# BiBL: AMR Parsing and Generation with Bidirectional Bayesian Learning

**Ziming Cheng**[1,2], **Zuchao Li**[1,2,*], **and Hai Zhao**[1,2,*]
[1]Department of Computer Science and Engineering, Shanghai Jiao Tong University
[2]MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University
{kk.cheng, charlee}@sjtu.edu.cn, zhaohai@cs.sjtu.edu.cn

## Abstract

Abstract Meaning Representation (AMR) offers a unified semantic representation for natural language sentences. Thus transformation between AMR and text yields two transition tasks in opposite directions, i.e., Text-to-AMR parsing and AMR-to-Text generation. Existing AMR studies only focus on one-side improvements despite the duality of the two tasks, and their improvements are greatly attributed to the inclusion of large extra training data or complex structure modifications which harm the inference speed. Instead, we propose data-efficient Bidirectional Bayesian learning (BiBL) to facilitate bidirectional information transition by adopting a single-stage multitasking strategy so that the resulting model may enjoy much lighter training at the same time. Evaluation on benchmark datasets shows that our proposed BiBL outperforms strong previous seq2seq refinements without the help of extra data which is indispensable in existing counterpart models. We release the codes of BiBL at: https://github.com/KHAKhazeus/BiBL.

## 1 Introduction

Abstract Meaning Representation (AMR) (Banarescu et al., 2013) is a formalism that could capture the semantic meaning of a natural language sentence in a graph representation. An AMR graph example is shown in Figure 1. Leaves in the AMR graph denote concepts in the text sentence while other nodes within the graph are entities. Two entities could be connected with relations defined under a common standard, and instance relations connect entities and corresponding concepts. The flexibility and semantic invariance characteristics make AMR applicable to various natural language processing (NLP) downstream tasks and achieve solid performance.

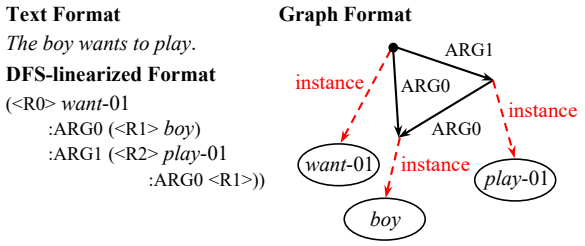


Figure 1: AMR graph and DFS-linearized formats for representing a text sentence "The boy wants to play". Instance relations are denoted with dotted red lines while other relations are denoted with black lines. Concepts in AMR graphs are enclosed by oval shapes. The indentation in the DFS-linearized format is only for clarity purposes.

The transformation between AMR graphs and texts forms two basic tasks offering helpful AMR information for the downstream tasks. Traditional approaches treat AMR-to-Text generation and Text-to-AMR parsing separately. However, utilizing graph linearization and the power of pre-trained language models, SPRING (Bevilacqua et al., 2021) casts two tasks into a unified sequence-to-sequence (seq2seq) structure and outperforms all traditional approaches proposed before. Such progress consolidates the prevalent position of seq2seq approaches tackling the AMR transition tasks.

Recently, many methods have been proposed to further enhance the seq2seq model. In order to improve the model performance on Text-to-AMR parsing, Zhou et al. (2021) incorporates an action-pointer mechanism. Yu and Gildea (2022) adds ancestor information of AMR graphs into the model. Chen et al. (2022) introduces a data augmentation approach by transforming data excerpted from semantic role labeling and dependency parsing into pseudo-AMR data, and then constructing extra training tasks using these data. Hoang et al. (2021) proposes graph ensembling to combine different models in order to achieve better results. For the AMR-to-Text generation task, Anonymous (2021)

*Corresponding author. This work was supported by Key Projects of National Natural Science Foundation of China (U1836222 and 61733011).



*Proceedings of the 29th International Conference on Computational Linguistics*, pages 5461–5475
October 12–17, 2022.

designs two auxiliary training objectives, i.e., relationship prediction and distance prediction of nodes in AMR graphs. Ribeiro et al. (2021) utilizes data from the KG2Text task, which is transforming knowledge graphs into texts, and masked language modeling to pre-train the model. Bai et al. (2022) proposes six graph pre-training tasks to obtain a better pre-trained model. Though the above-mentioned refinements are effective, applying auxiliary techniques, including ensembling and adding graph information during training, could greatly harm the training or inference speed. Meanwhile, obtaining a better pre-training model using extra silver data or data augmentation techniques greatly increases the time consumption and design complexity of the whole model training phase.

Instead of using extra data or complicated techniques which harm the training or inference speed, we propose a novel solution for AMR transition tasks. The solution consists of two auxiliary tasks to help the model grasp the joint probability of the AMR graphs and texts. By interweaving the two auxiliary tasks with the main transduction task, the extra knowledge learned in the two auxiliary tasks could boost the model understanding of the main task. Therefore, single-stage multitask learning without fine-tuning could be adopted for BiBL. Based on our proposed BiBL, the training paradigm could be greatly simplified, and under the same model settings, our implemented models achieve new state-of-the-art performances in AMR-to-Text generation. Compared with previous refinements, further comparison experiments show that BiBL is more efficient both during the training and inference phases.

## 2 Related Work

There are two tasks when concerning the transformation between AMR graphs and texts. However, it is not always that these two tasks are jointly considered.

### 2.1 Text-to-AMR Parsing

**Graph-based approaches** Many graph-based approaches (Zhang et al., 2019a,b; Cai and Lam, 2020a,b; Zhou et al., 2020) have been proposed to solve the transition from texts to AMR graphs. These approaches merge graph structures into their model design, and graph recategorization technique (Zhang et al., 2019a; Zhou et al., 2020; Cai and Lam, 2020a) is introduced to unify different graph representations that are identical in semantics. Despite the complex graph structure design, these models can achieve decent performances due to the explicit incorporation of AMR graph features.

