ontology-mediated queries are used to determine if there is a critical situation in the data, and specific features triggering the situation are selected out (Borgida et al., 2003; Bienvenu and Ortiz, 2015).

For example (see Figure 3.3), the ontology contains statements like

[2]This part is done by Alisa Kovtunova.

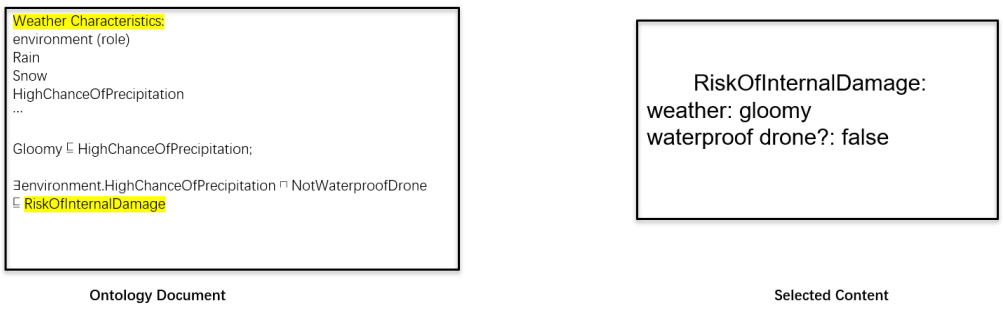


Figure 3.3: An example of how the query work based on the ontology.

- Gloomy $\sqsubseteq$ HighChanceOfPrecipitation
- $\exists$environment.HighChanceOfPrecipitation $\sqcap$ $\exists$NotWaterproofDrone $\sqsubseteq$ RiskOfInternalDamage

which characterize gloom as a high chance of precipitation weather and describe a critical risk of internal damage situation triggered by a not waterproof drone flying in gloomy weather. In this example, the query predicts **Risk Of Internal Damage**, which indicates an increased level of criticality.

Consequently, this annotation step with DL can be summarized as follows: Based on the sensory data records scheme and other studies about hand-over control in the drone domain, assumptions and rules for the hand-over mechanism are designed. Both data records and the mechanism act as the description logic ontology. With the ontology as well as the queries regarding 4 types of warning in Table 3.2, DL reasoning can automatically generate the type of criticality for every data record. As in our case, those parts of the input record which trigger the criticality are helpful for later processes. DL justifications (Horridge, 2011) are used to extract them. Then, this information is encoded into DL expression in the form of grounded DNF formulas.

3.3.3 Annotation of Natural Language Utterance

Given the collection of data records and their related snapshots in videos, we invited native English speakers to label each snapshot with a corresponding utterance representing the detected critical state(s). The types of criticality classify and determine the content of the utterance. For instance, the criticality type **Risk Of Physical Damage** informs about potential physical damage like a crash to other objects. In contrast, **Risk Of Internal Damage** informs about potential internal device damage like a non-waterproof drone flying in a rainy situation. Moreover, the criticality type is also important information for the pilot, so the criticality type is also labeled in the utterance. Furthermore, the second utterances set are annotated with more various syntax and vocabulary to provide the neural model with a more diverse training dataset to learn. **Ref-1** refers to utterance type 1 and **Ref-2** refers to utterance type 2.

In Figure 3.4, an example of a labeled utterance set for one data record is demonstrated. As we can see, the utterance can be varied in length because it depends on whether the snapshots trigger the hand-over situation or not. If it does, the utterance at that time step is added along with the time step referring to the exact time. Otherwise, no utterance

will be added up. What is more, the first utterance without time step indicates that the hand-over situation begins at (00:00) and is triggered by internal drone status only.

Utterance1	Utterance2
Risk of physical damage! The drone has a damaged frame and low battery. It is flying at an altitude of 20m. (0:02) Risk of physical damage! There are stairs in the drone's flight path at a distance of 3m. There's also a building only 2m away. (0:03) Risk of physical damage! Now the building is only 1m away. Now the stairs are only 1.5m away. (0:13) Risk of physical damage! Now the building is in the drone's flight path at a distance of 1m. (0:14) Risk of physical damage! Now the building is only 1m away.	Risk of physical damage! The drone's frame is damaged and the battery is running low. It's also flying with an altitude of 20m. (0:02) Risk of physical damage! There are stairs 3m away and in the drone's flight path, and a building only 2m away. (0:03) Risk of physical damage! The drone is now flying 1m away from the building and 1.5m from the stairs. (0:13) Risk of physical damage! Now the building in the flight path and only 1m away. (0:14) Risk of physical damage! The drone is flying 1m close to the building now.

Figure 3.4: An example of labeled utterance set for one data record. Utterance2 (Ref-2) is more diverse in syntax and vocabulary than utterance1 (Ref-1)

Consequently, each video contains the annotated sensory data like Figure 1.2, and the annotated utterances like Figure 3.4.

3.3.4 Further Selection and Refinement

As it is shown in Table 3.3, Table 3.4 and Figure 3.4, one data record contains information for the whole video which is last for around 10 seconds. To simulate the real use case of a hand-over system, we separate the data by second. Moreover, since the time step records are the obstacle information, they will be similar in syntax and content. We select the first time step record as well as the drone status record only. Consequently, one single data record contains one drone status record and one time step record (the first snapshot containing obstacle) after selection. Obviously, one selected data record is represented as the first two sentences in Figure 3.4 at maximum instead of all after selection. Figure 3.5 shows examples of the current dataset for further implementation.

Utterance1	Utterance2
Risk of physical damage! The drone has a damaged frame and low battery. It is flying at an altitude of 20m.	Risk of physical damage! The drone's frame is damaged and the battery is running low. It's also flying with an altitude of 20m.
(0:00) Risk of physical damage! There is a wall in the drone's flight path at a distance of 5m.	(0:00) Risk of physical damage! A wall is in the drone's flight path and only 5m away.
(0:01) Risk of physical damage! There's a wall in the drone's path at a distance of 5m. There's also a bridge only 1.5m away.	(0:01) Risk of physical damage! There's a wall in 5m in the flight path. The drone is also flying only 1.5m from a bridge.
Risk of physical damage! The drone has a damaged frame and empty battery. The drone is also flying indoors and at an altitude of 8m. (0:00) Risk of physical damage! There's a railing in the drone's flight path at a distance of 2m.	Risk of physical damage! The frame is damaged and the battery is empty. The drone is also flying at an altitude of 8m while indoors. (0:00) Risk of physical damage! There's a railing in the drone's path in 2m.

Figure 3.5: Four examples of labeled utterance per data record after selection in Ref-1 and Ref-2, respectively.

To this end, the annotation process is finished. Each data record from the dataset contains the reference utterance(s) as target output. Moreover, features triggering the reference utterance(s) are also selected as the input data. The input data are also converted to SGD dataset (Rastogi et al., 2020) format and flat MR format for further implementation.

Chapter 4
NLG System Design and Implementation

In the previous chapter, we discussed how we set up our corpus, from collecting data to annotating. As is shown in Table 3.1, 316 data records are generated within the limit of our labor. Nevertheless, since many of the steps are performed manually, it stops us from generating a large and diverse corpus with limited resources and labor. In order to scale up the size of our corpus and accelerate the NLG process, we try to extend our work by designing and implementing a neural NLG system to perform our NLG task automatically in this chapter. To this end, we will provide an overview of the design and implementation underlying the NLG system. Our NLG system consists of 2 components. In the first component, we performed linearization to our data record. Second, we implemented the Text-to-Text Transfer Transformer framework as the utterance generation rewriter.

We describe the complete NLG system architecture and pipeline in Section 4.1. In Section 4.2, We describe the first main component, the selection and linearization of data in the system. We present the other main component of the neural model implementation and perform the fine-tuning process in Section 4.3.

4.1 Pipeline of our NLG System

In this section, we will discuss the architecture of the whole neural NLG system for our NLG task.

Given the complete setup and annotation procedure in Chapter 3, it is obvious that the utterance annotation procedure costs most of the resources and labor and keeps it from scaling up to a large size. Consequently, that is where we want to replace it with a neural model. After investigating the annotation process and the characteristics in our corpus, we define the pipeline of our neural model.

Firstly, we start with linearization, that is, converting tabular data records to nicer formats

suitable for few-shot learning. Three different formats (schemes) are converted such that further comparison can be discussed. We don't replace the annotation from videos to data because the following reason: the annotation from videos to data can be regarded as the simulation of a realistic real-time drone hand-over situation, which means this tabular data can be obtained from real-time sensors instantly. Consequently, there is no need to annotate them manually in the real use case.

Secondly, we implement the rewriting part, namely, implementing the neural model training process using our data from the linearization process. Here, we introduce the unified Text-To-Text Transfer Transformer (T5) framework to perform this step. Because it is shown that it performs well with the linearization schemes for few-shot learning implemented in the linearization process, we can summarize these two steps as follows:

- **Data Linearization**
- **Rewriting with neural model**

Figure 4.1 shows a better picture of our system pipeline. While the upper one is the manual pipeline we discussed already in Chapter 3, the lower one is the one for our neural NLG system.

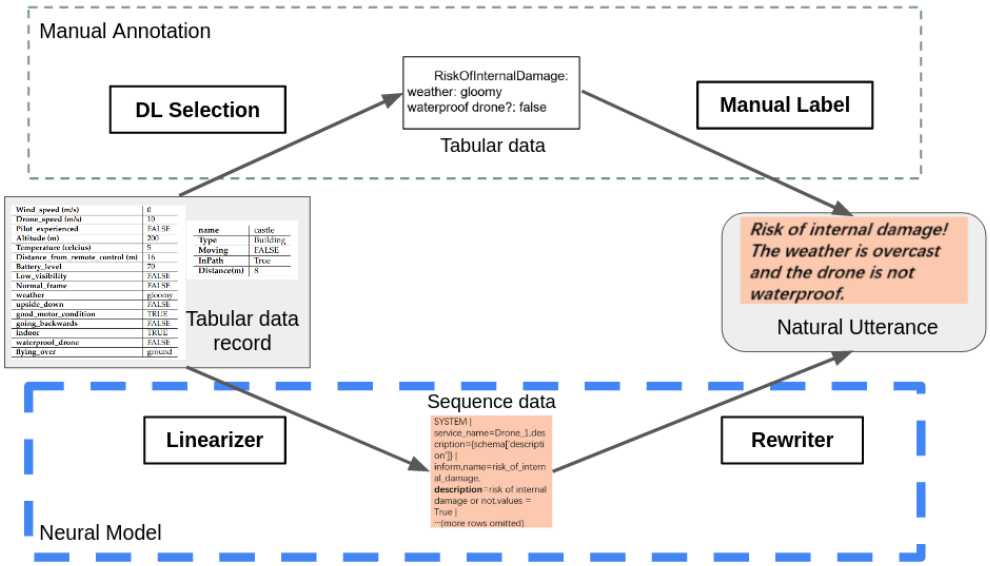


Figure 4.1: Our neural NLG system pipeline is in blue compared to the original pipeline in grey.

4.2 Data Linearization

To perform the linearization is to convert our tabular dataset into a certain scheme as input for the neural model. The most basic format is providing the data as Meaning Representations (MR) pairs similar to feature-value pairs, for example, the E2E restaurant dataset. However, it's not a suitable scheme for our dataset for two main reasons. On the one hand, the size of our dataset is quite small compared to the E2E restaurant dataset which contains 50k data instances. On the other hand, many features in our dataset

are Boolean values (indoor = true/false, waterproof drone = true/false, etc.) which make it more difficult to generate nice phrases from these Boolean-value MR. As a result, after investigating methods and scientific research in similar areas, the few-shot learning scheme method (Kale and Rastogi, 2020) for NLG tasks in virtual assistants, such as Alexa and Siri, is introduced into our linearization step.

The following subsection will demonstrate the reasons to convince that the method applied in the virtual assistant area is applicable in our area. Then, the details of the methods and implementation will be discussed.

4.2.1 Reasons on Adapting Methods from Virtual Assistant NLG Task

Traditional NLG systems of virtual assistants base heavily on templates to produce system utterances because the template can control the output utterance in a rigorous way and keep away from errors. However, with the increasing of multiple APIs in multiple domains in virtual assistants and the demand for high-quality user experiences, the templated method becomes difficult to scale to new domains and APIs as well as tedious to users. Consequently, data-driven approaches using neural network model have gained prominence in virtual assistant NLG task, and some research on it are also discussed in the previous relative work chapter. These systems need less effort and can generate more natural utterances with novel patterns (Kale and Rastogi, 2020). Based on the NLG background in the virtual assistant and our task, the reasons for adopting the following methods into our corpus to sequence our data can be concluded as follows:

- Firstly, as our corpus is for drone assistants aiming to convey the warning message to users in a natural language way, it's similar to the use of the virtual assistant to convey information to users also in a natural language way.
- Secondly, the target-generated utterances in both cases have the same features: short, neat, data divided as several features-values pairs representing as dialogue actions. The boolean value is acceptable as dialogue action in the existing virtual assistant NLG dataset.
- Thirdly, there is no doubt that drone assistants will involve more and more user interactions while developing. Namely, more and more multiple APIs will be added compared to the limited API in our corpus. So it is suitable to adopt a method that can be scalable and low maintenance. The first refers to easily supporting new APIs without requiring skilled developers, while the second refers to the seamless addition of new APIs without retraining.
- Finally, as our annotated corpus is rather small due to a lack of human labor, methods providing good performances in few-shot learning will be suitable for our NLG system. Regarding this, the following scheme methods show considerably good performances in few-shot learning in several datasets.

To this end, adopting the following scheme method to our drone assistant NLG system is considered promising.

4.2.2 Scheme Methods

Regarding the issue of scaling the data-driven generative model up to multiple APIs in multiple domains, Kale's paper addressed this issue as zero-shot and few-shot NLG

and proposed three scheme methods, two of which are new. The first method is similar to MR pairs but with structure and is called Naive representation. The second method, Schema-Guided NLG, represents each MR as a slot using its natural language description. The third method, Template Guided Text Generation (T2G2), utilizes template-based representations for each system actions (Kale and Rastogi, 2020).

As all of these scheme methods are proposed based on virtual assistant NLG area, and the main benchmarking dataset is SGD dataset (Rastogi et al., 2020) which uses dialogue action to represent MR, dialogue actions, corresponding to feature-value pairs mentioned above, are introduced to further explain each method.

