Improving the Semantic Parsing of Building Regulations through Intermediate Representations

Fuchs S[1], Dimyadi J [1, 2], Witbrock M[1], Amor R[1]
[1]The University of Auckland, New Zealand
[2]CAS Limited, Auckland, New Zealand
sffc348@aucklanduni.ac.nz

Abstract. Recent developments show that large transformer-based language models have the capability to generate coherent text and source code in response to user prompts. This capability can be used in the construction domain to interpret building regulations and convert them into a semantic representation usable for automated compliance checking. While base-size models can already be taught to perform semantic parsing with decent quality, this paper shows how intermediate representations (IR) can be used to improve the semantic parsing quality. With reversible IRs, the training time could be reduced to almost a quarter of the initial duration, and through adding a hierarchical parsing step, improvements of up to 6.6% on F1-Scores were reached.

1. Introduction

Semantic parsing has many applications, from program synthesis to interpreting robotic commands and querying in natural language. Another potential use case is the interpretation of norms such as regulations and standards, which is essential for automated compliance checking (ACC) applications. ACC is an active area of research, particularly in the architectural, engineering, and construction (AEC) domains. The primary challenges are the absence or lack of digital regulations to convey the highly complex legalese, the availability and accessibility of information from Building Information Models (BIM), the diversity of required checking procedures (e.g., spatial reasoning), and non-conformity of terms used in BIM and regulations (Solihin and Eastman, 2015). A common approach to address the machine-processability of natural language regulations is translating them into a computable form, which an ACC system can then utilise. Both the translation process and various formats' representation capabilities have received significant attention (Fuchs and Amor, 2021; Zhang and El-Gohary, 2022b; Zhang et al., 2023). Approaches to translating building regulations range from manual expert interpretation to fully automated translation. While most previous approaches divided the translation process into one or more information extraction (e.g., entities, properties, relations, exceptions) and transformation tasks, Fuchs et *al*. (2022) proposed to tackle the problem as a seq2seq task using a transformer-based semantic parser. A seq2seq model is trained with pairs of input and target sequences, allowing the model to learn the mapping between the two. It is commonly used in tasks such as language translation, summarisation, and question-answering.

While semantic parsing is a promising direction, the performance of existing semantic parsers for building regulations still falls short of expectations. Significant improvements in semantic parsing of natural language expressions into SQL and SPARQL were achieved by coarse-to-fine-grained parsing (Dong and Lapata, 2018) and using reversible and lossy intermediate representations (IR) (Herzig et al., 2021). Both papers used the strategy to generate a simplified representation as an intermediate step towards the final output representation. They retain the benefits of seq2seq approaches, such as potentially reducing cascading errors and utilising the strength of pre-trained language models while reducing the complexity of the task. This paper investigates the potential of such methods to improve translating building regulations to LegalRuleML (LRML), an emerging XML-based representation standard (Dimyadi et *al.*,

2017). Furthermore, we investigate suitable IRs for this task and whether multi-task learning or training separate specialised models performs better. This paper contributes a method for a faster and more accurate translation of building regulations into LRML and unlocks new possibilities for interpretability and semi-automation using IRs.

2. Background

2.1 Transformer-based Semantic Parsing of LegalRuleML

Semantic parsing is the process of transforming natural language into a formal representation of its meaning. Such representations range from semantic representations such as Abstract Meaning Representation (AMR) to logical representations such as First-Order Logic or Lambda Calculus, query languages such as SQL and SPARQL, and programming languages such as Python and Java. LRML is a markup language designed to represent norms in a machine-processable format. It incorporates both semantic and logical representations to enable computers to reason about legal requirements. Fuchs et al. (2022) proposed to use a T5 model (Raffel et al., 2020) pre-trained on AMR parsing to translate building regulations into the short version of LRML shown in Listing 1. T5 uses the transformer architecture proposed by Vaswani et al. (2017). The model consists of an encoder to generate a latent representation of the input and a decoder to combine the output generated so far with the input representation through an attention mechanism and generate the following output token. The primary mechanism to generate the latent representation is self-attention, i.e., each token is contextualized by incorporating the relevant information of all other tokens in the sequence. The speciality of the T5 model is that it formalizes each task into a text-to-text format by formulating the task with an initial prompt, e.g., "translate English to German", and all answers are transformed into a textual representation, e.g., "True" and "False", or "cat" and "dog", rather than binary values 0 and 1 for classification tasks. This strategy makes the model easily applicable to a broad range of tasks and suitable for multi-task learning, i.e., learning to do multiple tasks in parallel.

```
Input:  3.4.2 The floor waste shall have a minimum diameter of 40 mm.

Output: if(expr(fun(exist), atom(var(floorWaste)))),
        then(obligation(expr(
             fun(greaterThanEqual),
             atom(rel(diameter), var(floorWaste)),
             data(baseunit(prefix(milli), kind(metre)), value(40.0)))))
```