**Pure seq2seq approaches** Pure seq2seq approaches for the Text-to-AMR task are data-hungry in nature. Therefore, in previous works, several methods have been explored to counter the data sparsity problem (Konstas et al., 2017; van Noord and Bos, 2017). However, these approaches either introduce a large amount of data that is not closely related to the parsing task or reduce the capacity of the model by ignoring and recovering fine-grained details through data pre-processing and post-processing. Utilizing BART (Lewis et al., 2020) as the model backbone, SPRING (Bevilacqua et al., 2021) provides a new seq2seq solution for Text-to-AMR parsing without removing details during the training process. Nonetheless, SPRING obtains solid model performance gains compared with previous methods and consequently becomes the backbones of many recent refinements.

### 2.2 AMR-to-Text Generation

**Graph2seq approaches** AMR-to-Text generation is the reverse process of Text-to-AMR parsing. Taking graph characteristics into consideration, several graph2seq methods (Beck et al., 2018; Guo et al., 2019; Wang et al., 2020a,b; Song et al., 2018; Zhu et al., 2019) utilize graph neural networks to tackle the problem. However, the performances of graph2seq approaches are inferior to the state-of-the-art approach, i.e., SPRING, which is purely seq2seq.

**Pure seq2seq approaches** Similar to pure seq2seq approaches in the Text-to-AMR parsing task, graph linearization is the key to transforming the task into a seq2seq formulation. Based on this formulation, Mager et al. (2020) adapts a pre-trained Transformer decoder to the generation task through fine-tuning. SPRING (Bevilacqua et al., 2021) fine-tunes BART on the generation task and proves that the encoder-decoder structure of BART greatly contributes to the final model performance. Since SPRING is powerful for both transduction directions, BiBL inherits the model backbone of SPRING, which guarantees a solid basis of model performance.

## 3 Method

As the seq2seq model only receives and emits data in sequential forms, we have to take the necessary pre-processing and post-processing to perform conversions between AMR graphs and their sequential forms. Then, the seq2seq transduction is completed using sequence generation models of encoder-decoder structure.

On top of the seq2seq model design, we enhance bidirectional model performance through two auxiliary tasks which facilitate the model's understanding of the joint probability between texts and AMR graphs.

### 3.1 Graph Linearization

For both AMR-to-Text generation and Text-to-AMR parsing, graph linearization is a necessary pre-processing task in seq2seq modeling. First, we replace the variables in the AMR graphs with specially designed tokens to avoid unnecessary tokenization. These tokens are added to the model as special tokens. Then, we linearize the graphs following a depth-first search (DFS) fashion. We use parentheses to indicate visit depth and use blank spaces to separate variables and actual concepts. A DFS-linearized example is shown in Figure 1.

Since the capacity of the model backbone is fairly adequate and graph recategorization mentioned in Section 2.1 harms the model performance in out-of-distribution settings, we do not incorporate graph recategorization into the pre-processing procedures of BiBL.

### 3.2 Task Formulation

Following the design of graph linearization, the bidirectional transition task could be transformed into a seq2seq task formulation. Given a linearized AMR graph $Y = [y_1, ..., y_k]$ and the corresponding sentence $X = [x_1, ..., x_n]$ where the length of the linearized graph is $k$ and the length of the text sentence is $n$, the goal of Text-to-AMR parsing is generating $Y$ given $X$, and the goal of AMR-to-Text generation is generating $X$ given $Y$. Specifically, the target of Text-to-AMR parsing is to maximize $P(Y|X)$, and the target of AMR-to-Text generation is to maximize $P(X|Y)$.

Sequence generation in NLP tasks is autoregressive in nature. Therefore, the two conditional probabilities mentioned above could be decomposed according to the following formulations:

$$\begin{aligned} P(Y|X) &= \prod_{t=1}^{k} P(y_t|X, Y_{<t}) \\ P(X|Y) &= \prod_{t=1}^{n} P(x_t|Y, X_{<t}) \end{aligned} \tag{1}$$

where $t$ denotes the index of the decoding step, $Y_{<t}$ and $X_{<t}$ denote the tokens before $t$ position in the linearized AMR graph and the text sentence, respectively. $P(y_t|X, Y_{<t})$ and $P(x_t|Y, X_{<t})$ denote the probability distributions of each word in the target vocabulary being the next candidate.

### 3.3 Post-processing

For the Text-to-AMR parsing task, post-processing is required to ensure the validity of the generated AMR sequences. We restore parenthesis parity, remove tokens that are not possible descendants of the previous token and remove tokens that are obvious repetitions through the rule-based filter mechanism proposed in SPRING. Moreover, since each kind of wiki tag is sparse in the whole data distribution, it is hard for the seq2seq model to attach correct wiki concepts. Therefore, we use the BLINK Entity Linker (Wu et al., 2020) to overwrite the wiki tags generated by BiBL.

### 3.4 Encoder-Decoder for Sequence Generation

To model the probability distribution $P(y_t|X, Y_{<t})$ and $P(x_t|Y, X_{<t})$ in Equation 1, we adopt an encoder-decoder model structure. In detail, the encoder first turns the input tokens into embeddings, and then multi-head attention modules transform the embeddings into hidden representations. For the simplicity of the formulation, we ignore the begin-of-sentence `[BOS]`, end-of-sentence `[EOS]`, and separation tokens `[SEP]` in the input tokens. We could represent the process of the encoder as follows:

$$H^e = \text{Encoder}(H^i, \theta_e) \tag{2}$$

where $\theta_e$ denotes trainable parameters of the encoder module, $H^i \in \mathcal{R}^{l \times d}$ denotes the input embeddings of length $l$, $H^e \in \mathcal{R}^{l \times d}$ denotes the output embeddings of the encoder, and $d$ denotes the hidden size of the whole model.

