One SPRING to Rule Them Both: Symmetric AMR Semantic Parsing and Generation without a Complex Pipeline

Michele Bevilacqua **Rexhina Blloshmi** **Roberto Navigli**
Sapienza NLP Group
Department of Computer Science, Sapienza University of Rome
{bevilacqua,blloshmi,navigli}@di.uniroma1.it

Abstract

In Text-to-AMR parsing, current state-of-the-art semantic parsers use cumbersome pipelines integrating several different modules or components, and exploit graph recategorization, i.e., a set of content-specific heuristics that are developed on the basis of the training set. However, the generalizability of graph recategorization in an out-of-distribution setting is unclear. In contrast, state-of-the-art AMR-to-Text generation, which can be seen as the inverse to parsing, is based on simpler seq2seq. In this paper, we cast Text-to-AMR and AMR-to-Text as a symmetric transduction task and show that by devising a careful graph linearization and extending a pretrained encoder-decoder model, it is possible to obtain state-of-the-art performances in both tasks using the very same seq2seq approach, i.e., SPRING (*Symmetric PaRsIng aNd Generation*). Our model does not require complex pipelines, nor heuristics built on heavy assumptions. In fact, we drop the need for graph recategorization, showing that this technique is actually harmful outside of the standard benchmark. Finally, we outperform the previous state of the art on the English AMR 2.0 dataset by a large margin: on Text-to-AMR we obtain an improvement of 3.6 Smatch points, while on AMR-to-Text we outperform the state of the art by 11.2 BLEU points. We release the software at `github.com/SapienzaNLP/spring`.

1 Introduction

In recent years Abstract Meaning Representation (Banarescu et al. 2013, AMR) has become an influential formalism for capturing the meaning of a given utterance within a semantic graph (parsing) and, vice versa, producing text from such a graph (generation). AMR's flexibility has resulted in promising improvements in Machine Translation (Song et al. 2019), Text Summarization (Hardy and Vlachos 2018; Liao, Lebanoff, and Liu 2018), Human-Robot Interaction (Bonial et al. 2020) and Information Extraction (Rao et al. 2017). Moreover, AMR-to-Text and Text-to-AMR systems represent an effective bridge between natural language and symbolic representations (which can be manipulated by both humans and computer programs), thus providing a step towards the decoupling of content planning – *what to say* – from language competence – *how to say it*.

Recent state-of-the-art approaches to Text-to-AMR semantic parsing feature very complex pre- and postprocessing pipelines, in which the output of several different components is integrated. Additionally, they employ fine-grained, content-specific heuristics developed on the basis of the training set that, as a consequence, can be very brittle across domains and genres. The parsing performance of simpler, full Sequence-to-Sequence (seq2seq) methods has hitherto lagged behind, mainly because they are less data-efficient than their alternatives.

When it comes to AMR-to-Text generation, which can be seen as the inverse task to Text-to-AMR parsing, vanilla seq2seq methods have, instead, achieved state-of-the-art results. This architectural asymmetry is not observed in other bidirectional transduction tasks such as machine translation, where the same architecture is used to handle the translation from language X to language Y, and vice versa. Motivated by this, a key goal of this paper is to achieve symmetry in AMR parsing and generation as well, by providing the same architecture for both. Moreover, we reduce the complexity of Text-to-AMR architectures by disposing of the need for content-modifying pipelines and additional syntactic and semantic features, which often depend on external components and data-specific heuristics. We achieve this by linearizing the AMR graph efficiently and by extending a pretrained seq2seq model, i.e., BART (Lewis et al. 2020), to handle both AMR-to-Text and Text-to-AMR. In fact, the only external resource consistently beneficial for our model is an off-the-shelf system for Entity Linking – a task that is hard to perform robustly with pure seq2seq models.

Our contributions are the following:

1. We extend a pretrained Transformer encoder-decoder architecture to generate either an accurate linearization of the AMR graph for a sentence or, vice versa, a sentence for a linearization of the AMR graph.
2. Contrary to previous reports (Konstas et al. 2017), we find that the choice between competing graph-isomorphic linearizations does matter. Our proposed Depth-First Search (DFS)-based linearization with special pointer tokens outperforms both the PENMAN linearization and an analogous Breadth-First Search (BFS)-based alternative, especially on AMR-to-Text.
3. We propose a novel Out-of-Distribution (OOD) setting for

estimating the ability of the Text-to-AMR and AMR-to-Text approaches to generalize on open-world data.

4. We show that graph recategorization should be avoided on open-world data because, although it slightly boosts the performance in the standard benchmark, it is not able to generalize in the OOD setting.
5. We outperform the previously best reported results in AMR 2.0 by 11.2 BLEU points for the generation task, and by 3.6 Smatch points for the parsing task.

2 Related Work

Our work is concerned with Text-to-AMR parsing, AMR-to-Text generation, and with how to use pretrained seq2seq models to handle both of these tasks.

2.1 Text-to-AMR Parsing

Pure seq2seq Seq2seq approaches model Text-to-AMR parsing as a transduction of the sentence into a linearization of the AMR graph. Due to their end-to-end nature, such approaches are appealing for this task. However, since seq2seq-based approaches are data-hungry, their performances for AMR parsing have, until now, been rather unsatisfactory, due to the relatively small amount of annotated sentence-AMR pairs. To overcome data sparsity, various different techniques have been employed: self-training using unlabeled English text (Konstas et al. 2017), character-level networks (van Noord and Bos 2017), and concept recategorization as a preprocessing step to reduce the open vocabulary components, e.g., named entities and dates (Peng et al. 2017; van Noord and Bos 2017; Konstas et al. 2017). Moreover, seq2seq-based models often incorporate features such as lemma, POS, or Named Entity Recognition (NER) tags, as well as syntactic and semantic structures (Ge et al. 2019).

To counteract sparsity, we employ transfer learning by exploiting BART (Lewis et al. 2020) – a recently-released pretrained encoder-decoder – to generate a linearized graph incrementally with a single auto-regressive pass of a seq2seq decoder. In fact, the base Transformer encoder-decoder of BART is similar to that of Ge et al. (2019), which differs, however, in that it trains the AMR parsing architecture from scratch.

