Improving DRS-to-Text Generation Through Delexicalization and Data Augmentation

Improving DRS-to-Text Generation through Delexicalization and Data Augmentation

Muhammad Saad Amin[1 [0000−0002−7002−9373]], Luca Anselma[2 [0000−0003−2292−6480]], and Alessandro Mazzei[3 [0000−0003−3072−0108]]

Department of Computer Science, University of Turin, Turin, Italy
{muhammadsaad.amin, luca.anselma, alessandro.mazzei}@unito.it

Abstract. Text generation from Discourse Representation Structure (DRS), is a complex logic-to-text generation task where lexical information in the form of logical concepts is translated into its corresponding textual representation. Delexicalization is the process of removing lexical information from the data which helps the model be more robust in producing textual sequences by focusing on the semantic structure of the input rather than the exact lexical content. Implementation of delexicalization is even harder in the case of the DRS-to-Text generation task where the lexical entities are anchored using WordNet synsets and thematic roles are sourced from VerbNet. In this paper, we have introduced novel procedures to selectively delexicalize proper nouns and common nouns. For data transformations, we propose to use two types of lexical abstractions (1): WordNet supersense-based contextually categorized abstraction; and (2): abstraction based on the lexical category associated with named entities and nouns. We present many experiments for evaluating the hypotheses of delexicalization in the DRS-to-Text generation task by using state-of-the-art neural sequence-to-sequence models. Furthermore, we also explored data augmentation through delexicalization while evaluating test sets with different abstraction methodologies i.e., with and without supersenses. Our experimental results proved the effectiveness of model generalizability through delexicalization while comparing it with the results of fully lexicalized DRS-to-Text generation. Delexicalization resulted in an improved translation quality with a significant increase in evaluation scores.

Keywords: Delexicalization · Data augmentation · Discourse representation structure · Formal meaning representation · Neural DRS-to-Text generation · Super senses.

1 Introduction

Delexicalization is the process of removing lexical knowledge from the data to make it generalized by emphasizing more on the syntactic structure and sentence patterns rather than the specific semantic content [1]. It is a very well-known technique in audio and speech processing where delexicalization and relexicalization procedures are used to improve dialogue systems through the preservation

of prosodic features for text-to-speech applications [2]. Contributions of data delexicalization are also becoming very popular in many applications of natural language processing. The motivation for using delexicalization is to enhance the model's ability to generate more natural sentences with better grammar and syntactic structure [3]. This also highlights the model's ability to perform well for unseen or out-of-vocabulary words [4]. Recent neural approaches to language generation have achieved peak performance through end-to-end training, they have also gained popularity in a variety of natural language generation (NLG) applications, including concept-to-text generation, machine translation (MT), and summarization [5].

Commonly used procedures of delexicalization in NLG include named entity dependent or exact delexicalization, language agnostic delexicalization, and delexicalization through pre-trained language models [6]. It is important to understand that the requirement to create a pragmatically correct text while maintaining the semantic and syntactic structure of the sentence makes data delexicalization even harder. In fact, grammatically inaccurate or out-of-scope delexicalized textual input might lead to poor model performance [7]. In the delexicalization process, lexical entities are replaced with a placeholder e.g., "Tom knocked at the door.", considering nouns only, the modified example is "[placeholder] knocked at the [placeholder].". Here, the placeholder can be any generalized tag depending on the type of delexicalized applied to the data.

Using transformers and encoder-decoder-based neural models, researchers working on text generation from concept or meaning representations –i.e. graph-based abstract meaning representations (AMR) [8,9,10], RDF-triples [11], or discourse representation structures (DRS) [12,13,14,15,16,17]– have recently focused on generating text from logical representations and vice versa. In this paper, we emphasize the role played by data delexicalization in formal meaning representation (DRS), in the context of neural DRS-to-Text generation tasks. DRS originated from discourse representation theory (DRT) that lists formal meaning representation in the form of first-order logic (FOL) [18]. Textual information is represented in DRS as events, concepts, and entities. For example, names are discourse referents that are usually represented as variables in DRS, along with the logical relations that exist between these entities, such as quantifiers, conjunctions, negations, disjunctions, etc. Lexical information like nouns, adjectives, and adverbs are represented as logical concepts that are associated with English WordNet, and verbs are represented as VerbNet roles [19,20,21]. As an example, a graphical representation of the DRS for the text "Tom was carrying a bucket of water." is shown in Fig. 1.

In a wide spectrum of data-to-text generation tasks having different flavors of input representations including an AMR graph [8,9,10], an RDF triple [11], or a table [22], neural DRS-to-Text generation is an application of the same stream having DRS as the data. The neural model takes a logical representation (DRS) as an input and generates the corresponding text as output [15,16].

