Logic-Guided Message Generation from Raw Real-Time Sensor Data

Ernie Chang, Alisa Kovtunova, Stefan Borgwardt, Vera Demberg, Kathryn Chapman, Hui-Syuan Yeh
Department of Computer Science, Saarland Informatics Campus, Germany
Chair for Automata Theory, Technische Universität Dresden, Germany
{cychang@coli,s8kachap@teams,demberg@lst,yehhui@coli}.uni-saarland.de
firstname.lastname@tu-dresden.de

Abstract
Natural language generation in real-time settings with raw sensor data is a challenging task. We find that formulating the task as an end-to-end problem leads to two major challenges in content selection – the sensor data is both redundant and diverse across environments, thereby making it hard for the encoders to select and reason on the data. We here present a new corpus for a specific domain that instantiates these properties. It includes handover utterances that an assistant for a semi-autonomous drone uses to communicate with humans during the drone flight. The corpus consists of sensor data records and utterances in 8 different environments. As a structured intermediary representation between data records and text, we explore the use of description logic (DL). We also propose a neural generation model that can alert the human pilot of the system state and environment in preparation of the handover of control.

Keywords: message generation, content selection, domain variability, low resources, description logic, experiment

1. Introduction
Sensor technology has evolved in the last decade driven by the need to delegate routine tasks to machines. An example of such delegation is the Internet of Things (IoT) represented in wearables, smart homes, autonomous driving, etc. In the high-stakes applications, the system must continuously monitor the sensor data stream to detect any situation deterioration and react on it promptly. Moreover, in the case of a constantly changing environment, faithful data records can be very diverse and contain a lot of superfluous information. In this setting, it becomes challenging to detect an abnormality automatically.

In this work, we consider an example that embodies both of these aspects of sensor data: redundancy and diversity. Particularly, as a recent technological advance, drones with impressive features, advanced sensors and capabilities have become commonplace (Fuhrman et al., 2019) (e.g. for aerial surveys, mapping, aerial movies and even selfi-drones). The amount of sensor information routinely processed during a flight such as altitude, wind speed, air pressure, temperature, etc. is enormous. This is related to the fact that drones are extremely useful in the most remote and hard-to-reach places where very little can be controlled by human operators. As these drones are used for an increasingly wide range of tasks, interacting with drones becomes more important. To enable these interactions, it is essential to devise a natural language generation (NLG) setup that can flexibly connect to a variety of data records collected by the drone and convey information reliably. In this paper, we propose a neural generation model (or drone assistant) that verbalizes messages from sensor data records in order to perform a controlled handover to a human drone pilot (see Figure 1). Recent data-driven methods have achieved good performance on various NLG tasks (Liu et al., 2018; Freitag and Roy, 2018; Chen et al., 2019). However, most studies focus on surface descriptions of simple record sequences, for example, attribute-value pairs of fixed or very limited schema, such as E2E (Novikova et al., 2017) and WikiBio (Lebret et al., 2016). In contrast, there is a much larger variety of data records available in the present setup, and the content selection task is substantially harder (only critical information, not all available information, should be mentioned at handover time).

![Figure 1. We focus on the drone handover as the main communicative function.](image)

An on-device drone utterance generation model is thus faced with two challenges due to its diverse and large sensor data inputs (see Figure 2): (1) In real world scenarios, deployed drone dialogue systems are constantly exposed to drastically different environments; therefore, the ability of the system to generalize to diverse as well as unseen environments is desirable. (2) The redundancy of raw sensor records adds overheads to the encoder, and result in texts with low fidelity, where wrong facts are selected to be verbalized or even hallucinated.