Hybrid approaches State-of-the-art results in Text-to-AMR have been attained by approaches that use more complex multi-modular architectures. These combine seq2seq methods with graph-based algorithms in either two-stage (Zhang et al. 2019a) or incremental one-stage (Zhang et al. 2019b; Cai and Lam 2020a) procedures. Moreover, they integrate similar processing pipelines and additional features such as the above-mentioned seq2seq approaches (Konstas et al. 2017), including fine-grained graph recategorization (Zhang et al. 2019a,b; Zhou et al. 2020; Cai and Lam 2020a), which all contribute significantly to the performances achieved.

In contrast, our model relies almost exclusively on seq2seq, does not need extra features, and employs a barebone postprocessing pipeline only for ensuring graph validity. Nonetheless, we significantly outperform previous state-of-the-art approaches. Additionally, we show that the extensive recategorization techniques, while boosting performance on the traditional in-domain benchmarks, are harmful in the OOD setting. Moreover, while other approaches have employed pretrained *encoders*, such as BERT (Devlin et al. 2019), in order to have powerful features for a parsing architecture (Zhang et al. 2019a,b; Cai and Lam 2020a), we are the first to show that pretrained *decoders*, too, are beneficial for AMR parsing, even though the pretraining only involves English, and does not include formal representations.

2.2 AMR-to-Text Generation

AMR-to-Text generation is currently performed with two main approaches: explicitly encoding the graph structure in a graph-to-text transduction fashion (Song et al. 2018; Beck, Haffari, and Cohn 2018; Damonte and Cohen 2019; Zhu et al. 2019; Cai and Lam 2020b; Yao, Wang, and Wan 2020), or as a purely seq2seq task through AMR graph linearization (Konstas et al. 2017; Mager et al. 2020). Recent graph-based approaches rely on Transformers to encode AMR graphs (Zhu et al. 2019; Cai and Lam 2020b; Wang, Wan, and Yao 2020; Song et al. 2020; Yao, Wang, and Wan 2020). The model of Mager et al. (2020) is a pretrained Transformer-based decoder-only model fine-tuned on a sequential representation of the AMR graph. Instead, we use an encoder-decoder architecture, which is more suitable for handling conditional generation and casts AMR-to-Text as symmetric to Text-to-AMR, therefore disposing of the need for a task-specific model.

2.3 Linearization Information Loss

Previous approaches to Text-to-AMR parsing (Konstas et al. 2017; van Noord and Bos 2017; Peng et al. 2017; Ge et al. 2019) use seq2seq methods in conjunction with lossy linearization techniques, which, in order to reduce complexity, remove information such as variables from the graph. This information is restored heuristically, making it harder to produce certain valid outputs. In contrast, we propose two linearization techniques which are completely isomorphic to the graph, and do not incur any information loss.

2.4 BART

BART is a Transformer-based encoder-decoder model which is pretrained through a denoising self-supervised task, i.e., reconstructing an English text which has been modified through shuffling, sentence permutation, masking and other kinds of corruption (Lewis et al. 2020). BART has shown significant improvements in conditioned generation tasks where the vocabulary of the input and output sequences largely intersect, such as question answering and summarization. Similarly, a large amount of AMR labels are drawn from the English vocabulary – despite the fact that AMR aims to abstract away from the sentence – and, therefore, we hypothesize that BART's denoising pretraining should be suitable for AMR-to-Text and Text-to-AMR as well. Moreover, it is possible to see a parallel between BART's pretraining task and AMR-to-Text generation, since the linearized AMR graph can be seen as a reordered, partially corrupted

version of an English sentence, which the model has to reconstruct.

3 Method

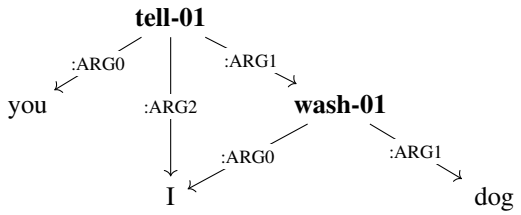
We perform both Text-to-AMR parsing and AMR-to-Text generation with the same architecture, i.e., SPRING, which exploits the transfer learning capabilities of BART for the two tasks. In SPRING AMR graphs are handled symmetrically: for Text-to-AMR parsing the encoder-decoder is trained to predict a graph given a sentence; for AMR-to-Text generation another specular encoder-decoder is trained to predict a sentence given a graph.

In order to use the graphs within the seq2seq model, we transform them into a sequence of symbols using various different linearization techniques (Section 3.1). Furthermore, we modify the BART vocabulary in order to make it suitable for AMR concepts, frames and relations (Section 3.2). Finally, we define lightweight, non content-modifying heuristics to deal with the fact that, in parsing, seq2seq may output strings which cannot be decoded into a graph (Section 3.3).

3.1 Graph Linearizations

In this work we use linearization techniques which are fully graph-isomorphic, i.e., it is possible to encode the graph into a sequence of symbols and then decode it back into a graph without losing adjacency information. We propose the use of special tokens `<R0>`, `<R1>`, ..., `<Rn>` to represent variables in the linearized graph and to handle co-referring nodes. Just as happens with variable names in PENMAN, i.e., the encoding that is used in the release files of AMR, whenever such special tokens occur more than once it is signaled in our encoding that a given node fulfills multiple roles in the graph. By means of this modification we aim to address the confusion arising from the use of seq2seq with PENMAN (PM), which does not allow a clear distinction to be made between constants and variables, as variable names have no semantics. Our special tokens approach is used in combination with two graph traversal techniques based on, respectively, DFS and BFS; in addition, we also experiment with PENMAN. In Figure 1 we show the linearizations of the AMR graph for "You told me to wash the dog".

DFS-based DFS, on which PENMAN is based, is very attractive as it is quite closely related to the way natural language syntactic trees are linearized: consider, e.g., the sentence "the dog which ate the bone which my father found is sleeping", where the noun *dog* is far removed from its head verb, *is sleeping*, because the dependents of *dog* are "explored" completely before the occurrence of the head verb. Thus, we employ a DFS-based linearization with special tokens to indicate variables and parentheses to mark visit depth. Moreover, we dispose of the redundant slash token (`/`). These features significantly reduce the length of the output sequence compared to PENMAN, where variable names are often split into multiple subtokens by the subword tokenizer. This is important for efficient seq2seq decoding with Transformers, which are bottlenecked by the quadratic complexity of attention mechanisms.