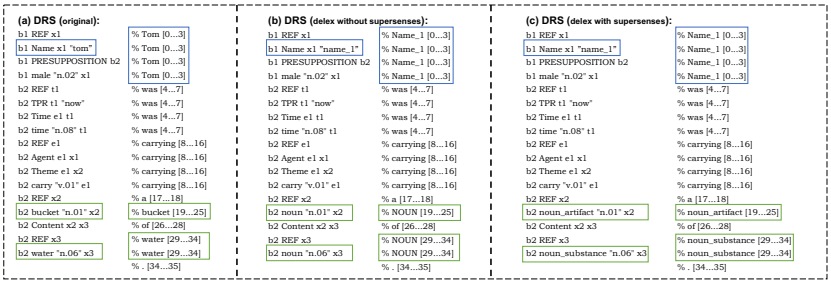

Fig. 1. Graphical representation of the DRS with lexical (a) and delexical without (b) and with (c) supersenses for the text "Tom was carrying a bucket of water.".

Compared to different logical or conceptual representations of data i.e., AMR and RDF, we choose DRS because it is more expressive and it can represent a wide range of semantic phenomena that can express logical relations for long sentences. Furthermore, using DRS can provide a very fine-grained investigation of the logical form of individual sentences, for DRS-to-Text generation, focusing more on syntactic or discourse-level structure rather than semantic contents only. Stated otherwise, "by modifying a DRS's meaning in a controlled manner, the systems' robustness can be closely observed and evaluated accordingly." [15]. However, since large language models (LLMs) have no prior knowledge of DRS, they may introduce noise in the data if used for delexicalization or relexicalization. This resilience property discourages the use of LLMs for data delexicalization–see Section 4.2.

In this study, we specifically develop and evaluate data delexicalization for the categories of (i) proper nouns (PNs) and (ii) common nouns (CNs), utilizing the robustness aspect of neural DRS-to-text generation. Specifically, we developed and evaluated the process of substituting the actual lexical information of the original DRS-text pairs with the placeholders by modifying the proper nouns and common nouns in the dataset. We investigate different approaches, with and without the use of supersenses, to generate new delexicalized training sentences. These approaches aim to investigate the importance of lexical/semantic information and the curial role played by the syntactic structure of logical representation in the task of neural DRS-to-text generation. Furthermore, we also applied a novel approach of data augmentation through delexicalization by combining lexical and delexical flavors of data. We want to understand the important role played by semantic knowledge in the identification of the syntactic structure of the sentence while evaluating a delexicalized test set.

This brings us to the following research questions:

1. How to delexicalize logical data representations such as DRS where lexical entities are strongly connected with external lexical databases like WordNet and VerbNet?
2. Can supersense contribute to enhancing the generalization power of the neural models when used for nouns?

3. What would be the behavior of the model if we augment logically delexicalized data with fully lexicalized one?
4. Can delexicalization and augmentation increase the model performance?
5. What is the behavior of seq-to-seq neural models in the case of pre-training (bi-LSTM) and fine-tuning (byT5)?
6. How do general-purpose large language models like chatGPT and Claude incorporate DRS when given as a prompt?

To the best of our knowledge, this study is the first attempt to explore data delexicalization in neural DRS-to-Text generation. Apart from some initial works on data augmentation of verbs and nouns in DRS-to-Text generation [16,17], this is the first work of data augmentation through delexicalization to evaluate the syntactic structure of the generated text through relexicalization.

The statistical structure of the neural network makes it difficult to analyze the type of information that the system has actually learned. Generally, the network learns that the verb follows the subject (e.g. grammatical competence) when we give a concrete example, such as "Brad Pitt is an actor", and/or that men can be actors (semantic and pragmatic knowledge) or that a particular man is an actor (world knowledge). How can we take advantage of the multi-level structure of neural learning? Our study has the side effect of raising these theoretical questions for further investigation.

The remaining paper is structured as follows: in Section 2, we describe the procedure used for noun delexicalization with and without supersenses. In Section 3, we give an insight into the architecture of the neural DRS-to-text pipelines. In Section 4, we describe the experimental results of DRS-to-text generation through (1) automatic metric-based and pre-trained model-based evaluations on a standard test set, (2) a reduced test set comparing our neural systems with two general LLMs, and (3) an error analysis of the model-generated text; Finally, the paper is concluded in Section 5 with a leading section of limitations.

2 Logical Delexicalization with Nouns

When it comes to the application of neural DRS-to-Text generation, logical data delexicalization seems to be a complex task. Each example in the training set consists of a logical input (DRS) and the corresponding text that goes with it. Both types of data representations need to be monitored when making methodological changes to the training data, as the neural network treats them as pairs of input values. The data transformations should therefore take into account the order of the meaning representations and the text translations and should be equal and balanced for both elements. We used different delexicalization approaches to generalize PNs and CNs in the DRS-to-Text generation task. We used the Parallel Meaning Bank[1] (PMB) dataset, which is created in the standard train-dev-test split, in its gold version.

[1] The PMB is developed at the University of Groningen as part of the NWO-VICI project "Lost in Translation – Found in Meaning" (Project number 277-89-003), led by Johan Bos.

Table 1. Different flavors of delexicalization applied to the dataset referring to data transformations without and with supersense. (Transf. = Transformation; PN placeholders in blue; CN placeholders in green).