To this end, we argue that it is necessary to leverage intermediate content representations to achieve faithful and controllable logical generation in such real-time settings with redundant data. In this paper, these repre-
Recent data-driven methods tend to conflate the pipeline modules into end-to-end neural networks, such as (Liu et al., 2018) [Wiseman et al., 2017, Wiseman et al., 2018, Gong et al., 2019]. However, purely neural models often suffer from problems with content fidelity (omission or hallucination of facts) (Dušek et al., 2018). More recent work has begun to focus on preserving the fidelity of the generation, such as (Dhingra et al., 2019, Tian et al., 2019). Their work obtains good performance on surface-level NLG. In contrast, our work focuses on reducing content selection overheads for complex input data with high variability.

Recent NLG datasets mostly focus on surface-level generation. This includes WeatherGov (Liang et al., 2009), E2E (Novikova et al., 2017), WikiBio (Lebret et al., 2016), and ToTTo (Parikh et al., 2020). However, these datasets contain natural language sentences which are simple restatements of data records, and involve no abstract logical inference. In fact, the model in (Chen et al., 2020a) only obtains a 20% factual correctness rate based on human evaluation, which is far from an acceptable level in real-world systems. In contrast, our work focuses on the logical formulations executed on complex data records that can be derived from real-time systems realistically. To this end, we believe our new dataset can help future development of on-device real-time drone assistants.

### 3. Drone Sensor Data

This section describes the collected corpus and the simulated environments. We first describe the collection process in Section 3.1 then discuss the data schema in Section 3.2 and annotations (Section 3.4) and Section 3.5. In this work, the drone assistant is used in handover situations, where it sends a message to human pilots when there is a problem and the drone cannot continue flying autonomously. The type of handover is also categorized according to the level of criticality, which describes the drone’s environment and corresponds to how urgent it is for control to be handed over to the human drone pilot.
3.1. Video Data Collection

We collected drone videos in 8 different environments: Disturbance (Di), Urban (Ur), Rural (Ru), Ocean (Oc), Desert (De), Island (Is), Factory (Fa) and Miscellaneous (Mi). These drone videos are recorded from the perspective of drones from either real drone manoeuvres or a drone simulator. The environments have drastically different settings: a detailed analysis is provided in Section 5. We split the original records into 316 snapshot videos of 10 seconds each. They are selected based on human judgement of whether the level of criticality rises to the point where a handover is required.

3.2. Data Record Schema

Each snapshot video from Section 3.1 is then manually annotated with realistic data records, which are based on the supposed sensor data that a drone can capture. We show an example of the data in Figure 2 which consists of a time step record of nearby objects, and a separate drone status record. The time step data reports 9 attributes that show the dynamics of the surrounding objects; for example, the object type, along with other information related to the flight path, such as InPath or Moving. The time step data are collected at 1-second intervals. The drone status record remains the same during the snapshot, as it indicates information of more permanence; for an example see Table 2. Together, they constitute a snapshot of data covering up to a 10-second interval. Snapshots are used as input data to the drone assistant.

In this section, examples of such data are written in bold.

3.3. Challenges

An end-to-end model using the raw snapshots as inputs faces the following problems.

1. The data record contains variable length information as the number and types of detected objects change between videos.

2. As the data is permutation invariant (Lee et al., 2019), the output of the model should not change under any permutation of the elements in the input data record.

3. Snapshots are long-form (containing at least 30 cells each). Irrelevant information in the data will tend to confuse the model.

4. By its design, transformer-based models are unable to process long sequences due to their self-attention operation, which scales quadratically with the sequence length.

To address these challenges, we incorporate DL reasoning in our drone assistant.

3.4. Annotation with Description Logic

Here, we describe the process of criticality annotation, where the type (see Table 1) and level (“informative”, “warning” or “advisory”) of criticality as well as DL expressions are added to each snapshot, in order to achieve more robust text generation.

Criticality prediction determines the type of utterance intent. To determine the type and the level of criticality, we employ description logic reasoning (Baader et al., 2007), based on an ontology consisting of axioms that describe background knowledge about drones and surrounding objects. For our test scenarios, the hand-crafted ontology contains 62 predicates and 55 axioms. We use ontology-mediated queries that determine whether a certain critical situation is present in the input data (Borgida et al., 2003; Bienvenu and Ortiz, 2015).