PM `( t / tell-01 :ARG0 ( y / you ) :ARG1 ( w / wash-01 :ARG0 i :ARG1 ( d / dog ) ) :ARG2 ( i / i ) )`

DFS `( <R0> tell-01 :ARG0 ( <R1> you ) :ARG1 ( <R3> wash-01 :ARG0 <R2> :ARG1 ( <R4> dog ) ) :ARG2 ( <R2> i ) )`

BFS `<R0> tell-01 :ARG0 <R1> you :ARG1 <R3> wash-01 :ARG2 <R2> i <stop> <R3> :ARG0 <R2> :ARG1 <R4> dog <stop>`

Figure 1: The AMR graph for the sentence "You told me to wash the dog." with the three different linearizations.

BFS-based The use of BFS traversal is motivated by the fact that it enforces a locality principle by which things belonging together are close to each other in the flat representation. Additionally, Cai and Lam (2019) suggest that BFS is cognitively attractive because it corresponds to a core-semantic principle which assumes that the most important pieces of meaning are represented in the upper layers of the graph. To this end, we present a BFS-based linearization which, just like our DFS-based one, uses special tokens to represent co-reference. We apply a BFS graph traversal algorithm which starts from the graph root r and visits all the children w connected by an edge e, appending to the linearization the pointer token to r, e, and then a pointer token if w is a variable, or its value in case w is a constant. The first time a pointer token is appended, we also append its `:instance` attribute. At the end of the iteration at each level, i.e., after visiting the children w, we append a special `<stop>` token to signal the end node exploration. In Figure 1, the visit starts with `tell-01`, iterates over its children, then, after the `<stop>`, goes on to `wash-01`.

Edge ordering All the above linearizations are decoded into the same graph. However, in the PENMAN-linearized gold annotations, an edge ordering can be extracted from each AMR graph. There has been a suggestion (Konstas et al. 2017) that annotators have used this possibility to encode information about argument ordering in the source sentence. Our preliminary experiments confirmed that imposing an edge ordering different from PENMAN has a big negative effect on the evaluation measures of AMR-to-Text generation, due to their order-sensitive nature. To control this, we have carefully designed the linearizations to preserve order information.

3.2 Vocabulary

BART uses a subword vocabulary and its tokenization is optimized to handle English, but it is not well-suited for AMR symbols. To deal with this problem we expand the tokenization vocabulary of BART by adding i) all the relations and frames occurring at least 5 times in the training corpus; ii) constituents of AMR tokens, such as `:op`; iii) the special tokens that are needed for the various graph linearizations. Moreover, we adjust the embedding matrices of encoder and decoder to include the new symbols by adding a vector which is initialized as the average of the subword constituents. The addition of AMR-specific symbols in vocabulary expansion avoids extensive subtoken splitting and thus allows the encoding of AMRs as a more compact sequence of symbols, cutting decoding space and time requirements.

Recategorization Recategorization is a popular technique to shrink the vocabulary size for handling data sparsity. It simplifies the graph by removing sense nodes, wiki links, polarity attributes, and/or by anonymizing the named entities. To assess the contribution of recategorization, we experiment with a commonly-used method in AMR parsing literature (Zhang et al. 2019a,b; Zhou et al. 2020; Cai and Lam 2020a). The method is based on string-matching heuristics and mappings tailored to the training data, which also regulate the restoration process at inference time. We direct the reader to Zhang et al. (2019a) for further details. We note that following common practice we use recategorization techniques only in parsing, due to the considerably higher information loss that could result in generation.

3.3 Postprocessing

In our approach we perform light postprocessing, mainly to ensure the validity of the graph produced in parsing. To this end, we restore parenthesis parity in PENMAN and DFS, and also remove any token which is not a possible continuation given the token that precedes it. For BFS, we recover a valid set of triples between each subsequent pair of `<stop>` tokens. Our approaches remove content limited to a few tokens, often repetitions or hallucinations. We notice that non-recoverable graphs are very rare, roughly lower than 0.02% in out-of-distribution data, with a negligible effect on overall performance. In addition, we integrate an external Entity Linker to handle wikification, because it is difficult to handle the edge cases with pure seq2seq. We use a simple string matching approach to search for a mention in the input sentence for each `:wiki` attribute that SPRING predicted in the graph, then run the off-the-shelf BLINK Entity Linker (Wu et al. 2020) and overwrite the prediction.

4 Experimental Setup

We now describe the setup of the experiments that we perform to evaluate SPRING in both Text-to-AMR parsing and AMR-to-Text generation.

4.1 Datasets

In-Distribution We evaluate the strength of SPRING on the standard evaluation benchmarks, which we refer to as the In-Distribution (ID) setting. The data that we use in this setting are the **AMR 2.0** (LDC2017T10) and **AMR 3.0** (LDC2020T02) corpora releases, which include, respectively 39,260 and 59,255 manually-created sentence-AMR pairs. AMR 3.0 is a superset of AMR 2.0. In both of them the training, development and test sets are a random split of a single dataset, therefore they are drawn from the same distribution.

Out-of-Distribution While the ID setting enables a comparison against previous literature, it does not allow estimates to be made about performances on open-world data, which will likely come from a different distribution of that of the training set. Motivated by common practice in related semantic tasks, such as Semantic Role Labeling (Hajič et al. 2009), we propose a novel OOD setting.

In this evaluation setting we assess the performance of SPRING when trained on OOD data, contrasting it with the ID results. We employ the AMR 2.0 training set, while for testing we use three distinct Out-of-Distribution (OOD) benchmarks, covering a variety of different genres: i) **New3**, a set of 527 instances from AMR 3.0, whose original source was the LORELEI DARPA project – not included in the AMR 2.0 training set – consisting of excerpts from newswire and online forums; ii) **TLP**, the full AMR-tagged children's novel *The Little Prince* (ver. 3.0), consisting of 1,562 pairs; iii) **Bio**, i.e., the test set of the Bio-AMR corpus, consisting of 500 instances, featuring biomedical texts (May and Priyadarshi 2017).

Silver In order to determine whether silver-data augmentation, another commonly used technique, is beneficial in both ID and OOD, we follow Konstas et al. (2017) and create pretraining data by running the SPRING parser using DFS (trained on AMR 2.0) on a random sample of the Gigaword (LDC2011T07) corpus consisting of 200,000 sentences.

4.2 Models

SPRING relies on BART with the augmented vocabulary, as discussed in Section 3.2. We use the same model hyperparameters as BART Large (or Base, when specified), as defined in Huggingface's `transformers` library. Models are trained for 30 epochs using cross-entropy with a batch size of 500 graph linearization tokens, with RAdam (Liu et al. 2020) optimizer and a learning rate of 1×10^{-5}. Gradient is accumulated for 10 batches. Dropout is set to 0.25.