Transf. Type	Lexicalized Text	Delexicalized Text
Delex w/o SS	Brad Pitt is an actor. The Mona Lisa hung above the antique table. Paris is a beautiful city. Noah and Sophia watched a movie at the local theater.	Name_1 is a NOUN. The Name_1 hung above the antique NOUN. City_1 is a beautiful NOUN. Name_1 and Name_2 watched a NOUN at the local NOUN.
Delex with SS	Brad Pitt is an actor. The Mona Lisa hung above the antique table. Paris is a beautiful city. Noah and Sophia watched a movie at the local theater.	Name_1 is a noun_person. The Name_1 hung above the antique noun_artifact. City_1 is a beautiful noun_location. Name_1 and Name_2 watched a noun_communication at the local noun_artifact.

Fig. 1 displays the graphical representations of the DRS transformations from fully lexical representation (a), to delexical without supersense (b), and delexical with supersenses (c) highlighting PN transformation (in blue) and CN (in green). The DRS (a) generates the sentence "Tom was carrying a bucket of water." reflecting a fully lexical translation of the DRS, while the DRS (b) generates "Name_1 was carrying a NOUN of NOUN." representing a more generalized delexicalization approach, and the DRS (c) generates "Name_1 was carrying a noun_artifact of noun_substance." producing text with the contextual control over delexicalized placeholder through the use of supersenses. Other textual examples are listed in Table 1 to have a clear understanding of the proposed delexicalized approaches.

2.1 Proper Noun Delexicalization

The proper names of a person (PER), which includes both male and female names, and of a place (GPE), which includes city, state, country, and island names, are the two specific Named Entity (NE) categories that we examined for PNs. To extract proper names from the text, we used the spaCy NE Recognizer https://spacy.io. For PER and GPE, there is a total of 3773 instances of PNs. The proper nouns are further divided into the following categories: person names account for 57%, city names 30%, state names 6%, country names 6%, and other types, such as island names, 1%.

While dexicalizing PNs, we have adopted two approaches to replace named entities with placeholders to analyze the impact of model generalizability by removing lexical information from the dataset. (1) Replacing named entities with custom placeholders i.e., person_name, city_name, state_name, country_name, etc. This substitution resulted in 6 different custom placeholders for all the named entities under observation in the dataset. (2) Replacing named entities with spaCy-defined placeholders i.e., PER and GPE. This substitution resulted in only 2 placeholders for all named entities in the dataset. For example, in "Tom is living in Boston now.", through the first approach we get "Name_1 is living in city_1 now." while with the second approach, we get "PER is living in GPE now.".

While experimenting with delexicalized PNs, we found custom delexicalization more useful as compared to spaCy-oriented delexicalization. This is because, for model evaluation through relexicalization, custom delexicalization helps to sustain true pragmatics of the logical input while preserving true semantic correlation between delexicalized named entities. While in the case of spaCy-defined placeholders, the model often confuses the exact location of the named entity placeholder in the delexicalized translation of the meaning representation. For example, in "Tom went to London and called Mary.", through custom delexicalization, we get "Name_1 went to City_1 and called Name_2." which is semantically more understandable to the neural model. While in the case of the spaCy-oriented placeholder, the model confused the order of semantic entities and generated "PER went to PER and called GPE." or "GPE went to PER and called PER.". Therefore, for all of our further experiments, we have used custom delexicalization for named entities. Some extra examples demonstrating the delexicalization procedures are listed in Table 1.

2.2 Common Noun Delexicalization

CNs sustaining the true contextual sense of the sentence are very important lexical entities, especially in the logical input representations like DRS. To extract CNs from the text, we used spaCy again which resulted in the extraction of 6193 lexical entities from the dataset. For delexicalization, we have adopted two different procedures for replacement. First, replacing all lexical entities of CNs with one spaCy-based placeholder i.e., NOUN. This type of delexical substitution makes the data fully generalized with only one placeholder for all lexical entities of CNs. Second, a novel WordNet-based supersense tagging (SST) approach that proved helpful in sustaining the categorical and contextual sense of the sentence. With supersenses, we identified CNs from the 26 lexicographic categories of WordNet based on data instances. These categories included noun_act, noun_artifact, noun_body, noun_cognition, noun_communication, noun_event, noun_feeling, noun_food, noun_group, and noun_motion, etc. A graphical representation of the categorical distribution of the CN through supersenses is shown in Fig. 2.

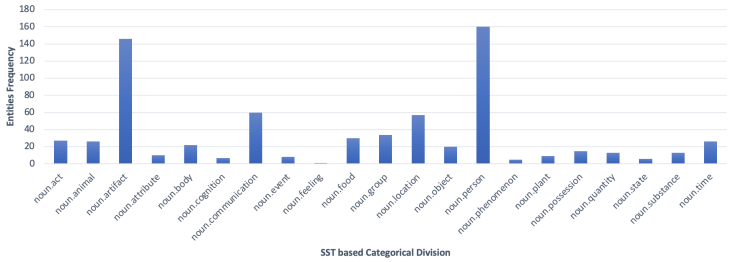

Fig. 2. Supersense-based categorical division of common nouns in Gold-PMB dataset.