In the following text, ontology axioms and queries are in sans serif. For example, the ontology contains axioms Foggy ⊑ LowVisibility and Env.LowVisibility ⊓ ¬near.Object ⊑ RiskOfPhysicalDamage, which characterize fog as a visibility impairment and describe a critical situation of the drone flying close to another object in a low-visibility environment. The query predicate RiskOfPhysicalDamage indicates an increased criticality.

The general process of DL annotation works as follows. Based on the data record schema from Section 3.2, domain experts create a mapping from the records to the DL ontology predicates. Using the four query predicates from Table 1 for each snapshot DL reasoning can then automatically derive which type of criticality holds. For the most common criticality in the video collection, RiskOfPhysicalDamage, we distinguish three levels of urgency depending on the ontology axiom triggering the criticality. We break ties between multiple reasons for criticality by keeping the most compelling one, i.e. between “informative” and “advisory” we choose the latter. Additionally, DL justifications (Horridge, 2011) are used to extract those parts of the input record that are responsible for the criticality. This information is encoded here into DL expressions, which takes the form of grounded DNF formulas (disjunctive normal form) expressing all reasons for the positive evaluation of the criticality queries. In the prototype implementation, since the ontology and the criticality queries are fixed, we did not use a DL reasoner to perform query answering. Instead we implemented the whole procedure as macros inside an annotation platform.

Example The partial status report in Table 2 is received from a defective drone steered by an inexperienced pilot inside a relatively cold room with different objects logged in Figure 2. Some of these data instances, on their own or in combination with others, indicate that safe piloting is not possible. The system must promptly recommend a handover. For

https://cloud.perspicuous-computing.science/s/zLoBagLxo2fgqw4
### Table 1. Information on the types of criticality.

<table>
<thead>
<tr>
<th>Types of Criticality</th>
<th>Description</th>
<th>Example DL Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>RiskOfPhysicalDamage</td>
<td>Potential physical damage (e.g. crash)</td>
<td>Altitude (m): 20 Battery_level: 30 OR InPath: true Distance: 3 at 00:02</td>
</tr>
<tr>
<td>RiskOfInternalDamage</td>
<td>Potential internal damage</td>
<td>weather: gloomy waterproof_drone: false</td>
</tr>
<tr>
<td>RiskOfHumanDamage</td>
<td>Risk of injuring nearby humans</td>
<td>indoor: true Distance: 0.5 Type: Human at 00:16</td>
</tr>
<tr>
<td>LostConnection</td>
<td>Drone connectivity/signal strength</td>
<td>Distance_from_remote_control (m): 162 Battery_level: 0</td>
</tr>
</tbody>
</table>

### Table 2. Sample of a status report that is part of a data record.

<table>
<thead>
<tr>
<th>Wind_speed (m/s)</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drone_speed (m/s)</td>
<td>10</td>
</tr>
<tr>
<td>Pilot_experienced</td>
<td>FALSE</td>
</tr>
<tr>
<td>Altitude (m)</td>
<td>20</td>
</tr>
<tr>
<td>Temperature (celcius)</td>
<td>5</td>
</tr>
<tr>
<td>Distance_from_remote_control (m)</td>
<td>16</td>
</tr>
<tr>
<td>Battery_level</td>
<td>70</td>
</tr>
<tr>
<td>Low_visibility</td>
<td>FALSE</td>
</tr>
<tr>
<td>Normal_frame</td>
<td>FALSE</td>
</tr>
<tr>
<td>weather</td>
<td>sunny</td>
</tr>
<tr>
<td>upside_down</td>
<td>FALSE</td>
</tr>
<tr>
<td>good_motor_condition</td>
<td>TRUE</td>
</tr>
<tr>
<td>going_backwards</td>
<td>FALSE</td>
</tr>
<tr>
<td>indoor</td>
<td>TRUE</td>
</tr>
<tr>
<td>waterproof_drone</td>
<td>FALSE</td>
</tr>
<tr>
<td>flying_over</td>
<td>ground</td>
</tr>
</tbody>
</table>