Hyperparameter search We report in Table 1 the final hyperparameters used to train and evaluate both the Text-to-AMR and AMR-to-Text models. To pick these parameters, we used random search with about 25 Text-to-AMR trials in the search space indicated in the third column. Text-to-AMR training requires about 22 and 30 hours on AMR 2.0 and AMR 3.0 using one 1080 Ti GPU, respectively; AMR-to-Text requires 13 and 16.5 hours on AMR 2.0 and AMR 3.0, respectively. At prediction time, we set beam size to 5 following common practice in neural machine translation (Yang, Huang, and Ma 2018).

SPRING variants We include models trained with the three linearizations, indicated as SPRING$^{[lin]}$, where [lin]

Parameter	Pick	Search Space
Optimizer	RAdam	-
Epochs	30	-
LR	$5 * 10^{-5}$	1/5/10/50 $*10^{-5}$
Betas	0.9, 0.999	-
Dropout	0.25	0.1 to 0.25, (+0.05)
W. Decay	0.004	0.001 to 0.01, (+0.001)
LR sched.	constant	-
Grad. accum.	10	1/5/10/15/20
Beam size	5	[1,5]

Table 1: Final hyperparameters and search space for the experiments.

is one of the linearizations: PENMAN (PM), DFS- (DFS) or BFS-based (BFS). In addition, we include variants of SPRING$^{\text{DFS}}$ using i) BART Base (base); ii) graph recategorization (+recat); iii) pretrained silver AMR data (+silver).

BART baseline We also report results on a vanilla BART baseline which treats PENMAN as a string, uses no vocabulary expansion and tokenizes the graph accordingly.

4.3 Comparison Systems

In-Distribution In the ID setting, we use the AMR 2.0 benchmark to compare SPRING variants against the best models from the literature. To this end, we include the following Text-to-AMR parsers: i) Ge et al. (2019, Ge+), an encoder-decoder model which encodes the dependency tree and semantic role structure alongside the sentence; ii) Lindemann, Groschwitz, and Koller (2019, LindGK), a compositional parser based on the *Apply-Modify* algebra; iii) Naseem et al. (2019, Nas+), a transition-based parser trained with a reinforcement-learning objective rewarding the Smatch score; iv) Zhang et al. (2019b, Zhang+), a hybrid graph- and transition-based approach incrementally predicting an AMR graph; v) Zhou et al. (2020, Zhou+), an aligner-free parser (Zhang et al. 2019a) enhanced with latent syntactic structure; vi) Cai and Lam (2020a, CaiL), a graph-based parser iteratively refining an incrementally constructed graph.

For AMR-to-Text, instead, we include the following: i) Zhu et al. (2019, Zhu+), a Transformer-based approach enhanced with structure-aware self-attention; ii) Cai and Lam (2020b, CaiL), a graph Transformer model which relies on multi-head attention (Vaswani et al. 2017) to encode an AMR graph in a set of node representations; iii) Wang, Wan, and Yao (2020, Wang+), a Transformer-based model generating sentences with an additional structure reconstruction objective; iv) Zhao et al. (2020, Zhao+), a graph attention network which explicitly exploits relations by constructing a line graph; v) Yao, Wang, and Wan (2020, Yao+), a graph Transformer-based model which encodes heterogeneous subgraph representations; vi) Mager et al. (2020, Mag+), a fine-tuned GPT-2 model (Radford et al. 2019) predicting the PENMAN linearization of an AMR graph.

For AMR 3.0, which is a recent benchmark, there are no previous systems to compare against. Thus, we train the previous state-of-the-art parsing model of Cai and Lam (2020a) on AMR 3.0 and perform the corresponding evaluation.

Out-of-Distribution In the OOD setting we compare the SPRING$^{\text{DFS}}$ variants when trained on AMR 2.0 and test on OOD data (New3, Bio and TLP) against the best of the same variants trained on the corresponding ID training set when available (i.e., New3 and Bio).

4.4 Evaluation

We evaluate on the Text-to-AMR parsing benchmarks by using Smatch (Cai and Knight 2013) computed with the tools released by Damonte, Cohen, and Satta (2017), which also report fine-grained scores on different aspects of parsing, such as wikification, concept identification, NER and negations. As regards AMR-to-text, we follow previous approaches and evaluate using three common Natural Language Generation (NLG) measures, i.e., BLEU (Papineni et al. 2002, BL), chrF++ (Popović 2017, CH+), and METEOR (Banerjee and Lavie 2005, MET), tokenizing with the script provided with JAMR (Flanigan et al. 2014). Additionally, as AMR abstracts away from many lexical and syntactic choices, we report the scores with untokenized BLEURT (Sellam, Das, and Parikh 2020, BLRT), i.e., a recent regression-based measure which has shown the highest correlation with human judgements in machine translation.

5 Results

We now report the results of our experiments. First, we evaluate SPRING on AMR 2.0 parsing and generation; then, we show, for the first time, the figures on the new AMR 3.0 benchmark. Finally, we tackle our proposed OOD setting.

5.1 AMR 2.0

Text-to-AMR The results on the AMR 2.0 benchmark are reported in Table 2. Among the three different simple linearization models, i.e., SPRING$^{\text{DFS}}$, SPRING$^{\text{BFS}}$, and SPRING$^{\text{PM}}$, the DFS-based one achieves the highest overall Smatch, obtaining slightly better results than the second-best one, the PENMAN, and a wider margin over the BFS one. All our configurations, however, outperform previous approaches by a large margin, with SPRING$^{\text{DFS}}$ outscoring the recategorized model of Cai and Lam (2020a) by 3.6 F1 points. The score gains are spread over most of the fine-grained categories of Damonte, Cohen, and Satta (2017), shown in the third column block in Table 2. The only notable exceptions are wikification and negations, where the score of SPRING$^{\text{DFS}}$ is lower than that of the previous state of the art, i.e., Cai and Lam (2020a), which handles both wiki links and negations heuristically. When we use recategorization, i.e., in SPRING$^{\text{DFS}}$+recat, we obtain a significant boost in performance, which is especially notable in the two above-mentioned categories. Moreover, SPRING$^{\text{DFS}}$+recat achieves the best reported overall performance so far, i.e., 84.5 Smatch F1 points. Regarding the other variants of SPRING$^{\text{DFS}}$, we inspect the contribution of silver data pretraning, i.e., SPRING$^{\text{DFS}}$+silver, and notice a significant improvement over SPRING$^{\text{DFS}}$, suggesting that warm-starting the learning is beneficial in this setting. Indeed, the model