For example, in "A cat is sitting on the chair.", if we delexicalize with a 1 spaCy placeholder, we have "A NOUN is sitting on the NOUN.". But if we perform delexicalization through supersenses, we have "A noun_animal is sitting on the noun_artifact.". The motivation for using delexicalization is to only extract lexical knowledge from the data without breaking semantic correlation and sentence structure. Through supersense delexicalization, we are facilitating the model to understand the delexicalized input structure more precisely as compared to spaCy-based delexicalized data having one placeholder for all CNs. For noun delexicalization, we have experimented with both types of delexicalization procedures i.e., with and without supersenses. Some examples demonstrating the delexicalization procedures with and without supersenses are listed in Table 1.

3 Three Neural DRS-to-Text Pipelines

DRS-to-text generation is a complex task that requires computationally efficient neural models. In this scope, we have used three different neural architectures in our implementation pipelines. The encoder-decoder-oriented recurrent sequence-to-sequence neural network with (1): character-based lexical encoding (CB-bi-LSTM henceforth); and (2): word-based lexical encoding (WB-bi-LSTM henceforth), both having bidirectional Long short-term memory (LSTM) layers [16]. In addition, we have also fine-tuned byT5: a state-of-the-art version of the Transformer family to improve DRS-to-text generation (FT-byT5 henceforth) [17]. Our first two models point towards the procedure of training a neural model from scratch and our third model focuses on fine-tuning a pre-trained LLM for task-specific applications.

We decided to adopt also the Transformer-based model for our experimental implementation because we are aware that the most advanced neural models for generating text from structured input representations use complex transformer-oriented architectures. However, the purpose of this paper is not to present the best-performing system but to analyze the consequences of data delexicalization in the context of neural DRS-to-text generation.

It should be noted that the main differences between CB-bi-LSTM and WB-bi-LSTM are based on the representations of the input and output data, i.e. characters or words, and how well they can handle words that are not in the dictionary (OOV). Since CB-bi-LSTM analyses character sequences, it can easily handle OOV words, while WB-bi-LSTM may have difficulty handling OOV words, as this depends more on the included vocabulary. We believe that the impact of certain data delexicalization strategies may be influenced by these two different approaches.

The model architecture and hyperparameters used in our experiments for sequence-to-sequence implementation are likewise [16], that focus on LSTM-based encoder-decoder cells with an epoch-based learning decay method that uses Adam as an optimizer. The validation metric we use is cross entropy, while the cost function is ce-mean. Table 2 contains other important hyperparameters that we have used for bi-LSTM experiments. When using AdamW as the

optimizer and fine-tuning the model over 15 epochs, we used the default hyperparameter settings of byT5 in our transformer-based version and made minor adjustments to the batch size, update steps, and learning rates. Table 2 contains a list of all hyperparameter settings of the FT-byT5 model just like used in [17].

Table 2. Hyperparameter settings for CB-bi-LSTM, WB-bi-LSTM, and FT-byT5.

bi-LSTM (CB|WB)		FT-byT5	
HyperParameters	**Values**	**HyperParameters**	**Values**
Embedding Dimensions	300	Batch size	15
Enc/Dec Cell	LSTM	Update steps	8
Enc/Dec Depth	2	Max learning Rate	1e-4
Mini-batch	48	Min learning Rate	1e-5
Normalization Rate	0.9	Warmup updates	3000
lr-decay	0.5	Max decay steps	30000
lr-decay-strategy	Epoch	No. of epochs	15
Optimizer	Adam	Optimizer	AdamW
Validation Metric	Cross-Entropy		
Cost-Type	ce-mean		
Beam Size	10		
Learning Rate	0.002		

We have used the Parallel Meaning Bank (PMB) dataset in its English edition. Among the different dataset flavors–Gold, Silver, and Bronze–we focused on the Gold dataset, which has a fully manual annotation and correction process. Gold-PMB uses training, development, and test files with 6620, 885, and 898 data instances, respectively, according to the standard split of the dataset. Two sample transformation methods were used to delexicalize the dataset. (1) Delexicalization of PNs through custom placeholder (see section 2.1) and CNs with a spaCy-oriented placeholder (see Section 2.2) (delex1 henceforth). (2) Delexicalization of PNs through custom placeholders and CNs with supersense-based placeholders (delex26 henceforth). For experimentation, we have delexicalized DRS-text pairs of train and development sets and only DRS for the test set. We kept the text of the test set the same, i.e., fully lexical, so that after performing relexicalization to the model-generated text, we can evaluate the model performance through comparison with the gold test set. A graphical representation of the pipeline is shown in Fig. 3.

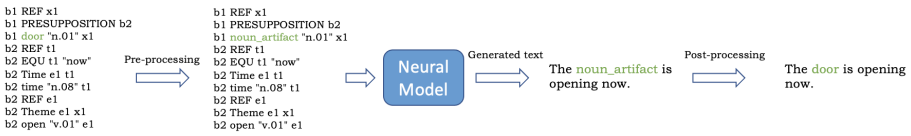

Fig. 3. Implementation pipeline of DRS-to-Text generation through the pre-processing (delexicalization) and post-processing (relexicalization) of the text "The door is opening now.".