Naive representation: This method is similar to basic MR pairs as it uses the most basic representation of each action $action_i$. e.g, $action_i(slot_i)$, $action_i(slot_i = value_i)$. These basic representations are concatenated as a sequence representation of utterance $\mathcal{A}$. This representation is simple to convert and is shown that it can give state-of-the-art good results for several well-known data-to-text benchmarks (Kale and Rastogi, 2020). However, it still requires a large corpus to convey the semantics of each action and to convey this structure as natural language text in the neural model training phrase discussed in the previous chapter. Two new methods are proposed based on the cons of the naive method.

Schema Guided NLG: This method is the extension of the Naive representation method by simply adding a natural language description to each action name. The action representations are $action_i$, $action_i(description(slot_i))$ and $action_i(description(slot_i) = value_i)$, where $description(slot)$ represents a natural language description of the action (Kale and Rastogi, 2020). This method is flexible as the descriptions which encode the semantics meaning of each action can be customized to get better performance. It's shown that this method can not only perform well in low-resource NLG tasks but also convey the semantics of the action. The descriptions we defined for each action in our NLG system are shown in Appendix A.2.

Template Guided Text Generation: Instead of only sequencing as action representations, this method directly converts the set of action outputs into a natural language utterance. The idea here is that, by initializing a minimal set of manually defined templates, simple utterances (action outputs) can be generated as the input to the pre-trained framework. In order to utilize this method, a converting template from action representations to action outputs is defined. The representation of $\mathcal{A}$ is now obtained by concatenating the corresponding action output of each action representation in $\mathcal{A}$ (Kale and Rastogi, 2020). Based on the designing rule for converting template of this method in the paper, converting template for our NLG system is defined as Table 4.1.

4.2.3 Linearization Implementation

In order to implement the scheme methods mentioned above with our dataset, the pipeline of data implementation in the paper is investigated. As the benchmarking dataset SDG (Rastogi et al., 2020) used by the paper is dialogue-action-based JSON format which is used in APIs protocols for a virtual assistant, conversion from our feature-value-based .xlsx format corpus into dialogue-action-based JSON format is performed. These JSON objects are similar to dialogues which are represented as a list of turns, and each turn contains a complete service (dialogue). Dialogue actions from each utterance are regarded as one frame, and frames from the same service are grouped into one turn. JSON objects grouped by environments (urban, rural, etc., eight environments in total) contain all dialogues from the same environment. Namely, each JSON object contains

Action Representation	Action Output
INFORM!!object_distance!!@	at a distance of @ m,
INFORM!!altitude!!@	at an altitude of @ m,
INFORM!!drone_speed!!@	at a speed of @ m/s,
INFORM!!object_name!!@	There is a @,
INFORM!!object_moving!!@	moving @,
INFORM!!object_ID!!@	none @,
INFORM!!object_type!!@	none @,
INFORM!!object_inpath!!@	@ in the drone's flight path,
INFORM!!object_timestamp!!@	none @,
INFORM!!wind_speed!!@	@ in strong wind,
INFORM!!pilot_experienced!!@	@ pilot is inexperienced,
INFORM!!temperature!!@	@ temperature setting,
INFORM!!distance_from_remote_control!!@	@m away from the remote control,
INFORM!!battery_level!!@	@ low battery,
INFORM!!low_visibility!!@	@ in a low visibility setting,
INFORM!!normal_frame!!@	@ has a damaged frame,
INFORM!!weather!!@	the weather is @,
INFORM!!upside_down!!@	@ is upside down,
INFORM!!good_motor_condition!!@	none @,
INFORM!!going_backwards!!@	is @ going backwards,
INFORM!!indoor!!@	is @ flying indoors,
INFORM!!waterproof_drone!!@	the drone is not @ waterproof,
INFORM!!flying_over!!@	flying over @,
INFORM!!lost_connection!!@	@ lost connection,
INFORM!!risk_of_internal_damage!!@	@ risk of internal damage,
INFORM!!risk_of_human_damage!!@	@ risk of human damage,
INFORM!!risk_of_physical_damage!!@	@ risk of physical damage,

Table 4.1: T2g2 scheme method converting template for our NLG system.

the following key information:

- **dialogue id** - the identifier for each dialogue
- **service** - the service present in the dialogue, here Drone is the only service
- **turns** - only one turn in each dialogue in our corpus

Our corpus contains only the warning messages from the drone system. There is no interaction utterance between the user and drone that is annotated yet, so only one utterance per turn. To this end, each turn consists of the following fields:

- **speaker** - the value is "SYSTEM" in our corpus
- **utterance** - only one utterance as warning message in each turn in our case
- **frame** - a list of annotated dialogue actions, only one frame in each turn in our case

In our case, the layout of each frame is much more clean and neat. Each frame consists of the following fields:

- **service** - service name corresponding to the frame, the value is Drone in our case
- **actions** - a list of actions from the system. Each action contains:
  - **act** : the type of action. Only INFORM action is presented in our case because the interaction is not considered in our corpus, and the warning messages aim to inform the user of some detail of the emergency situation
  - **canonical values** : the values in their canonicalized form, for example, $[True, False]$.
  - **slot** : a slot argument for the action
  - **values** : the value assigned to the slot, for example, $True, building$ etc.

Our corpus is rather small, and each flying environment has a major preference for warning messages. For example, the main warning message type in an urban environment is about obstacles, while in a rural environment, it is about connection. To deal with this, each flying environment is separated as a specific domain for further experiments. Consequently, there are eight folders related to 8 flying environments. Furthermore, each of these folders contains three sub-folders (train, dev, test) with one JSON object, respectively. These JSON objects are divided by the original environment-grouped JSON object, and the percentage are 80%, 10%, and 10%, respectively.

After the division, now the origin JSON objects are prepared. This origin JSON files, along with the structure template for **schema guided** method (JSON format) and template for **template guided** method (TSV format) for our drone system, which we defined shown in the previous Appendix A.2 and Table 4.1 are all needed to generate the final linearization sequence for each of the three methods above. Kale's paper (Kale and Rastogi, 2020) uploaded the code[3] for generating the sequences. Modifications are made based on their code to adapt to our case as the training is offline instead of using Google TPU, and the final file format used for further training is CSV instead of TSV in our case. Consequently, the new linearization dataset has the same original file structure. However, each sub-folder (train, test and evaluation) contains 3 CSV files regarding three methods (naive, schema, template) instead of one JSON object. Each CSV file contains two columns. The column "data" contains the representation sequences, while the column "utterance" contains the corresponding target utterances.

As above, the selection and linearization process is investigated and implemented. Table 4.2 shows an example of the representation sequences corresponding to one utterance in each scheme method after Description Logic (DL). Implementation is also done in the data without using DL, and the main difference is that all 21 dialogue actions (features) in all of these three methods for each data instance without using DL to select important features.

4.3 Neural Model Implementation

In this section, the implementation of the rewriting process for the dataset obtained in the last process is discussed. Firstly, as our model is based on the pre-trained t5-small model, there will be a short introduction to the T5 model, t5-small. Then comes the detail of the implementation.

[3]https://github.com/google-research/task-oriented-dialogue/tree/main/generation

Scheme Method	Representation Sequence
Naive	SYSTEM | service_name=Drone_1 | inform,risk_of_physical_damage, values = informative | inform,object_inpath, values = true | inform,object_distance, values = 5 | inform,object_name, values = obstruction
Schema Guided	SYSTEM | service_name=Drone_1,description={schema['description']} | inform,name=risk_of_physical_damage,description=risk of physical damage or not, values = informative | inform,name=object_inpath,description=Object is in path or not,examples =true,false, values = true | inform,name=object_distance,description=Distance of object, values = 5 | inform,name=object_name,description=Name of the object, values = obstruction
Template Guided	SYSTEM | informative risk of physical damage, true in the drone's flight path, at a distance of 5 m, There is a obstruction,
Actual Utterance	Risk of physical damage! There's an obstruction in the drone's flight path at a distance of 5m.

Table 4.2: An example shows the representation sequences by utilizing three scheme methods in data using Description Logic (Chang et al., 2022).

4.3.1 T5-small Model

T5-small is the smaller version of t5-base pre-trained model, which has fewer layers and parameters. In t5-small, the vocabulary size is by default 32128, encoder layers and pooler layer size is by default 512, the size of the intermediate feed-forward layer in each T5Block is by default 2048, the number of hidden layers in the Transformer encoder is 6, same as the number of hidden layers in the decoder. Moreover, its number of attention heads for each attention layer in the Transformer encoder is 8, its number of buckets for each attention layer is 32, its ratio for all dropout layers is 0.1. Its initializing factor for all weight matrices is 1 and the type of feed-forward layer is "relu".To this end, t5-small has around 60 million parameters (Raffel et al., 2020). Since t5-small is widely used in text summarization and translation, some prefixes, e.g., "translate English to German, " can be applied to identify different tasks.

4.3.2 Implementation

This section will present the detailed implementation to train our neural model and inference of the output sentences.

The pre-trained model t-5 small is from the open source AI community huggingface, which implements the t5 model in PyTorch. By this means, more freedom can be obtained for transfer training and fine-tuning as there are many interfaces and functions available in the huggingface library.

The major functions are implemented as follows:

The main function *t5trainer* passes all the parameters and performs other sub-functions calls, indicating that the input is various parameters and data. Its input includes the path to our dataset, column index for source and target utterance in our dataset, model output path, and list of model parameters. The model parameters list is designed and made up of pre-trained model types (t5-small, t5-base, or t5-large are officially provided in the huggingface t5 module), training batch size, validation batch size, training epochs, validation epochs, learning rate, max length of the source text, max length of the target text and seed for reproducibility.

With all these inputs, the following steps will be performed, and related functions will be called:

- **setup**: this step is performed in *t5trainer* itself to set random seeds, tokenizer based on the selected per-trained model, define the model by adding a language model layer on top of the selected pre-trained model

- **data preparation**: this step is to prepare our dataset for further training and validation, including encoding the original text-based dataset by calling *customizeddataset* class and defining torch data loaders for testing, validation, and training.
- **training**: this step is to define an optimizer and perform a training loop based on the number of training epochs by calling *trainingepoch* function per loop. During the loop, validation will be performed based on the validation epoch, and further information will be logged. Last but not least, the model will be saved after the training loop. The validation function is similar to *trainingepoch* function by setting the model to evaluation mode instead of training mode.
- **inference**: at last step, inference for test dataset will be carried out by calling *inference* function. By this means, output sentences will be generated and saved.

Algorithm 1 further outlines the implementation details on for *t5trainer*.

Algorithm 1 t5trainer

```
1: procedure SETUP
2:     set up random seed, tokenizer, pre-trained model, optimizer
3: procedure DATA PREPARATION
4:     for data in [trainset, valset, testset] do
5:         customset = customizeddata(data)
6:         loader = Dataloader(customset)
7: procedure MODEL TRAINING
8:     for 1, 2, ... number of training epochs do
9:         model = trainingepoch(list_of_parameters)
10:        evaluate on valset and log the loss on valset
11:        save model, tokenizer
12: procedure INFERENCE
13:    predicts, actuals = inference(list of parameters)
14:    save predicts, actuals
```

The sub-function *trainingepoch* mentioned above, with the input of tokenizer, model, dataloader and optimizer, is called for training one epoch by loading the training dataloader in the unit of batch size into the model and updating the model parameters. Details are shown in Algorithm 2.

The other sub-function *inference*, with the input of tokenizer, model and dataloader, is called to generate a predicted utterance from the test dataloader. The generation is to generate the torch tensor first with our trained model and then decode the tensor as a string using a tokenizer decoder. Its details are presented in Algorithm 3.

The class *customizeddata* is the class to reconstruct our dataset. As the original dataset is string-based, reconstruction is needed to get the tensor-based data with information on the data itself and its attention mask. This step can be done by calling the function in the tokenizer. The visualization of *customizeddata* class is shown in Figure 4.2.

To this end, the neural model training process is implemented. Based on our implementation of linearization and neural model training process, the pipeline of our NLG system is implemented completely.

Algorithm 2 trainingepoch

Input *tokenizer, model, dataloader ...*
$totaloss = 0$
set model to train
for data **in** dataloader **do**
    $input\_idx, attention\_mask = data[source\_ids], data[source\_mask]$
    $labels = ignore\_pad\_token\_id(data[target\_ids])$
    $output = model(input\_ids, attention\_mask, labels)$
    $totaloss+ = output[0]$
    set gradients of optimizer as 0
    compute gradients for the backward pass
    update parameters based on gradients
$totalloss = totalloss/len(loader)$
log important information

Algorithm 3 inference

Input *tokenizer, model, dataloader ...*
set *predictions, actuals* as lists
set model to eval
for data **in** dataloader **do**
    $y, ids, mask = data[target\_ids], data[source\_ids], data[source\_mask]$
    $generated\_ids = model\_generate(ids, mask, parameters\_for\_beam\_search)$
    $pred = tokenizer\_decode(generated\_ids)$
    $target = tokenizer\_decode(y)$
    append *pred, target* to *predictions, actuals*
return *predictions, actuals*

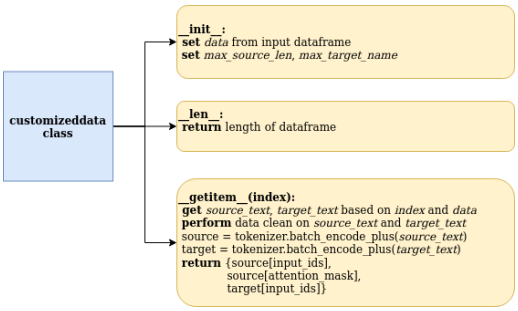


Figure 4.2: Architecture of class *customizeddata* implementation.

4.3.3 Fine-tuning

As the implementation for training (fine-tuning), our neural model is done in the above section. Because a new language model layer as an output layer is added on top of the pre-trained model, and our dataset is small and new. The fine-tuning step is performed in this section.

As shown previously, there are two variables in our dataset, using the Description Logic or not, and the reference type is Ref-1 or Ref-2. In this way, four datasets are gained. The

following fine-tuning experiments are performed in dataset **nodl1**, that is, with Ref-1 reference type (more uniform in syntax) and without Description Logic (all the DAs are presented without selection). That is because, in the pre-fine-tuning, the results among all four datasets were similar, and datasets with longer input (not using DL) met some edge cases. Solving this issue gives a deeper understanding of applying the fine-tuning in the t5 pre-trained model.