Model	Recat.	Smatch	Unlab.	NoWSD	Conc.	Wiki.	NER	Reent.	Neg.	SRL
Ge+ (2019)	N	74.3	77.3	74.8	84.2	71.3	82.4	58.3	64.0	70.4
LindGK (2019)**	N	75.3	-	-	-	-	-	-	-	-
Nas+ (2019)**	N	75.5	80.0	76.0	86.0	80.0	83.0	56.0	67.0	72.0
Zhang+ (2019b)**	Y	77.0	80.0	78.0	86.0	86.0	79.0	61.0	77.0	71.0
Zhou+ (2020)*	Y	77.5	80.4	78.2	85.9	86.5	78.8	61.1	76.1	71.0
CaiL (2020a)*	N	78.7	81.5	79.2	88.1	81.3	87.1	63.8	66.1	74.5
CaiL (2020a)*	Y	80.2	82.8	80.0	88.1	86.3	81.1	64.6	78.9	74.2
SPRING^{DFS}	N	83.8	86.1	84.4	90.2	84.3	**90.6**	70.8	74.4	79.6
SPRING^{BFS}	N	83.2	85.7	83.7	90.3	83.5	90.2	70.9	70.9	78.2
SPRING^{PM}	N	83.6	86.1	84.1	90.1	83.1	90.2	71.4	72.7	79.4
BART baseline	N	82.7	85.1	83.3	89.7	82.2	90.0	70.8	72.0	79.1
SPRING^{DFS} (base)	N	82.8	85.3	83.3	89.6	83.5	89.9	70.2	71.5	79.0
SPRING^{DFS} +recat	Y	**84.5**	**86.7**	**84.9**	89.6	**87.3**	83.7	72.3	**79.9**	79.7
SPRING^{DFS} +silver	N	84.3	**86.7**	84.8	**90.8**	83.1	90.5	**72.4**	73.6	**80.5**

Table 2: Text-to-AMR parsing results (AMR 2.0). Row blocks: previous approaches; SPRING variants; baseline + other SPRING^{DFS}. Columns: model; recategorization (Y/N); Smatch; Fine-grained scores. The best result per measure across the table is shown in bold. The best result per measure within each row block is underlined. Models marked with */** rely on BERT Base/Large.

of Ge et al. (2019), which does not exploit pretraining, performs considerably worse. We note, however, that in addition to the powerful initialization of BART, our extensions also provide a significant improvement over the BART baseline, ranging from 0.5 (SPRING^{BFS}) to 1.1 (SPRING^{DFS}) Smatch points. Finally, even when we limit the number of parameters, and use BART Base instead, we outperform the previous state of the art, obtaining 82.8 Smatch F1 points.

Finally, we compute the significance of performance differences among SPRING variants using the non-parametric approximate randomization test (Riezler and Maxwell 2005), which is very conservative and appropriate for corpus-level measures. The improvement of SPRING^{DFS} against SPRING^{BFS} and BART baseline is significant with $p < 0.005$, while it is not significant when considering PENMAN linearization.

AMR-to-Text We report in Table 3 the AMR 2.0 AMR-to-Text results. SPRING^{DFS} achieves 45.3 BLEU points, improving the previous state of the art (Yao, Wang, and Wan 2020) by 11 points, and obtains very significant gains in chrF++ and METEOR as well. As far as linearization is concerned, SPRING^{DFS} proves to be significantly stronger than both SPRING^{PM} and SPRING^{BFS} in 3 out of the 4 measures.

This could be due to the fact that DFS is closer to natural language than BFS, and is more compact and efficient than PENMAN (see Section 3.1). Similarly to the Text-to-AMR task results, the pretraining with silver data boosts the performance, with SPRING^{DFS}+silver improving the baseline by 0.6 BLEU points. Finally, there is a big gain against the fine-tuned GPT-2 model of Mager et al. (2020), demonstrating that using a pretrained decoder on its own is suboptimal. As in Text-to-AMR, we compute the significance of results using the non-parametric approximate randomization test. The performance gap between SPRING^{DFS} and the alternatives in AMR-to-Text, i.e., SPRING^{PM}, SPRING^{BFS}, and BART baseline, is significant with $p < 0.001$.

	BL	CH+	MET	BLRT
Zhu+ (2019)	31.8	64.1	36.4	-
CaiL (2020b)	29.8	59.4	35.1	-
Wang+ (2020)	32.1	64.0	36.1	-
Zhao+ (2020)	32.5	-	36.8	-
Mag+ (2020)	33.0	63.9	37.7	-
Yao+ (2020)	34.1	65.6	38.1	-
SPRING^{DFS}	45.3	73.5	41.0	56.5
SPRING^{BFS}	43.6	72.1	40.5	54.6
SPRING^{PM}	43.7	72.5	41.3	56.0
BART baseline	42.7	72.2	40.7	54.8
SPRING^{DFS} +silver	**45.9**	**74.2**	**41.8**	**58.1**

Table 3: AMR-to-Text generation results (AMR 2.0). Row blocks: previous approaches; SPRING variants; baseline +silver. Columns: measures. Bold/underline as in Table 2.

5.2 AMR 3.0

The results on AMR 3.0 (Table 4) confirm that SPRING^{DFS} obtains the best performance. However, the important thing to note here is that graph recategorization, without significant human effort in expanding the heuristics,[1] is not able to scale on a more diverse benchmark such as AMR 3.0: SPRING^{DFS}+recat achieves lower performances than the non-recategorized counterpart, with the exception of negations, whose heuristics are probably more resilient to change in data distribution. Note that the harmful impact of recategorization outside of AMR 2.0 is noticeable even with the

[1]We use the heuristics designed by Zhang et al. (2019a) which were optimized on the AMR 2.0 training set.