Next, for the decoding step of $t$ during training, the decoder module takes the encoder output embeddings $H^e$ and the ground truth tokens before $t$ position as inputs. Then, it generates a hidden

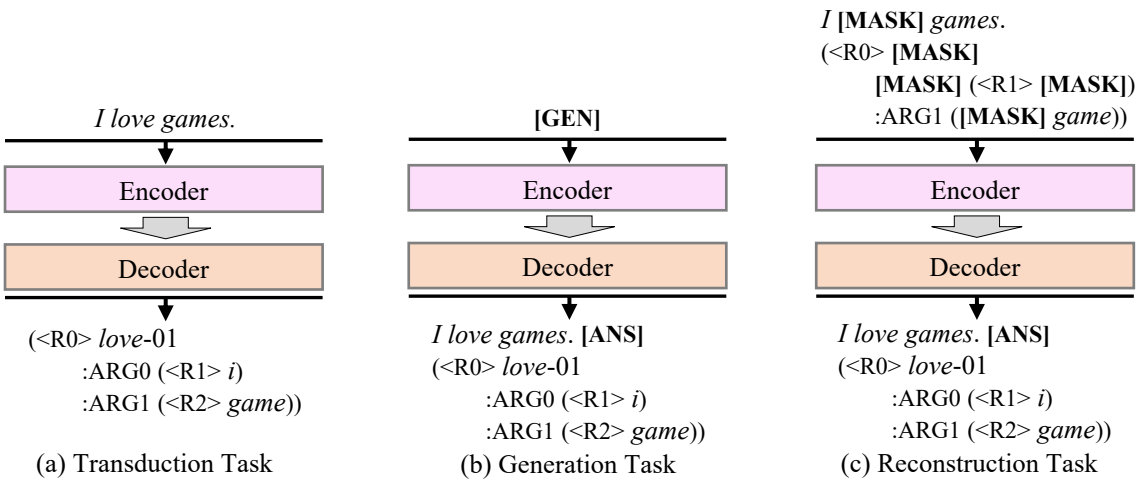


Figure 2: Visualization of the BiBL's multitask learning paradigm. All texts and graphs are linearized in implementations. The indentation in the figure is only for understanding and clarity purposes. Words in the text sentence are italicized, and special tokens are shown in bold.

vector as the output. The computation process of the decoder could be represented as follows:

$$h_t^d = \text{Decoder}(H^e, T_{<t}, \theta_d) \tag{3}$$

where $\theta_d$ denotes trainable parameters of the decoder module, $H^e$ denotes the output embeddings of the encoder, $T_{<t}$ denotes the tokens before $t$ position in the target sequence, and $h_t^d \in \mathcal{R}^d$ denotes the output hidden vector. During inference, $T_{<t}$ is replaced with tokens generated by the decoder module before $t$ time step $\hat{T}_{<t}$.

The final step of sequence generation is to generate a probability distribution for the next candidate token utilizing the output hidden vector. Combining the definitions of the encoder and decoder module, the formulation of the whole process is as follows:

$$\text{SG}_t(H^i, \theta_e, \theta_d) = \text{softmax}(h_t^d W^{LM}) \tag{4}$$

where $\text{SG}_t(H^i, \theta_e, \theta_d)$ denotes the probability distribution vector output of $t$ decoding step, $W^{LM} \in \mathcal{R}^{d \times \|V\|}$ denotes a trainable classification matrix that turns the output hidden vector of the decoder $h_t^d$ into a probability distribution of size $\|V\|$, the target vocabulary size.

### 3.5 Transduction Loss

Following the design of graph linearization and the adoption of the encoder-decoder sequence generation model, we could model the paired transduction tasks between graphs and text sequences as follows. Taking Text-to-AMR parsing as an example, the encoder input should be the text sequence $X = [x_1, ..., x_n]$ and subsequently the encoder outputs a contextualized representation of $X$. For the training step $t$, the decoder takes the linearized graph before $t$ position $Y_{<t}$ as the input and autoregressively generates the next graph token while taking the contextualized output of the encoder into account. Similarly, AMR-to-Text generation could be modeled by swapping the text sequence and the linearized graph. An example of the transduction task is shown in Figure 2(a).

Given model parameters $\theta$, two conditional probabilities mentioned in Equation 1 could be formulized as:

$$\begin{aligned} P(Y|X,\theta) &= \prod_{t=1}^{k} P(y_t|X, Y_{<t}, \theta) = \prod_{t=1}^{k} \text{SG}_t(H_X^i, \theta_e, \theta_d) \\ P(X|Y,\theta) &= \prod_{t=1}^{n} P(x_t|Y, X_{<t}, \theta) = \prod_{t=1}^{n} \text{SG}_t(H_Y^i, \theta_e, \theta_d) \end{aligned} \tag{5}$$

where $H_X^i \in \mathcal{R}^{n \times d}$ and $H_Y^i \in \mathcal{R}^{k \times d}$ denote the input embeddings of sequence $X$ and sequence $Y$, respectively.