In Kale's paper (Kale and Rastogi, 2020), the training process and hyperparameters they applied for the SGD Dataset are a learning rate of $1e-3$ and batch size of 256. Also, the early-stop approach is applied in their training process. For inference, the beam search algorithm with a width of 4 and length penalty of 0.6 are applied. Based on their prior experience from this paper and huggingface T5 documentation several possible values of these hyperparameters are chosen, and the fine-tuning is performed as follows:

Learning Rate

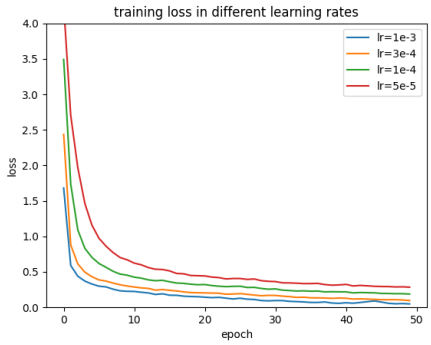
The learning rate is a hyperparameter that determines how much to change the model based on the estimated error whenever the model weights are updated. If the value of the learning rate is too small, a long training process may be needed, whereas a large value may result in an unstable training process which leads to convergence difficulty. In the transfer-learning area, since the pre-trained model contains various features from large datasets, the one that needs training is the last new-added output layer. Consequently, a low learning rate is expected in transfer learning. In the documentation of T5 in the huggingface community, the learning rates $1e-4$ and $3e-4$ are recommended, as they are proven to work well for most problems, including classification, summarization, translation, and question generation. In order to identify the suitable learning rate for our model. Experiments are carried out to compare the training loss based on different learning rates, $1e-3$, $3e-4$, $1e-4$, and $5e-5$, on our three different scheme methods.

As shown in the three subfigures in Figure 4.3 about the training loss based on the epoch increasing in different learning rates regarding different scheme methods, all four learning rate values can lead to convergence stably. For the largest learning rate $1e-3$, it can provide faster convergence compared to other values. To this end, the largest learning rate $1e-3$ will be chosen as the one for our model.

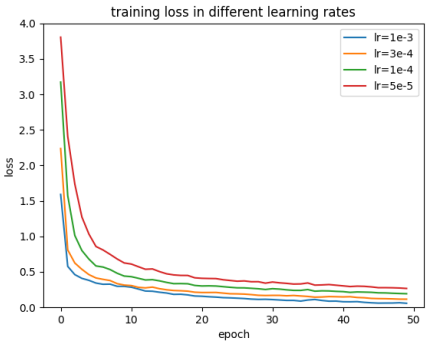
Batch Size

Batch size is also a significant hyperparameter for the training process because suitable batch size can lead to fast and smooth convergence. There is always a trade-off for choosing a suitable batch size. A large batch size is more efficient and faster in training speed. Moreover, in general, the larger the batch size is, the more accurate the descent direction is, and the less training oscillation is caused. However, the batch size should not be increased blindly. A large batch size value will make the memory capacity insufficient and reduce the number of iterations required for each epoch. Thus the update of the model parameters will be slow. Finally, when it is increased to a certain level, the determined direction of descent will not change anymore. Due to the considerably small size of our corpus, the optimal batch size for training from prior experience is full batch learning, that is, all training data in one batch and one batch per epoch. Nevertheless, as the batch size of 256 is used in Kale's paper for the SDG dataset, it can show that a rather large batch size value can be considered.

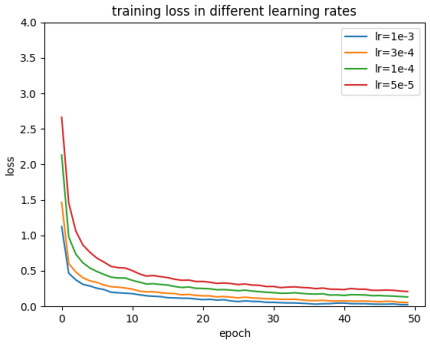
However, as mentioned before, a large batch size can lead to insufficient processor



(a) Naive



(b) Schema guided



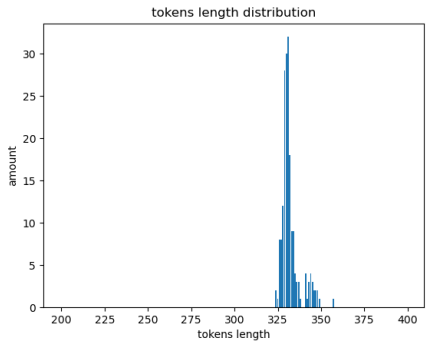
(c) Template guided

Figure 4.3: Training loss based on training epochs in different learning rates.

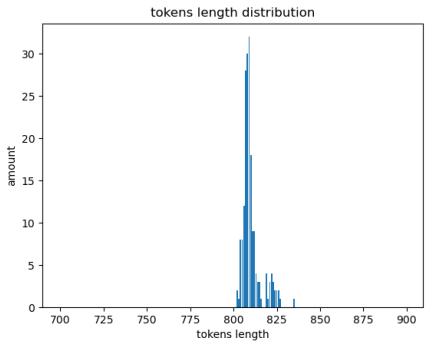
memory capacity. This factor becomes significant for the pre-trained model, t5, used in our case because of the following. Firstly, there is no limit on the max input tokens length for t5 because relative position embedding is used in t5. Secondly, the mechanism of relative position defines the amount of position embedding equal to the amount of the **input tokens length**, so doubling the input tokens length leads to a quadruple of the memory requirements. With our computation resource, the largest batch size for the max input length of 512 is 24. In the previous experiments on learning rate, the initial setting of batch size is 8, and the max input length is 1080, which ensures that the input data will not be truncated. As shown in the previous training loss vs. learning rate figures, our initial setting can already result in a stable and considerable fast convergence.

In order to further optimize the batch size, the distributions of the number of input tokens length regarding three different scheme methods in **nodl1** dataset are reported in Figure 4.4. The statistics are obtained by using the tokenizer from t5-small pre-trained model. In the figure, all three scheme methods centralize their distribution in a small range, **naive** is around 325 to 375, **schema guided** is around 800 to 850 and **template guided** is around 125 to 200. Consequently, **schema guided** is the one that can not be tokenized to a max length of 512. To this end, the values of batch size 24, 16, 8, 4, are compared with the setting of max input length of 512 for **naive** and **template guided**, while the values of batch size 8, 4, 2 are compared with the setting of max input length of 1024 for **schema guided**.

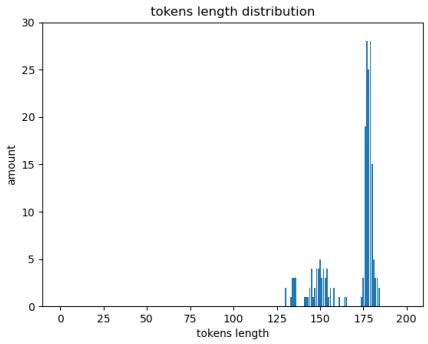
As shown in Figure 4.5 about the training loss based on the epoch increasing in different



(a) Naive



(b) Schema guided



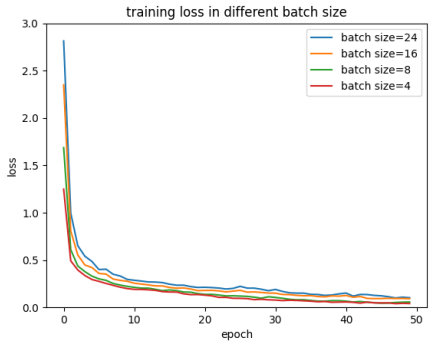
(c) Template guided

Figure 4.4: Number of tokens length distribution among different scheme methods in **nodl1** dataset.

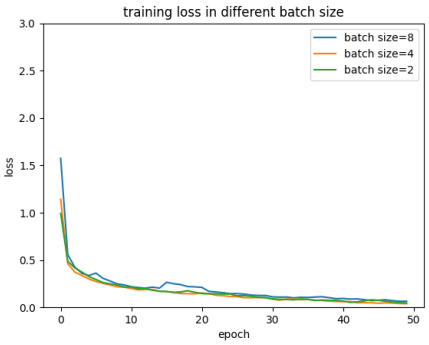
values of batch size regarding different scheme methods, all values of batch size can lead to convergence when the number of training epochs increases. For the scheme methods **naive** and **template guided**, training with a larger batch size has more unstable convergence and needs more epochs for convergence. That is, a smaller batch size fits model training in these two scheme methods. However, on the one hand, the plot with a batch size value of 8 nearly overlaps with the one with a batch size value of 4. On the other hand, training with a batch size value of 4 is much slower than 8. The batch size value of 8 is picked for these two scheme methods. For the scheme method **schema guided**, all three plots overlap mostly, and the plot with the largest value of batch size has more fluctuation. Consequently, a larger batch size with smoother convergence can be chosen for faster computation. That is, the batch size value of 4 can be chosen.

Beam Width and Length Penalty for Inference

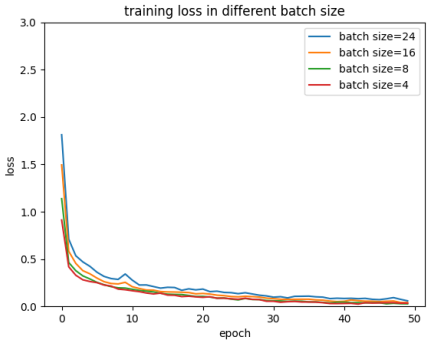
After training, parameters in inference are also needed to be fine-tuned. The inference is typical for natural language generation task, which predicts a sequence of words. The basic idea of inference is to output the probability distribution over each word in the vocabulary for words in the output sequence one by one and to transform the probabilities into a final sequence of words. As the vocabulary of the pre-trained model is often hundreds of thousands of words or even more, it is obvious that the search



(a) Naive



(b) Schema guided



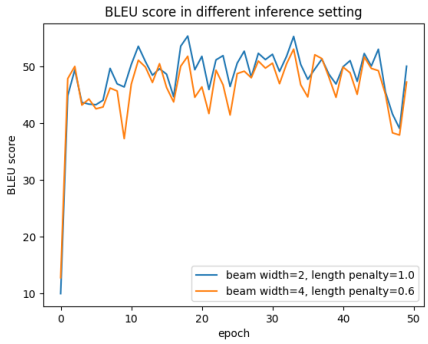
(c) Template guided

Figure 4.5: Training loss based on training epochs in different batch sizes.

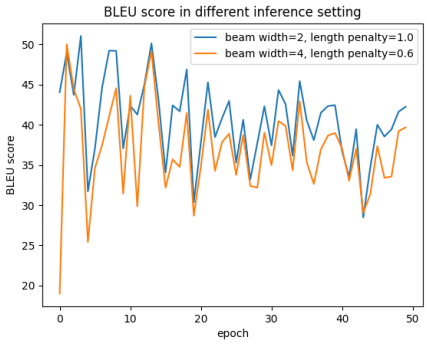
problem is exponential based on the output sequence length and is intractable to search completely. In the t5 pre-trained model used in our task, the inference heuristic is beam search which begins with k randomly generated states and keeps track of the k best states for each step. The main parameters of beam search are **beam width**, the k states which are kept track, and the **length penalty**, that is, the regularization parameter to solve the length bias issue among the different lengths of output sequences. In Kale's paper, the beam width 4 and the length penalty 0.6 are picked. And in other NLG tasks using t5 small as a pre-trained model, the beam width 2 and the length penalty 1 are picked. These two combinations are compared in our case to pick the suitable one for different scheme methods on their model after using the suitable learning rate, batch size, and 20 epochs for training.

As shown in Figure 4.6 about the BLEU score based on the epoch increasing in different inference settings regarding different scheme methods, it's evident that the blue setting (beam width = 2, length penalty = 1.0) performs better than the orange setting (beam width = 4, length penalty = 0.6) in all three scheme methods. Eventually, the blue setting for inference is chosen.

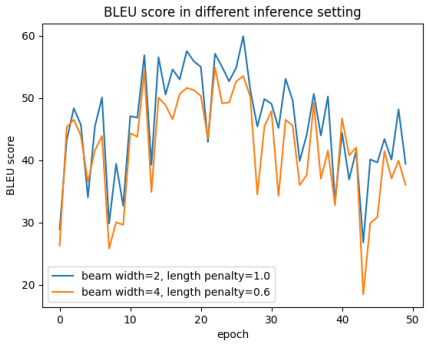
To this end, suitable hyperparameters are chosen. Nevertheless, the above figures all show that training for 20 epochs is already sufficient for convergence. For ease of comparison, 20-epoch updates for all training are performed.



(a) Naive



(b) Schema guided



(c) Template guided

Figure 4.6: BLEU score based on training epochs in different inference settings.

Chapter 5
Experiments and Results

In the previous chapter, our neural NLG system is implemented and fine-tuned with hyperparameter tuning. As shown before, three different scheme methods for few-shot learning and two different annotation utterance types, along with content selection with Description Logic, act as variables in our NLG system. In this chapter, more experiments and comparisons will be carried out to evaluate the performance of our NLG system. The evaluation will be presented in the following aspects: (A) Benchmarking. (B) Domain Generalization and Few-shot Learning Ability. (C) Robustness against Input Noise.

In Section 5.1, benchmarking will be performed for our NLG system. Since our corpus is brand new, implementation and details for two other NLG models acting as baselines, as well as evaluation metrics, will be presented. We further analyze the domain generalization and few-shot learning capability for our NLG system in Section 5.2. Lastly, the robustness of our NLG system to input noise will be analyzed by introducing various perturbation methods.

5.1 Benchmarking

This section will be performed among different state-of-the-art neural models and our model using data from our corpus. Before the result is presented and discussed, architecture of the baseline neural models will be introduced briefly.