### Table 3. Examples of original (O) text and its three T5-paraphrases (P).

- **O** Risk of physical damage! There is a skyscraper in the flight path of the drone at a distance of 2m.
- **P** Risk of physical damage! The drone has a damaged frame and is flying indoors. There's a skyscraper in the drone path at a distance of 2m.

- **O** There is a damaged frame and a dangerous floor in the drone's flight path at a distance of 2m.
- **P** The drone has a damaged frame and is flying indoors. Risk of physical damage! There’s a skyscraper on the flight path of the drone at a distance of 2m.

### 3.5. Collection of Natural Utterances

As ground truth, we employed human experts to label each snapshot with an utterance that describes the situation detected by DL. As discussed in Section 3.4, the type and level of criticality already determine the character of the utterances. However, the example above demonstrates that the criticality can be created by various combined reasons. For instance, an “advisory” criticality type RiskOfPhysicalDamage is intended to alert the human pilot to make prompt decisions regarding the flight course. An “informative” RiskOfPhysicalDamage communicates a suboptimal internal state of the drone, such as a low power level, after which the human pilot can decide how to act on it. At this stage, the human experts are able to prioritise and aggregate the information e.g. an utterance could contain a solution recommendation or a partial situation report containing the data to be changed.

#### Paraphrase Augmentation.

To enrich the variability of the texts, we use T5 to generate paraphrases of the texts. For each utterance, we generate an additional three sentences by varying the beam size during decoding. By obtaining 10 sentences initially, linguistic experts were prompted to select the top 3 sentences based on their fluency and the perceived textual similarity with the original reference. We display some examples in Table 3.

We next describe the approach for automatically generating such utterances.

### 4. The Approach

The neural drone assistant primarily consists of two modules described in detail in Sections 4.1-4.2 and in Figure 3. The first one is a data record linearizer where table-formatted records are converted into a linear string a formal ontology to encode background knowledge. Since it has its own format, it is independent of the platform or the programming language. This allows an ontology to be viewed, extended, and debugged by domain experts regardless of the end application. Moreover, in general, such ontologies can also be learned (semi-automatically) from other sources such as annotated data, text, and alignment with high-level ontologies (Lehmann and Völker, 2014). Existing ontology editing platforms, e.g. Protegé[4] also incorporate tools for visualisation, automatic analysis and reasoning, such as query answering.

https://protege.stanford.edu/
sequence along with auxiliary information. The second module, a DL-to-text rewriter, transforms the linearized sequence into a human-readable sentence. Finally, at the end of this section, we summarize the benefits of incorporating the DL reasoning seen from the NLG perspective.

### 4.1. Data Record Linearization

To linearize data records into sequences, we employ a technique previously used (Kale and Rastogi, 2020), where slot descriptions are added to each slot so as to ease the generation process (see Figure 4). While the slot descriptions are easy to obtain, it remains difficult to encode the semantics of large data records that contain irrelevant cells and duplicate information. Thus, we propose an extension of the schema-guided representation (Kale and Rastogi, 2020) by replacing the slot names with their natural language descriptions and also selecting them based on DL expressions (Section 3.4) so as to only focus on the relevant data records.

### 4.2. DL-to-Text Rewriting

The goal of the rewriting module is to convert DL expressions (generated as described in Section 3.4) to a natural language response with the same semantic content. Thus, we finetune a Text-to-Text Transfer Transformer (T5) (Raffel et al., 2019) model, which is a pre-trained sequence-to-sequence transformer, to generate the natural language response using the linearized DL expression sequence as input. Figure 3 depicts the resulting framework. For ease of comparison, we perform 100-epoch updates for all training, as was empirically found to be sufficient for convergence.