In order to maximize these two conditional probabilities, transduction loss is defined as one of the training objectives:

$$\begin{aligned} l_{td}^{A2T} &= -\sum_{t=1}^{n} \log P(x_t|Y, X_{<t}, \theta) \\ l_{td}^{T2A} &= -\sum_{t=1}^{k} \log P(y_t|X, Y_{<t}, \theta) \end{aligned} \tag{6}$$

where $l_{td}^{A2T}$ and $l_{td}^{T2A}$ denote the sample-wise transduction loss defined for AMR-to-Text generation and Text-to-AMR parsing, respectively.

$P(y_t|X, Y_{<t}, \theta)$ and $P(x_t|Y, X_{<t}, \theta)$ denote the probability of generating $y_t$ and $x_t$, which are the ground truth labels in respective transduction tasks.

### 3.6 Bidirectional Bayesian Learning

Inspecting previous refinements and other general methods for bidirectional enhancement including dual learning (He et al., 2016), their improvements are mainly due to the inclusion of large pre-training data or complex structure modifications. However, our proposed method is data-efficient and can achieve great model improvements without extra pre-training data and model structure changes.

According to the Bayesian rules, we could find that two conditional probabilities in Equation 1 are interconnected through a common factor $P(X, Y)$:

$$\begin{aligned} P(Y|X) &= P(X, Y)/P(X) \\ P(X|Y) &= P(X, Y)/P(Y) \end{aligned} \tag{7}$$

Therefore, our motivation is that the incorporation of joint probability $P(X, Y)$ could facilitate the model's understanding of the joint probabilistic space of AMR graphs and texts. The model performance could be enhanced on both sides of the transduction task with proper design. The extra design should first clearly model the joint probability $P(X, Y)$ and then should take measures to avoid causing excessive confusion.

We propose two ways to model the joint probability $P(X, Y)$, and two auxiliary loss functions are included in our model design, i.e., generation loss and reconstruction loss. Utilizing the concatenation of text sentences and corresponding linearized graphs, two tailored auxiliary tasks help the model grasp the mutual information between two data forms, which is beneficial for the basic transduction task. Meanwhile, various effective measures are taken to avoid causing excessive confusion. Visualizations of the model input and output for these two auxiliary tasks are shown in Figure 2(b,c).

Inspired by LAMOL (Sun et al., 2020), we feed the model encoder with a specially designed `[GEN]` token. Denoting the `[GEN]` token as $s_g$, the encoder input is represented as $G = [s_g]$. Then the encoder generates an output signal which the decoder attends to, and the decoder would begin a special generation process. For training step $t$, denoting the concatenation of a text sequence and the corresponding linearized graph as $Z = [X; Y]$. To form the concatenation, we insert another special `[ANS]` token between $X$ and $Y$ sequences to separate them, creating discrimination from the original task. The decoder is expected to re-generate $Z$ in an auto-regressive fashion. We adopt teacher forcing for the training of the selected encoder-decoder. For generation step $t$, the decoder is fed with ground truth labels before the $t$ position. Consequently, it is possible for the model to re-generate paired data without much information provided to the encoder. The corresponding joint probability and the sample-wise generation loss are defined as follows:

$$\begin{aligned} P([X; Y]) &= \prod_{t=1}^{n+k} P(z_t|G, Z_{<t}, \theta) = \prod_{t=1}^{n+k} \mathrm{SG}_t(H_G^i, \theta_e, \theta_d) \\ l_g &= -\sum_{t=1}^{n+k} \log P(z_t|G, Z_{<t}, \theta) \end{aligned} \tag{8}$$

where $H_G^i \in \mathcal{R}^{1\times d}$ denotes the input embedding of $G$, and $l_g$ denotes the auxiliary generation loss. This auxiliary generation task could help the decoder grasp the underlying joint probability distribution of the graph-text data and facilitates the model to gain more knowledge from the training data. The `[GEN]` token input for the encoder works as a special signal for the decoder to maintain clear discrimination between the generation of the concatenated sequences in the generation task and the target sequences in the original transduction task. Discriminated by `[GEN]` token, it would not impede the learning of the encoder-decoder attention mechanism in the main transduction task. Therefore, the auxiliary generation task could facilitate the main transduction task and avoid causing excessive confusion.

Besides having a thorough understanding of the joint probability distribution of the training data pairs, BiBL needs to learn a universal joint distribution that is applicable to open-world data. Inspired by masked language modeling (Devlin et al., 2019), we adopt the following reconstruction task. First, we concatenate the text sentence and the linearized graph to get $Z = [X; Y]$. Similar to the generation auxiliary task, `[ANS]` token is inserted between $X$ and $Y$ sequences, which could prevent excessive confusion brought to the original transduction task. Then we randomly mask $50\%$ of the concatenation to get a masked concatenation $M = [x_1, ..., s_m, ..., x_n, s_a, y_1, ..., s_m, ..., y_k]$, where $s_m$ denotes the `[MASK]` token and $s_a$ denotes the `[ANS]` token. Taking this masked concatenation as input, the encoder generates a contextualized masked representation. Attending to

the representation generated by the encoder, the decoder is expected to auto-regressively reconstruct the original concatenation $Z$. In this auxiliary task, the reconstruction loss would be computed over all tokens generated by the decoder. The corresponding joint probability and the sample-wise reconstruction loss are defined as follows:

$$P([X;Y]) = \prod_{t=1}^{n+k} P(z_t|M, Z_{<t}, \theta) = \prod_{t=1}^{n+k} \mathrm{SG}_t(H_M^i, \theta_e, \theta_d)$$
$$l_r = -\sum_{t=1}^{n+k} \log P(z_t|M, Z_{<t}, \theta) \tag{9}$$

where $H_M^i \in \mathcal{R}^{(n+k)\times d}$ denotes the input embedding of the masked input $M$ and $l_r$ denotes the auxiliary reconstruction loss. This auxiliary task reinforces the decoder to understand the condensed representation produced by the encoder. Meanwhile, since masks could be applied simultaneously to texts and AMR graphs, the reconstruction of masked tokens helps the model adapt to the special structures of the graph linearization results while facilitating possible information alignment between two forms of data. This knowledge could be extended to general situations and is applicable to open-world data.