5.1.1 Architecture of Baseline Models

Seq2Seq Model

The first neural model implemented is a Seq2Seq model with encoder-decoder architecture (Bahdanau et al., 2015). The tokenizer applied for this model is $en\_core\_web\_sm$ from SpaCy, which is primarily sufficient for standard use cases. The inference phase is simply taking the output with maximum probability each time. Adding the attention mechanism, packed padded sequences, and masking for improvement, the Seq2Seq

model architecture is as follows:

- **Encoder**: it is a single-layer GRU with a bidirectional RNN.
- **Decoder**: the decoder has an attention layer and a single layer GRU with a single direction RNN. Its input, which includes a weighted source vector from the attention vector, the previous decoder hidden state and embedded input word, are passed to the decoder RNN to get the predictions word by word.
- **Attention layer**: it is added to the decoder to transfer information from encoder to decoder. This layer will take the input from the previous hidden state of the decoder and all of the stacked forward and backward hidden states from the encoder. Its output is an attention vector after softmax with the same length as the source sentence. That is, each element is between 0 to 1, and its sum is 1.

Figure 5.1 shows the visual architecture for this Seq2Seq model. As it is the initial state for the decoder in the figure, the input for the attention layer (box in light purple and named "a") is z by default, and it will turn to s_1 for the next state. Figure 5.2 shows more architecture details for it.

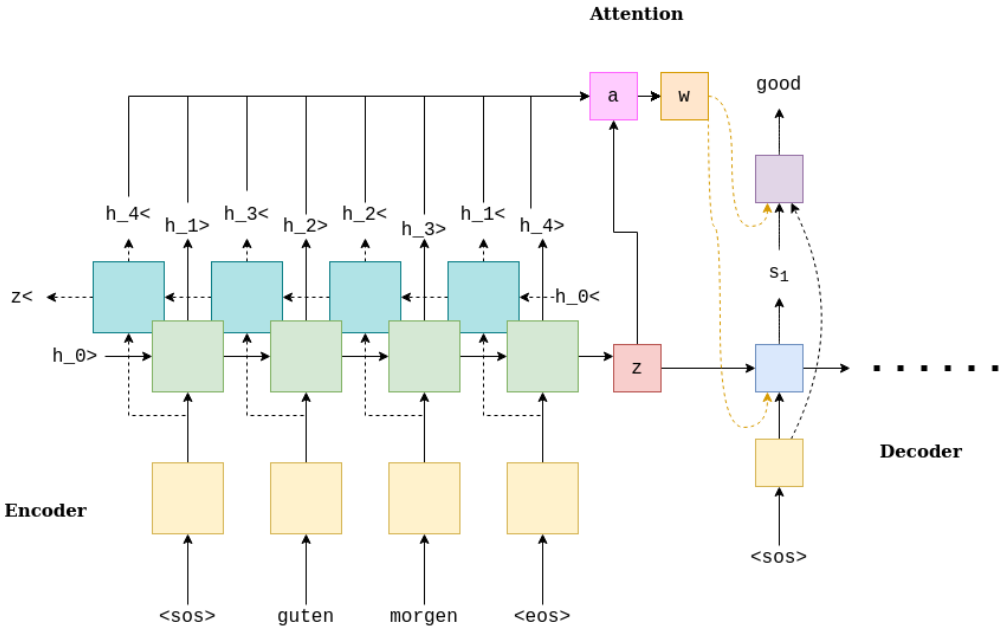


Figure 5.1: Visual architecture of the baseline Seq2Seq model.

Based on the open source resource[4], the implementation is made up of converting our dataset to MR pairs, training the model for reasonable epochs, and generating output utterances from the test dataset.

TGen

The other baseline model is TGen, a statistical natural language generator specially for spoken dialogue systems. TGen is much more like a complete and more complicated

[4] https://github.com/bentrevett/pytorch-seq2seq

```
Seq2Seq(
 (encoder): Encoder(
        (embedding): Embedding(7853, 256)
        (rnn): GRU(256, 512, bidirectional=True)
        (fc): Linear(in_features=1024, out_features=512, bias=True)
        (dropout): Dropout(p=0.5, inplace=False)
 )
 (decoder): Decoder(
        (attention): Attention(
        (attn): Linear(in_features=1536, out_features=512, bias=True)
        (v): Linear(in_features=512, out_features=1, bias=False)
        )
        (embedding): Embedding(5893, 256)
        (rnn): GRU(1280, 512)
        (fc_out): Linear(in_features=1792, out_features=5893, bias=True)
        (dropout): Dropout(p=0.5, inplace=False)
 )
)
```

Figure 5.2: Architecture details of the baseline Seq2Seq model.

NLG generator, namely a NLG system, than the previous Seq2Seq model. It consists of two different algorithms a statistical sentence planner based on A*-style search and a Seq2Seq RNN architecture (Dušek and Jurčíček, 2016).

It has been demonstrated in Dusek's paper (Dušek and Jurčíček, 2016) that TGen can be trained to produce correct and natural output sentences on different benchmark datasets in spoken dialogue system domain. Based on the code resource[5] which is already well implemented for the TGen generator, our implementation is converting our dataset to the suitable DA-pairs format so that it can fit into TGen's training process.

5.1.2 Evaluation Metrics

BLEU is the most common evaluation metric in NLG, which attempts to match variable length (normally from 1 to 4) phrases between output and reference utterances. To determine the final score, averaging scores from different length is performed. Consequently, the higher the **BLEU** score is, the higher probability of co-occurrence phrases between output and reference, and the closer this output utterance is to its reference. The other metric, **NIST**, is the improved version of **BLEU** by rewarding the output of infrequently used words. **METEOR**, the metric for Evaluation of Translation with Explicit Ordering, is another improvement based on the drawbacks of **BLEU**. It considers the recall issue in previous **BLEU** and measures sequence order of n-grams with word alignment method. The next one **ROUGE-L**, Recall-Oriented Understudy for Gisting Evaluation. is f-measure statistic based on the co-occurrence accuracy and recall of the longest common sub-sequence. The higher the **ROUGE-L** score is, the longer common subsequence exists between output and reference. **CIDEr** is a combination of **BLEU** and vector space model. It considers each sentence as a document and then computes the TF-IDF (each term is an n-gram instead of a word). Taking this TF-IDF as a vector, the similarity between the candidate and reference sentences can be further computed. By exploiting the TF-IDF algorithm, different n-grams can have suitable weights considering their length and frequency.

Besides common metrics mentioned above, a more novel, transfer learning-based automatic metric, **BLEURT**, is also applied in our evaluation. Because common automation metrics compute the exact word similarities only, **BLEURT** is introduced as an improvement. It can capture higher level of semantic similarities between input and reference by

[5] https://github.com/UFAL-DSG/tgen

being trained on a public collection of ratings as well as additional customized ratings (Sellam et al., 2020). Figure 5.3 shows an example that **BLEURT** captures NLG quality beyond surface overlap. In this example, it shows that **BLEURT** can evaluate the real semantic meaning instead of simply word-level overlap between input and reference.

Input: Bud Powell était un pianiste de légende. **Reference:** Bud Powell was a legendary pianist.	**sentence BLEU** *(0-100)*	**BLEURT**
Candidate 1: Bud Powell was a legendary pianist.	100	*1.01*
Candidate 2: Bud Powell was a historic piano player.	46.7	*0.71*
Candidate 3: Bud Powell was a New Yorker.	54.1	*-1.49*

Figure 5.3: An example of BLEURT captures semantic similarities beyond surface overlap (Sellam et al., 2020).

Moreover, based on our dialogue-action structured dataset, **SER** (Slot Error Rate) is introduced into our evaluation metrics. From the prior work (Kale and Rastogi, 2020; Makhoul et al., 1999), its concept is simply the ratio of the number of slot errors divided by the number of slots in the reference. The lower the **SER** score, the more the percentage of correctly copied slots from the structured data. **SER** is also implemented as an automatic metric, but this automatic implementation can not deal with binary slots or paraphrases of slots. Because a certain amount of slots are in binary value in our case, **SER** is taken as human evaluation. Here is the definition of our **SER** metric after adjustment:

$$SER = 1 - \frac{TA + 0.5H - 0.5R}{RE} \tag{5.1}$$

Where RE is the total number of slots in reference, TA is the number of correct slots in target, H is the number of semi-correct slots (e.g., slot name is mentioned but the corresponding value is wrong) in target and R is the number of redundant slots in the target. Similar to the original **SER**, the lower the score, the better it performs.

As **SER** is a human evaluation metric in our case, due to the limitation of labor, it will be performed on 10% of the test data, and its final score is determined by averaging the results.

5.1.3 Performance

In this section, the performance of different models will be analyzed to benchmark our NLG system with other state-of-the-art NLG methods. As mentioned above, there are three variables for our system, using DL or not, type of scheme methods, and type of reference. Table 5.1 and table 5.2 show the models performance in different evaluation metrics in 2 types of reference respectively. Figure 5.4 and Figure 5.5 show some examples of the generated utterances respectively. All the score is determined by averaging three times of experiments with different random seeds.

According to scores from various evaluation metrics in these two tables, it can be concluded that: (A) No matter in which type of reference, the set of settings in **t5+DL** can outperform the other settings by a large extent. **T5+DL+sc** performs the best in Ref-1

	Reference Type1 (Ref-1)						
	BLEU	**BLEURT**	**SER**	**NIST**	**METEOR**	**ROUGE_L**	**CIDEr**
Seq2Seq+all	41.74	42.78	66.17	3.19	38.25	59.32	0.37
Seq2Seq+DL	33.31	37.77	72.95	2.65	29.40	48.74	0.17
TGen+all	31.80	37.87	79.48	3.02	27.78	52.06	0.16
TGen+DL	30.91	38.94	66.41	3.13	29.06	54.92	0.17
t5+all+na	52.79	67.74	27.00	4.32	34.33	64.31	2.47
t5+all+sc	40.60	56.96	50.00	2.38	25.88	45.74	0.75
t5+all+tg	55.48	69.54	28.69	4.94	37.31	64.20	2.22
t5+DL+na	63.33	74.94	**11.11**	5.51	41.74	75.42	4.66
t5+DL+sc	**68.70**	**78.62**	13.02	**5.68**	**45.32**	**81.80**	**5.54**
t5+DL+tg	64.20	74.85	20.38	4.04	37.99	73.74	4.44

Table 5.1: Performance of different models and scheme methods in Ref-1 dataset. Except NIST and CIDEr, all other metrics are in percentage. The best score regarding each metric is shown in bold. all: without DL selection. na: naive scheme. sc: schema guided scheme. tg: template guided scheme.

	Reference Type2 (Ref-2)						
	BLEU	**BLEURT**	**SER**	**NIST**	**METEOR**	**ROUGE_L**	**CIDEr**
Seq2Seq+all	20.03	34.72	84.33	1.16	18.40	43.91	0.06
Seq2Seq+DL	14.94	28.82	86.48	1.37	16.48	29.21	0.02
TGen+all	10.62	27.75	88.35	1.19	15.27	38.74	0.04
TGen+DL	23.91	31.09	78.13	2.44	20.50	43.32	0.05
t5+all+na	24.43	51.98	44.38	2.13	20.43	35.34	0.55
t5+all+sc	22.84	50.90	66.09	1.01	17.06	33.60	0.21
t5+all+tg	24.15	59.24	30.63	3.74	26.99	40.99	0.82
t5+DL+na	24.34	61.61	29.52	2.98	26.54	49.07	1.39
t5+DL+sc	22.66	63.24	**19.80**	3.04	26.18	48.04	1.21
t5+DL+tg	**26.24**	**64.43**	20.23	**3.43**	**28.64**	**50.25**	**1.44**

Table 5.2: Performance of different models and scheme methods in Ref-2 dataset. Except NIST and CIDEr, all other metrics are in percentage. The best score regarding each metric is shown in bold. all: without DL selection. na: naive scheme. sc: schema guided scheme. tg: template guided scheme.

dataset and **t5+DL+tg** performs the best in Ref-2 dataset. (B) Best scores of SER in both Ref-1 and Ref-2 also fall in the set of setting in **t5+DL**. **T5+DL+na** gets the best SER score in Ref-1 and **t5+DL+sc** gets the best SER score in Ref-2. Although SER's two best scores settings are not the same as other automatic metrics, SER is computed on only 10% of test data. SER results can still be regarded as similar to other automatic metrics results. As SER is more explicit than the other metrics in our case and the average slots number in our dataset is around 7, a SER score of around 10% means less than 1 slot is not mentioned correctly in output averagely, which is quite acceptable. (C) By comparing the examples from generated utterances in Figure 5.4 and Figure 5.5, it is obvious that without using the DL method, **t5+all** settings can have a reasonable performance with our small size of training data. In contrast to the baseline models, which can not generate a complete and meaningful sentence, **t5+all** settings are promising. As a result, there might be a possibility to get better performance with a larger data size. (D) Although