A further original contribution of this paper is the analysis of data augmentation through logical data delexicalization. We want to understand the impor-

tant role played by data augmentation thus focusing on the data-hungry nature of the neural models. For this implementation, we have conducted four different experiments. (1) Concatenating fully lexical and fully delexicalized (delex1) data examples and evaluating model performance for the delex1 test set. (2) Concatenating fully lexical and fully delexicalized (delex26) data examples and evaluating model performance for the delex26 test set. (3) Concatenating fully lexical, fully delexicalized (delex1), and fully delexicalized (delex26) data examples and evaluating model performance for the delex1 test set. (4) Concatenating fully lexical, fully delexicalized (delex1), and fully delexicalized (delex26) data examples and evaluating model performance for the delex26 test set.

4 Experimental Results

For experimental implementation, we have used 3 different neural models with the motivation of pre-training and fine-tuning a sequence-to-sequence model. Table 3, Table 4, and Table 5 list the experimental findings of all the experiments for pre-training characters and words tokenization oriented bi-LSTM models and fine-tuning transformer-based byT5 model respectively. Based on the models used for experiments, we categorize the results in the sense that we list all CB-biLSTM results in Table 3, all WB-biLSTM results are mentioned in Table 4, and Table 5 displays all FT-byT5 results having delexicalized flavors of logic-text pairs with and without supersenses. Furthermore, we have applied data augmentation through delexicalization by concatenating fully-lexical and respective flavors of delexicalized datasets to analyze the model performance. For experimental evaluations, we perform automatic metric-based evaluation (see Section 4.1) and comparison of large language models (LLMs) like chatGPT-3.5 and Claude-2.0 (see Section 4.2) with our best-delexicalized model to compare powerful general-purpose LLMs with our delexicalized model.

4.1 Evaluation with automatic metrics

For a clear understanding of the results in Table 3, Table 4, and Table 5, we have split the evaluation scores into 4 different blocks. The first block (exp. 01) is our baseline with the DRS-to-text generation results on the model trained or fine-tuned on a fully lexical dataset i.e., without delexicalization. Our second block (exp. 02-03) represents results for 2 different delexicalization approaches with supersenses (exp. 02) and without supersenses (exp. 03). Here the model is pre-trained or fine-tuned on only delexicalized data samples. In the third block, we have results for data augmentation through delexicalization by concatenating fully lexical data examples with one flavor of delexicalization i.e., with supersenses (exp. 04) and without supersenses (exp. 05) respectively. Finally, in the fourth block, we have compound augmentation results with the concatenation of fully lexical and delexicalized data examples with and without supersenses. With the same training examples, we evaluate two different test sets: (1) a delexicalized test set with supersenses (exp. 06), and (2) a delexicalized test set without supersenses (exp. 07).

Table 3. CB-biLSTM results for delexicalization with supersenses (delex26) and without supersenses (delex1) on Gold-PMB dataset. (Note: MET. = METEOR; RUG. = ROUGE; CMT. = COMET; B.Scr = BERT-Score; CB = character-based; tst = testing)

Exp.	Implementation Type	BLEU	chrF	MET.	RUG.	CMT.	B.Scr
CB-01	Fully Lexical	46.80	66.22	39.12	72.54	79.33	95.31
CB-02	Delex26	48.51	63.58	40.34	74.24	75.37	94.67
CB-03	Delex1	*51.10*	61.24	*40.80*	*74.43*	74.11	94.26
CB-04	Lex+delex26	*60.45*	***71.12***	*46.46*	*80.78*	*82.48*	*96.19*
CB-05	Lex+delex1	57.68	69.85	44.33	78.62	81.63	95.94
CB-06	Lex+delex26+delex1(tst delex26)	60.95	70.52	46.25	80.70	81.94	96.10
CB-07	Lex+delex26+delex1(tst delex1)	***61.38***	*70.66*	***46.53***	***81.41***	***82.60***	***96.20***

Table 3 lists our results for the pre-training of the bi-LSTM model for the char-based tokenization approach (CB-biLSTM). Compared to the baseline (CB-01), CB-biLSTM wins in all aspects of experiments run through delexicalization with and without supersenses and augmentation. In fact, data augmentation always helps in improving model performance by increasing model generalization ability. In the case of individual data delexicalization (CB-02, CB-03), delexicalization without supersenses wins (CB-03, in italics). With data augmentation, delexicalization with supersenses shows a good improvement and gains significantly (CB-04 in italics) as compared to delexicalization without supersenses (CB-05). Finally, with compound augmentation, the CB-biLSTM model generalizes more for a test set that does not contain any supersenses (CB-07, in bold and italics). CB-07 also indicates the highest score in all the experimental formats of the CB-biLSTM model.