### 4.3. DL Operation As Set Transformation

With the challenges from Section 3.3 in mind, we describe how the DL expression reduces the data complexity for T5 in a way that is functionally similar to a set transformer (Lee et al., 2019).

T5 follows an encoder-decoder structure using stacked self-attention layers for both the encoder and decoder. Self-attention layers typically map one variable-length sequence of symbol representations \( X = (x_1, \ldots, x_n) \) to another sequence of equal length \((z_1, \ldots, z_n)\), with \( x_i, z_i \in \mathbb{R}^d \), for \( d \) being the embedding dimension of a word. The per-layer computational complexity of self-attention is \( O(n^2d) \) (Vaswani et al., 2017). By applying the permutation invariant data record linearization function based on DL expressions to the input data sequence, DL(\( X \)) = \( (x_1, \ldots, x_m) \), as a pre-processing step, we can decrease the length of layer input. Indeed, since DL(\( \cdot \)) is basically a filter, it guarantees that \( m \leq n \) and, in practice (e.g. see the first two lines of Table 2), \( m \ll n \). This results in much lower processing times.

---

In the transformer models, attention weights are calculated using all the words in the input sequence at once. Therefore, for estimating computational benefits of a new input size, we can observe a difference already in the self-attention complexity.
and maintains a high level of representational power, thus yielding better robustness w.r.t. diverse input data. Similarly to how the approaches (Moryossef et al., 2019; Hua and Wang, 2019; Koncel-Kedziorski et al., 2019), the function DL(.) performs an over-approximated text planning, which includes the selection of relevant content (what to say). Thus, controllability, measured by whether the generation correctly reflects the key semantic information in the input, improves naturally over the target output.

5. Corpus Analysis

This section and Table 4 present the details of our corpus. We begin by describing the high-level characteristics for each environment, then analyze both the data record complexity and lexical richness of the utterance.

Characteristics of Environments. Generally speaking, the generated types of criticalities as in Section 3.4 vary between environments due to differences in the types and numbers of objects, settings, distance between remote control and drone, etc. Those environments which contain more humans (e.g. Ur) present more human obstacles, which detect criticalities based on a different set of parameters than the non-human obstacles. The environments in more desolate areas (e.g. De, Oc) generally have fewer objects, thereby provoking fewer obstacle/nearby object warnings. Certain environments with open, outdoor settings (e.g. Ru, Is) contain more instances of long-distance remote drone control, and thus produce more LostConnection warnings than those in more enclosed environments (e.g. Fa, Mi). Lastly, some environments (e.g. Mi, Di) trigger substantially more RiskOfInternalDamage criticalities than others, which are typically prompted by a non-waterproof drone flying in a wet environment. Further, the two aforementioned environments are the only ones containing moving obstacles/nearby objects which warrant RiskOfPhysicalDamage warnings.

Data Record Complexity. Two crucial properties of the given data records are their redundancy and variable length when linearized. The performance of a neural drone assistant will very much be influenced by these factors since such records cannot be properly processed in a low resource setting. For instance, we observe that some attributes such as going backwards are rarely used in the DL expressions or in the utterance: it should only add overhead to the encoding process in the rare occasion of the drone flying backwards. Thus, we compute the average number of cells per data record to get a sense of the distribution of raw data redundancy. We found that some environments (e.g. Di) tend to have more objects perceived by the drone, and so tend to have a larger time step record. Importantly, Table 4 indicates that in average the number of relevant cells for all environments is significantly reduced with the use of DL-transformation.

Lexical Richness. We used the Lexical Complexity Analyser (Lu, 2012) to measure various dimensions of lexical richness of the utterances from Section 5.5. We complement the traditional measure of lexical diversity type-token ratio (TTR) with the more robust measure of mean segmental TTR (MSTTR) (Lu, 2012). The higher the value of MSTTR, the more diverse is the measured text. Table 4 shows that the highest MSTTR value is for Is and Mi while Di and Fa has the lowest value. In addition, we measure lexical sophistication (LS) also known as lexical rareness and find that Oc has the highest LS score.