	CaiL	CaiL+r	S^{DFS}	S^{DFS}+s	S^{DFS}+r
Text-to-AMR					
Smatch	78.0	76.7	**83.0**	**83.0**	80.2
Unlab.	81.9	80.6	**85.4**	**85.4**	83.1
NoWSD	78.5	77.2	**83.5**	**83.5**	80.7
Conc.	88.5	86.5	**89.8**	89.5	87.7
Wiki.	75.7	77.3	**82.7**	81.2	77.8
NER	83.7	74.7	**87.2**	87.1	79.8
Reent.	63.7	62.6	70.4	**71.3**	69.7
Neg.	68.9	72.6	73.0	71.7	**75.1**
SRL	73.2	72.2	78.9	**79.1**	78.1
AMR-to-Text					
BL	-	-	44.9	**46.5**	-
CH+	-	-	72.9	**73.9**	-
MET	-	-	40.6	**41.7**	-
BLRT	-	-	57.3	**60.8**	-

Table 4: Text-to-AMR and AMR-to-Text results on AMR 3.0. Best in bold. $S^{[lin]}$ = $SPRING^{[lin]}$. +s/r = +silver/recat.

	New3	TLP	Bio
Text-to-AMR			
$SPRING^{DFS}$ (ID)	78.6	-	79.9
$SPRING^{DFS}$	**73.7**	77.3	**59.7**
$SPRING^{DFS}$+recat	63.8	76.2	49.5
$SPRING^{DFS}$+silver	71.8	**77.5**	59.5
AMR-to-Text			
$SPRING^{DFS}$ (ID)	61.5	-	32.3
$SPRING^{DFS}$	**51.7**	**41.5**	5.2
$SPRING^{DFS}$+silver	50.2	40.4	**5.9**

Table 5: OOD evaluation on Text-to-AMR (Smatch) and AMR-to-Text (BLEURT). Best in bold.

pretrained model of Cai and Lam (2020a).

5.3 Out-of-Distribution

Finally, we show in Table 5 the results of the evaluation on the OOD datasets. As can be seen, there is constantly a big difference between the score achieved by the OOD models and the best ID counterparts (see OOD paragraph in Section 4.3), indicated as $SPRING^{DFS}$ (ID). Interestingly enough, not using recategorization results in consistently higher performances than using it. This is especially notable for Bio, which, in addition to being OOD with respect to the AMR 2.0 training set, is also out-of-domain. On this dataset $SPRING^{DFS}$ (ID) model outperforms $SPRING^{DFS}$ by over 20 Smatch points, and $SPRING^{DFS}$+recat by over 30 points. On New3, which is not out-of-domain, the difference with ID is noticeably narrower compared to $SPRING^{DFS}$ (4.9 Smatch points), but considerably larger against the $SPRING^{DFS}$+recat. Recategorization is not as harmful in TLP, perhaps because the text of the underlying children's story is simpler. Differently from the results on AMR 2.0, $SPRING^{DFS}$ +silver does not show consistent improvements over $SPRING^{DFS}$. We attribute this to the fact that the pretraining corpus, i.e., Gigaword, is similar in distribution to AMR 2.0, so that the boost in performance in AMR 2.0 benchmark comes due to overfitting on some genres and is not general.

6 Case Study Analysis: Negation

Through the OOD and AMR 3.0 benchmark evaluation, we demonstrated the harmful impact of recategorization rules based on training sets. Interestingly, across experiments, the breakdown scores (Damonte, Cohen, and Satta 2017) for many aspects of meaning were consistently better without recategorization, with the exception of negations. Negations are handled by a commonly-used rule-based method (Zhang et al. 2019a): `:polarity` attributes are discarded during training – causing a loss of information – and are restored by i) identifying the negated lemmas usually associated with negative polarity words such as *no, not* and *never*; ii) aligning the lemma to the corresponding node in the graph by string-matching heuristics; iii) adding the `:polarity` attribute to the aligned node. Hand-crafted rules lead to high precision due to the frequency of common patterns. However, there are many cases which the heuristics cannot handle correctly, while fully-learned approaches are able to, as they do not constrain the possible outputs they produce.

In Table 6 we contrast the predictions of $SPRING^{DFS}$ with $SPRING^{DFS}$ +recat, trained on AMR 2.0, on several edge cases which heuristics fail to handle. Example (1) shows a standard negation with *don't* + verb, which the designed heuristics handle easily. However, simply changing a word, as in example (2), makes the rule-based system crucially depend on word-to-node alignment, which is non-trivial when the same lemma (*say*) appears multiple times. Thus, in this case, the heuristics misalign the negated occurrence of *say*, and introduce `:polarity` at a lower level in the graph. Additionally, syntax makes it such that assumptions based on word order may easily fail: in example (3) heuristics negate the closest lemma to the negation, i.e., *pupil*, instead of the root of graph *love-01*, which corresponds to a word occurring further along in the sentence. However, even if the heuristics were rewritten to take syntax into account, it would still be difficult to handle cases like example (4): the negation *don't* takes large scope over the conjunction, resulting in many `:polarity` edges in the AMR graph. Finally, while due to space constraints the analysis here is limited to negations, similar problems tend to appear whenever fine-grained rules are applied to the input sentence, e.g., for entities, dates or politeness markers.

7 Conclusion

In this paper we presented a simple, symmetric approach for performing state-of-the-art Text-to-AMR parsing and AMR-to-Text generation with a single seq2seq architecture. To achieve this, we extend a Transfomer encoder-decoder model pretrained on English text denoising to also work with AMR. Furthermore, we put forward a novel AMR graph

SPRING^{DFS}	SPRING^{DFS}+recat
(1) *I didn't say he believes that.*	

`(s / say-01`
`  :polarity -`
`  :ARG0 (i / i)`
`  :ARG1 (b / believe-01`
`  :ARG0 (h / he)`
`  :ARG1 (t / that)))`
	`(s / say-01`
	`  :polarity -`
	`  :ARG0 (i / i)`
	`  :ARG1 (b / believe-01`
	`  :ARG0 (h / he)`
	`  :ARG1 (t / that)))`

(2) *I didn't say he said that.*	

`(s / say-01`
`  :polarity -`
`  :ARG0 (i / i)`
`  :ARG1 (s2 / say-01`
`  :ARG0 (h / he)`
`  :ARG1 (t / that)))`
	`(s / say-01`
	`  :ARG0 (i / i)`
	`  :ARG1 (s2 / say-01`
	`  :polarity -`
	`  :ARG0 (h / he)`
	`  :ARG1 (t / that)))`