### 3.7 Single-stage Multitask Learning

A multitask learning paradigm is used in our training process to combine three loss functions mentioned above, i.e., transduction loss $l_{td}^{A2T}$ or $l_{td}^{T2A}$, generation loss $l_g$ and reconstruction loss $l_r$. The formulae are as follows:

$$l^{A2T} = l_{td}^{A2T} + \lambda_g l_g + \lambda_r l_r$$
$$l^{T2A} = l_{td}^{T2A} + \lambda_g l_g + \lambda_r l_r \tag{10}$$

where $l^{A2T}$ and $l^{T2A}$ denote the loss functions of AMR-to-Text generation and Text-to-AMR parsing, respectively, $\lambda_g$ is the weight of the generation loss, and $\lambda_r$ is the weight of the reconstruction loss. The loss is computed for every sample while averaged over a single batch. Moreover, the input sequences in the transduction task are placed in front of the target sequences for concatenation.

Since the joint probability distribution learned in two auxiliary tasks is conducive to the understanding of the conditional probability that corresponds to the original transduction task, it should be better to interweave the auxiliary tasks with the transduction task. Meanwhile, three measures are taken to avoid causing disturbances over the main transduction task. First, for the generation task, the special `[GEN]` token input for the encoder creates discrimination between the generation task and the original task. Second, for both auxiliary tasks, we utilize another special `[ANS]` token to separate two sequences, creating further discrimination from the original task. Third, by choosing the best loss weight configuration, the relationship among the three tasks could be adjusted and unnecessary confusion could be minimized. Therefore, different from all previous refinements that adopt a two-stage training paradigm of pre-training and fine-tuning, a single-stage multitask learning paradigm is explored. Results listed in the ablation studies 6 could show that this learning paradigm can greatly simplify the training process while retaining the extra performance boost by incorporating the extra knowledge.

## 4 Experimental Setup

Based on the evaluation system of previous works (Bevilacqua et al., 2021; Chen et al., 2022; Hoang et al., 2021), we inspect the performance of BiBL in two settings: In-Distribution and Out-of-Distribution. The in-distribution setting shows the power of BiBL on standard benchmarks. Since the bidirectional transitions between AMR graphs and texts should be modeled as a unified process without dependence on language contexts, it is also important to evaluate our model in out-of-distribution settings. The experiment benchmark system uses Nvidia Titan RTX as GPU and Intel Xeon E5-2637@3.50Ghz as CPU. The system is built with 256GB RAM.

### 4.1 Datasets

**In-Distribution** The data we use in the in-distribution setting are **AMR 2.0** (LDC2017T10) and **AMR 3.0** (LDC2020T02) corpora releases. AMR 3.0 is an expanded version of AMR 2.0 with additional AMR-text data to cover more general situations. AMR 3.0 contains $55,635$ pairs of graphs and texts, and AMR 2.0 contains $36,521$ pairs.

**Out-of-Distribution** Following the benchmark proposed in SPRING, we incorporate three evaluation datasets in the out-of-distribution setting: i) **TLP**, an AMR-tagged dataset containing $1,562$ data pairs constructed from the famous children's novel *The Little Prince*.; ii) **Bio**, full data from the Bio-AMR corpus (May and Priyadarshi, 2017) containing $6,952$ data pairs. While in SPRING, only the test data are used in the evaluation, we

use all data splits to conduct a thorough evaluation; iii) **New3**, a set of data excerpted from AMR 3.0 containing 527 data pairs, whose original source is the LORELEI DARPA project.

**Silver Data** Silver data is generated according to the following process. First, we obtain the best Text-to-AMR parsing BiBL model trained on the target dataset (AMR 2.0 or AMR 3.0). Then, we choose **English Gigaword** (LDC2003T05) as an unlabeled dataset. We use the previously trained Text-to-AMR BiBL model to generate annotations over randomly chosen sentences from the unlabeled dataset. Finally, a newly annotated dataset for extra training is acquired. For BiBL, the silver data we generate contains 200k sentence-graph pairs.

### 4.2 Models

**Model Setting** BART with augmented vocabulary is chosen as the pre-trained encoder-decoder model for BiBL. We inherit the model hyperparameters from BART Large, defined in Huggingface's `transformers` library. For training-related hyperparameters, our models are trained for 30 epochs using cross-entropy with a batch size of 500 graph linearization tokens, RAdam (Liu et al., 2020) optimizer, and a learning rate of $1 \times 10^{-5}$. The gradient is accumulated for 10 batches. The dropout is set to 0.25.

**BiBL variants** During experiments, we have done an empirical study on the best weight configuration for generation and reconstruction loss. Details are shown in Appendix A. For Text-to-AMR parsing, we set $\lambda_g$ as 1.0 and $\lambda_r$ as 0.5. For AMR-to-Text generation, we set $\lambda_g$ as 0.15 and $\lambda_r$ as 0.5. Models with $\lambda_g$ or $\lambda_r$ as 0 are included for ablation studies. BiBL variants trained with extra silver data are also included in the experiments. The silver data will be directly included in the single-stage multitask learning without further fine-tuning.