6. Experiments

We conduct experiments on the collected drone corpus that is split into training, validation and testing sets as reported in Table 5.

Vocabulary. We use SentencePiece (Kudo and Richardson, 2018) to encode text as WordPiece tokens. For all experiments, we use a vocabulary of 32,000 wordpieces as in T5 (Raffel et al., 2019), which is shared across both the input and output of our model. Specifically, both the encoder and decoder consist of 12 blocks, and each block comprises of self-attention, optional encoder-decoder attention, and a feed-forward network layer. The “key” and “value” matrices of all attention mechanisms have an inner dimensionality of 64 and all attention mechanisms have 12 heads, resulting in a model with about 220 million parameters. For regularization, we use a dropout probability of 0.1 everywhere dropout is applied in the model.

Configurations. Our baseline model: T5 is designed so that the encoder and decoder are each similar in size and configuration as in the previous work (Devlin et al., 2018). Specifically, both the encoder and decoder consist of 12 blocks, and each block comprises of self-attention, optional encoder-decoder attention, and a feed-forward network layer. The “key” and “value” matrices of all attention mechanisms have an inner dimensionality of 64 and all attention mechanisms have 12 heads, resulting in a model with about 220 million parameters. For regularization, we use a dropout probability of 0.1 everywhere dropout is applied in the model.

Test Scenarios. Based on the split in Table 5, we design four different test scenarios for the drone assistant model with different training and testing sets. For all inference scenarios, we either test the model on the data records of environments previously seen in the training set (seen) or not (unseen). In all, we train the model on data derived from all environments so that it obtains inductive bias that is more diverse and robust to environmental changes. We also simulate the scenario where we have training data from one environment (ind.) or all except one. In the latter case, the model will be exposed to this unseen environment for testing. This is to simulate the real-time scenario where the drone assistant model is situated in new environments. For all scenarios, the drone assistant is to generate a handover message in relation to the input data record.

Benchmark Comparisons. In Table 6, we compare our model with Fairseq (Ott et al., 2019) seq2seq baseline, a simple retrieval method by using the data record and text pairs as a dictionary, and retrieving the text

*It divides the corpus into successive segments of a given length and then calculates the average TTR of all segments

*It is calculated as the proportion of lexical word types not on the list of 2,000 most frequent words from the British National Corpus.
of the closest\textsuperscript{7} data record representation at inference time. A closely-related method is the template-based generator (\textit{template}) where we construct variations of templates based on the training set. Lastly, we use KGPT [Chen et al., 2020b] which is a pretrained data-to-text model that learns to generate text from various types of structured data. We finetune this model on the training set.

We compare our approach, \textit{T5+DL}, described in Section\textsuperscript{4} with a baseline \textit{T5}. In the latter, the transformer model is simply finetuned on the linearized input sequence from each specified training set and tested on the target environments.

### 7. Main Results

Here we present the experiment results and analysis by first (A) comparing different models within the same environments, so as to provide a more comprehensive comparison of benchmark systems and our proposed model. (B) We then examine how differences in environments influence the performance of each model. (C) Lastly, we also examine the impact of removing the testing environment from the training data, as a way to test the generalizability of the models. This is crucial to the development of real-time drone assistants as there should be no assumptions made about the type of environment that it will be exposed to.