Listing 1: LRML Rule Example from NZ_NZBC-G13AS1

2.2 Intermediate Representations

Any representation that falls between the input text and the target representation will be considered an intermediate representation (IR). Representations that have the potential to be generated more easily or that can support the generation of the target representation are of interest. Accordingly, IRs can be related to hierarchical information processing, which was applied to building regulations by Zhou and El-Gohary (2022) and Zhang and El-Gohary (2022a). Considering the hierarchically complex structure of legalese, IRs are a great candidate to support the semantic parsing of building regulations.

Herzig et al. (2021) distinguished between reversible and lossy IRs. Reversible IRs are deterministic transformations of the original representation that can be undone by applying the reverse function. An example of such an IR is combining entities and properties using a dot notation: `atom(rel(width), var(wall)) -> atom(wall.width)`. Lossy IRs are further distant

from the target representation. Given a target representation, the lossy IR can be generated by removing certain information. For example, to generate Python code, Dong and Lapata (2018) replaced variable names with the NAME placeholder and values with their data type, e.g., NUMBER or STRING. Alternatively, a lossy IR can be generated from the input text. An alignment between the input text and the target representation might be required, or the work-intensive task of manually creating a ground truth must be conducted. The closest to our work is Ferraro et al. (2020), who used the coarse-to-fine-grained parsing by Dong and Lapata (2018) to semantically parse RegTech, a dataset with a lambda calculus representation of 140 sentences from general regulations, including building regulations, shown in Listing 2. Compared to their work, we generate LRML rather than parsing the clause into a representation used as a step towards producing the computer-processable rules. Furthermore, we utilize pre-trained transformer models rather than training an LSTM model from scratch and explore a broad range of settings to investigate the potential of IRs.

```
Input:  A large building is any building with a net lettable area greater than
        300 m2.
Output: lambda $0 (if (a large building:$0) then (is any building with a lettable
        area greater than ($0 300m2)))
```

Listing 2: RegTech Example (Ferraro et al. 2020)

3. Methodology

The first step in our work is to find reversible IRs for the verbose LRML representation shown in Listing 1. The resulting reversible IR will be used throughout the rest of the paper. The second step is to introduce lossy IRs as used in previous literature. Thirdly, a paraphrased legal clause is generated as an intermediate step. Finally, the model's capability to improve previous predictions will be tested.

3.1 Reversible Intermediate Representations

A new simplified representation is predicted instead of the original representation. Then the reverse function is applied to evaluate the quality of the translation. Accordingly, there is a common measure to compare all results. Potential reversible IRs are identified by inspecting the LRML dataset and the process used to create the initial data set (Dimyadi et al. 2020).

Unit: An automatic unit conversion enriched the LRML dataset with additional metadata. By reverting the unit conversion, this overhead can be removed from the semantic parsing task, and the existing unit conversion can be applied again afterwards:

```
data(baseunit(prefix(milli), kind(metre)), value(40.0)) -> data(40 mm)
```

And/Or: Many occasions were identified where expressions needed to be repeated for different data values. To avoid this redundancy, we allowed conjunctions and disjunctions in the data field and combined consecutive expressions with such duplication. E.g.,

```
data(or(metal, wood))
```

Atom: The nesting of atoms inside expressions increases the length further. This was addressed by expressing subjects and their properties using a dot notation:

```
atom(rel(diameter), var(floorWaste)) -> atom(floorWaste.diameter)
```

Expression: The remaining expressions can be simplified further by utilising the function terms directly and inserting the atom and data values as arguments:

```
expr(fun(greaterThanEqual), atom(floorWaste.diameter), data(40 mm) ->
greaterThanEqual(floorWaste.diameter, 40 mm)
```

Loop: Fuchs et al. (2023) identified loops as especially problematic due to their complex representation. Accordingly, the following simplification was applied:

```
expr(rulestatement(forEach(…)), appliedstatement(is(…))) -> loop(forEach(…), is(…))
```

These simplifications lead to a shorter representation with the following expected benefits for semantic parsing: 1) faster training and inference, 2) fewer syntactic mistakes, 3) reduced complexity, and 4) increased training stability.