Table 4. WB-biLSTM results for delexicalization with supersenses (delex26) and without supersenses (delex1) on Gold-PMB dataset. (Note: MET. = METEOR; RUG. = ROUGE; CMT. = COMET; B.Scr = BERT-Score; WB = word-based; tst = testing)

Exp.	Implementation Type	BLEU	chrF	MET.	RUG.	CMT.	B.Scr
WB-01	Fully Lexical	40.36	56.06	33.42	65.26	73.66	94.44
WB-02	Delex26	*52.32*	*62.77*	*41.67*	*75.37*	*76.94*	*94.95*
WB-03	Delex1	49.80	58.47	40.33	73.32	73.94	94.45
WB-04	Lex+delex26	*56.49*	***67.90***	***44.24***	***78.73***	*80.93*	95.74
WB-05	Lex+delex1	53.98	65.98	41.99	75.85	80.17	*95.75*
WB-06	Lex+delex26+delex1(tst delex26)	57.05	67.11	44.00	78.16	***80.94***	95.87
WB-07	Lex+delex26+delex1(tst delex1)	***57.07***	67.38	*44.11*	*78.42*	80.59	***95.76***

Table 4 shows the results for the WB-biLSTM model pre-trained with the delexicalized and augmented training examples with and without supersenses. Just like the CB-bi-LSTM model, the WB-bi-LSTM model also wins in all the

aspects of experimental implementation when compared with the baseline (WB-01). Unlike the CB-biLSTM model, the WB-biLSTM model shows significantly different results. For individual delexicalization results (WB-02, WB-03), the model trained on supersense-based delexicalized data wins (WB-02, in italics). Surprisingly, compound data augmentation does not seem effective for the WB-biLSTM model, as the model gets the highest scores, apart from the BLEU score, for data augmentation with supersense-based delexicalized data (WB-04 in bold and italics). While the highest BLEU score is for the compound augmentation testing the sub-set having delexicalized data without supersenses (WB-07).

Table 5. FT-byT5 results for delexicalization with supersenses (delex26) and without supersenses (delex1) on Gold-PMB dataset. (Note: MET. = METEOR; RUG. = ROUGE; CMT. = COMET; B.Scr = BERT-Score; T5 = fine-tuned byT5; tst = testing)

Exp.	**Implementation Type**	**BLEU**	**chrF**	**MET.**	**RUG.**	**CMT.**	**B.Scr**
T5-01	Fully Lexical	51.88	73.16	43.55	76.04	86.89	96.74
T5-02	Delex26	62.44	77.93	*47.50*	*82.18*	*89.47*	*97.47*
T5-03	Delex1	*62.73*	*77.85*	47.41	81.76	88.96	97.37
T5-04	Lex+delex26	*62.94*	*78.48*	*47.67*	*82.20*	*89.72*	97.40
T5-05	Lex+delex1	62.72	78.09	47.30	82.06	88.95	*97.43*
T5-06	Lex+delex26+delex1(tst delex26)	63.54	78.85	47.91	82.47	89.72	97.60
T5-07	Lex+delex26+delex1(tst delex1)	***64.22***	***78.87***	***48.07***	***82.90***	***90.16***	***97.63***

Finally, in Table 5, we show our results for the byT5 model fine-tuned on delexicalized data with and without supersenses. Overall model generalization ability is enhanced with the adaptation of data delexicalization. Furthermore, augmentation helped the model to achieve the best results in the case of compound augmentation while testing the delexicalized subset without supersenses (T5-07 in bold and italics). The influence of individual delexicalization procedures i.e., with and without supersenses (T5-02, T5-03) does not significantly affect the model's generalization power as the results are very close. Similarly, in the case of data augmentation (T5-04, T5-05), the model is not significantly improved with different flavors of delexicalization along with lexical augmentation.

Comparing the overall performance of all models, byT5 shows the best results with a fine-tuning perspective (see Table 5, T5-07) when compared with the pre-training based biLSTM models (see Table 3 and Table 4). This highlights the need for state-of-the-art sequence-to-sequence models for complex task-specific applications, e.g., DRS-to-text. In the next sections about the comparison of neural DRS-to-Text generation model with LLMs (Section 4.2) and for error analysis (Section 4.3), we will use the text generated by our best model, i.e., FT-byT5, evaluated for test set without supersenses (see Table 5, T5-07).

4.2 Comparing Neural DRS-to-Text Generation and LLMs

To get a better understanding of how well our delexicalized neural approach performs, compared to a general-purpose LLM that has not been fine-tuned for a specific task, we compare the quality of the generated text of our neural DRS-to-text systems with two LLMs, ChatGPT 3.5 [23] and Claude 2.0 [24]. We examined the performance of the LLM using both few-shot and zero-shot learning techniques. The behavior of ChatGPT did not improve even in the case of few-shot learning, while Claude gained a lot in few-shot as compared to zero-shot approaches (see Table 6).