**(A) Benchmark Comparisons.** Comparing our proposed approach with the benchmarks, we see that our proposed technique \textit{T5+DL+all} outperforms all methods. In particular, \textit{Retrieval+all} and \textit{Template+all} achieve the worst performance; while \textit{Seq2seq+all} and \textit{KGPT+all} do slightly better. We find that our baseline approach without DL, \textit{T5+all}, already generates utterances with more surface overlap with the reference than other techniques. However, the significant improvement is brought about with the inclusion of DL – with differences up to 37.36 BLEU points. This correlates with our other observation in Figure\textsuperscript{5} where \textit{Data Record+DL} is much shorter than the raw data records. It is also reflected in the poorer text quality of \textit{T5+all}, which produces a shorter utterance and is missing some essential attributes. This shows that the combination of linearization technique, additional slot descriptions and DL transformation are highly beneficial for generating utterances with high reference surface overlap.

**(B) Impact of Environments On Performance.** Since the length of data records and vocabulary distribution is not homogeneous across different environments, it also results in differences in model performance. We observe that the length of the data records influences the text quality very much, as indicated by the improvements in terms of automatic evaluation \textit{i.e.} BLEU-4 scores on \textit{T5+DL+all} are generally higher than \textit{T5+all}.

\begin{table}[h]
\centering
\begin{tabular}{lcccccccc}
\hline
  & Di & Ur & Ru & Oc & De & Is & Fa & Mi \\
\hline
Average number of cells & 168.85 & 74.90 & 39.46 & 27.92 & 12.56 & 27.6 & 34.0 & 52.25 \\
Average number of cells (DL) & 3.26 & 2.05 & 2.24 & 2.44 & 3.16 & 2.36 & 1.68 & 2.68 \\
LS & 0.60 & 0.58 & 0.59 & 0.63 & 0.60 & 0.61 & 0.62 & 0.62 \\
MSTTR & 0.56 & 0.60 & 0.58 & 0.59 & 0.60 & 0.61 & 0.56 & 0.61 \\
\hline
\end{tabular}
\caption{Corpus statistics across all environments. Lexical sophistication (LS) and mean segmental type-token ratio (MSTTR) are defined in Section\textsuperscript{5}. Average number of cells is defined as the average number of values per data record in the corpus.}
\end{table}

\begin{table}[h]
\centering
\begin{tabular}{lcccccccc}
\hline
  & Di & Ur & Ru & Oc & De & Is & Fa & Mi \\
\hline
Train. & 48 & 10 & 80 & 14 & 12 & 15 & 15 & 30 \\
Valid. & 6 & 5 & 10 & 5 & 5 & 5 & 5 & 5 \\
Test. & 6 & 5 & 10 & 5 & 5 & 5 & 5 & 5 \\
Total & 60 & 20 & 100 & 24 & 22 & 25 & 25 & 40 \\
\hline
\end{tabular}
\caption{Dataset splits in each environment.}
\end{table}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure5.png}
\caption{An example of data records before and after DL-transformation. We display generation outputs of \textit{T5+all} and \textit{T5+DL+all} and show them side-by-side with the reference text.}
\end{figure}

\textsuperscript{7}This is based on the string similarity.
Table 6. Performance in BLEU-4 (Papineni et al., 2002) on testing sets derived from the seen scenario across different environments. Ind. means that the training set is only drawn from the target environment. Scores of proposed approach are statistically-significant based on the two-tailed t-test with $p < 0.05$.