3.2 Lossy Intermediate Representations

Given the simplified LRML representation, the lossy IRs are generated by dropping some elements or extracting others. Separate models are trained as demonstrated in Listing 3. Model 1 predicts the IR, and Model 2 the final LRML rule given the legal clause and the predicted IR. As alternatives, Model 2 can be trained using the oracle IR or both the predicted and oracle IR. This experiment shall indicate whether it is more critical for Model 2 to receive a correct IR during training or if it is better to prepare the model for data closer to what is expected at test time. Since a model pre-trained on a related task can benefit the final semantic parsing task, Model 2 is either trained from scratch (i.e., T5-AMR) or based on Model 1. Finally, an investigation is conducted on whether and how well Model 2 utilises the IR. The evaluation includes three versions of the test set: 1) Input plus predicted IR, 2) Input plus oracle IR, and 3) Input only. As an intuition, the parsing quality should increase if the predicted IR is good enough, the quality should be close to perfect given the oracle IRs (depending on the closeness of the IR), and in the last case, the model should still be able to predict meaningful LRML rules, given it does not entirely rely on the IR. The following IRs were explored:

Entity extraction: Most previous approaches to automate the translation of building regulations were based on entity extraction. A natural way to begin the manual translation is to identify the main entities and properties in a legal clause and their logical connections. We extract a unique set of all subjects, properties, and objects from the LRML representation. Listing 3 shows how the models are trained using this IR.

```
Model 1:
Input:  extract LegalRuleML entities: G13AS1 3.4.2 The floor waste shall have a
        minimum diameter of 40 mm.
Output: floorWaste, diameter, 40 mm

Model 2:
Input:  translate English to LegalRuleML: G13AS1 3.4.2 The floor waste[…]<sep>
        floorWaste, diameter, 40 mm
Output: if(exist(floorWaste)), then(obligation(greaterThanEqual(floorWaste.diam…
```

Listing 3: Example input and output for lossy IRs

Expressions only: Identifying the correct logical connections between expressions is of particular importance. Having expressions falsely assigned to the antecedent or consequent, having expressions connected with conjunctions rather than disjunctions, or missing negations can result in rules being not or falsely triggered and compliance issues being falsely reported or missed. There is a high likelihood for such cases to happen in machine learning because such labels are often unbalanced, and a model might learn to predict the more common case. This difficulty is counteracted by predicting only the expressions in the first step, i.e., removing if then, and, or, not, obligation, etc. We hypothesise that it will be easier to identify those logical connections after all expressions were identified, e.g., `exist(floor…), greaterThanEqual(…)`

3.3 Paraphrases as Intermediate Representation

Instead of generating the IR from LRML, simplifying the natural language legal clause might enhance the parsing quality. We manually constructed paraphrases, in a way similar to controlled natural languages, by applying the following transformations to the text:

If-then structure: Reorder the clauses and embed them into an if-then structure: `If a floor waste exists, then…`

Remove unnecessary text: Remove text not represented in LRML, including statements describing the reason for a particular legal requirement and the simplification of long phrases.

Coreference resolution: Repeat entities if references occur over a longer distance, e.g.: `If a floor waste exists, then the floor waste shall…`

Scope of conjunctions and disjunctions: It can become unclear how disjunctions and conjunctions are nested in long clauses. This problem is remediated by explicitly adding the connectors and indicating the start of disjunction using phrases such as `either…or…or…`.

References: References to documents, clauses, tables, and figures are prevalent in legal text. The LRML rules contain the document code for such references, while in the legal clauses, references to the same document are often implicit: e.g., `this document`, `Table 4.5`. Document codes are added to such instances to support the parser (`G13/AS1`, `C/AS2 Table 4.5`).

In contrast to lossy IRs, the paraphrases were not appended to the Model 2 input, but Model 2 was trained to predict LRML directly from the paraphrased clause as in Herzig et al. (2021). While this decision puts more importance on the quality of the paraphrases, it removes the need to share attention between the two inputs.