Model	Generated Sequences
	of physical damage! The drone has a damaged frame and low battery. It's flying at an altitude of 5m. Risk of physical damage! There's a tree in the drone's flight path at a distance of 5m. Risk of physical damage! There's
Seq2Seq+all	of physical damage! The drone has a damaged frame and low battery. It's flying at an altitude of 5m. Risk of physical damage! There's a tree in the drone's flight path at a distance of 5m. Risk of physical damage! There's
	of physical damage! The drone has a damaged frame and low battery. It's flying at an altitude of 5m. Risk of physical damage! There's a
	of physical damage ! The drone has a damaged frame and low battery . It 's flying at an altitude of physical damage ! There 's a distance of physical damage ! There 's a distance of physical damage ! There 's a distance of physical damage ! There
Seq2Seq+DL	of physical damage ! The drone has a damaged frame . It 's flying at an altitude of physical damage ! There 's a distance of physical damage ! There 's a distance of 5 m. There 's a distance of 5 m. There 's a distance of 5
	of physical damage ! The drone has a damaged frame . It 's flying at an altitude of physical damage ! There 's a distance of physical damage ! There 's a distance of physical m. There 's a distance of 5 m. There 's a distance of 5
	Risk of physical damage! The drone has has damaged It's flying at at altitude of Risk of Risk of physical damage! There's There's the the drone's drone's flight path at distance of distance of .
TGen+all	Risk of physical physical damage! The drone has has damaged battery. It's flying at at altitude of Risk of Risk of physical damage! There's There's in the drone's drone's flight path at distance of of a .
	Risk of physical damage! The drone has has damaged battery. It's flying at at altitude of Risk of Risk of physical physical damage! There's There's the the the drone's flight path at distance of of .
	Risk of physical physical damage! The drone has has battery. battery. It's at at altitude of Risk of Risk of physical damage! damage! There's in in the drone's drone's path at at distance of of .
TGen+DL	Risk of physical damage! The drone has has a at at at altitude of Risk of of physical damage! There's a in the drone's drone's path at at distance of of of .
	Risk of physical damage! damage! There's a the drone's drone's path at distance of .
	The drone has a damaged frame and low battery. It's flying at an altitude of 1m. Risk of physical damage! There's a cell pole only 2m away.
t5+all+naive	The drone has a damaged frame and empty battery. It's flying at an altitude of.5m. Risk of physical damage! There's a railing in the drone's flight path at a distance of 5m.
	The drone has a damaged frame and low battery. It's flying at an altitude of 30m. Risk of physical damage! There's a waterfall only.5m away.
	The drone has a damaged frame and low battery. It's flying at an altitude of 1m. Risk of physical damage! There's a telephone pole only 2m away.
t5+all+schema	The drone is flying indoors at an altitude of 1m. Risk of physical damage! There's a railing in the drone's flight path at a distance of 5m.
	The drone has a damaged frame and low battery. It's flying at an altitude of 30m. Risk of physical damage! There's a waterfall only.5m away.
	The drone has a damaged frame and low battery. It's flying at an altitude of 1m. Risk of physical damage! There's a telephone pole in the drone's flight path at a distance of 5m.
t5+all+t2g2	The weather is overcast and the drone is not waterproof. Risk of physical damage! There's a train in the drone's flight path at a distance of 5m.
	The drone has a damaged frame. Risk of physical damage! There's a rock only 0.5m away.
	1m away from the remote control. Risk of physical damage! There's a tree in the drone's flight path at a distance of 8m.
t5+DL+naive	The drone has a damaged frame and empty battery. It's flying at an altitude of 6m in a low-temperature setting. Risk of physical damage! There's a freight train train in the drone's flight path at a distance of 5m.
	The drone has a damaged frame. Risk of physical damage! There's a rock only.5m away.
	The drone has a damaged frame and low battery. It's flying at an altitude of 1m. The drone is 4876m away from the remote control. Risk of physical damage! There's a tree in the drone's flight path at a distance of 8m.
t5+DL+schema	Fire Train has a damaged frame and empty battery. It's flying at an altitude of 6m in a low-temperature setting. Risk of physical damage! There's a freight train in the drone's flight path at a distance of 5m.
	The drone has a damaged frame. Risk of physical damage! There's a rock only.5m away.
	The drone has a damaged frame and low battery. It's flying at an altitude of 1m. The drone is 4878m away from the remote control. Risk of physical damage! There's a tree in the drone's flight path at a distance of 8m.
t5+DL+t2g2	The drone has a damaged frame and empty battery. It's flying at an altitude of 20m in a low-temperature setting. Risk of physical damage! There's a freight train in the drone's flight path at a distance of 5m.
	The drone has a damaged frame. Risk of physical damage! There's a rock only.5m away.
	Risk of physical damage and lost connection! The drone has a damaged frame and low battery. It's flying at an altitude of 1m. The drone is 4778m away from the remote control. Risk of physical damage! There's a tree in the drone's flight path at a distance of 8m.
Reference Typ1	Risk of physical damage! The drone has a damaged frame and empty battery. It's flying in a low-temperature setting. Risk of physical damage! There's a freight train in the drone's flight path at a distance of 5m.
	Risk of physical damage! The drone has a damaged frame. Risk of physical damage! There's a rock only.5m away.

Figure 5.4: A few examples of generated utterances from different models in Ref-1 dataset.

the DL method is regarded as an improvement for our NLG system, when it comes to a pure Seq2Seq model, it performs worse according to the scores from **Seq2Seq+DL** and

Model	Generated Sequences
	Risk of physical damage! The drone is flying at a is flying at a in the drone is flying at a in the drone is flying
Seq2Seq+all	Risk of physical damage! The drone is flying at a is flying at a in the drone is flying at a in the drone is flying
	Risk of physical damage! The drone is flying at a is flying at a
	Risk of physical damage ! The drone is flying is flying . m altitude . m altitude . m altitude . m altitude . m altitude . m altitude . m altitude . m altitude . m altitude . m altitude . m altitude . m altitude . m
Seq2Seq+DL	Risk of physical damage ! The drone is flying is flying . m altitude . m altitude . m altitude . m altitude . m altitude . m altitude . m altitude . m altitude . m altitude . m altitude . m altitude . m altitude . m
	Risk of physical damage ! The drone is flying is flying . m altitude . m altitude . m altitude . m altitude . m altitude . m altitude . m altitude . m altitude . m altitude . m altitude . m altitude . m altitude . m
	Risk of physical physical physical damage! The The The is is is a a a a a the Risk Risk of physical physical damage! The The The The The is is flying a a a in in in .
TGen+all	Risk of physical physical physical damage! The The is is is flying a a in in in .
	Risk of physical physical physical damage! The The The The is is is flying a a a a a in .
	Risk Risk of physical damage! The The drone is is flying flying is flying at flying at flying at at at at Risk Risk of physical physical damage! The drone is is and a a a a a the the the in .
TGen+DL	Risk of physical damage! The drone is is flying flying is flying flying at flying at at Risk Risk of of physical damage! The drone is a a in in .
	Risk Risk of physical damage! The drone is is flying is flying a a a a a .
	The drone is flying at a 1m altitude with low battery. Risk of physical damage! There's a telephone pole 5m away which is in the flight path.
t5+all+naive	The frame is in need of repair. Risk of physical damage! The drone is flying toward a rail train 5m away.
	In.5m there's a waterfall in the drone's path.
	The drone is flying 1m high with a low battery. Risk of physical damage! The drone is flying toward a telephone pole in its path 5m away.
t5+all+schema	The drone is flying at a 20m altitude with low battery. Also, it's overcast and the drone isn't waterproof. Risk of physical damage! In 1m there's a cargo train in the flight path.
	The drone is flying next to rocks only 0.5m away.
	The drone's frame is damaged and the battery is running low, while flying at a 1m altitude. Risk of physical damage! There's a telephone pole 5m away and in the flight path.
t5+all+t2g2	The drone's frame is in need of repair. Risk of physical damage! The drone is flying toward a freight train which is only 5m away.
	The drone's frame is damaged and the battery is running low, while flying at a 30m altitude. Risk of physical damage! The drone is flying close to a rock.5m away.
	The drone is flying at a 1m altitude with an empty battery. Risk of physical damage! There's a tree in 8m in the flight path.
t5+DL+naive	The drone is flying 6m high with a damaged frame and empty battery. Risk of physical damage! There's a freight train in 5m in the flight path.
	The frame is in need of repair. Risk of physical damage! The drone is flying near a rock which is.5m away.
	The drone is flying at a 1m altitude. Risk of physical damage! There's a tree in 8m in the drone's path.
t5+DL+schema	The drone is flying at a 5m altitude in a cold environment. Risk of physical damage! The drone is flying toward a freight train in its path 5m away.
	The drone's frame is in need of repair. Risk of physical damage! The drone is flying near a rock only 0.5m away.
	The drone is flying at a 1m altitude with a damaged frame and low battery. Risk of physical damage! There's a tree in the flight path in 8m.
t5+DL+t2g2	The drone is flying toward a freight train which is 5m away.
	The drone is flying a rock which is.5m away.
	Risk of physical damage and lost connection! The drone is flying 1m high with a damaged frame. It's on low battery and 4778m front he remote. Risk of physical damage! The drone is approaching a tree in its path in 8m.
Reference Typ2	Risk of physical damage! The frame on the drone is in need of repair and the battery is empty. The outdoor temperature is also low. Risk of physical damage! There's a freight train in the path in 5m.
	Risk of physical damage! The drone has a damaged frame. Risk of physical damage! The drone is flying near a rock.5m away.

Figure 5.5: A few examples of generated utterances from different models in Ref-2 dataset.

Seq2Seq+all. As the DL method selects the slots based on our designed rules and logic, the sequence of slots in input data is mostly not the same sequence of slots appearing in the target reference. This might be the main factor why **Seq2Seq+DL** performs worse than **Seq2Seq+all**. It suggests that rearranging the sequence to match the target reference might be a good way to improve the DL method further. (E) In Table 5.2 for Ref-2, there is no huge difference on BLEU scores for **t5+all** and baseline models. On the other hand,

BLEURT score can detect and show a huge difference between them. However, when it comes to the examples shown in Figure 5.5, it is obvious that **t5+all** performs way better than baseline models. This can be attributed to the various syntax and vocabulary in Ref-2. BLEURT can capture the semantic similarities beyond surface overlap, which is useful in this case. (F) Based on the scores from both tables mentioned above, three different scheme methods **t5+DL** settings report no huge difference in our case. However, when it comes to **t5+all**, **t5+all+sc** scores the worst among 3 methods in both case. This can be attributed to the long input data length of **t5+all+schema** (around 800 tokens on average), which has already been discussed in the previous fine-tuning process.

5.2 Domain Generalization and Few-shot Learning Capability

In the previous section, we have already evaluated the performance of our NLG system, and it shows that the three scheme methods show no vast difference in the general performance. As these three methods are for few-shot learning, experiments about few-shot and zero-shot learning capability are performed in this section to evaluate these three different methods further.

5.2.1 Domain Generalization Capability

Domain generalization capability, similar to zero-shot learning capability, is the capability for the NLG model to handle domains that were not exposed to during training. As an ideal NLG model, it is promising that it can perform well even in a brand-new domain. What is more, although eight domains are clarified in our case, including one miscellaneous domain, it is no doubt that various environment domains can not be enumerated. Additionally, in different environment domains, there is a new combination of slots as input and new vocabulary. Domain generalization capability is undoubtedly significant for our NLG system.

Results are reported in Table 5.3 and Table 5.4 for two types of references Ref-1 and Ref-2 respectively. The **Seen** means the model is trained by all training data, while the **Unseen** means the model is trained by all training data, excluding the corresponding domain dataset. In order to be more accurate, the scores are computed only in the corresponding domain test data. **Unseen - Seen** represents the BLEU scores decrease between Unseen and Seen. The lower **Unseen - Seen** is, the worse the Unseen model performs compared to the Seen model. All scores are determined by averaging two times experiments with different random seeds.

From the results, it can be concluded that: (A) The lowest decrease (Unseen - Seen) appears mostly in the Rural and Disturbance domains, which is consistent with the fact that they both occupy a large portion of data in the whole dataset. Consequently, the BLEU score drops significantly when these domains are excluded from training data. (B) Except for the above two domains, most of the decreases remain trivial (less than -5). Some of them even report positive values. This suggests that these three scheme methods all have the capability for zero-shot learning, which is also known as domain generalization capability. (C) Setting **t5+DL+tg** outperforms other methods in both cases. In Ref-1 dataset it scores a positive value 3.82, and in Ref-2 dataset it scores -0.01. This suggests that the simple template-based scheme method **t2g2** has superior generalization capability for zero-shot learning.

t5+DL+na	de	di	fa	is	mi	oc	ru	ur	average
Unseen	74.18	48.09	60.19	52.45	73.63	60.39	59.13	49.50	
Seen	76.72	51.02	64.13	55.19	73.24	62.50	70.90	52.66	
Unseen - Seen	-2.54	-2.93	-3.94	-2.74	0.39	-2.11	**-11.77**	-3.16	-3.60
t5+DL+sc	**de**	**di**	**fa**	**is**	**mi**	**oc**	**ru**	**ur**	**average**
Unseen	67.22	41.89	57.39	51.37	76.45	68.47	57.23	45.96	
Seen	82.38	47.71	57.21	53.57	80.31	80.85	92.03	50.17	
Unseen - Seen	-15.16	-5.82	0.18	-2.20	-3.86	-12.38	**-34.80**	-4.21	-9.78
t5+DL+tg	**de**	**di**	**fa**	**is**	**mi**	**oc**	**ru**	**ur**	**average**
Unseen	77.71	42.47	57.71	54.31	72.15	59.40	76.34	58.51	
Seen	75.75	53.53	41.32	48.32	69.82	55.95	69.85	53.51	
Unseen - Seen	1.96	**-11.06**	16.39	5.99	2.33	3.45	6.49	5.00	3.82

Table 5.3: BLEU scores in Ref-1 dataset among different domains settings. The lowest **Unseen - Seen** among all domains is in bold, and the highest one among three models is underlined. Domain names refer to: de: Desert, di: Disturbance, fa: Factory, is: Island, mi: Miscellaneous, oc: Ocean, ru: Rural, ur: Urban.

t5+DL+na	de	di	fa	is	mi	oc	ru	ur	average
Unseen	25.61	13.90	28.33	19.09	27.33	23.98	17.89	15.50	
Seen	24.37	24.06	32.30	19.74	31.92	25.81	19.25	17.55	
Unseen - Seen	1.24	**-10.16**	-3.97	-0.65	-4.59	-1.83	-1.36	-2.05	-2.92
t5+DL+sc	**de**	**di**	**fa**	**is**	**mi**	**oc**	**ru**	**ur**	**average**
Unseen	25.02	10.25	24.48	14.10	25.21	19.64	26.90	16.41	
Seen	20.38	14.45	25.75	15.04	26.45	23.31	30.09	17.25	
Unseen - Seen	4.64	**-4.20**	-1.27	-0.94	-1.24	-3.67	-3.19	-0.84	-1.34
t5+DL+tg	**de**	**di**	**fa**	**is**	**mi**	**oc**	**ru**	**ur**	**average**
Unseen	21.76	17.80	38.00	21.23	33.01	28.18	24.65	19.31	
Seen	17.89	16.74	41.71	25.11	33.22	30.99	22.96	15.43	
Unseen - Seen	3.87	1.06	-3.71	**-3.88**	-0.21	-2.81	1.69	3.88	-0.01

Table 5.4: BLEU scores in Ref-2 dataset among different domains settings. The lowest **Unseen - Seen** among all domains is in bold, and the highest one among three models is underlined. Domain names refer to: de: Desert, di: Disturbance, fa: Factory, is: Island, mi: Miscellaneous, oc: Ocean, ru: Rural, ur: Urban.

5.2.2 Few-shot Learning Capability

As a virtual assistant for drone domain, it is essential for our NLG system to support a constantly increasing number of domains and APIs which means the few-shot learning capability. For example, when a new type of sensor data is added for the drone, a sound NLG system with superior few-shot learning capability can adapt to this new type as a new slot by training with a few example data. In this section, experiments are carried out to analyze the trade-off between the number of training examples and the performance of our NLG system.