<table>
<thead>
<tr>
<th>Model</th>
<th>Di</th>
<th>Ur</th>
<th>Ru</th>
<th>Oc</th>
<th>De</th>
<th>Is</th>
<th>Fa</th>
<th>Mi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Template+all</td>
<td>22.43</td>
<td>41.87</td>
<td>37.62</td>
<td>34.27</td>
<td>26.58</td>
<td>22.11</td>
<td>34.82</td>
<td>38.63</td>
</tr>
<tr>
<td>Retrieval+all</td>
<td>18.91</td>
<td>45.20</td>
<td>40.80</td>
<td>31.89</td>
<td>24.62</td>
<td>21.89</td>
<td>33.34</td>
<td>42.77</td>
</tr>
<tr>
<td>Seq2seq+all</td>
<td>24.03</td>
<td>51.55</td>
<td>48.71</td>
<td>38.93</td>
<td>32.92</td>
<td>25.52</td>
<td>45.51</td>
<td>50.93</td>
</tr>
<tr>
<td>KGPT+all</td>
<td>25.14</td>
<td>54.50</td>
<td>50.28</td>
<td>40.01</td>
<td>34.52</td>
<td>25.97</td>
<td>41.68</td>
<td>47.27</td>
</tr>
<tr>
<td>T5+all</td>
<td>27.89</td>
<td>56.28</td>
<td>52.07</td>
<td>42.63</td>
<td>36.95</td>
<td>31.18</td>
<td>47.42</td>
<td>52.59</td>
</tr>
<tr>
<td>T5+DL+all</td>
<td>65.25</td>
<td>65.88</td>
<td>78.61</td>
<td>52.45</td>
<td>47.15</td>
<td>59.65</td>
<td>63.30</td>
<td>71.89</td>
</tr>
<tr>
<td>T5+ind.</td>
<td>15.88</td>
<td>33.90</td>
<td>40.47</td>
<td>28.16</td>
<td>33.04</td>
<td>22.06</td>
<td>46.15</td>
<td>35.56</td>
</tr>
<tr>
<td>T5+DL+ind.</td>
<td>42.52</td>
<td>39.26</td>
<td>73.09</td>
<td>29.06</td>
<td>38.30</td>
<td>32.56</td>
<td>42.61</td>
<td>44.69</td>
</tr>
</tbody>
</table>

Table 7. Human Evaluation on the sampled outputs (100 instances) for model comparison on REF-B for both seen and unseen scenarios across all environments. The first row is the (human) utterances from Section 3.5. We abbreviate Naturalness and Wrong as Nat and Wr.

<table>
<thead>
<tr>
<th>Model</th>
<th>seen Nat</th>
<th>seen Miss</th>
<th>seen Wr</th>
<th>unseen Nat</th>
<th>unseen Miss</th>
<th>unseen Wr</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human</td>
<td>4.76</td>
<td>0</td>
<td>0</td>
<td>4.58</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Retrieval</td>
<td>3.32</td>
<td>49</td>
<td>63</td>
<td>2.95</td>
<td>57</td>
<td>48</td>
</tr>
<tr>
<td>Template</td>
<td>4.21</td>
<td>41</td>
<td>47</td>
<td>4.35</td>
<td>59</td>
<td>51</td>
</tr>
<tr>
<td>KGPT</td>
<td>3.45</td>
<td>46</td>
<td>66</td>
<td>4.23</td>
<td>48</td>
<td>44</td>
</tr>
<tr>
<td>Seq2seq</td>
<td>4.10</td>
<td>43</td>
<td>57</td>
<td>4.10</td>
<td>39</td>
<td>45</td>
</tr>
<tr>
<td>T5</td>
<td>4.20</td>
<td>45</td>
<td>51</td>
<td>4.15</td>
<td>49</td>
<td>51</td>
</tr>
<tr>
<td>T5+DL</td>
<td>4.38</td>
<td>39</td>
<td>44</td>
<td>4.27</td>
<td>31</td>
<td>39</td>
</tr>
</tbody>
</table>

8. Conclusion

In this work, we present the task of message generation from real-time sensor data records and release a new language generation corpus that differs from previous corpora in terms of number and diversity in data records. Our results demonstrate the difficulty of the task such that it can serve as baseline for similar tasks where texts are generated from raw data records. Furthermore, we showed that description logic reasoning is able to transform sensor data records and reduce the difficulty of the encoding process to obtain better generation outputs.

9. Acknowledgements.

This work was supported by the DFG in grant 389792660 (TRR 248) (see [http://perspicuous-computing.science](http://perspicuous-computing.science)).
10. Bibliographical References


Tian, R., Narayan, S., Sellam, T., and Parikh, A. P. (2019). Sticking to the facts: Confident decod-