Table 6. Evaluation of DRS-to-Text generation text for LLMs reporting scores for ChatGPT 3.5, Claude 2.0, the baseline (without delexicalization), and our best (FT-byT5) model. (Note: LLM = Large Language Model; MET. = METEOR; RUG. = ROUGE; CMT. = COMET; B.Scr = BERT Score)

LLM Type	Implementation Type	BLEU	chrF	MET.	RUG.	CMT.	B.Scr
Claude-2.0	Zero-shot learning	11.33	44.15	29.39	42.43	69.83	92.31
	Few-shot learning	*27.25*	*58.72*	*38.58*	*64.25*	*87.17*	*95.37*
ChatGPT-3.5	Zero-shot learning	*9.82*	*43.69*	*27.91*	*39.80*	*68.80*	*91.98*
	Few-shot learning	9.58	40.46	26.01	37.40	66.17	91.54
byT5	Fully lexical model	47.55	71.47	42.90	74.56	86.49	96.52
	FT-byT5 (our best model)	***61.00***	***75.96***	***45.70***	***80.02***	***88.52***	***97.29***

We considered a sample of 215 examples from the test set that were (1) evaluated using the best DRS-to-text neural models, i.e., FT-byT5 (see Table 5), and (2) in response to prompts from Claude 2.0 and ChatGPT 3.5 to obtain text produced by the models. We evaluated the generated text using automatic evaluation metrics (see Table 6). The italic score represents the best results for each model type while the bold and italics results show the best results when compared to overall model types which in this case is our best model, i.e., FT-byT5. The experimental evaluation clearly shows that task-specific neural models are required for DRS-to-text generation, as LLMs, which are generative general-purpose models, do not perform well on complex domain-specific applications.

4.3 Error Analysis

We further examine the model-generated text by manual inspection to gain a deep understanding of the experimental evaluations. To do so, we compare the same instances generated by different models, e.g., fully lexical, LLMs (ChatGPT and Claude), and FT-byT5. Examples of different features of semantics, syntax, missing information, the hallucinatory behavior of LLMs, additional information, and perfectly generated examples are shown in Table 7. The text generated by the model was evaluated based on the following criteria: (1) incorrect information (highlighted in red), (2) additional irrelevant information (underlined), and (3) semantically correct but with different textual representations (blue).

Table 7. Error analysis of different model-generated examples with reference text.

Reference Text	Model Type	Model Generated Text
You can't live on that island.	Fully Lexical	Everybody can't live in the island.
	ChatGPT-3.5	If a person does not live on an island, it is possible.
	Claude-2.0	If x1 is a person, then x1 does not live on an island, which is presumed.
	FT-byT5	You cannot live on that island.
It will cost around 10000 yen.	Fully Lexical	It will cost about 1000 yen.
	ChatGPT-3.5	The cost of the entity referred to by x1 is presupposed to be now and is equal to 10,000 yen per unit.
	Claude-2.0	An entity costs 10000 yen now.
	FT-byT5	It will cost around 10000 yen.
Have you googled her?	Fully Lexical	You googled her.
	ChatGPT-3.5	The hearer (you), who is a female, is currently googling something at the present time.
	Claude-2.0	The hearer is currently googling a presumed female.
	FT-byT5	Have you googled her?

Three key elements of text generation in natural language – negation, question, and quantity – are shown in Table 7. The fully lexical model had difficulty capturing the actual semantics of the phrases in the context of the examples listed in the table (completely incorrect semantics are marked in red). The exact quantity and grammatical structure of the phrases were also difficult for the model to determine (see examples in Table 7 for the full lexical model).

Both ChatGPT and Claude underperformed because they were unable to produce accurate translations for the DRS examples. Analysis of the samples shows that, rather than producing a literal translation, the models began by explaining the logical representation of the DRS (text that is deemed extraneous is underlined). This, we assume, is because these LLMs were trained without the use of any semantic or formal meaning representation. Furthermore, the few-shot learning approach is also not helping these models to generalize in a better way. We have selected samples from the best models, such as few-shot text for Claude and zero-shot text for chatGPT, to use during the manual inspection of LLM-generated text (see LLM results in Table 6 for few-shot and zero-shot).

Although it struggled a little to replicate the exact information presented in the test set, our best model was able to capture the semantic and grammatical representation in the best feasible way. Due to exact word overlaps between text pairs, these small changes (highlighted in blue) in the model-generated text will not affect the human evaluation; however, because the generated text maintains the exact meaning, semantics, and grammatical structure of the sentences, they will result in low scores for automatic evaluations.

5 Conclusion

We performed data delexicalization of the DRS for common and proper nouns through WordNet supersenses and Named Entities-based lexical abstractions. Individually both delexicalization procedures, compared to fully lexical ones, resulted in enhancing the generalization ability of the neural model. The use of lexical data augmentation along with data delexicalization further improves the robustness capabilities and adds to the performance gain. Our experiments with biLSTM and byT5 neural sequence-to-sequence models showed promising

results with the best scores for the fine-tuned byT5 model. We found that data delexicalization helps the model to focus more on the syntactic structure of complex meaning representation thus generating correct textual sequences. General-purpose LLMs (ChatGPT and Claude) hallucinate and explain the DRS rather than generating the correct textual sequences. This highlights the true need for task-specific models for complex domain-specific applications.