3.4 Self-reflection

In Section 3.2, expressions were isolated as IR, leaving out logical connections. This IR is very close to the target representation, which raises the question: What if the target representation is appended instead of the IR and the model has the chance to correct mistakes made during the initial predictions? Can the model recognize such mistakes given the additional right-hand context? These questions relate to the recent successes of GPT-4 in showing signs of self-reflection with the ability to improve its previous outputs by being prompted about the correctness of its responses (Shinn et al. 2023). While such behaviour might exist for large language models, we evaluate if smaller T5 models can also improve themselves and if this method can be used for ensembling generation models.

These questions are tested by training separate models as described in Section 3.2. Model 1 predicts LRML as IR, given the regulation clause. Model 2 predicts or improves the LRML rule, given the regulation clause and the LRML rule predicted by Model 1. The main intention is to train Model 2 to correct mistakes made by Model 1. But if Model 1 is overfitting the training data, it might not produce any output with errors. A separate experiment is conducted, where IR predictions for the training data are generated before overfitting the training data. As in previous experiments, the test and validation IR predictions are from the best Model 1.

4. Experimental Results

T5-AMR was trained as per Fuchs et al. (2022). The following hyperparameters were changed to achieve more stable training: Learning rate: 1e-4, Batch size: 8, No Early Stopping. Following Fuchs et al. (2023), we report BLEU and LRML F1-Scores averaged over three runs with different random seeds and evaluate on random test splits (RTS) and document test splits

(DTS). The RTS is 8:1:1 for training, validation, and testing (i.e., 576/71/71 samples). Since the DTS in Fuchs et al. (2023) had fewer samples, we randomly selected training and validation samples to match the RTS (i.e., 576/71/55 samples). We adjust the generation arguments to limit duplicated outputs: Repetition penalty: 1.2, No repeat n-gram size: 9 and 13 (after and before shortening the expressions in Section 3.1), Beam size: 3. Post-processing of model outputs was conducted to fix the tree structure based on the if- and then-keywords.

The future work proposed by Fuchs et al. (2023) was addressed to improve the consistency of the LRML rules by addressing the following issues:

- Logical consistency: Ensure that if-statements contain only selection and applicability criteria (Ilal and Günaydın 2017).
- LRML order: The order of LRML expressions is aligned to the legal text.
- Translation consistency: The same types of phrases are translated the same way.
- Granularity: The granularity of concepts is aligned to IFC, UniClass and Omniclass.
- Remove clauses deemed untranslatable: Remove clauses such as the ones in B1/AS1 describing textual changes to referenced standards (Fuchs et al. 2023).
- Complex expressions: Define variables and use loops only where necessary.

4.1 Reversible Intermediate Representations

The transformations discussed in Section 3.1 were applied step by step. Since the And/Or-transformation was the most questionable, the experiments were conducted with and without this transformation. Table 1 shows the results of the experiments. Most significantly, the training time could be decreased to nearly a quarter of the initial time. Furthermore, the F1-Score for the random split increased by 2% when applying all transformations. The unit and expression transformations achieved the most significant improvements. The DTS led to somewhat counterintuitive and unstable results. The Atom-transformation with the And/Or-transformation leads to worse outcomes, but without the And/Or-transformation, it leads to the best result. Because of the inconsistent results, the smaller size, and the high sample similarity of the DTS, and because F1-Scores were used for the model selection, we rely primarily on the RTS F1-Score and use the LRML representation with all transformations for the experiments in Sections 4.2 to 4.4.

Table 1: Results for reversible IRs. We report the average Runtime, BLEU and F1-Score for DTS and RTS and the standard deviation in brackets.

IR	Runtime	DTS - BLEU	DTS - F1-Score	RTS - BLEU	RTS – F1-Score
Baseline	4672	65.1% (1.1)	55.9% (0.3)	68.4% (0.6)	64.4% (0.4)
+ Unit	4063	66.3% (1.1)	57.0% (2.0)	71.2% (1.0)	65.2% (0.6)
+ And/Or	2437	62.9% (2.0)	55.6% (1.1)	71.6% (1.9)	65.3% (0.9)
+ Atom	4703	64.9% (3.5)	55.6% (1.2)	71.7% (1.6)	65.2% (0.9)
+ Atom (w/o And/Or)	4256	68.1% (1.0)	**58.5% (0.8)**	70.5% (1.4)	65.4% (1.0)
+ Expression	1484	62.7% (5.0)	56.4% (1.2)	71.7% (0.7)	65.8% (1.3)
+ Expression (w/o And/Or)	1952	66.2% (0.2)	55.5% (0.6)	71.6% (3.0)	64.9% (1.9)
+ Loop	1270	64.7% (0.3)	55.4% (1.2)	71.1% (1.1)	**66.4% (0.3)**
+ Loop (w/o And/Or)	1548	66.3% (2.1)	56.6% (0.8)	71.2% (0.4)	65.9% (0.6)