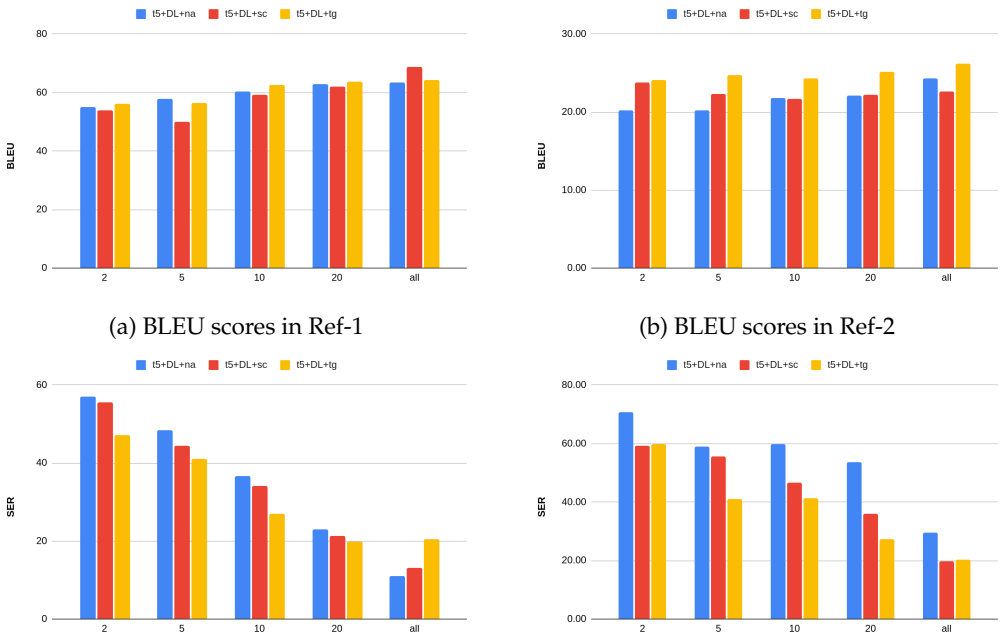
In the experiments, K-shot subsets for varying values of K $[2, 5, 10, 20]$ are defined. Based on it, each of the eight domains from the training set offers K random samples. To this end, the amount of data is $[16, 40, 80, 160]$ in the training set, respectively. (When the original training set for a specific domain is insufficient, the rest will be randomly chosen from the dev set randomly). The test set is left untouched. The automatic BLEU metric

and human SER metric are both carried out for evaluation. All the scores are determined by averaging two times experiments with different random seeds.

Results are reported in Figure 5.6 referring to Table 5.5. It can be concluded that: (A) It is clearly shown that the performance is improving as more and more data is available, which results in the BLEU scores increasing and SER declining when K increases. (B) In most cases, setting **t5+DL+tg** outperforms the other two settings with a higher BLEU score and a lower SER score in every K-shot setting. Although the extreme 2-shot setting (only 16 training samples) is not obvious, in the 5-shot setting (only 40 training samples), the SER scores for **t5+DL+tg** in both Ref-1 and Ref-2 can be controlled to around 40% already. (C) Remarkably, in Ref-2 dataset, **t5+DL+tg** with 5-shot setting outperforms the **t5+DL+na** model trained with 20-shot setting, which is 4 times larger. It can be taken as evidence that **t2g2** scheme method offers superior few-shot learning capability.

BLEU											
	Ref-1						**Ref-2**				
K	**2**	**5**	**10**	**20**	**all**	**K**	**2**	**5**	**10**	**20**	**all**
t5+DL+na	55.06	57.72	60.41	62.79	63.33	**t5+DL+na**	20.27	20.18	21.76	22.09	24.34
t5+DL+sc	53.83	50.1	59.15	61.96	68.7	**t5+DL+sc**	23.84	22.30	21.72	22.17	22.66
t5+DL+tg	56.21	56.44	62.56	63.63	64.2	**t5+DL+tg**	24.16	24.73	24.36	25.12	26.24
SER											
	Ref-1						**Ref-2**				
K	**2**	**5**	**10**	**20**	**all**	**K**	**2**	**5**	**10**	**20**	**all**
t5+DL+na	57.06	48.33	36.67	23.08	11.11	**t5+DL+na**	70.83	59.03	59.72	53.74	29.52
t5+DL+sc	55.56	44.41	34.21	21.22	13.02	**t5+DL+sc**	59.38	55.56	46.53	36.11	19.80
t5+DL+tg	47.22	40.97	26.98	19.84	20.38	**t5+DL+tg**	59.72	40.97	41.31	27.43	20.23

Table 5.5: BLEU and SER scores among different K-shot settings.



(a) BLEU scores in Ref-1

(b) BLEU scores in Ref-2

(c) SER scores in Ref-1

(d) SER scores in Ref-2

Figure 5.6: Performance in few-shot learning referring to Table 5.5.

5.3 Robustness to Input Noise

The previous sections show that our NLG system has reasonably good performance and offers superior zero-shot and few-shot capability, especially using **t2g2** scheme method. However, recent studies show that NLG system performing well on different benchmarks may not be equal to working well in a real situation that contains unpredictable noise. Some high-performance NLG model may be sensitive to trivial input changes and generate erroneous predictions. Therefore, in order to evaluate the robustness of the NLG model, experiments by adding noise into input data should be carried out instead of bechmarking only on common well-designed datasets (Moradi and Samwald, 2021). Since our data domain is in real-world sensory data from drones, robustness to input noise becomes a significant factor to be considered. In this section, experiments are carried out to analyze the robustness of our NLG system against input perturbations.

5.3.1 Perturbations Methods and Implementation

Based on the investigation of Moradi's paper (Moradi and Samwald, 2021), nine perturbation types from both character-level and word-level are introduced and implemented to generate noise for our dataset.

- **Character-level perturbation**: This perturbation type focus on a randomly selected word W, and perturbations are performed to its characters. It consists of the following types:
  - **Insertion**: For W contains three characters or more, a randomly selected character is inserted to W in a random position except for the first and last one.
  - **Deletion**: For W contains three characters or more, a character from W is randomly selected and deleted except the last or the first character.
  - **Replacement**: A character from W is randomly selected and replaced by a random character.
  - **Swapping**: A character from W is randomly selected and swapped with its right or left character.
  - **Repetition**: A random position is selected, except the first or last position, and the character in that position is copied and inserted right after it.
  - **Letter case changing**: Either the first or all the characters' letter case of W is toggled.
- **Word-level perturbation**: In this type of perturbation, it is performed to the randomly selected word W itself. It consists of the following types:
  - **Deletion**: A randomly selected word W is removed from the input sample.
  - **Ordering**: Two consecutive words from the sample are randomly selected, and their order is changed.
  - **Repetition**: A word W is randomly selected, and its copy is inserted right after it.

With the above perturbation types and Moradi's implementation[6], adjustments are made to fit in our dataset, which contains not only characters but also symbols. Figure 5.7 presents an example for every perturbation method. Furthermore, variable PPS (Perturbation Per Sample) in the range $[1, 2, 3]$ is introduced, indicating how many times a specific perturbation type is performed per data sample. After generating noise for test data with all the perturbation methods mentioned above, they are evaluated using the BLEU metric in original models trained in Section 5.1.

Perturbation	Perturbated Text
	Character-level
deletion	SYSTEM | service_name=Drone_1 | inform,risk_of_physical_damage ,**vaues** = warning | inform,normal_frame ,values = 0.0
insertion	**SmYSTEM** | service_name=Drone_1 | inform,risk_of_physical_damage ,values = warning | inform,normal_frame ,values = 0.0
Icc	SYSTEM | service_name=Drone_1 | inform,risk_of_physical_damage ,**Values** = warning | inform,normal_frame ,values = 0.0
repetition	SYSTEM | service_name=Drone_1 | inform,risk_of_physical_damage ,**vallues** = warning | inform,normal_frame ,values = 0.0
replace	**STSTEM** | service_name=Drone_1 | inform,risk_of_physical_damage ,values = warning | inform,normal_frame ,values = 0.0
swapping	**YSSTEM** | service_name=Drone_1 | inform,risk_of_physical_damage ,values = warning | inform,normal_frame ,values = 0.0
	Word-level
deletion	SYSTEM | service_name=Drone_1 | inform,risk_of_physical_damage ,values = warning | ,normal_frame ,values = 0.0
ordering	**| SYSTEM** service_name=Drone_1 | inform,risk_of_physical_damage ,values = warning | inform,normal_frame ,values = 0.0
repetition	SYSTEM | service_name=Drone_1 | inform,risk_of_physical_damage ,values = warning | **inform inform** ,normal_frame ,values = 0.0
Original text	SYSTEM | service_name=Drone_1 | inform,risk_of_physical_damage ,values = warning | inform,normal_frame ,values = 0.0

Figure 5.7: Character-level and word-level perturbations examples in our implementation. Perturbation is in bold.

5.3.2 Performance

Results are reported in Figure 5.8 referring to Table 5.6, 5.8, 5.7, 5.9 which the lowest decrease (average - original) models are in bold. It can be concluded that: (A) **T5+DL+sc** models outperform others against noise to a large extent, based on the fact that there is a lower decrease (average - original) compared to 2 other models and no decreasing trend when PPS increasing. It suggests that scheme method **schema guided** offers superior robustness against noise. It can be attributed to the structure of **t5+DL+sc**. As its structure consists of a slot-value pair plus slot description per slot, noise cannot easily disturb the information of a slot under this structure. (B) It is obvious that **t5+DL+tg** reports the largest decreasing trend of (average - original) when PPS increasing in both Ref-1 and Ref-2. It suggests that **t5+DL+tg** performs worst against noise. It is also consistent with the fact that, unlike the other two structured scheme methods, **t2g2** directly connects every shot with its representation and value. (C) When focusing on Ref-1 which the models score significantly higher BLEU score (+40 BLEU) than Ref-2, **t5+DL+na** and **t5+DL+sc** both are more sensitive to word-level noise instead of character-level which is vice versa for **t5+DL+tg**. This could be attributed to the fact that word-level perturbation can break data structure more easily than character-level.

[6] https://github.com/mmoradi-iut/NLP-perturbation

Character-level Perturbation for Ref-1									
	PPS = 1								
	orginal	**deletion**	**insertion**	**lcc**	**repetition**	**replace**	**swapping**	**average**	**average-original**
t5+DL+na	63.33	49.15	48.05	48.41	48.35	47.30	44.48	47.62	-15.71
t5+DL+sc	68.70	67.08	67.12	66.25	66.94	67.32	66.73	66.91	**-1.79**
t5+DL+tg	64.20	58.45	59.68	60.37	55.14	57.56	60.31	58.59	-5.62
	PPS = 2								
t5+DL+na	63.33	45.91	47.85	45.24	47.35	48.16	45.33	46.64	-16.69
t5+DL+sc	68.70	67.35	66.15	67.31	66.72	66.40	66.61	66.76	**-1.94**
t5+DL+tg	64.20	55.15	59.28	59.68	55.51	52.51	58.80	56.82	-7.38
	PPS = 3								
t5+DL+na	63.33	44.59	46.65	47.94	50.71	49.61	45.92	47.57	-15.76
t5+DL+sc	68.70	66.72	66.25	66.69	66.30	65.84	66.56	66.39	**-2.31**
t5+DL+tg	64.20	50.38	56.42	57.88	52.17	51.34	51.01	53.20	-11.00

Table 5.6: BLEU scores in different character-level perturbation settings for Ref-1. The lowest decrease **(average - original)** among all models is in bold. PPS indicates how many times the perturbation is performed per data sample.

Character-level Perturbation for Ref-2									
	PPS = 1								
	orginal	**deletion**	**insertion**	**lcc**	**repetition**	**replace**	**swapping**	**average**	**average-original**
t5+DL+na	24.34	24.01	22.95	23.95	23.80	23.73	23.54	23.66	-0.68
t5+DL+sc	22.66	21.45	21.72	21.86	23.77	21.91	22.17	22.15	**-0.51**
t5+DL+tg	26.24	28.94	25.64	23.82	25.16	22.94	24.48	25.16	-1.08
	PPS = 2								
t5+DL+na	24.34	23.58	23.41	24.74	24.28	22.82	23.74	23.76	-0.58
t5+DL+sc	22.66	20.81	23.17	20.89	24.01	22.63	21.25	22.13	**-0.53**
t5+DL+tg	26.24	26.59	25.01	26.04	22.59	24.32	24.33	24.81	-1.43
	PPS = 3								
t5+DL+na	24.34	23.93	22.06	23.87	25.43	25.22	23.54	24.01	-0.33
t5+DL+sc	22.66	21.48	23.48	20.76	24.52	23.45	21.37	22.51	**-0.15**
t5+DL+tg	26.24	23.68	23.01	23.97	22.70	20.95	26.36	23.45	-2.79

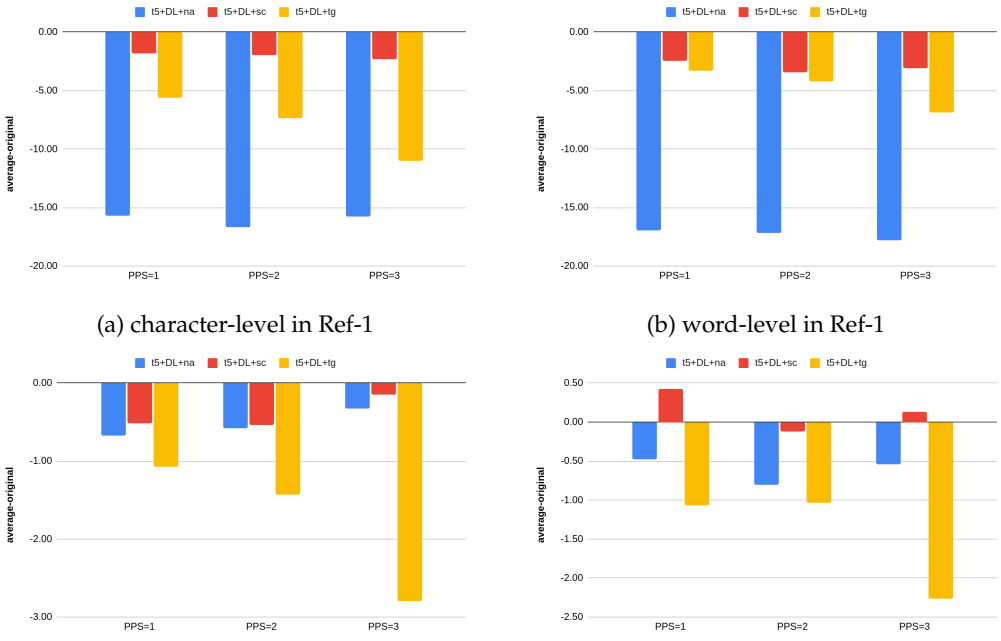
Table 5.7: BLEU scores in different character-level perturbation settings for Ref-2. The lowest decrease **(average - original)** among all models is in bold. PPS indicates how many times the perturbation is performed per data sample.