(3) *Don't the pupils who have come last year love to study?*	

`(l / love-01`
`  :polarity -`
`  :mode interrogative`
`  :ARG0 (p / pupil`
`  :ARG1-of (c / come-01`
`  :time (y / year`
`  :mod (l / last))))`
`  :ARG1 (s / study-01`
`  :ARG0 p))`
	`(l / love-01`
	`  :mode interrogative`
	`  :ARG0 (p / pupil`
	`  :polarity -`
	`  :ARG1-of (c / come-01`
	`  :time (y / year`
	`  :mod (l / last))))`
	`  :ARG1 (s / study-01`
	`  :ARG0 p))`

(4) *Don't eat or drink*	

`(o / or`
`  :op1 (e / eat-01`
`  :mode imperative`
`  :polarity -`
`  :ARG0 (y / you))`
`  :op2 (d / drink-01`
`  :mode imperative`
`  :polarity -`
`  :ARG0 y))`
	`(o / or`
	`  :op1 (e / eat-01`
	`  :mode imperative`
	`  :polarity -`
	`  :ARG0 (y / you))`
	`  :op2 (d / drink-01`
	`  :mode imperative`
	`  :ARG0 y))`

Table 6: Example of graphs parsed by SPRING^{DFS} and SPRING^{DFS}+recat for different sentences involving negations.

DFS-based linearization which, in addition to being more compact than its alternatives, does not incur any information loss. Most importantly, we drop most of the requirements of competing approaches: cumbersome pipelines, heavy heuristics (often tailored to the training data), along with most external components. Despite such cutting down on complexity, we strongly outperform the previous state of the art on both parsing and generation, reaching 83.8 Smatch and 45.3 BLEU, respectively. We also propose an Out-of-Distribution setting, which enables evaluation on different genres and domains from those of the training set. Thanks to this setting, we are able to show that the integration of recategorization techniques or silver data – popular techniques for boosting performances – harm the performances in both parsing and generation. Employing a simpler approach like ours, based on lighter assumptions, allows for more robust generalization. Here we show the generalizability of the models on different data distributions and across domains, while leaving the extension across languages as in Blloshmi, Tripodi, and Navigli (2020) and across formalisms (Navigli 2018) for future work. Finally, we invite the community to use the OOD evaluation to enable the development of more robust automatic AMR approaches. Furthermore, we believe our contributions will open up more directions towards the integration of parsing and generation. We release our software at `github.com/SapienzaNLP/spring`.

Acknowledgments

The authors gratefully acknowledge the support of the ERC Consolidator Grant MOUSSE No. 726487 and the ELEXIS project No. 731015 under the European Union's Horizon 2020 research and innovation programme.

This work was partially supported by the MIUR under the grant "Dipartimenti di eccellenza 2018-2022" of the Department of Computer Science of Sapienza University.

References

Banarescu, L.; Bonial, C.; Cai, S.; Georgescu, M.; Griffitt, K.; Hermjakob, U.; Knight, K.; Koehn, P.; Palmer, M.; and Schneider, N. 2013. Abstract Meaning Representation for Sembanking. In *Proc. of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, 178–186.

Banerjee, S.; and Lavie, A. 2005. METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. In *Proc. of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, 65–72. Ann Arbor, Michigan.

Beck, D.; Haffari, G.; and Cohn, T. 2018. Graph-to-Sequence Learning using Gated Graph Neural Networks. In *Proc. of ACL 2018*, 273–283. Melbourne, Australia.

Blloshmi, R.; Tripodi, R.; and Navigli, R. 2020. XL-AMR: Enabling Cross-Lingual AMR Parsing with Transfer Learning Techniques. In *Proc. of EMNLP 2020*, 2487–2500. Online.

Bonial, C.; Donatelli, L.; Abrams, M.; Lukin, S. M.; Tratz, S.; Marge, M.; Artstein, R.; Traum, D. R.; and Voss, C. R. 2020. Dialogue-AMR: Abstract Meaning Representation for Dialogue. In *Proc. of LREC 2020*, 684–695.

Cai, D.; and Lam, W. 2019. Core Semantic First: A Top-down Approach for AMR Parsing. In *Proc. of EMNLP-IJCNLP 2019*, 3799–3809. Hong Kong, China.

Cai, D.; and Lam, W. 2020a. AMR Parsing via Graph-Sequence Iterative Inference. In *Proc. of ACL 2020*, 1290–1301. Online.

Cai, D.; and Lam, W. 2020b. Graph Transformer for Graph-to-Sequence Learning. In *Proc. of AAAI 2020*, 7464–7471.

Cai, S.; and Knight, K. 2013. Smatch: an Evaluation Metric for Semantic Feature Structures. In *Proc. of ACL 2013*, 748–752.

Damonte, M.; and Cohen, S. B. 2019. Structural Neural Encoders for AMR-to-text Generation. In *Proc. of NAACL 2019: Human Language Technologies*, 3649–3658. Minneapolis, Minnesota.

Damonte, M.; Cohen, S. B.; and Satta, G. 2017. An Incremental Parser for Abstract Meaning Representation. In *Proc. of EACL 2017*, 536–546. Valencia, Spain.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proc. of NAACL 2019: Human Language Technologies*, 4171–4186. Minneapolis, Minnesota. doi:10.18653/v1/N19-1423.

Flanigan, J.; Thomson, S.; Carbonell, J.; Dyer, C.; and Smith, N. A. 2014. A Discriminative Graph-Based Parser for the Abstract Meaning Representation. In *Proc. of ACL 2014*, 1426–1436. Baltimore, Maryland.

Ge, D.; Li, J.; Zhu, M.; and Li, S. 2019. Modeling Source Syntax and Semantics for Neural AMR Parsing. In *Proc. of IJCAI 2019*, 4975–4981.

Hajič, J.; Ciaramita, M.; Johansson, R.; Kawahara, D.; Martí, M. A.; Màrquez, L.; Meyers, A.; Nivre, J.; Padó, S.; Štěpánek, J.; Straňák, P.; Surdeanu, M.; Xue, N.; and Zhang, Y. 2009. The CoNLL-2009 Shared Task: Syntactic and Semantic Dependencies in Multiple Languages. In *Proc. of CoNLL 2009: Shared Task*, 1–18. Boulder, Colorado.

Hardy; and Vlachos, A. 2018. Guided Neural Language Generation for Abstractive Summarization using Abstract Meaning Representation. In *Proc. of EMNLP 2020*, 768–773.