### 4.3 Evaluation Metrics

**Text-to-AMR Parsing** Fine-grained Smatch (Cai and Knight, 2013) is chosen as the evaluation metric for Text-to-AMR parsing since previous works all use this metric, and it reports fine-grained scores on different aspects of the parsed AMR graphs.

**AMR-to-Text Generation** Following previous works, we evaluate BiBL with three common Natural Language Generation measures, i.e., BLEU (Papineni et al., 2002), chrF++ (Popović, 2017), and METEOR (Banerjee and Lavie, 2005).

### 4.4 Comparison System

**Text-to-AMR Parsing** SPRING (Bevilacqua et al., 2021), Ancestor (Yu and Gildea, 2022), Graphene4S (Hoang et al., 2021), AMRize (Chen et al., 2022) and GraphPre (Bai et al., 2022) mentioned in Section 1 are included.

**AMR-to-Text Generation** SPRING (Bevilacqua et al., 2021), ReconCov (Anonymous, 2021), STA (Ribeiro et al., 2021) and GraphPre (Bai et al., 2022) mentioned in Section 1 are included.

## 5 Results

**Text-to-AMR parsing** According to Tables 1 and 2, under the setting of no extra training data and no complex ensembling techniques applied, BiBL outperforms all other refinements on AMR 3.0. Moreover, for AMR 2.0 and AMR 3.0, BiBL outperforms all previous refinements on the fine-grained Named Entity Recognition (NER) metric by a large margin. Especially for AMR 3.0, the performance metric of NER is increased by 4.3 even compared with AMRize. Besides, Wikification and topology-related Unlabeled metrics are also improved. This shows that the extra reconstruction task helps the model grasp the alignment between texts and AMR graphs. These alignments include special entity mappings and structural mappings, which are beneficial to NER and topology-related unlabeled metrics.

**AMR-to-Text generation** According to Tables 3 and 4, outperforming all previous methods by a significant margin (1.7 BLEU for AMR 2.0 and 1.5 BLEU for AMR 3.0), BiBL with silver data achieves a new state-of-the-art performance on AMR-to-Text generation. Meanwhile, BiBL with no silver data also surpasses all previous refinements under the same setting, proving that BiBL is the most effective refinement without the need for extra training data and the pre-training setting.

**Out-of-Distribution** Table 6 shows that BiBL outperforms several models (Graphene and SPRING) under the out-of-distribution setting without extra training data or graph ensembling techniques. This confirms that BiBL facilitates the model to grasp the joint probability distribution of AMR graphs and texts which is applicable to open-world data.

**Bidirectional Performance Enhancement** Inspecting BiBL on the more comprehensive AMR 3.0 dataset, under the setting of no extra data and no ensembling techniques, BiBL achieves the best results on Text-to-AMR parsing. For AMR-to-Text

| Model | Extra Data | Ensemble | Smatch | NoWSD | Wiki. | Conc. | NER | Neg. | Unlab. | Reent. | SRL |
|---|---|---|---|---|---|---|---|---|---|---|---|
| SPRING (2021) | N | N | 83.8 | 84.4 | **84.3** | 90.2 | 90.6 | **74.4** | 86.1 | 70.8 | 79.6 |
| *Ancestor (2022) | N | N | **84.8** | **85.3** | 84.1 | 90.5 | 91.8 | 74.0 | **88.1** | **75.1** | **83.4** |
| **BiBL** (ours) | N | N | 84.6 | 85.1 | 83.6 | 90.3 | **92.5** | 73.9 | 87.8 | 74.4 | 83.1 |
| SPRING (2021) | 200k | N | 84.3 | 84.8 | 83.1 | 90.8 | 90.5 | 73.6 | 86.7 | 72.4 | 80.5 |
| Graphene4S (2021) | 200k | Y | 84.8 | 85.3 | **83.9** | 90.6 | 92.2 | **75.2** | 88.0 | 71.4 | 83.5 |
| *AMRize (2022) | 40k | Y | 85.3 | 85.7 | **83.9** | 90.7 | 92.2 | 75.0 | **88.4** | **75.0** | **83.6** |
| *GraphPre (2022) | 200k | N | **85.4** | **85.8** | 81.4 | **91.2** | 91.5 | 74.0 | 88.3 | 73.5 | 81.5 |
| **BiBL** (ours) | 200k | N | 84.7 | 85.1 | 83.2 | 90.4 | **92.6** | 75.0 | 87.8 | 74.6 | 83.3 |

Table 1: Text-to-AMR parsing results on AMR 2.0 dataset. Row blocks: baseline + approaches without any extra training data and ensembling techniques; approaches with extra training data or ensembling techniques. Columns: Model; Smatch; Fine-grained scores. The best results within each block are shown in bold. Models with * denote contemporary works.

| Model | Extra Data | Ensemble | Smatch | NoWSD | Wiki. | Conc. | NER | Neg. | Unlab. | Reent. | SRL |
|---|---|---|---|---|---|---|---|---|---|---|---|
| SPRING (2021) | N | N | 83.0 | 83.5 | 82.7 | **89.8** | 87.2 | **73.0** | 85.4 | 70.4 | 78.9 |
| *Ancestor (2022) | N | N | 83.5 | 84.0 | 81.5 | 89.5 | 88.9 | 72.6 | 86.6 | **74.2** | **82.2** |
| **BiBL** (ours) | N | N | **83.9** | **84.3** | **83.7** | **89.8** | **93.2** | 68.1 | **87.2** | 73.8 | 81.9 |
| SPRING (2021) | 200k | N | 83.0 | 83.5 | 81.2 | 89.5 | 87.1 | 71.7 | 85.4 | 71.3 | 79.1 |
| Graphene4S (2021) | 200k | Y | 83.8 | 84.2 | 81.9 | 90.1 | 88.3 | **74.6** | 86.9 | 70.2 | 82.5 |
| *AMRize (2022) | 40k | Y | 84.0 | 84.5 | 80.7 | 90.0 | **88.9** | 73.1 | **87.1** | **73.9** | **82.6** |
| *GraphPre (2022) | 200k | N | **84.2** | **85.8** | 81.4 | **90.2** | 88.5 | 72.1 | **87.1** | 72.4 | 80.3 |
| **BiBL** (ours) | 200k | N | 83.5 | 84.0 | **82.1** | 89.6 | **88.9** | 73.3 | 86.7 | 73.7 | 82.2 |