Word-level Perturbation for Ref-1						
	PPS = 1					
	orginal	**deletion**	**ordering**	**repetition**	**average**	**average-original**
t5+DL+na	63.33	47.43	43.29	48.35	46.36	-16.97
t5+DL+sc	68.70	65.67	66.38	66.72	66.26	**-2.44**
t5+DL+tg	64.20	61.53	60.29	60.96	60.93	-3.27
	PPS = 2					
t5+DL+na	63.33	49.22	43.68	45.61	46.17	-17.16
t5+DL+sc	68.70	65.38	65.91	64.40	65.23	**-3.47**
t5+DL+tg	64.20	58.14	60.38	61.48	60.00	-4.20
	PPS = 3					
t5+DL+na	63.33	42.74	47.89	46.06	45.56	-17.77
t5+DL+sc	68.70	64.47	66.14	66.12	65.58	**-3.12**
t5+DL+tg	64.20	56.81	55.40	59.83	57.35	-6.85

Table 5.8: BLEU scores in different word-level perturbation settings for Ref-1. The lowest decrease **(average - original)** among all models is in bold. PPS indicates how many times the perturbation is performed per data sample.

Word-level Perturbation for Ref-2						
	PPS = 1					
	orginal	**deletion**	**ordering**	**repetition**	**average**	**average-original**
t5+DL+na	24.34	22.81	23.95	24.82	23.86	-0.48
t5+DL+sc	22.66	22.80	23.89	22.58	23.09	**0.43**
t5+DL+tg	26.24	24.21	25.39	25.93	25.18	-1.06
	PPS = 2					
t5+DL+na	24.34	23.19	23.85	23.56	23.53	-0.81
t5+DL+sc	22.66	21.04	23.60	22.99	22.54	**-0.12**
t5+DL+tg	26.24	23.92	25.90	25.79	25.20	-1.04
	PPS = 3					
t5+DL+na	24.34	20.48	25.09	25.82	23.80	-0.54
t5+DL+sc	22.66	20.73	25.53	22.12	22.79	**0.13**
t5+DL+tg	26.24	22.36	23.98	25.60	23.98	-2.26

Table 5.9: BLEU scores in different word-level perturbation settings for Ref-2. The lowest decrease **(average - original)** among all models is in bold. PPS indicates how many times the perturbation is performed per data sample.



(a) character-level in Ref-1

(b) word-level in Ref-1

(c) character-level in Ref-2

(d) word-level in Ref-2

Figure 5.8: Mean values of (average - original) among different models in different perturbation settings referring to table 5.6, 5.8, 5.7, 5.9.

Chapter 6
Conclusion

In this chapter, the essential aspects of the work done in this thesis will be concluded regarding the achievements of the research goal. Furthermore, the possible future work on this thesis will be presented.

6.1 Conclusions

It is motivated by the demand for a NLG system in the drone assistant domain that can generate fluent and native utterances from primary sensory data as warning messages. The research goal for this thesis is to implement and evaluate the neural NLG system for this domain, which can be further extended to other hand-over assistants that generate warning messages from sensory data.

We started by investigating related work in other NLG tasks to address the challenges for our own NLG task. The investigation is conducted on the fundamental concept of NLG, some recent NLG tasks and development from template to neural E2E system, recent NLG methods stressing content fidelity problems and other existing data-driven NLG datasets as well as the architecture of well-known neural network models in NLG. After analyzing the difference between research output from related work and our NLG task, the following three challenges are addressed and solved in this thesis:

The first challenge is how to design rules and logic to annotate the initial corpus scientifically and reasonably. In order to solve the first challenge, the setup and annotation for an initial dataset are presented in Chapter 3. Before the utterance annotation, the collection of original data from video chips to feature-value pairs, as well as the design of warning message category and grade, is proposed. Then the annotation procedure is presented, which consists of annotation from video to data, annotation with Description Logic, and annotation of natural language utterance. To this end, the initial corpus consists of feature-value pairs with corresponding utterances. It has variables of applying DL or not, in Ref-1 or Ref-2 (the latter is more diverse in syntax and vocabulary).

The second challenge is how to implement a suitable NLG system based on limited data resources. Regarding the second challenge, Chapter 4 is presented to solve it.

Firstly, a pipeline of our NLG system is presented to further identify two components in the system: the linearizer and the rewriter. Linearizer performs data selection and linearization, where three scheme methods for zero-shot learning and few-shot learning are implemented in our dataset including **na**, **sc** and **tg**. As transfer learning from a pre-trained model is suitable with our limited data, our rewriter is implemented by fine-tuning a pre-trained t5-small model with our data. The detailed implementation and architecture are presented, and fine-tuning is performed to set up hyperparameters for model training. The hyperparameters are shown as follows: learning rate is $1e-3$, the batch size is 8 (downgraded to 4 in one special case **t5+all+sc**), and beam search inference with beam width is 2 and length penalty is 1.0. For ease of comparison, 20 epoch updates for all training are performed.

The last challenge is how to evaluate the NLG system in different aspects. Chapter 5 is presented in order to deal with the last challenge. Firstly, benchmarking is performed to evaluate and compare our NLG system to others. Two other NLG models, the Seq2Seq model and TGen, are introduced and implemented with our dataset as baseline models. In order to perform an objective and comprehensive evaluation among all NLG models, the automatic metrics BLEU, NIST, METEOR, ROUGE-L, and CIDEr are computed. Moreover, another state-of-the-art transfer learning-based metric, BLEURT, is computed. In addition, a human evaluation metric SER (Slot Error Rate) is adjusted from previous work and performed so as to evaluate the correctness of the output sentence intuitively. Results show that the setting **t5+DL+sc** and **t5+DL+tg** can outperform baseline models to a large extent (around +30 BLEU and -60 SER). Based on their high performance, evaluation focusing on the other two aspects is further carried out among three scheme methods in the basic setting of **t5+DL**. On the one hand, domain generalization and few-shot learning capability are the first to be considered. Through the experiments on blocking specific domains and reducing data amount in training data, we found that setting **t5+DL+tg** offers superior zero-shot and few-shot learning capability compared to two other scheme methods. On the other hand, robustness against input noise is also considered. Through the experiments on adding nine different perturbation types in both character-level and word-level into test data, we concluded that **t5+DL+sc** can outperform others to a large extent while **t5+DL+tg** performs worst. In addition, **t5+DL+sc** is more sensitive to word-level perturbation which is vice versa for **t5+DL+tg**. It suggests that there is a trade-off applying these two high-performance few-shot learning scheme methods **sc** and **tg** in real-world data.

To this end, the neural NLG system in the drone assistant domain for generating warning messages from sensory data is designed and implemented. It can outperform other baselines and offer superior few-shot learning capability as well as good robustness.

6.2 Future Work

Lastly, some ideas and future improvements that would continue the work of this thesis are proposed.

First of all, more comprehensive evaluation metrics can be further implemented, which is suitable for our NLG task. As our output is a sentence, it is not easy to evaluate whether it is correct and can be evaluated in various dimensions. Although many commonly used automatic evaluation metrics like BLEU have already been applied in our evaluation, the fact that they all have their focus suggests that they might not be the most suitable evaluation metric. That is why the novel metric **BLEURT** is computed, and the human

evaluation metric **SER** is proposed and performed. In order to improve performance for **BLEURT**, which uses a well-trained model to identify semantic similarities, further fine-tuning with customized data focusing on the drone domain can be performed in its model. **SER** is an intuitive metric in our NLG task and has been implemented as an automatic metric in prior work. However, **SER** automatic implementation can not evaluate Boolean slot values or identify paraphrases of slot values. As a result, it is taken as human evaluation after adjustment in our case. With the limited labor, **SER** is performed in a certain amount instead of all test data, which results in significant fluctuation. It will be promising to implement automatic **SER** solving the issues mentioned above.

Another interesting part that would be investigated is the **t5+all** setting which each training input contains all the features. As mentioned before, Description Logic is one part of the annotation process, selecting the features that should be mentioned in the reference. It's no doubt that **t5+DL** can outperform **t5+all**. Nevertheless, the performance of **t5+all** in both BLEU and SER can be considered reasonable with our limited data size (200 data samples for training data). It suggests that our neural model might be powerful enough to learn the hand-over mechanism. It will be a good try to extend our dataset size in order to evaluate whether **t5+all** can perform on par with **t5+DL** or not.

Lastly, the high-performance scheme method **schema guided** could be further improved in our NLG task. In the previous evaluation, scheme method **schema guided** shows good robustness to input noise and offers competent few-shot and zero-shot capability due to its complete structure and natural descriptions for every inserted slot. However, representations using this scheme method are mostly more prolonged than others in length, and this factor keeps it from maintaining good performance in many-slots-in-one-representation situations. As a result, further experiments on its structure template modification are promising to improve performance. Firstly, the intuitive idea is to examine performance on different descriptions per slot to find the best performance. Also, in our dataset, binary slots, which are straightforward and have no need for description to explain, appear frequently. It could be promising to reduce its length by deleting the description for binary slots or grouping all binary slots into one slot. Moreover, it might be a good idea to merge relative slots into one. On the other hand, besides modifying its structure template, mixing two high-performance scheme methods by replacing the description in schema guided with the corresponding template from **t2g2** method would also be a good try.