Limitations. We are also working on performing lexical abstraction on all lexical entities in the meaning representation. We have not expanded our implementation to include other low-resource languages like Italian, Dutch, and German.

Acknowledgments. We thank "High-Performance Computing for Artificial Intelligence (HPC4AI) at the University of Turin" for providing GPU support.

Disclosure of Interests. The authors declare no conflict of interest.

References

1. Nai-xing, W. (2007). Shared Meaning and Delexicalization. Journal of PLA University of Foreign Languages.
2. Vainio, M., Suni, A., Raitio, T., Nurminen, J., Järvikivi, J., and Alku, P. (2009). New method for delexicalization and its application to prosodic tagging for text-to-speech synthesis. , 1703-1706. https://doi.org/https://doi.org/10.21437/Interspeech.2009-514.
3. Sharma, S., He, J., Suleman, K., Schulz, H., and Bachman, P. (2016). Natural Language Generation in Dialogue using Lexicalized and Delexicalized Data. ArXiv, abs/1606.03632.
4. Shimorina, A., and Gardent, C. (2018). Handling Rare Items in Data-to-Text Generation. , 360-370. https://doi.org/https://doi.org/10.18653/v1/W18-6543.
5. Dušek, O., Novikova, J., and Rieser, V. (2018). Findings of the E2E NLG challenge. arXiv preprint arXiv:1810.01170.
6. Zhou, G., and Lampouras, G. (2020). WebNLG challenge 2020: Language agnostic delexicalisation for multilingual RDF-to-text generation. In Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+) (pp. 186-191).
7. Hao Dong, Jingqing Zhang, Douglas McIlwraith, and Yike Guo. 2017. I2t2i: Learning text to image synthesis with textual data augmentation. In IEEE international conference on image processing (ICIP), pages 2015–2019. vol. 2017. IEEE.
8. Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. in Proc., 7:178–186.
9. Angela Fan and Claire Gardent. 2020. Multilingual AMR-to-text generation. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 2889–2901, Online. Association for Computational Linguistics.
10. Jeffrey Flanigan, Chris Dyer, Noah A Smith, and Jaime G Carbonell. 2016. Generation from abstract meaning representation using tree transducers. in Proc., 2016:731–739.

11. Thiago Castro Ferreira, Claire Gardent, Nikolai Ilinykh, Chris van der Lee, Simon Mille, Diego Moussallem, and Anastasia Shimorina. 2020. The 2020 bilingual, bi-directional WebNLG+ shared task: Overview and evaluation results (WebNLG+ 2020). In Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+), pages 55–76, Dublin, Ireland (Virtual). Association for Computational Linguistics
12. Valerio Basile and Johan Bos. 2011. Towards generating text from discourse representation structures. in ENLG', 11:145–150.
13. Rik van Noord, Lasha Abzianidze, Hessel Haagsma, and Johan Bos. 2018. Evaluating scoped meaning representations. In Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018), Miyazaki, Japan. European Language Resources Association (ELRA).
14. Rik van Noord. 2019. Neural boxer at the IWCS shared task on DRS parsing. in Proc. IWCS Shared Task on Semantic Parsing, Gothenburg, Sweden. Association for Computational Linguistics.
15. Chunliu Wang, Rik van Noord, Arianna Bisazza, and Johan Bos. 2021. Evaluating text generation from discourse representation structures. In Proceedings of the 1stWorkshop on Natural Language Generation, Evaluation, and Metrics (GEM 2021), pages 73–83, Online. Association for Computational Linguistics.
16. Muhammad Saad Amin, Alessandro Mazzei, and Luca Anselma. 2022. Towards data augmentation for drsto- text generation. In Proceedings of the Sixth Workshop on Natural Language for Artificial Intelligence (NL4AI 2022) co-located with 21th International Conference of the Italian Association for Artificial Intelligence (AI*IA 2022), Udine, November 30th, 2022, volume 3287 of CEUR Workshop Proceedings, pages 141–152. CEUR-WS.org.
17. Muhammad Saad Amin, Luca Anselma, and Alessandro Mazzei. 2024. Exploring Data Augmentation in Neural DRS-to-Text Generation. In Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers), pages 2164–2178, St. Julian's, Malta. Association for Computational Linguistics.
18. Kamp, Hans, and Uwe Reyle. From discourse to logic: Introduction to model-theoretic semantics of natural language, formal logic, and discourse representation theory. Vol. 42. Springer Science and Business Media, 2013.
19. Johan Bos. 2021. Quantification annotation in discourse representation theory. in ISA 2021-17th Workshop on Interoperable Semantic Annotation, Groningen/Virtual, Netherlands.
20. Hans Kamp and Uwe Reyle. 1993. From Discourse to Logic: Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory. Kluwer Academic Publishers, Dordrecht.
21. Katarzyna Jaszczolt. 2023. Semantics, Pragmatics, Philosophy: A Journey Through Meaning. New York, NY: Cambridge University Press.
22. Ankur Parikh, XuezhiWang, 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.
23. OpenAI. 2023. Gpt-4 technical report.
24. Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. 2023. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting.