Comparing Neural Meaning-to-Text Approaches for Dutch

Chunliu Wang[*] CHUNLIU.WANG@RUG.NL
Johan Bos[*] JOHAN.BOS@RUG.NL

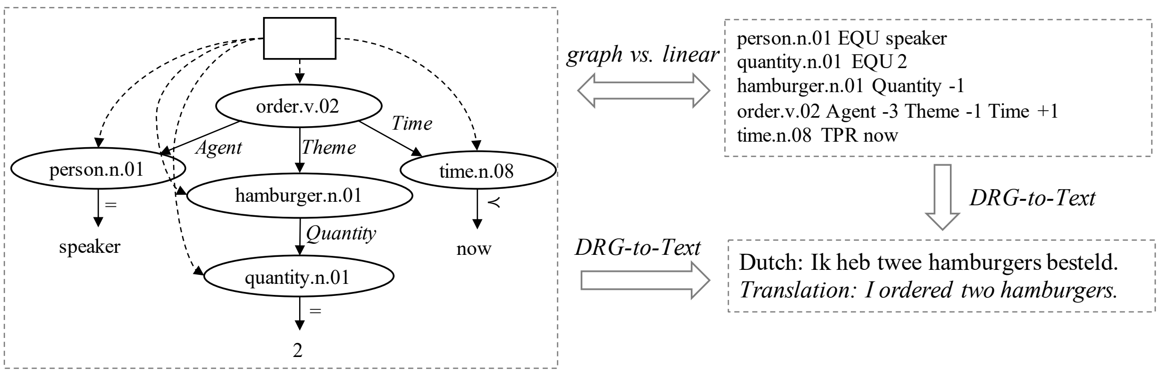
[*] *Center for Language and Cognition, University of Groningen, Netherlands*

Abstract

The neural turn in computational linguistics has made it relatively easy to build systems for natural language generation, as long as suitable annotated corpora are available. But can such systems deliver the goods? Using Dutch data of the Parallel Meaning Bank, a corpus of (mostly short) texts annotated with language-neutral meaning representations, we investigate what challenges arise and what choices can be made when implementing sequence-to-sequence or graph-to-sequence transformer models for generating Dutch texts from formal meaning representations. We compare the performance of linearized input graphs with graphs encoded in various formats and find that stacking encoders obtain the best results for the standard metrics used in natural language generation. A key challenge is dealing with unknown tokens that occur in the input meaning representation. We introduce a new method based on WordNet similarity to deal with out-of-vocab concepts.

1. Introduction

Meaning-to-Text generation is an active research task in the Natural Language Generation community, in which the goal is to generate textual descriptions from structured data, such as Abstract Meaning Representation, AMR (Banarescu et al. 2013), Discourse Representation Structures, DRS (Wang et al. 2021), and minimal recursion semantics, MRS (Copestake et al. 1997). Due to the limitation of available annotated corpora, previous work usually focuses on English. Instead, we investigate meaning-to-text generation for Dutch using a recently made available corpus that pairs formal meaning representations with short Dutch sentences. In this paper, we use Discourse Representation Graphs (DRG) as our research meaning representation data, which is a notational variant of DRS (Kamp and Reyle 1993) and can be represented as simple directed acyclic graphs or in a linear variable-free notation (Bos 2021). Figure 1 visualizes the task of generating Dutch from meaning representations.



```
person.n.01 EQU speaker
quantity.n.01 EQU 2
hamburger.n.01 Quantity -1
order.v.02 Agent -3 Theme -1 Time +1
time.n.08 TPR now
```

Dutch: Ik heb twee hamburgers besteld.
Translation: I ordered two hamburgers.

Figure 1: Generating Dutch from formal meaning representations.

We focus on neural network approaches for generating text. A key issue in neural Meaning-to-Text generation is how to encode the input data (the meaning representation, see Figure 1). The current focus is on two typical approaches: sequential encoders and graph encoders. Sequential encoders are primarily used for data represented in linearized format (Konstas et al. 2017), and encoders are usually based on the Long-short Term model, LSTM (Hochreiter and Schmidhuber 1997) or a Transformer model (Vaswani et al. 2017). Graph encoders are usually applied to graph-structured data, aiming at producing sentences that preserve both the meaning and structured information of the input graphs because linearized data may not always capture the structure inherent in meaning representations. In addition, there are also hybrid approaches of adding sequential information to graph encoders, such as using stacking encoders (Damonte and Cohen 2019) or adding sequential graphs to original graphs data (Guo et al. 2019). However, few works explicitly compare these methods, especially for non-English meaning-to-text generation tasks. In this paper, we aim to take Dutch as the target language and compare various input representation possibilities for neural network approaches. This comparison is facilitated by the data resource that we use, which offers meaning representations both in linear and in graph format (Section 3).

While recently a substantial amount of meaning-to-text research is carried out on AMR (Konstas et al. 2017, Damonte and Cohen 2019, Guo et al. 2019, Ribeiro et al. 2019), we motivate our choice of DRG here. Firstly, DRG is a more expressive formalism than AMR, as it models more semantic phenomena (negation, presupposition) and can handle document-level representations by using explicit discourse relations. Secondly, DRG is language neutral (even though an English WordNet is used to represent concepts), and the annotated DRG corpus is available for multiple languages, while AMR is slightly biased towards English. Thirdly, although not perfect, the alignment between text and meaning in the linearized DRGs is more smoothly than in AMR. Imperfect alignments can make it difficult for the linear recurrent neural networks to induce the original connections between words and nodes in the graph (Song et al. 2018). And, finally, there is an annotated corpus available that pairs Dutch sentences with DRGs (Abzianidze et al. 2017). To the best of our knowledge, no such corpus exists for AMR. With this as background, we aim to answer the following research questions:

1. Which type of encoder in sequential and graph encoders achieves the best performance in DRG-to-text generation for Dutch?
2. Is it useful to add sequential information to graph2Seq models and in which way is this best done?
3. What are the major challenges in DRG-to-text generation for Dutch?

Before we introduce the various methods for generating sentences from meaning representations, we first have a closer look at the semantic formalism of our choice: Discourse Representation Graphs.

2. Discourse Representation Graphs

DRG is a variant of DRS (Kamp and Reyle 1993) that allows us to represent meaning in two different ways: as a sequential notation, or as a graph. Both representations are semantically equivalent. But the sequence notation, which is a linearisation of the graph, contains more information, since the order of the concepts reflects the order in which they are introduced in the corresponding sentences. This order information is normally not present in the graph notation. There are five types of semantic information that can be found: concepts, roles, constants, comparison operators and discourse relations. In more detail, they are defined as:

1. The concepts are represented by WordNet synsets (Christiane 1998) (for nouns, verbs, adjectives and adverbs), indicating the lemma, part-of-speech and sense number (ball.n.04, kill.v.01, old.a.02, ...);

2. The roles are represented by the thematic relations proposed in VerbNet (Kipper et al. 2008) (Agent, Theme, Time, ...), which are used to mark relations between concepts;

3. The constants are used to represent discourse deictics (speaker, hearer, now), unknown information (?), names, and quantities;

4. Comparison operators are used to relate and compare concepts or constants, such as EQU (=, equality), APX ($\sim$, approximately), and TPR ($\prec$, temporal precedence);

5. Discourse relations are used to indicate discourse relations between different scopes of a DRG, such as NEGATION, CONTINUATION, ELABORATION and CONTRAST.

The Parallel Meaning Bank, PMB (Abzianidze et al. 2017), provides a large corpus of sentences annotated with DRS for several languages including Dutch, as shown in Figure 2. The box format used in Discourse Representation Theory (Kamp and Reyle 1993) is perhaps intuitive and easy to read but not convenient for modeling with deep learning methods. As a result, DRS is often preprocessed in a format such as clauses (van Noord et al. 2018, Wang et al. 2021), graphs (Fancellu et al. 2019) or trees (Liu et al. 2021) that can be modeled by neural network models.

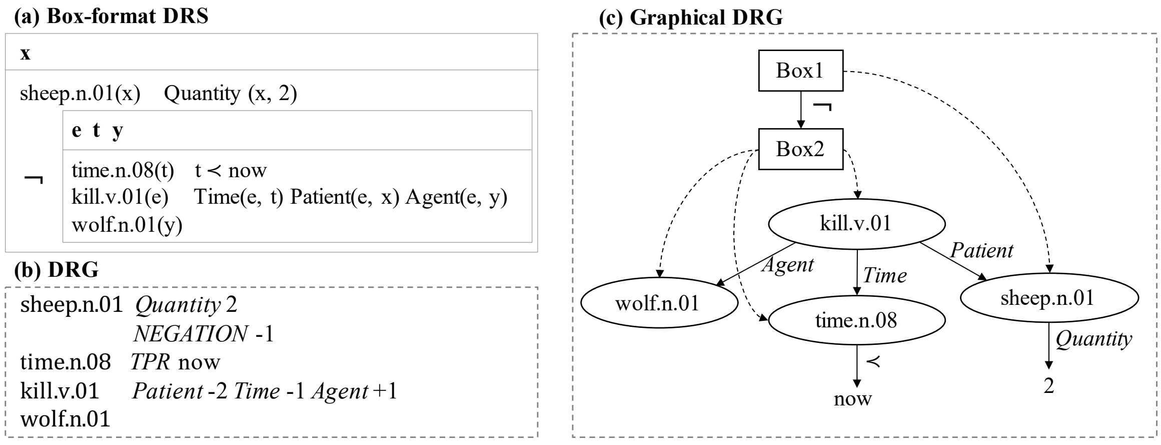


Figure 2: (a) The box-format DRS for the Dutch sentence: *Twee schapen werden niet gedood door de wolf. (Translation: Two sheep were not killed by the wolf.)*, (b) the corresponding sequential notation DRS and (c) the corresponding visualized graphical DRG.

These conversion procedures of graph format and tree format often require complicated and cumbersome rules, and the obtained structure is also difficult to understand. On the other hand, DRG (see Figure 2b), in particular the sequential notation, enables a straightforward way to manually encode a DRS as a sequence of elements without resorting to explicit variable names and removing several notational redundancies. It is simpler and easier to understand than clause format and can be converted to graph structure directly, as shown in Figure 2c). The role of variables is taken over by relative positive and negative indices, where negative indices refer to concepts introduced before in the discourse, and positive indices refer to concepts mentioned after (Bos 2021). For instance, "Patient -2" in Figure 2b connects the killing event to the concept introduced two positions before in the sequence, i.e., sheep.n.01.

VerbNet roles connect the relationship between concepts and are regarded as edge labels in the DRG, while concepts form the nodes. The comparison operators are also represented as edge labels in the graph, while the constants such as named entities are terminating nodes. All the roles and comparison operators following the concepts are treated as outgoing edges of that concept. There are three types of nodes: entities, constants, and contexts (i.e., DRSs). The contexts are indexed (Box$_1$, Box$_2$, and so on, in order to distinguish them from each other in the graph structures. Note that contexts are implicit in the sequence notation (Bos 2021).

3. Methods

In this section we show how we represent DRG data for the different types of encoders. We use two types of data representations and apply them to the corresponding baseline encoders. We also summarize three hybrid approaches that combine sequential information in linearized data with structured information in graph data. Furthermore, we introduce three state-of-the-art graph encoders for our comparative experiments: Gated Graph Neural Networks (GGNN), Graph Attention Networks (GAT), and Graph Isomorphic Networks (GIN). Finally, we present our methods for dealing with the out-of-vocabulary (OOV) for DRG data.

3.1 Input Representations

Linearized DRG This type of input representation is the linearized sequential notation of DRS data provided in release 4.0.0 of the Parallel Meaning Bank (PMB) (Abzianidze et al. 2017) We remove the alignment between meaning and text in the original PMB files (Figure 3a) and replace all newlines with spaces to ensure that the meaning representation is a sequence on one line. We also remove quotation marks contained in named entities to reduce the sparsity of the data (Figure 3b). In sum, this is essentially a type of input representation that requires hardly any pre-processing.

Graph-structured DRG This type of input representation requires substantial pre-processing. Based on Section 2, the sequential notation DRS can be transformed into a directed labeled graph. A directed acyclic graph can be represented as $\boldsymbol{G} = \langle \boldsymbol{V}, \boldsymbol{E} \rangle$, where $\boldsymbol{V}$ is the set of graph nodes and $\boldsymbol{E}$ corresponds to all graph edges. Each edge $e \in \boldsymbol{E}$ is a triple $(v_i,\ l,\ v_j)$, showing labelled relation between two connected nodes v_i and v_j, where v_i is the parent node, l is the edge label and v_j is the child node.

Following previous work (Beck et al. 2018), a directed acyclic graph can be converted into the directed unlabeled Levi graph used for graph encoders, where each labeled edge $e = (v_i, l, v_j)$ is transformed into two unlabeled edges $e_1 = (v_i,\ l)$ and $e_2 = (l,\ v_j)$. Figure 3c illustrates this idea. For instance, the single edge between be.v.08 and time.n.08 with label Time is replaced by two new edges: an unlabeled edge between be.v.08 and Time and another one between Time and time.n.08, where Time become a new node.

This approach is also convenient for representing compound named entities, where the direction of edges corresponds to the order of the tokens comprising the name, without using additional edge labels. For instance, the labeled edge $e = (\text{male.n.01}, \text{Name}, \text{"Stephen Hawking"})$ is converted to three unlabeled edges $e_1 = (\text{male.n.01}, \text{Name})$, $e_2 = (\text{Name}, \text{Stephen})$, and $e_3 = (\text{Stephen}, \text{Hawking})$, where Name and "Stephen Hawking" become three new nodes in an unlabeled graph.

The final step of the process outlined is to produce the graph in the form of an adjacency matrix (Beck et al. 2018). Figure 3d illustrates a simple directed unlabeled graph with self-loops. In this matrix, a 1 or 0 is in position according to whether graph vertices are adjacent or not.

3.2 Baselines Encoders

Sequential Encoders We use bidirectional LSTM encoders as our baseline sequential encoders, where the sequential linearization is the input to a bidirectional LSTM network (Konstas et al. 2017), which is widely used in various NLP tasks such as machine translation. They learn the input sequence in two directions and then concatenate the states for each time step:

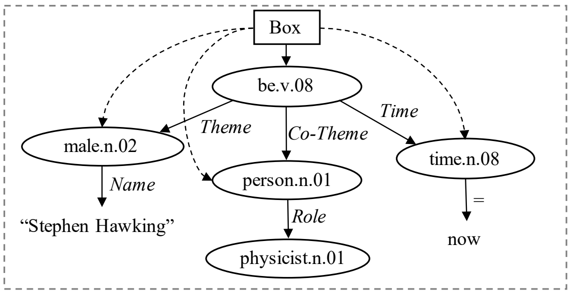
(a) DRG aligned with text

male.n.02 Name "Stephen Hawking"	% Stephen Hawking [0-15]
be.v.08 Theme -1 Time +1 Co-Theme +2	% is een [16-22]
time.n.08 EQU now	%
person.n.01 Role +1	% natuurkundige. [23-37]
physicist.n.01	%

(b) Linearized DRG

male.n.02 Name Stephen Hawking be.v.08 Theme -1 Time +1 Co-Theme +2 time.n.08 EQU now person.n.01 Role +1 physicist.n.01

(c) Graphical DRG



(d) Graph-structured DRG

	Box	male.n.02	Name	Stephen	be.v.08	Hawking	Theme	Time	Co-Theme	time.n.08	EQU	now	person.n.01	Role	physicist.n.01
Box	1	1	0	0	1	0	0	0	0	1	0	0	1	0	1
male.n.02	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
Name	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
Stephen	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0
Hawking	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
be.v.08	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0
Theme	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0
Time	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0
Co-Theme	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0
time.n.08	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
EQU	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
now	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
person.n.01	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
Role	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
physicist.n.01	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 3: (a) An example of sequential notation DRS data for the sentence of *Stephen Hawking is een natuurkundige. (Translation: Stephen Hawking is a physicist.)* (b) Linearized DRG for sequential encoders neural models. (c) Graphical DRG for the sentence of *Stephen Hawking is een natuurkundige.* (d) Input representation with graph-structured information for graph encoders.

$$\overleftarrow{h_j} = LSTM(\overleftarrow{h_{j+1}}, x_j) \tag{1}$$

$$\overrightarrow{h_j} = LSTM(\overrightarrow{h_{j-1}}, x_j) \tag{2}$$

where x_j is the input vector at time step j, and $\overleftarrow{h_j}$ and $\overrightarrow{h_j}$ are hidden states generated based on x_j and the previous hidden states $\overleftarrow{h_{j+1}}$ and $\overrightarrow{h_{j-1}}$.

Graph Encoders Previous studies have pointed out that sequence-to-sequence (seq2seq) models lack explicit modeling of syntax or any language hierarchy structure, while graph-to-sequence (graph2seq) models have been proposed to incorporate structured information in neural graph encoders (Song et al. 2018). Specifically, we rely on Graph Convolutional Networks (GCNs) as our baseline graph encoders, which is a recent class of multilayer neural networks operating on graphs (Thomas and Max 2017). Each node $v \in \boldsymbol{V}$ with a feature vector $x_i \in R^d$, and the GCN sums over the embeddings of the immediate neighbor of each node following:

$$h_i^{(l+1)} = \text{ReLU}(\sum_{j \in N(i)} W_{dir(j,i)}^{(l)} h_j^{(l)} + b_{dir(j,i)}^{(l)}) \tag{3}$$

where $W_{dir(j,i)}^{(l)}$ represents the weight matrix of the l^{th} layer and $b_{dir(j,i)}^{(l)}$ represents the bias vector of the l^{th} layer, which are direction-specific parameters, where $dir(j,i) \in \{$*default, reverse, self*$\}$. *default*, *reverse*, and *self* refer to the original edges, the reversed edges to the original edges, and the self-loop edges (Marcheggiani and Titov 2017). Further more, $N(i)$ is the set of immediate neighbors of x_i, ReLU is the activation function, and h_j is the embedding representation of node $x_j \in V$ at layer l.

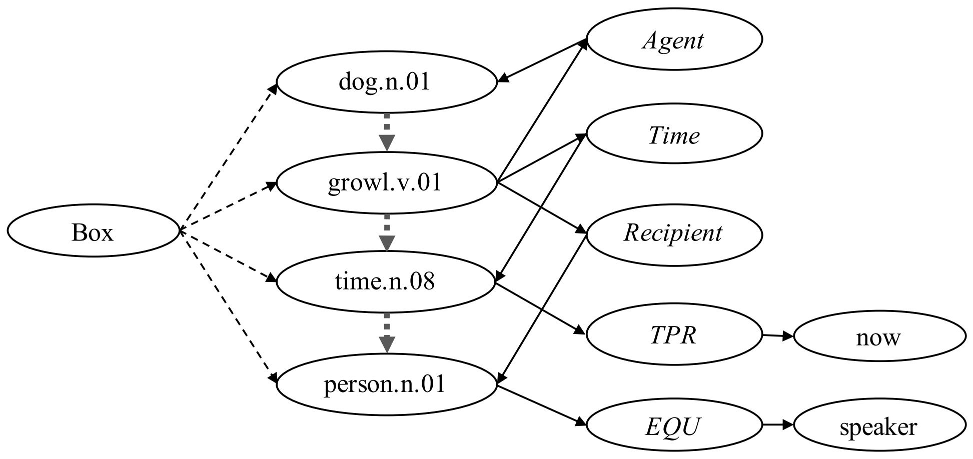


Figure 4: The Levi graph of DRG with sequential connections between concepts (gray dashed lines). The full graph also contains reverse and self edges, which are omitted in the figure.

3.3 Hybrid Encoders

In order to better integrate the advantages of the two types of baseline encoders, we also experiment with hybrid approaches where we combine graph methods with sequence methods.

Graph Encoders with Sequential Graph The input of graph-structured data is initially a surface form containing sequential information, which is lost when it is modeled using graph encoders, especially for data like DRG whose concepts are aligned with words in the text. Inspired by the work of syntax-based neural machine translation (Beck et al. 2018), we add a sequential graph to the original graph (`GCN+seq`), which uses additional sequential edges connecting concepts in DRG, as shown in Figure 4. That means the node embeddings in a graph are computed by Equation 3, where $dir(j,i) \in \{default, reverse, self, sequence\}$ instead of $dir(j,i) \in \{default, reverse, self\}$. On this basis, the sequential information in the original format data can be modeled.

Stacking Graph Encoders with Sequential Encoders Another approach combining graph-structured information with sequential information is to stack graph encoders with sequential encoders. In this paper, we apply stacking to LSTM encoders and GCN encoders. This method was shown to be effective for syntax-aware neural machine translation tasks (Bastings et al. 2017), and allows us to easily test the contributions of stacking components. There are two ways to implement stacking encoders (Damonte and Cohen 2019): using graph encoders on top of sequential encoders (`GCN+LSTM`), where the graph encoders are applied to create refined input embeddings and then passed through LSTM, and its reverse (`LSTM+GCN`), using sequential encoders on top of graph-structured encoders.

3.4 Other Typical Graph Encoders

We apply three current state-of-the-art graph encoders to DRG data as our comparative experiments. Below we give a brief summary of each of these approaches.

Gated Graph Neural Networks GGNNs use a Gated Recurrent Unit (GRU) to facilitate information propagation between local layers, which was proposed for the reason that the GCN model has difficulty learning deep layers of node information. GGNN can be seen as a multi-layer GCN where layer-wise parameters are tied and gating mechanisms are added. With this, the model can propagate node information between long-distance nodes in the graph (Yujia et al. 2016). In particular, the l^{th} layer of a GGNN is calculated as:

$$h_i^{(l+1)} = \mathrm{GRU}(h_i^{(l)}, \sum_{j \in N_i} W^{(l)} h_j^{(l)}) \tag{4}$$

where $W^{(l)}$ represents the weight matrix of the l-th layer which is a direction-specific parameter. ρ is the activation function, and we use ReLU in the experiments. h_j is the embedding representation of node $j \in V$ at layer l, and GRU is a gated recurrent unit, a combination function.

Graph Attention Networks Some researchers consider it is unreasonable to assign equal importance to all adjacent nodes of a node and proposed GAT, which updates each node representation by incorporating the attention mechanism to calculate the importance of adjacent information (Veličković et al. 2018).

$$h_i^{(l+1)} = \rho(\sum_{j \in N_i} \alpha_{ji}^{(l)} W^{(l)} h_j^{(l)}) \tag{5}$$

where $W_{dir(j,i)}^{(l)}$ represents the direction-specific weight matrix, and $\alpha_{ji}^{(l)}$ is the normalized attention coefficient at l^{th} layer computed by the attention mechanism as follows:

$$\alpha_{ji}^{l} = \mathrm{softmax}(\sigma(a^{T}[W^{(l)} h_i^{(l)} || W^{(l)} h_j^{(l)}])) \tag{6}$$

where σ is a activation function, $||$ is the concatenation function and α is a model parameter for computing attention function.

Graph Isomorphic Networks Due to the limited understanding of graph representation properties by the above graph models, GIN was proposed to analyze the expressive ability of graph neural networks to capture different graph structures and proved to be as powerful as the Weisfeiler-Lehman (WL) graph isomorphism test (Xu et al. 2019). GIN updates node representations as:

$$h_i^{(l+1)} = \mathrm{MLP}^{(l+1)}(h_i^{(l)} + \sum_{j \in N_i} h_j^{(l)}) \tag{7}$$

where MLP is a multi-layer perceptron. Different from the above graph networks, GIN simply aggregates the node along with its neighbors without a combination aggregated neighborhood feature.

3.5 Dealing with Out-of-Vocabulary Tokens

Models are typically trained with a closed output vocabulary derived from the training data, and can never give non-zero probability to a specific token not seen during training (Pappas et al. 2020). The tokens that are not in the vocabulary are unknown tokens, and the ubiquity of unknowns inevitably leads to data sparsity and limits the learning capacity of models. So we need appropriate strategies to deal with unknowns. In the case of Discourse Representation Graphs (see Section 2), the unknowns could be concepts, roles, constants, comparison operators, and discourse relations. Because of the relatively small (and finite) inventories of roles, comparison operators and discourse relations, these three semantic categories never occur as unknowns as they are all part of the DRG training data, so we can safely ignore them. But this is not the case for concepts and constants (e.g., numbers, names), of which there is a relative large number of rare instances occurring during test time (more than 20% of the concepts and about 25% of the constants are not seen during training). We consider several strategies for dealing with Out-of-Vocabulary (OOV) issues for constants and concepts.

OOV Constants Regarding meaning-to-text generation tasks, the strategies for alleviating the data sparsity can by and large be divided into (a) non-word-level tokenization (van Noord et al. 2018, Wang et al. 2021), (b) anonymization (Marcheggiani and Titov 2017, Damonte and Cohen 2019), and (c) the copy mechanisms (Song et al. 2018, Ribeiro et al. 2019). First we evaluate whether these strategies can be used in our comparative experiments.

Non-word-level tokenization is an "open up" vocabulary approach that models sequences of bytes, characters, or subwords instead of conventional word tokens. This technique is often used in seq2seq models but rarely in graph2seq models because it makes the graphs considerably larger. Hence we discard this option here.

Anonymization is a general method that works by replacing named entities and numerical values with specific tokens with type information (e.g. person, location, date) (Konstas et al. 2017). If two entities of the same type in a given input will be given a numerical suffix, e.g. PERSON_0 and PERSON_1. Anonymization can alleviate data sparsity well, especially for data sets with a large number of named entities. For data containing compound named entities, anonymized data can be well applied to graph models. However, this approach typically requires a large number of manual rules for covering all types of open-class tokens and is not easily adaptable to new domains (Song et al. 2018), so we ignore this method in experiments.

The copy mechanism works on top of an attention-based RNN decoder by integrating the attention distribution into the final vocabulary distribution (Gu et al. 2016), which favors generating words such as dates, numbers, and named entities that appear in the graphical or linearized input. Therefore, in the experiments below, we adopt the copy mechanism for both sequential encoders and graph encoders to deal with unknown constants.

OOV Concepts Unlike the constants above, the mapping between concepts and words in the corresponding text is not a simple copy of the surface text. Although the copy mechanism always provides state-of-the-art adaptation performance for dealing with constants, it cannot solve the problem of rare occurrences of concepts. Concepts in DRG are represented by WordNet synsets as explained in Section 2, and WordNet represents nouns, verbs, adjectives and adverbs in the form of a network through connections between synonyms (synsets) (Christiane 1998, Agirre et al. 2009, Orkphol and Yang 2019). The interconnections denote conceptual-semantic and lexical relations, including hyponymy, hypernymy, synonymy, antonymy, and so on. Our basic idea is to replace unknown concepts in DRG by utilizing the interlink between synsets to help models generate natural and coherent text.

Table 1: The NLTK interfaces used in replacement process for different POS concepts

Lexical Relation	**Nouns**	**Verbs**	**Adj.**	**Adv.**
Synonym	✓	✓	✓	✓
Hypernym	✓	✓	✗	✗
Hyponym	✓	✓	✗	✗
Entailment	✗	✓	✗	✗
Verb_group	✗	✓	✗	✗
Similar_to	✗	✗	✓	✗

More specifically, we use NLTK[1] toolkit to implement the above idea, which provides easy-to-use interfaces to access WordNet version 3.0. When we test the model, we first automatically check the unknown concepts in the test set, and then employ the NLTK interface to obtain concept replacement candidates for the OOV concepts. The acquisition of candidates is done via synonyms, hypernyms, hyponyms and similar-to concepts of the OOV concepts depending on the different parts of speech (POS). The specific NLTK interfaces shown for each lexical relation are shown in Table 1. We use the

1. NLTK 3.7 release: http://www.nltk.org/.

above concepts as replacement concepts candidates, and find a concept in training vocabulary with the most similar meaning to replace the OOV concepts, where the similarity between two concepts can be obtained by measuring the path between two synsets in WordNet. For example, grill.v.01 is an OOV concept in the DRG test set and its part of speech is a verb. Then, we search for its synonyms, hypernyms, hyponyms and verb_groups according to Table 1 and get the candidates {barbeque.v.01, change.v.02} that are in the vocabulary of the training set. The hyponym barbeque.v.01 gets the best similarity and is used to replace grill.v.01. Figure 5 summarises the process of handling OOV concepts.

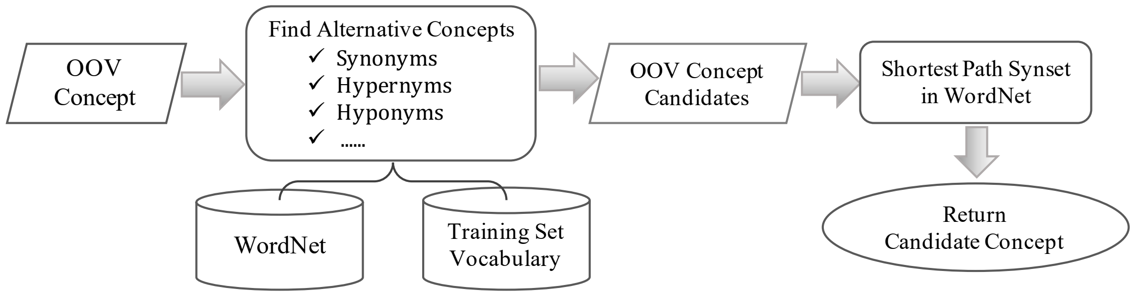


Figure 5: The steps of handling OOV concepts (WordNet synsets) in meaning representations.

4. Experiments and Results

All the models we used were implemented based on OpenNMT-py (Klein et al. 2017) and Pytorch Geometric (Matthias and E. 2019). We adopt the standard fully batched attention-based LSTM decoder (Dzmitry et al. 2015), where the attention memory is the concatenation of the attention vectors among all input words.[2]

4.1 Experimental Setup

Data We use DRG data for Dutch from release 4.0.0 of Parallel Meaning Bank (PMB) (Abzianidze et al. 2017), which contains 1,467 gold (fully manually verified) instances, 1,440 silver (partially corrected machine-generated) instances, and 28,265 bronze (uncorrected machine-generated) data. Each instance contains a DRG and corresponding Dutch sentence. There are 437 development instances and 491 test instances in the gold data. We use the remaining gold, silver and bronze data as training data. The Moses (Koehn et al. 2007) tokenizer is used to segment the reference sentences, i.e., the text corresponding to the DRG data.

Hyperparameters We construct vocabularies from all words in the training data. For the hyperparameters, we use the SGD optimizer with the initial learning rate set to 1 and decay 0.8. In addition, we set the dropout to 0.5 at the decoder layer to avoid overfitting. We set ReLU activations, and highway layers for graph models. For the LSTM model, we set a single layer, which is based on the performance of the development set. For the layer-wise graph models, we set one layer of the encoder, the number of graph layers are chosen from {2, 3, 4}, the encoder hidden dimensions are chosen from {300, 512, 750}, and the batch size are chosen from {32, 64, 128}. We show the performance of graph models training on two graph layers, 32 batch size and 750 hidden dimensions, which achieves the best performance on the GCN model.

Evaluation We use three standard metrics measuring word overlap between system output and references. They are BLEU (Papineni et al. 2002), METEOR (Lavie and Agarwal 2007), and

2. Our code and checkpoints are available at `https://github.com/wangchunliu/DRG-generation-Dutch`.

ROUGE-L (Lin 2004). These metrics are standard in machine translation evaluation and common in NLG. In Section 4.2, we show the results of the models' performance with the copy mechanisms for unknown constants but without the replacement strategy for unknown concepts. In Section 4.3, we give the performance of state-of-the-art model (GCN + LSTM) and compare it with the results after adding the replacement strategy.

4.2 Basic Results

Sequential Encoders versus Graph Encoders Table 2 summarizes the performance of baseline models for Dutch DRG data. The results show that the baseline LSTM sequential encoder outperforms the GCN graph encoders GCN by far, scoring better on all three metrics. It is worth noting that a hybrid encoder, obtained by adding sequential graphs to the graph encoder (GCN + seq in Table 2), can significantly improve the performance of the graph encoders training on original graphs, and achieves competitive performance compared to LSTM alone. In addition, we find that stacking encoders can improve performance over a single type of encoders, and the order of stacking encoders affects the results. Using the GCN model to encode first is better than using the LSTM model to encode first (Table 2).

Table 2: Comparison between graph encoders and sequential encoders.

Type	**Model**	**BLEU**	**METEOR**	**ROUGE**
Sequential Encoder (baseline)	LSTM	44.2	35.0	68.8
Graph Encoder (baseline)	GCN	37.7	32.0	64.6
Hybrid Encoder 1	GCN + Seq	43.9	34.7	68.6
Hybrid Encoder 2 (stacking)	LSTM + GCN	44.8	35.0	69.1
Hybrid Encoder 3 (stacking)	GCN + LSTM	**45.3**	**35.6**	**70.4**

Comparison of Different Graph Encoders As Table 2 shows, stacking works best. So it would be interesting to compare various types of popular graph encoders stacked with LSTM (Section 3.4). Table 3 shows the performances of three more types of graph encoders (GIN, GAT and GGNN) stacked with LSTM. Our experiments show that none of these new combinations outperform the hybrid encoder based on stacking GCN with LSTM.

Table 3: Comparison between different types of stacked hybrid encoders.

Type	**Model**	**BLEU**	**METEOR**	**ROUGE**
Hybrid Encoder 3 (stacking)	GCN + LSTM	**45.3**	**35.6**	**70.4**
Hybrid Encoder 4 (gated)	GGNN + LSTM	42.0	34.1	68.1
Hybrid Encoder 5 (attention)	GAT + LSTM	43.0	35.2	69.5
Hybrid Encoder 6 (isomorphic)	GIN + LSTM	42.7	34.6	68.7

4.3 Effects of Replacement Strategy for OOV Concepts

The results shown in Table 2 and Table 3 were obtained without applying our strategies for unknown concepts. The number of OOV concepts is considerable in our data set. Table 4 show the distribution of concepts (i.e., WordNet synsets) for nouns, verbs, adjectives, and adverbs in the training vocabularies and test set vocabularies, as well as the number of concepts included in the test set vocabulary but not in the training vocabulary. The concepts of nouns and verbs occupy most of the concepts in the corpus, and the OOV concepts are also mainly concentrated on them. Since there are many nouns and verbs in the vocabulary, it is relatively easy to find suitable alternative concepts for OOV noun concepts and verb concepts in the vocabulary, as shown in Table 4. Correspondingly, although there are not many OOV concepts for adjectives, it is difficult to find replacement concepts in the vocabulary. Unlike other part-of-speech concepts, adverb concepts are relatively rare in both WordNet and training vocabulary, and it is difficult to find alternative concepts using lexical relations.

Table 4: Distribution of OOV concepts and their replacement concepts via lexical relations.

Type	**Nouns**	**Verbs**	**Adj.**	**Adv.**	**Total**
Concept types in training	10,235	4,842	3,149	211	18,437
Concept types in test	333	234	93	9	669
OOV concept types in test	60	69	18	4	151
OOV concept tokens in test	61	74	19	4	158
OOV concept replaced by synonym	2	6	2	1	11
OOV concept replaced by hypernym	51	31	n.a.	n.a.	82
OOV concept replaced by hyponym	8	11	n.a.	n.a.	19
OOV concept replaced by verb_group	n.a.	5	n.a.	n.a.	2
OOV concept replaced by entailment	n.a.	2	n.a.	n.a.	5
OOV concept replaced by similar_to	n.a.	n.a.	8	n.a.	8
OOV concept without replacement	0	19	9	3	31

But what is the effect of this replacement strategy for unknown concepts? Using the best-performing hybrid model, we ran it on the test set after applying our strategy for unknown concepts. Our results, shown in Table 5, indeed show a slight increase in the automated evaluation metric results.

Table 5: Effect of WordNet-based replacement strategy of dealing with unknown concepts.

Type	**Model**	**BLEU**	**METEOR**	**ROUGE**
Hybrid Encoder 3 (unaltered)	GCN + LSTM	45.3	35.6	70.4
Hybrid Encoder 3 (replacing OOV concepts)	GCN + LSTM	45.9	36.0	70.7

In a way, big improvements of results cannot be expected with the current evaluation metrics. This is because when we use a substitution of a new concept for an unknown concept, the model generates the text corresponding to that new concept, not the text corresponding to the unknown concept. Even though two concepts may respond to similar meanings, it does not improve the word-overlap between generated text and the reference in which the standard metrics BLEU, METEOR and ROUGE are based. In this case we would gain more insight of the impact of our concept replacement strategy by looking at the data.

In Table 6, we give several examples of the effects of our replacement strategy. Although our method cannot improve the scores of automatic evaluation metrics very much, it can greatly improve the readability of generated text, and produce texts that are close in meaning to the original text in some cases. However, this method also has certain limitations. For individual rare concepts, it is also difficult to find a similar concept to replace it. When a hypernym is very far away from the unknown concept, the generated text may also face losing most of its meaning.

Table 6: Effect of the replacement strategy for unknown concepts using the GCN+LSTM model. Shown is the input DRG with OOV concepts in red and replaced concepts in blue, followed by the generated texts for the input without (red) and with replacing (blue).

Meaning	have.v.01 Time +1 Pivot +2 Theme +3 time.n.08 EQU now person.n.01 EQU hearer shoehorn.n.01 → device.n.01
Gen. text	Heeft u een klantenkaart?
Gen. text	Heeft u dat apparaat?
Ref. text	Hebt u een schoentrekker?
Meaning	company.n.01 Name FermiLab create.v.06 → produce.v.02 Agent -1 Time +1 Result +2 time.n.08 TPR now software.n.01 Name Linux LTS
Gen. text	FermiLab diende Linux LTS.
Gen. text	FermiLab produceert Linux LTS.
Ref. text	FermiLab ontwikkelde Linux LTS.
Meaning	person.n.01 Role +1 extremist.n.01→ person.n.01 kidnap.v.01 Agent -2 Time +1 Theme +2 time.n.08 TPR now person.n.01 Role +1 wife.n.01 Of +1 person.n.01 Role +1 president.n.03
Gen. text	De docent stelde de vrouw van de president.
Gen. text	Ze wilden de vrouw van de president.
Ref. text	Extremisten ontvoerden de vrouw van de president.
Meaning	tip.n.01 PartOf +1 spear.n.01 → weapon.n.01 time.n.08 TPR now dip.v.01→ enter.v.01 Theme -3 Time -1 Destination +2 deadly.a.01 → fatal.a.01 AttributeOf +1 poison.n.01
Gen. text	De kern van de brievenbus werd donderdag tot een rustige handelsonderneming.
Gen. text	De kern van de uitbraak werden binnen tot de tweezijdige weerstand.
Ref. text	De top van de speer was gedrenkt in een dodelijk vergif.
Meaning	alarm_clock.n.01 wake.v.05 → awaken.v.01 Causer -1 Time +1 Patient +2 time.n.08 ClockTime 07:00 TSU now person.n.01 EQU speaker
Gen. text	De wekker kozen me om zeven uur.
Gen. text	De wekker maakt me om zeven uur.
Ref. text	De wekker wekt me om zeven uur
Meaning	queen.n.02 → leader.n.01 female.n.02 Name Elizabeth Title -1 pass_away.v.01 → die.v.01 Patient -1 Time +1 time.n.08 YearOfCentury 1603 TPR now
Gen. text	Zelfs Elizabeth diende in 1603.
Gen. text	leider Elizabeth stierf in 1603.
Ref. text	Koningin Elizabeth overleed in 1603.
Meaning	entity.n.01 time.n.08 EQU now measure.n.02 Quantity 1000000 Unit +1 yen.n.02 worth.a.02 → valuable.a.01 Theme -4 Time -3 Value -2
Gen. text	Is een miljoen yen doorgebracht?
Gen. text	Het is een miljoen yen waard.
Ref. text	Dit is een miljoen yen waard.
Meaning	person.n.01 EQU speaker have.v.01 Pivot -1 Time +1 Theme +2 time.n.08 EQU now person.n.01 Quantity 2 Role +1 niece.n.01 → relative.n.01
Gen. text	Ik heb twee jezuïeten.
Gen. text	Ik heb twee familieleden.
Ref. text	Ik heb twee nichtjes.
Meaning	person.n.01 Name ? be.v.02 Theme -1 Time +1 Co-Theme +2 time.n.08 EQU now person.n.01 Role +1 leader.n.01 PartOf +2 country.n.02 Name algeria military_unit.n.01 → enemy.n.01 Name Gewapende Islamitische Groep Source -1
Gen. text	Wie is de leider van algeria nog een algeria?
Gen. text	Wie is de leider van algeria de algeria Gewapende Groep Groep Groep?
Ref. text	Wie is de leider van de Algerijnse Gewapende Islamitische Groep?
Meaning	NEGATION -1 hedgehog.n.02 → mammal.n.01 NEGATION -1 be.v.01 Theme -1 Co-Theme +2 small.a.01 AttributeOf +1 animal.n.01
Gen. text	Een dolfijn is een klein dier.
Gen. text	Een zoogdier is een klein dier.
Ref. text	De egel is een klein dier.
Meaning	male.n.02 Name Tom toss.v.01 → throw.v.01 Agent -1 Time +1 Theme +2 Destination +3 time.n.08 TPR now key.n.01 female.n.02 Name Mary
Gen. text	Tom diende de sleutel aan Mary.
Gen. text	Tom gooide de sleutel naar Mary.
Ref. text	Tom gooide de sleutels naar Mary.

5. Discussion

Most previous studies on data-to-text generation tasks have shown that graph-structured data trained with graph encoders can achieve performance similar to, or even better than, sequential encoders. This is often attributed to the loss of structural information after linearization of the graph (Damonte and Cohen 2019, Ribeiro et al. 2019, Guo et al. 2019). The use of graph neural networks can indeed preserve the structural information of data well. In addition, in linearized data based on AMR, the concept nodes after serialization often do not correspond to the surface word order, and it would be difficult for sequential neural networks to learn these long-range dependencies (Damonte and Cohen 2019). In our experiments (see Table 2), however, we show that sequential encoders (LSTM) can significantly outperform graph encoders (GCN) without sequential information because the serialized DRG data contains the order of concepts aligned with the information in the corresponding text. This sequential information is lost when treating the input as a graph. Furthermore, we are also interested in the impact of different data properties on the performance of the model. Therefore, we conduct additional experiments for more analysis. This section reports on the effect of the graph size, data size, and use of indices in DRG input on generating Dutch texts.

5.1 Effect of Graph Size

Large DRG graphs are always mapped into long sentences. In theory, graph encoders should be more favorable for long sentences because there are more long-distance syntactic dependencies in graph data, which may not be adequately captured by sequential encoders. However, large graphs may have more complicated structures, and a small amount of large graph data may not help the models learn more information, but instead brings noise. To figure it out, we merge the test set and development set, and then divide them into six buckets according to the length of reference tokens, from 3 to 8 words. Figure 6 shows the GCN model underperforms the LSTM model for any text length, and they have the same trend for changes in text length, with the best results at length 6. There is relatively little long text in Dutch in both the training and test sets, and longer data has the same impact on sequential and graphical models, making performance worse.

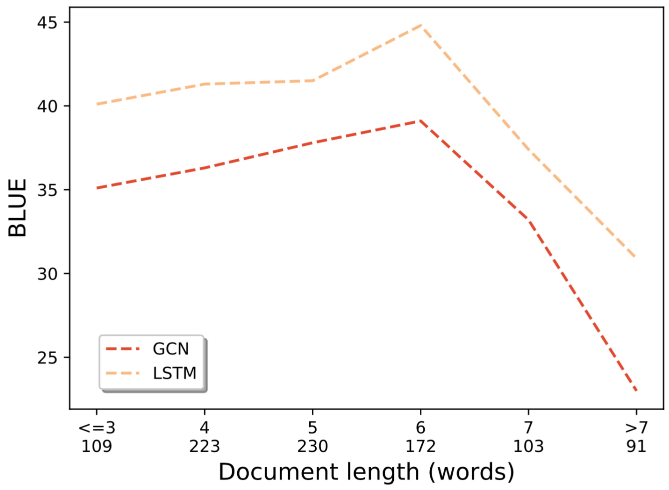


Figure 6: BLEU scores with respect to the sentence length on the combined dev and test set. The x-axis shows the document length in words (top) and the number of documents for that length (bottom).

5.2 Effect of Data Size

We assume that the performance of the models is largely affected by the size and quality of the data, and theoretically, the graph encoders are susceptible to noise in the graph data. Meanwhile, while DRG datasets are usually short sentences, silver data and bronze data contain DRG data with part of long sentences. We believe that such long data makes the size of the graph bigger, which is not conducive to the learning of the graph encoders. Different from the above experiments, we test this hypothesis by retraining the models based on different quality data, instead of using the trained model to test on the test set.

Table 7: Comparison for baseline encoders trained on different size of training data.

Type	**Model**	**Data**	**BLEU**	**METEOR**	**ROUGE**
Graph Encoders	GCN	*gold+silver*	17.0	19.8	43.9
		gold+silver+bronze	37.7	32.0	64.6
Sequential Encoders	LSTM	*gold+silver*	12.8	15.0	35.8
		gold+silver+bronze	44.2	35.0	68.8

Table 7 shows the graph encoders outperform the sequential encoders when using gold data and silver data, but with the addition of bronze data, the sequential encoders will far outperform the graph encoders. It is clear that sequential encoders can benefit more from large datasets than graph encoders, although it is still difficult to distinguish whether the performance of the graph encoders is greatly affected by the data noise or the data size.

5.3 Effect of Indices

The linearized DRG has many indices to represent the connection relationship between concepts. Since the performance of sequential encoders is better than graph encoders, we hypothesize that sequential encoders can learn the index information. To test this, we delete the index information in linearized DRG data and then use the LSTM model to train a new model. As shown in Table 8, the performance is almost unchanged when the training data is without indices.

Table 8: Comparison for LSTM models trained on linearized DRG with and without indices.

DRG Data	**BLEU**	**METEOR**	**ROUGE**
DRGs with indices (standard)	**44.2**	**35.0**	**68.8**
DRGs without indices	43.5	34.7	68.4
DRGs with shuffled concepts	32.8	29.6	59.1

Although the indices themselves do not contain semantic content, they act as pointers to represent semantic relationships between concepts. So it is surprising that the lack of this information doesn't have a significant impact on performance. Perhaps the sequence of concepts and (unconnected) roles comprises enough information for natural language generation, and a layer of indices hardly adds new information, at least for short sentences without long-distance dependencies. We argue that sequential encoders perform better than graph encoders because the linearized DRG contains sequential information for text. To test this hypothesis, we shuffle the order of concepts in DRG and use it as a new training corpus. Our results, see Table 8, show that its performance is substantially lower than that of graph models if they lose sequential information.

6. Conclusion and Future Work

In this study, we presented a meaning-to-text generation task for Dutch. We introduced the process for DRG-to-text generation to investigate the differences between various neural models. We found that with a limited amount of training data, the graph encoders outperform the sequential encoders and that as the amount of data increases, the sequential encoders benefit more and outperform the graph encoders. Hybrid encoders, where sequential information is added to graph encoders, can significantly improve the performance as standard graph encoders lack sequential information modeling.

Future work should address the limitation in size of the data set and whether the conclusion we draw in this work also generalizes to larger datasets, and to datasets with longer texts. It would also be interesting to investigate how the different architectures behave on certain linguistic phenomena such as negation, discourse relations, and long-range dependencies.

References

Abzianidze, Lasha, Johannes Bjerva, Kilian Evang, Hessel Haagsma, Rik van Noord, Pierre Ludmann, Duc-Duy Nguyen, and Johan Bos (2017), The Parallel Meaning Bank: Towards a multilingual corpus of translations annotated with compositional meaning representations, *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, Association for Computational Linguistics, Valencia, Spain, pp. 242–247. https://www.aclweb.org/anthology/E17-2039.

Agirre, Eneko, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa (2009), A study on similarity and relatedness using distributional and WordNet-based approaches, *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, Association for Computational Linguistics, Boulder, Colorado, pp. 19–27. https://aclanthology.org/N09-1003.

Banarescu, Laura, 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, *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, Association for Computational Linguistics, Sofia, Bulgaria, pp. 178–186. https://aclanthology.org/W13-2322.

Bastings, Jasmijn, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Sima'an (2017), Graph convolutional encoders for syntax-aware neural machine translation, *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Copenhagen, Denmark, pp. 1957–1967. https://aclanthology.org/D17-1209.

Beck, Daniel, Gholamreza Haffari, and Trevor Cohn (2018), Graph-to-sequence learning using gated graph neural networks, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Association for Computational Linguistics, Melbourne, Australia, pp. 273–283. https://aclanthology.org/P18-1026.

Bos, Johan (2021), Variable-free discourse representation structures, *Semantics Archive.*

Christiane, Fellbaum (1998), Wordnet: An electronic lexical database, *The MIT Press, Cambridge, Ma., USA.*

Copestake, Ann A., Dan Flickinger, Carl Pollard, and Ivan A. Sag (1997), Minimal recursion semantics: An introduction, *Research on Language and Computation* **3**, pp. 281–332.

Damonte, Marco and Shay B. Cohen (2019), Structural neural encoders for AMR-to-text generation, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Association for Computational Linguistics, Minneapolis, Minnesota, pp. 3649–3658. https://aclanthology.org/N19-1366.

Dzmitry, Bahdanau, Cho Kyunghyun, and Bengio Yoshua (2015), Neural machine translation by jointly learning to align and translate, *in* Yoshua, Bengio and LeCun Yann, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings.* http://arxiv.org/abs/1409.0473.

Fancellu, Federico, Sorcha Gilroy, Adam Lopez, and Mirella Lapata (2019), Semantic graph parsing with recurrent neural network DAG grammars, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Association for Computational Linguistics, Hong Kong, China, pp. 2769–2778. https://aclanthology.org/D19-1278.

Gu, Jiatao, Zhengdong Lu, Hang Li, and Victor O.K. Li (2016), Incorporating copying mechanism in sequence-to-sequence learning, *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Association for Computational Linguistics, Berlin, Germany, pp. 1631–1640. https://aclanthology.org/P16-1154.

Guo, Zhijiang, Yan Zhang, Zhiyang Teng, and Wei Lu (2019), Densely connected graph convolutional networks for graph-to-sequence learning, *Transactions of the Association for Computational Linguistics* **7**, pp. 297–312, MIT Press, Cambridge, MA. https://aclanthology.org/Q19-1019.

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

Kamp, Hans and Uwe Reyle (1993), *From Discourse to Logic; An Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and DRT*, Kluwer, Dordrecht.

Kipper, Karin, Anna Korhonen, Neville Ryant, and Martha Palmer (2008), A large-scale classification of english verbs, *Language Resources and Evaluation* **42**, pp. 21–40.

Klein, Guillaume, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush (2017), OpenNMT: Open-source toolkit for neural machine translation, *Proceedings of ACL 2017, System Demonstrations*, Association for Computational Linguistics, Vancouver, Canada, pp. 67–72. https://aclanthology.org/P17-4012.

Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst (2007), Moses: Open source toolkit for statistical machine translation, *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, Association for Computational Linguistics, Prague, Czech Republic, pp. 177–180. https://www.aclweb.org/anthology/P07-2045.

Konstas, Ioannis, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer (2017), Neural AMR: Sequence-to-sequence models for parsing and generation, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Association for Computational Linguistics, Vancouver, Canada, pp. 146–157. https://aclanthology.org/P17-1014.

Lavie, Alon and Abhaya Agarwal (2007), Meteor: An automatic metric for mt evaluation with high levels of correlation with human judgments, *Proceedings of the Second Workshop on Statistical Machine Translation*, StatMT '07, Association for Computational Linguistics, USA, p. 228–231.

Lin, Chin-Yew (2004), ROUGE: A package for automatic evaluation of summaries, *Text Summarization Branches Out*, Association for Computational Linguistics, Barcelona, Spain, pp. 74–81. https://www.aclweb.org/anthology/W04-1013.

Liu, Jiangming, Shay B. Cohen, and Mirella Lapata (2021), Text generation from discourse representation structures, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Association for Computational Linguistics, Online, pp. 397–415. https://aclanthology.org/2021.naacl-main.35.

Marcheggiani, Diego and Ivan Titov (2017), Encoding sentences with graph convolutional networks for semantic role labeling, *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Copenhagen, Denmark, pp. 1506–1515. https://aclanthology.org/D17-1159.

Matthias, Fey and Lenssen Jan E. (2019), Fast graph representation learning with PyTorch Geometric, *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

Orkphol, Korawit and Wu Yang (2019), Word sense disambiguation using cosine similarity collaborates with word2vec and wordnet, *Future Internet*. https://www.mdpi.com/1999-5903/11/5/114.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu (2002), BLEU: a method for automatic evaluation of machine translation, *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, Philadelphia, Pennsylvania, USA, pp. 311–318. https://aclanthology.org/P02-1040.

Pappas, Nikolaos, Phoebe Mulcaire, and Noah A. Smith (2020), Grounded compositional outputs for adaptive language modeling, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Association for Computational Linguistics, pp. 1252–1267. https://www.aclweb.org/anthology/2020.emnlp-main.96.

Ribeiro, Leonardo F. R., Claire Gardent, and Iryna Gurevych (2019), Enhancing AMR-to-text generation with dual graph representations, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Association for Computational Linguistics, Hong Kong, China, pp. 3183–3194. https://aclanthology.org/D19-1314.

Song, Linfeng, Yue Zhang, Zhiguo Wang, and Daniel Gildea (2018), A graph-to-sequence model for AMR-to-text generation, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Association for Computational Linguistics, Melbourne, Australia, pp. 1616–1626. https://aclanthology.org/P18-1150.

Thomas, N. Kipf and Welling Max (2017), Semi-supervised classification with graph convolutional networks, *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, OpenReview.net. https://openreview.net/forum?id=SJU4ayYgl.

van Noord, Rik, Lasha Abzianidze, Antonio Toral, and Johan Bos (2018), Exploring neural methods for parsing discourse representation structures, *Transactions of the Association for Computational Linguistics* **6**, pp. 619–633. https://www.aclweb.org/anthology/Q18-1043.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin (2017), Attention is all you need, *in* Guyon, I., U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, Curran Associates, Inc., pp. 5998–6008. http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.

Veličković, Petar, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio (2018), Graph Attention Networks, *International Conference on Learning Representations.* accepted as poster. https://openreview.net/forum?id=rJXMpikCZ.

Wang, Chunliu, Noord Rik van, Arianna Bisazza, and Johan Bos (2021), Evaluating text generation from discourse representation structures, *Proceedings of the 1st Workshop on Natural Language Generation, Evaluation, and Metrics (GEM 2021)*, Association for Computational Linguistics, Online, pp. 73–83. https://aclanthology.org/2021.gem-1.8.

Xu, Keyulu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka (2019), How powerful are graph neural networks?, *International Conference on Learning Representations.* https://openreview.net/forum?id=ryGs6iA5Km.

Yujia, Li, Tarlow Daniel, Brockschmidt Marc, and S. Zemel Richard (2016), Gated graph sequence neural networks, *in* Bengio, Yoshua and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings.* http://arxiv.org/abs/1511.05493.