4.2 Lossy Intermediate Representations

Table 2 shows the effectiveness of the lossy IRs described in Section 3.2. Using expressions as IR improved the DTS results by 1.7% but only by 0.5% for the RTS in the best setup. In contrast, for RTS, there are consistently high improvements using entities as IR, with up to 4.6% for the multi-task model trained on both the predicted and the oracle entities. Continuing the training from Model 1 worked in most cases better than training a new model from scratch. This might be the case since Model 1 has already learned to use the regulation clause and does not rely too much on the IR after refinement. This is especially noticeable for the expression IR since the separately trained model has F1-Scores of 0% when removing the IR. Also, this model scores close to 100% given the oracle expressions, which indicates it primarily relies on copying the inputs. A significant challenge is the correct generation of the IR. E.g., extracting the entities for the DTS had only a BLEU-Score of 45.3%. For the RTS, the BLEU-Score was 62.9%. These BLEU-Scores might also explain why this IR was only helpful for the RTS. The low BLEU-Scores and the oracle results indicate that we could improve the parsing performance by enhancing entity extraction. A dictionary-based approach could be investigated in future work.

Table 2: Results for lossy IRs in F1-Scores. We report two IR experiments: Entities and Expressions (EXPR). We differentiate between training separate models (SEP) and refining Model 1 (MUL).

IR	Model	Data	DTS-Oracle	RTS-Oracle	DTS-w/o IR	RTS-w/o IR	DTS	RTS
ENTITY	SEP	Prediction	75.6%	80.4%	38.6%	36.4%	55.5%	68.2%
		Oracle	77.0%	81.8%	35.7%	34.5%	55.1%	68.7%
		Combined	77.0%	82.2%	40.4%	38.5%	55.0%	69.7%
	MUL	Prediction	73.9%	79.2%	51.9%	58.1%	55.6%	69.6%
		Oracle	76.0%	82.3%	48.2%	55.6%	55.0%	68.9%
		Combined	74.9%	82.0%	51.5%	59.4%	55.1%	**71.0%**
EXPR	SEP	Prediction	92.0%	94.9%	0.0%	0.0%	56.4%	65.0%
		Oracle	97.6%	97.5%	0.0%	1.0%	56.0%	64.4%
		Combined	97.0%	96.9%	0.0%	0.3%	56.2%	65.1%
	MUL	Prediction	72.5%	85.3%	53.2%	62.1%	**57.1%**	66.9%
		Oracle	97.5%	97.6%	26.6%	32.1%	56.0%	64.5%
		Combined	91.1%	94.8%	51.8%	62.9%	56.3%	66.4%

4.3 Paraphrases as Intermediate Representation

Using paraphrases as IR gives improvements of up to 2.3% for the RTS (see Table 3). It is unclear if the main reason for the improvement is the IR or the additional training data since having two versions of input resulting in the same output could also be beneficial for learning. By training a model with both the original and paraphrased clauses, we can confirm that this might be the case (i.e., 70.1% F1-Score for the RTS). The DTS results deteriorated, possibly caused by the weak paraphrasing performance: 35.6% and 56.0% BLEU for DTS and RTS. Nevertheless, testing with the oracle paraphrases indicates that a simplified systematic formulation of legal texts, or potentially a natural language rewriting, might allow legal experts to produce LRML or other semantic representations without much knowledge engineering experience. This could also be integrated into a user-friendly semi-automated LRML translation interface.