Konstas, I.; Iyer, S.; Yatskar, M.; Choi, Y.; and Zettlemoyer, L. 2017. Neural AMR: Sequence-to-Sequence Models for Parsing and Generation. In *Proc. of ACL 2017*, 146–157. Vancouver, Canada.

Lewis, M.; Liu, Y.; Goyal, N.; Ghazvininejad, M.; Mohamed, A.; Levy, O.; Stoyanov, V.; and Zettlemoyer, L. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proc. of ACL 2020*, 7871–7880.

Liao, K.; Lebanoff, L.; and Liu, F. 2018. Abstract Meaning Representation for Multi-Document Summarization. In *Proc. of the 27th International Conference on Computational Linguistics*, 1178–1190.

Lindemann, M.; Groschwitz, J.; and Koller, A. 2019. Compositional Semantic Parsing across Graphbanks. In *Proc. of ACL 2019*, 4576–4585. Florence, Italy.

Liu, L.; Jiang, H.; He, P.; Chen, W.; Liu, X.; Gao, J.; and Han, J. 2020. On the Variance of the Adaptive Learning Rate and Beyond. In *ICLR 2020*.

Mager, M.; Fernandez Astudillo, R.; Naseem, T.; Sultan, M. A.; Lee, Y.-S.; Florian, R.; and Roukos, S. 2020. GPT-too: A Language-Model-First Approach for AMR-to-Text Generation. In *Proc. of ACL 2020*, 1846–1852. Online.

May, J.; and Priyadarshi, J. 2017. SemEval-2017 Task 9: Abstract Meaning Representation Parsing and Generation. In *Proc. of the 11th International Workshop on Semantic Evaluation*, 536–545. Vancouver, Canada.

Naseem, T.; Shah, A.; Wan, H.; Florian, R.; Roukos, S.; and Ballesteros, M. 2019. Rewarding Smatch: Transition-Based AMR Parsing with Reinforcement Learning. In *Proc. of ACL 2019*, 4586–4592. Florence, Italy.

Navigli, R. 2018. Natural Language Understanding: Instructions for (Present and Future) Use. In *Proc. of IJCAI 2018*, 5697–5702.

Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W.-J. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In *Proc. of ACL 2002*, 311–318. Philadelphia, Pennsylvania, USA.

Peng, X.; Wang, C.; Gildea, D.; and Xue, N. 2017. Addressing the Data Sparsity Issue in Neural AMR Parsing. In *Proc. of EACL 2017*, 366–375. Valencia, Spain.

Popović, M. 2017. chrF++: words helping character n-grams. In *Proc. of the 2nd Conference on Machine Translation*, 612–618. Copenhagen, Denmark.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; and Sutskever, I. 2019. Language Models are Unsupervised Multitask Learners. - .

Rao, S.; Marcu, D.; Knight, K.; and III, H. D. 2017. Biomedical Event Extraction using Abstract Meaning Representation. In *Proc. of BioNLP*, 126–135. Vancouver, Canada.

Riezler, S.; and Maxwell, J. T. 2005. On Some Pitfalls in Automatic Evaluation and Significance Testing for MT. In *Proc. of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, 57–64. Ann Arbor, Michigan.

Sellam, T.; Das, D.; and Parikh, A. 2020. BLEURT: Learning Robust Metrics for Text Generation. In *Proc. of ACL 2020*, 7881–7892. Online.

Song, L.; Gildea, D.; Zhang, Y.; Wang, Z.; and Su, J. 2019. Semantic Neural Machine Translation using AMR. *TACL* 7: 19–31.

Song, L.; Wang, A.; Su, J.; Zhang, Y.; Xu, K.; Ge, Y.; and Yu, D. 2020. Structural Information Preserving for Graph-to-Text Generation. In *Proc. of ACL 2020*, 7987–7998. Online.

Song, L.; Zhang, Y.; Wang, Z.; and Gildea, D. 2018. A Graph-to-Sequence Model for AMR-to-Text Generation. In *Proc. of ACL 2018*, 1616–1626. Melbourne, Australia.

van Noord, R.; and Bos, J. 2017. Neural Semantic Parsing by Character-based Translation: Experiments with Abstract Meaning Representations. *CoRR* abs/1705.09980.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, u.; and Polosukhin, I. 2017. Attention is All You Need. In *Proc. of NIPS 2017*, NIPS'17, 6000–6010. Red Hook, NY, USA.

Wang, T.; Wan, X.; and Yao, S. 2020. Better AMR-To-Text Generation with Graph Structure Reconstruction. In *Proc. of IJCAI 2020*, 3919–3925.

Wu, L.; Petroni, F.; Josifoski, M.; Riedel, S.; and Zettlemoyer, L. 2020. Scalable Zero-shot Entity Linking with Dense Entity Retrieval. In *Proc. of EMNLP 2020*, 6397–6407. Online.

Yang, Y.; Huang, L.; and Ma, M. 2018. Breaking the Beam Search Curse: A Study of (Re-)Scoring Methods and Stopping Criteria for Neural Machine Translation. In *Proc. of EMNLP 2018*, 3054–3059. Brussels, Belgium.

Yao, S.; Wang, T.; and Wan, X. 2020. Heterogeneous Graph Transformer for Graph-to-Sequence Learning. In *Proc. of ACL 2020*, 7145–7154. Online.

Zhang, S.; Ma, X.; Duh, K.; and Van Durme, B. 2019a. AMR Parsing as Sequence-to-Graph Transduction. In *Proc. of ACL 2019*, 80–94. Florence, Italy.

Zhang, S.; Ma, X.; Duh, K.; and Van Durme, B. 2019b. Broad-Coverage Semantic Parsing as Transduction. In *Proc. of EMNLP-IJCNLP 2019*, 3786–3798. Hong Kong, China.

Zhao, Y.; Chen, L.; Chen, Z.; Cao, R.; Zhu, S.; and Yu, K. 2020. Line Graph Enhanced AMR-to-Text Generation with Mix-Order Graph Attention Networks. In *Proc. of ACL 2020*, 732–741. Online.

Zhou, Q.; Zhang, Y.; Ji, D.; and Tang, H. 2020. AMR Parsing with Latent Structural Information. In *Proc. of ACL 2020*, 4306–4319. Online.

Zhu, J.; Li, J.; Zhu, M.; Qian, L.; Zhang, M.; and Zhou, G. 2019. Modeling Graph Structure in Transformer for Better AMR-to-Text Generation. In *Proc. of EMNLP-IJCNLP 2019*, 5459–5468. Hong Kong, China.