Table 2: Text-to-AMR parsing results on AMR 3.0 dataset. Row blocks/Columns/Bold/* as in Table 1.

| Model | Extra Data | BL | CH++ | MET |
|---|---|---|---|---|
| SPRING (2021) | N | 45.3 | 73.5 | 41.0 |
| *ReconCov (2021) | N | 45.4 | 73.6 | 42.4 |
| **BiBL** (ours) | N | **47.0** | **74.8** | **43.2** |
| SPRING (2021) | 200k | 45.9 | 74.2 | 41.8 |
| STA (2021) | 2000k | 49.7 | - | **45.4** |
| *GraphPre (2022) | 200k | 49.8 | 76.2 | 42.6 |
| **BiBL** (ours) | 200k | **51.5** | **77.6** | 45.2 |

Table 3: AMR-to-Text generation results on AMR 2.0 dataset. Row blocks: approaches without extra training data; approaches with extra training data. Columns: Model; BLEU; chrF++; METEOR. The best results within each block are shown in bold. Models with * denote contemporary works.

| Model | Extra Data | BL | CH++ | MET |
|---|---|---|---|---|
| SPRING (2021) | N | 44.9 | 72.9 | 40.6 |
| *ReconCov (2021) | N | 45.7 | 73.7 | 42.8 |
| **BiBL** (ours) | N | **47.4** | **74.5** | **43.4** |
| SPRING (2021) | 200k | 46.5 | 73.9 | 41.7 |
| *GraphPre (2022) | 200k | 49.2 | 76.1 | 42.3 |
| **BiBL** (ours) | 200k | **50.7** | **76.7** | **45.0** |

Table 4: AMR-to-Text generation results on AMR 3.0 dataset. Row blocks/Columns/Bold/* as in Table 3.

| Model | SMATCH | BLEU |
|---|---|---|
| **BiBL** (silver) | | |
| - two-stage | 84.70 | 51.49 |
| - single-stage | 84.65 | 51.45 |

Table 5: Experiments on training paradigms. AMR 2.0 is chosen as the dataset. The recorded results are the best results achieved among all BiBL variants.

generation, BiBL with single-stage multitask learning surpasses the current state-of-the-art GraphPre model. Since the two-stage training paradigm of pre-training and fine-tuning is adopted by GraphPre, the performance boost proves that the single-stage multitask learning could not only simplify the training process but also contribute to the final model performance. For both transduction directions, BiBL with no silver data even surpasses several refinements that need extra training data (SPRING, Graphene4S) in both in-distribution and out-of-distribution settings. These results show that our proposed method is data-efficient and effective on bidirectional transitions. A case study is included in Appendix B for further examination.

We have also conducted experiments on model efficiency. According to Table 8, the time spent pre-training and fine-tuning GraphPre, which is the current state-of-the-art refinement, is 8.6x of the time spent training BiBL through single-phase mul-

| Model | Extra Data | Ensemble | TLP | Bio | New3 |
|---|---|---|---|---|---|
| *Text-to-AMR* | | | | | |
| SPRING (2021) | N | N | 77.3 | 59.7 | 73.7 |
| Graphene4S (2021) | 200k | Y | 77.9 | **61.5** | 74.8 |
| *AMRize (2022) | 40k | N | **78.9** | 61.2 | 75.4 |
| *GraphPre (2022) | 200k | N | 76.9 | **63.2** | **76.9** |
| **BiBL** (ours) | N | N | 78.6 | 61.0 | 75.4 |
| **BiBL** (ours) | 200k | N | 78.3 | 61.1 | 75.4 |
| *AMR-to-Text* | | | | | |
| SPRING (2021) | N | N | 24.3 | 18.9 | 37.2 |
| *GraphPre (2022) | 200k | N | **29.1** | 20.7 | 44.8 |
| **BiBL** (ours) | N | N | 26.1 | 20.3 | 39.0 |
| **BiBL** (ours) | 200k | N | 28.2 | **24.4** | **45.0** |

Table 6: OOD evaluation on Text-to-AMR (Smatch) and AMR-to-Text (BLEU). All models are trained on the AMR 2.0 dataset. Bold/* as in Table 1.

| Model | Extra Data | SMATCH | NER | BLEU |
|---|---|---|---|---|
| **BiBL** | | | | |
| - $\lambda_g = 0.15 + \lambda_r = 0.5$ | N | - | - | 47.4 |
| - $\lambda_g = 1.0 + \lambda_r = 0.5$ | N | **83.9** | **93.2** | - |
| - $\lambda_g = 0.15 + \lambda_r = 0.5$ | 200k | - | - | **50.1** |
| - $\lambda_g = 1.0 + \lambda_r = 0.5$ | 200k | 83.5 | 88.9 | - |
| - $\lambda_g = 0 + \lambda_r = 0.5$ | N | 83.7 | 89.1 | 47.3 |
| - $\lambda_g = 0.15 + \lambda_r = 0$ | N | 82.8 | 88.2 | 45.2 |
| - $\lambda_g = 1.0 + \lambda_r = 0$ | N | 83.0 | 88.3 | 45.5 |

Table 7: Ablation studies on Text-to-AMR (Smatch and fine-grained NER) and AMR-to-Text (BLEU) using AMR 3.0 as the benchmark. Bold denotes the best results.

titask training under the same setting. According to Table 9, since the ensembling of Graphene4S is computed with CPUs, the time spent ensembling four graphs generated by Graphene4S is 22.8x of the time spent by BiBL for the inference of one training sample when the beam size is set to 5.