Table 3: Results for paraphrases (PARA) as IR measured in F1-Scores. SEP: Separate models; MUL: Refining Model 1

IR	Model	Data	DTS-Oracle	RTS-Oracle	DTS	RTS
PARA	SEP	Prediction	71.4%	82.1%	53.8%	66.7%
		Oracle	71.6%	81.5%	52.7%	66.4%
		Combined	73.0%	84.8%	53.9%	**68.9%**
	MUL	Prediction	69.2%	81.1%	53.3%	67.7%
		Oracle	70.6%	80.6%	53.0%	66.7%
		Combined	71.7%	83.8%	54.6%	**68.9%**

4.4 Self-reflection

The T5 model was tested on its ability to correct mistakes. As indicated in previous results and according to Table 4, a newly trained model does not seem to learn anything but copy the LRML rules. Also, the multi-task model trained with the oracle LRML rules only learned to copy the inputs. But it is also noticeable that in contrast to separately trained models, the multi-tasking model does not entirely unlearn predicting the LRML if no IR is given. Slight improvements were achieved in the multi-tasking setup and training with the predictions. Furthermore, our overfitting hypothesis is strengthened through the last experiment, where the F1-Score increased to 70.9% with the initial LRML rules predicted after three epochs. In this experiment, training with the oracle and predictions decreases the performance since the model starts copying the inputs. Nevertheless, the best-performing model was better when removing the predicted LRML and only slightly better with the oracle LRML. So, whether the model made meaningful corrections or whether the appended LRML rules improved the training process is questionable.

Table 4: Results for self-reflection experiments measured in F1-Scores. SEP: Separate models; MUL: Refining Model 1; REFLECT: LRML generated with the best IR model; OVERFIT: LRML generated after three epochs

IR	Model	Data	DTS-Oracle	RTS-Oracle	DTS-w/o IR	RTS-w/o IR	DTS-F1	RTS-F1
-	T5	Original	-	-	-	-	55.4%	66.4%
REFLECT	SEP	Prediction	98.6%	97.4%	0.0%	0.0%	55.3%	66.3%
		Oracle	99.3%	98.5%	0.0%	0.0%	55.5%	66.4%
		Combined	98.9%	98.2%	0.0%	0.0%	55.4%	66.5%
	MUL	Pred	76.1%	85.7%	53.1%	66.7%	55.9%	67.3%
		Oracle	98.8%	98.3%	45.2%	54.2%	55.4%	66.4%
		Combined	90.8%	95.0%	52.3%	65.5%	55.5%	67.7%
OVERFIT	SEP	Pred	73.6%	87.5%	0.0%	0.0%	56.4%	65.5%
		Oracle	99.3%	98.5%	0.0%	0.0%	55.5%	66.4%
		Combined	98.9%	98.2%	0.0%	0.0%	55.4%	66.5%
	MUL	Pred	59.8%	73.3%	56.4%	**71.0%**	**56.5%**	**70.9%**
		Oracle	98.8%	98.3%	45.2%	54.2%	55.4%	66.4%
		Combined	90.8%	95.0%	52.3%	65.5%	55.5%	67.7%

5. Discussion

Applying the reversible IR described in Section 3.1 reduced the number of sub-word tokens of the LRML rule from an average of 181 and a maximum of 911 to an average of 82 and a maximum of 408. This reduction can be considered an essential step towards the lossy IR and self-reflection experiments where the generated IR was appended to the input text. The entity IR and self-reflection then brought the main F1-Score gains.

A more general contribution is the experimental outcome of Section 4.4, which indicates that having knowledge of previous outputs can positively influence the training. This could be due to having an example translation, which, in our case, is based on the corresponding input. This phenomenon should be further researched, including the adaptability to related tasks.

There are two primary threats to the validity of this study. First, we used the DTS proposed in previous work to allow comparability between the results. Some of the results for the DTS do not reflect the conclusions drawn from the RTS. While we expect this to be related to the DTS test set's smaller size and diversity, further examination will be required. Second, the LRML dataset is still a work in progress and shall be extended in future work. Especially an independently created test set is planned to strengthen the trustworthiness of the results and remediate the issues related to the DTS.

6. Conclusions

This paper shows that reversible intermediate representations (IR) can reduce the training time to almost a quarter and improve the semantic parsing quality by 2%. Using entity extraction as IR leads to further improvements of up to 4.6%, and manual simplification of legal clauses could help enhance the parsing quality by 18.4% to 84.8%.

The significance of these results is threefold. First, shorter training and inference time allow potential collaboration between expert and machine. The semantic parser can generate initial translations, which experts can improve. Partial translations and auto-completion can limit the manual effort required for these improvements. Second, the increased quality of the LRML translation is a step towards fully automated translation. While the scores are not yet perfect, the outputs are very promising. Currently, the evaluation is against a single possible translation. In future work, we must consider logically equivalent solutions and add a more fine-grained assessment closer to the representation's end use. Third, while entities and paraphrases as IRs bring improvements by themselves, integrated into a user interface, they would be reviewable and improve interpretability. IRs also open a new way of expert collaboration by allowing experts to annotate entities to improve the translation or to paraphrase the legal clause instead of manipulating the LRML rule directly.