List of Figures

List of Tables

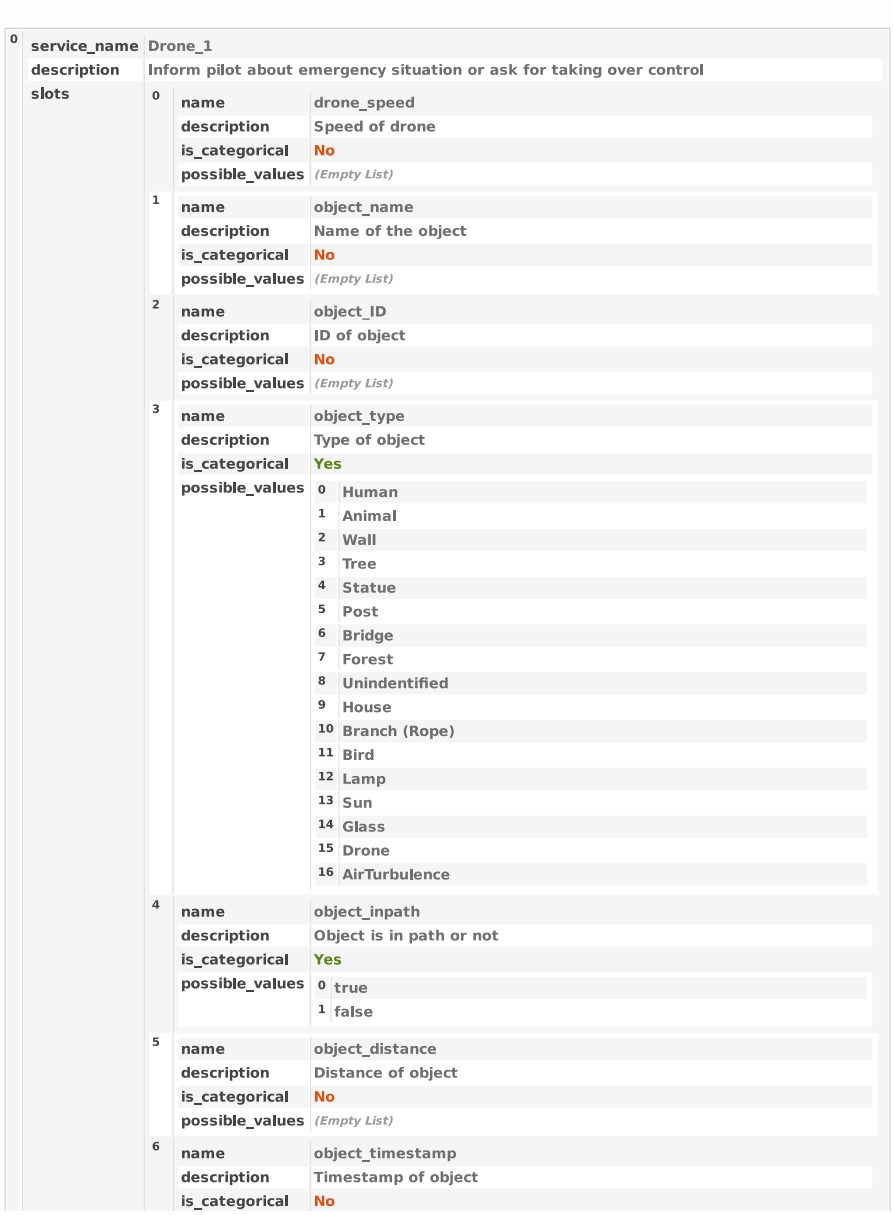
Appendix A
Appendix

A.1 Hand-over Situations Mechanism

```
Assumption: Ground, Water below are not Objects (i.e., risk of "object near" - not while
landing).

Main Warnings:
RiskOfPhysicalDamage
RiskOfInternalDamage (water, electricity)
RiskOfHumanDamage
LostConnection

Battery Characteristics:
EmptyBattery (<20%)
LowBattery (<45%)

Drone Characteristics:
(waterproof)
NotWaterproofDrone
    NotWaterproofDrone ⊓ WaterproofDrone ⊑ ⊥

Weather Characteristics:
    environment (role)
Rain
Snow
HighCHanceOfPrecipitation
Indoor
LowVisibility
LowTemperature (<10 grad Celcius)
StrongWind (> 12 metres/sec)

Altitude Characteristics:
    hasAltitude (role)
High (>70m)
Low (<50cm)
Flying (>0cm)

Speed Characteristics:
    hasSpeed (role)
Fast (>=13m/s)

Ground Characteristics:
    flyingAbove
Water
Ground

Pilot Characteristics:
    pilotedBy
Inexperienced

Distance Characteristics for Objects:
(roles)
near (<3m)
veryClose (<1m)
reachable (<10m)

inPath (means, the drone eventually needs to change direction in order to avoid the object)

Distance for Remote Control (concept):
AlmostOutOfRangeRC
OutOfRCRange
```

```
∃near.Object ⊓ ∃pilotedBy.Inexperienced ⊑ RiskOfPhysicalDamage
∃veryClose.Object ⊑ RiskOfPhysicalDamage
∃near.MovingObject ⊑ RiskOfPhysicalDamage
∃(inPath ⊓ reachable).Object. ⊑ RiskOfPhysicalDamage
GoingBackwards ⊑ RiskOfPhysicalDamage
UpsideDown ⊓ ∃pilotedBy.Inexperienced ⊑ RiskOfPhysicalDamage
∃environment.Indoor ⊑ RiskOfPhysicalDamage

Gloomy ⊑ HighChanceOfPrecipitation
∃environment.HighChanceOfPrecipitation ⊓ NotWaterproofDrone ⊑ RiskOfInternalDamage
∃environment.Gloomy ⊓ ∃hasAltitude.High ⊑ ∃environment.LowVisibility
Dark ⊑ LowVisibility
Foggy ⊑ LowVisibility
(Rain ⊔ Snow) ⊑ LowVisibility
∃environment.Rain ⊓ NotWaterproofDrone ⊑ RiskOfInternalDamage
∃environment.Snow ⊓ NotWaterproofDrone ⊑ RiskOfInternalDamage
DirtyLenses ⊑ ∃environment.LowVisibility
∃environment.LowVisibility ⊓ ∃near.Object ⊑ RiskOfPhysicalDamage
StrongWind ⊓ Rain ⊑ Hurricane
Hurricane ⊑ ExtremeWeather
Lightning ⊑ ExtremeWeather
∃environment.ExtremeWeather ⊓ ∃pilotedBy.Inexperienced ⊑ RiskOfPhysicalDamage ⊓
RiskOfInternalDamage
LowTemperature ⊓ LowBattery ⊑ RiskOfPhysicalDamage

FrontRightPropeller ⊑ ∃hasPart^- (range of the role hasPart)
RearRightPropeller ⊑ ∃hasPart^-
FrontLeftPropeller ⊑ ∃hasPart^-
RearLeftPropeller ⊑ ∃hasPart^-
Propeller ⊑ ∃hasPart^-
Battery ⊑ ∃hasPart^-

Bird ⊑ Animal
Animal ⊑ MovingObject
Object ⊓ Moving ⊑ MovingObject
Car ⊑ MovingObject
Drone ⊑ MovingObject %other drone
MovingObject ⊑ Object
FragileObject ⊑ Object
Human ⊑ MovingObject
AirTurbulence ⊑ SpaceAnomaly
SpaceAnomaly ⊑ MovingObject

Tree ⊑ Object
Wall ⊑ Object
Statue ⊑ Object
Post ⊑ Object
Bridge ⊑ Object
Forest ⊑ Object
Unindentified ⊑ Object
House ⊑ Object
Branch_Rope ⊑ Object         ← treat it as a special name

LowBattery ⊑ Battery
EmptyBattery ⊑ Battery

AlmostOutOfRCRange ⊓ ∃hasPart.LowBattery ⊑ LostConnection
OutOfRCRange ⊑ LostConnection

∃flyingAbove.Water ⊓ ∃hasAltitude.Low ⊓ NotWaterproofDrone ⊑ RiskOfPhysicalDamage ⊓
RiskOfInternalDamage

Flying ⊓ ∃hasPart.Damaged ⊑ RiskOfPhysicalDamage
Flying ⊓ ∃hasPart.EmptyBattery ⊑ RiskOfPhysicalDamage
∃flyingAbove.Water ⊓ ∃hasPart.EmptyBattery ⊓ NotWaterproofDrone ⊑ RiskOfInternalDamage
Flying ≡ ∃flyingAbove


∃veryClose.Human ⊑ RiskOfHumanDamage
∃environment.Indoor ⊓ ∃near.Human ⊑ RiskOfHumanDamage
Glass ⊑ FragileObject
Lamp ⊑ FragileObject
∃near.FragileObject ⊓ ∃near.Human ⊑ RiskOfHumanDamage
∃inPath.Sun ⊑ BlindingLight
∃inPath.Lamp ⊑ BlindingLight
∃veryClose.Lamp ⊑ RiskOfInternalDamage
BlindingLight ⊑ LowVisibility
∃hasAltitude.Low ⊓ ∃hasSpeed.Fast ⊑ RiskOfPhysicalDamage

∃environment.StrongWind ⊓ ∃hasPart.LowBattery ⊑ RiskOfPhysicalDamage
∃hasAltitude.High ⊓ ∃hasPart.LowBattery ⊑ RiskOfPhysicalDamage
```

A.2 Schema Guided Method Structure Template



	possible_values	*(Empty List)*
7	name	wind_speed
	description	Wind speed
	is_categorical	No
	possible_values	*(Empty List)*
8	name	pilot_experienced
	description	Pilot is experienced or not
	is_categorical	No
	possible_values	*(Empty List)*
9	name	altitude
	description	Flying altitude
	is_categorical	No
	possible_values	*(Empty List)*
10	name	temperature
	description	Flying temperature
	is_categorical	No
	possible_values	*(Empty List)*
11	name	distance_from_remote_control
	description	Distance from remote control
	is_categorical	No
	possible_values	*(Empty List)*
12	name	battery_level
	description	Battery level
	is_categorical	No
	possible_values	*(Empty List)*
13	name	low_visibility
	description	Low visibility or not
	is_categorical	Yes
	possible_values	0: 1.0 1: 0.0
14	name	normal_frame
	description	Normal frame or not
	is_categorical	Yes
	possible_values	0: 1.0 1: 0.0
15	name	weather
	description	Weather
	is_categorical	Yes
	possible_values	0: Sunny 1: Rainy 2: Gloomy 3: Snowy 4: Foggy 5: Dark 6: Unindentified 7: Lightning

16	name	upside_down
	description	Upside down or not
	is_categorical	Yes
	possible_values	0 1.0 1 0.0
17	name	good_motor_condition
	description	Good motor condition or not
	is_categorical	Yes
	possible_values	0 1.0 1 0.0
18	name	going_backwards
	description	Going backwards or not
	is_categorical	Yes
	possible_values	0 1.0 1 0.0
19	name	indoor
	description	Indoor or not
	is_categorical	Yes
	possible_values	0 1.0 1 0.0
20	name	waterproof_drone
	description	Waterproof drone or not
	is_categorical	No
	possible_values	*(Empty List)*
21	name	flying_over
	description	Where the drone flying over
	is_categorical	Yes
	possible_values	0 ground 1 water
22	name	lost_connection
	description	Lost connection or not
	is_categorical	Yes
	possible_values	0 1.0 1 0.0
23	name	risk_of_internal_damage
	description	risk of internal damage or not
	is_categorical	Yes
	possible_values	0 1.0 1 0.0
24	name	risk_of_human_damage
	description	risk of human damage or not
	is_categorical	Yes
	possible_values	0 1.0 1 0.0

25	**name**	**risk_of_physical_damage**
	description	**risk of physical damage or not**
	is_categorical	**No**
	possible_values	*(Empty List)*

Bibliography

Jay Alammar. 2018. The illustrated transformer. Accessed: 2022-10-06.

Franz Baader, Diego Calvanese, Deborah Mcguinness, Daniele Nardi, and Peter Patel-Schneider. 2007. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge university press.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.

Meghyn Bienvenu and Magdalena Ortiz. 2015. Ontology-mediated query answering with data-tractable description logics. In *Reasoning Web International Summer School*, pages 218–307. Springer.

Research BIS. 2021. Global drone delivery market: Focus on drone receptacle, drone type, package size, range, and application - analysis and forecast, 2023 to 2030. ReportLinker.

Alex Borgida, Maurizio Lenzerini, and Riccardo Rosati. 2003. Description logics for databases. *The description logic handbook: theory, implementation, and applications*, pages 462–484.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. MultiWOZ - a large-scale multi-domain Wizard-of-Oz dataset for task-oriented dialogue modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026, Brussels, Belgium. Association for Computational Linguistics.

Ernie Chang, Alisa Kovtunova, Stenfan Borgwardt, Vera Demberg, Kathryn Chapman, and Hui-Syuan Yeh. 2022. Logic-guided message generation from raw real-time sensor data. In *Proceedings of the 13th Conference on Language Resources and Evaluation (LREC 2022)*, page 6899–6908. LREC.

Zhiyu Chen, Harini Eavani, Wenhu Chen, Yinyin Liu, and William Yang Wang. 2020. Few-shot NLG with pre-trained language model. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 183–190, Online. Association for Computational Linguistics.

Bhuwan Dhingra, Manaal Faruqui, Ankur Parikh, Ming-Wei Chang, Dipanjan Das, and William Cohen. 2019. Handling divergent reference texts when evaluating table-to-text generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4884–4895, Florence, Italy. Association for Computational Linguistics.

Ondřej Dušek and Filip Jurčíček. 2016. Sequence-to-sequence generation for spoken dialogue via deep syntax trees and strings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 45–51, Berlin, Germany. Association for Computational Linguistics.

Ondřej Dušek, Jekaterina Novikova, and Verena Rieser. 2018. Findings of the E2E NLG challenge. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 322–328, Tilburg University, The Netherlands. Association for Computational Linguistics.

Ondřej Dušek, Jekaterina Novikova, and Verena Rieser. 2020. Evaluating the state-of-the-art of end-to-end natural language generation: The e2e nlg challenge. *Computer Speech & Language*, 59:123–156.

Markus Freitag and Scott Roy. 2018. Unsupervised natural language generation with denoising autoencoders. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3922–3929, Brussels, Belgium. Association for Computational Linguistics.

Tino Fuhrman, David Schneider, Felix Altenberg, Tung Nguyen, Simon Blasen, Stefan Constantin, and Alex Waibe. 2019. An interactive indoor drone assistant. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6052–6057. IEEE.

Milica Gašic, Dongho Kim, Pirros Tsiakoulis, Catherine Breslin, Matthew Henderson, Martin Szummer, Blaise Thomson, and Steve Young. 2014. Incremental on-line adaptation of pomdp-based dialogue managers to extended domains. *In Proceedings on InterSpeech*.

Jan Hajic, Martin Cmejrek, Bonnie Dorr, Yuan Ding, Jason Eisner, Daniel Gildea, Terry Koo, Kristen Parton, Gerald Penn, Dragomir Radev, et al. 2002. Natural language generation in the context of machine translation. In *Summer workshop final report, Johns Hopkins University*.

Matthew Henderson, Blaise Thomson, and Steve Young. 2014. Robust dialog state tracking using delexicalised recurrent neural networks and unsupervised adaptation. In *2014 IEEE Spoken Language Technology Workshop (SLT)*, pages 360–365. IEEE.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Matthew Horridge. 2011. *Justification based explanation in ontologies*. The University of Manchester (United Kingdom).

Daniel Jurafsky and James H. Martin. 2008. *Speech and language processing. Pearson International Edition*. Prentice Hall.

Mihir Kale and Abhinav Rastogi. 2020. Template guided text generation for task oriented dialogue. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6505–6520, Online. Association for Computational Linguistics.

Andrej Karpathy. 2015. The unreasonable effectiveness of recurrent neural networks. Accessed: 2022-10-06.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1203–1213, Austin, Texas. Association for Computational Linguistics.

Percy Liang, Michael Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 91–99, Suntec, Singapore. Association for Computational Linguistics.

Tianyu Liu, Kexiang Wang, Lei Sha, Baobao Chang, and Zhifang Sui. 2018. Table-to-text generation by structure-aware seq2seq learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

John Makhoul, Francis Kubala, Richard Schwartz, and Ralph Weischedel. 1999. Performance measures for information extraction. In *In Proceedings of DARPA Broadcast News Workshop*, pages 249–252.

Danilo Mirkovic, Lawrence Cavedon, Matthew Purver, Florin Ratiu, Tobias Scheideck, Fuliang Weng, Qi Zhang, and Kui Xu. 2011. Dialogue management using scripts and combined confidence scores. US Patent 7,904,297.

Milad Moradi and Matthias Samwald. 2021. Evaluating the robustness of neural language models to input perturbations. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1558–1570, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. 2017. The E2E dataset: New challenges for end-to-end generation. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 201–206, Saarbrücken, Germany. Association for Computational Linguistics.

Christopher Olah. 2015. Understanding lstm networks. Accessed: 2022-10-06.

Ankur Parikh, Xuezhi Wang, Sebastian Gehrmann, Manaal Faruqui, Bhuwan Dhingra, Diyi Yang, and Dipanjan Das. 2020. ToTTo: A controlled table-to-text generation dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1173–1186, Online. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2020. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8689–8696.

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

Thibault Sellam, Dipanjan Das, and Ankur Parikh. 2020. BLEURT: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892, Online. Association for Computational Linguistics.

Amanda Stent, Matthew Marge, and Mohit Singhai. 2005. Evaluating evaluation methods for generation in the presence of variation. In *Computational Linguistics and Intelligent Text Processing*, pages 341–351, Berlin, Heidelberg. Springer Berlin Heidelberg.

Ilya Sutskever, James Martens, and Geoffrey E Hinton. 2011. Generating text with recurrent neural networks. In *ICML*.

Ran Tian, Shashi Narayan, Thibault Sellam, and Ankur P. Parikh. 2019. Sticking to the facts: Confident decoding for faithful data-to-text generation. *CoRR*, abs/1910.08684.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. HMM-based word alignment in statistical translation. In *COLING 1996 Volume 2: The 16th International Conference on Computational Linguistics*.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1711–1721. Association for Computational Linguistics.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263, Copenhagen, Denmark. Association for Computational Linguistics.

Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179.

Steve J Young. 2000. Probabilistic methods in spoken-dialogue systems. *Philosophical Transactions: Mathematical, Physical and Engineering Sciences*, pages 1389–1402.

Alina Zeshan. 2019. Recurrent neural networks. Accessed: 2022-10-06.