In future work, the translation quality could be improved by introducing better entity extraction and using large language models or pre-training for text simplification. The self-reflection capability could be improved by using data augmentation to generate a more versatile set of translation mistakes, which the correction model should learn to fix. Additionally, the ability of large language models to generate LRML directly in a few-shot learning setup should be explored.

Acknowledgment

This research was funded by the University of Canterbury's Quake Centre's Building Innovation Partnership (BIP) programme, which is jointly funded by industry and the Ministry of Business, Innovation and Employment (MBIE).

References

Dong, L. and Lapata, M. (2018). Coarse-to-Fine Decoding for Neural Semantic Parsing. In: *Proc. of the 56th Annual Meeting of the Association for Computational Linguistics*, Volume 1, pp. 731-742.

Dimyadi, J., Governatori, G. and Amor, R. (2017). Evaluating LegalDocML and LegalRuleML as a standard for sharing normative information in the AEC/FM domain. In *Proceedings of the Joint Conference on Computing in Construction (JC3)* (Vol. 1, pp. 637-644).

Dimyadi, J., Fernando, S., Davies, K. and Amor, R. (2020). Computerising the New Zealand building code for automated compliance audit. *New Zealand Built Environment Research Symposium (NZBERS)*.

Ferraro, G., Lam, H.P., Tosatto, S.C., Olivieri, F., Islam, M.B., van Beest, N. and Governatori, G. (2020). Automatic extraction of legal norms: evaluation of natural language processing tools. In *New Frontiers in Artificial Intelligence: JSAI-isAI International Workshops, JURISIN, 2019, Revised Selected Papers 10* (pp. 64-81). Springer International Publishing.

Fuchs, S. and Amor, R. (2021). Natural Language Processing for Building Code Interpretation: A Systematic Literature Review. In: *Proc. of the Conference CIB W78*, Vol. 2021, pp. 11-15.

Fuchs, S., Dimyadi, J., Witbrock, M. and Amor, R. (2023). Training on digitised building regulations for automated rule extraction. In *ECPPM 2022-eWork and eBusiness in Architecture, Engineering and Construction 2022* (pp. 428-435). CRC Press.

Fuchs, S., Witbrock M., Dimyadi, J. and Amor, R. (2022). Neural Semantic Parsing of Building Regulations for Compliance Checking. In: *Proc. of the Conference CIB W78*, Vol. 2022.

Herzig, J., Shaw, P., Chang, M.W., Guu, K., Pasupat, P. and Zhang, Y. (2021). Unlocking compositional generalization in pre-trained models using intermediate representations. *arXiv preprint arXiv:2104.07478*.

Ilal, S.M. and Günaydın, H.M. (2017). Computer representation of building codes for automated compliance checking. *Automation in construction*, *82*, pp.43-58.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W. and Liu, P.J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, *21*(1), pp.5485-5551.

Shinn, N., Labash, B. and Gopinath, A. (2023). Reflexion: an autonomous agent with dynamic memory and self-reflection. *arXiv preprint arXiv:2303.11366*.

Solihin, W. and Eastman, C. (2015). Classification of rules for automated BIM rule checking development. *Automation in construction*, *53*, pp.69-82.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

Zhang, R. and El-Gohary, N. (2022a). Hierarchical Representation and Deep Learning–Based Method for Automatically Transforming Textual Building Codes into Semantic Computable Requirements. *Journal of Computing in Civil Engineering*, *36*(5), p.04022022.

Zhang, R. and El-Gohary, N. (2022b). Natural language generation and deep learning for intelligent building codes. *Advanced Engineering Informatics*, *52*, p.101557.

Zhang, Z., Nisbet, N., Ma, L. and Broyd, T. (2023). Capabilities of rule representations for automated compliance checking in healthcare buildings. *Automation in Construction*, *146*, p.104688.

Zhou, P. and El-Gohary, N. (2022). Semantic Information Extraction of Energy Requirements from Contract Specifications: Dealing with Complex Extraction Tasks. *Journal of Computing in Civil Engineering*, *36*(5), p.04022025.