## 6 Ablation Studies

Since AMR 3.0 is the superset of AMR 2.0, our ablation studies are based on the AMR 3.0 dataset so that a more comprehensive investigation could be conducted. Through a thorough analysis of Table 7, the following conclusions could be made:

• Inspecting all variants of BiBL for Text-to-AMR parsing on both AMR 2.0 and AMR 3.0, if the scope of the original dataset is limited, silver data could have minor contributions to BiBL on the Text-to-AMR parsing task. Otherwise, silver data may hurt the model performance. For the opposite AMR-to-Text generation task, the performance boost of silver data on BiBL is distinct.

• The combination of the auxiliary reconstruction and generation tasks achieves the best model performance, proving that both tasks are conducive and indispensable to the main transduction task.

| Model | Extra Data | Ensemble | Time | Multiplier |
|---|---|---|---|---|
| SPRING (2021) | N | N | 35.63 | 1.0x |
| *GraphPre[pre] (2022) | 200k | N | 267.23 | 7.5x |
| *GraphPre[fine] (2022) | N | N | 38.73 | 1.1x |
| *GraphPre[total] (2022) | 200k | N | 305.96 | 8.6x |
| **BiBL** (ours) | 200k | N | 128.38 | 3.6x |

Table 8: Training efficiency evaluation on different refinement approaches. Time is evaluated using the same computation resource and averaged over three batches. The unit of Time is a millisecond.

| Model | Ensemble | Time | Multiplier |
|---|---|---|---|
| *BARTlarge-based models* | - | 133.99 | 1.0x |
| SPRING (2021) | N | | |
| *GraphPre (2022) | N | | |
| **BiBL** (ours) | N | | |
| Graphene4S (2021) | Y | 3050.13 | 22.8x |

Table 9: Inference efficiency evaluation on different refinement approaches. Time is evaluated using the same computation resource and averaged over a single batch. The unit of Time is a millisecond.

BiBL enables the model to effectively capture the underlying information in the joint probabilistic space of AMR graphs and texts.

BiBL is trained with single-stage multitask learning. Further experiments are conducted to compare the chosen paradigm with the two-stage training paradigm. For two-stage training, we pre-train BiBL with auxiliary tasks and then fine-tune BiBL without them. Results listed in Table 5 show that the benefits brought by fine-tuning are rather trivial. This proves that through proper loss weight configuration, auxiliary tasks could help BiBL to achieve better performance on the main transduction task without causing confusion. Therefore, the training paradigm could be reduced to single-stage multitask learning without losing performance boost.

## 7 Conclusion

In this paper, we propose BiBL which is capable of enhancing the seq2seq approach to tackle the bidirectional transition task between AMR graphs and texts. BiBL could learn the joint probability of two data forms in a data-efficient way through single-stage multitask learning with auxiliary generation and reconstruction tasks. Eventually, model performances are greatly improved in in-distribution and out-of-distribution settings. Moreover, our model outperforms all previous refinements with no extra data on Text-to-AMR parsing and achieves the new state-of-the-art result on AMR-to-Text generation.

## References


Anonymous. 2021. Amr-to-text generation with graph structure reconstruction and coverage. In *Review of Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*.

Xuefeng Bai, Yulong Chen, and Yue Zhang. 2022. Graph pre-training for AMR parsing and generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Online. Association for Computational Linguistics.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria. Association for Computational Linguistics.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

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

Michele Bevilacqua, Rexhina Blloshmi, and Roberto Navigli. 2021. One SPRING to rule them both: Symmetric AMR semantic parsing and generation without a complex pipeline. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 12564–12573. AAAI Press.

Deng Cai and Wai Lam. 2020a. AMR parsing via graph-sequence iterative inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1290–1301, Online. Association for Computational Linguistics.

Deng Cai and Wai Lam. 2020b. Graph transformer for graph-to-sequence learning. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7464–7471. AAAI Press.

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 748–752, Sofia, Bulgaria. Association for Computational Linguistics.

Liang Chen, Peiyi Wang, Runxin Xu, Tianyu Liu, Zhifang Sui, and Baobao Chang. 2022. Atp: Amrize then parse! enhancing amr parsing with pseudoamrs.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *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)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Zhijiang Guo, 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*, 7:297–312.

Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tie-Yan Liu, and Wei-Ying Ma. 2016. Dual learning for machine translation. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

Thanh Lam Hoang, Gabriele Picco, Yufang Hou, Young-Suk Lee, Lam Nguyen, Dzung Phan, Vanessa López, and Ramon Fernandez Astudillo. 2021. Ensembling graph predictions for amr parsing. *Advances in Neural Information Processing Systems*, 34.

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

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. 2020. On the variance of the adaptive learning rate and beyond. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Manuel Mager, Ramón Fernandez Astudillo, Tahira Naseem, Md Arafat Sultan, Young-Suk Lee, Radu Florian, and Salim Roukos. 2020. GPT-too: A language-model-first approach for AMR-to-text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1846–1852, Online. Association for Computational Linguistics.

Jonathan May and Jay Priyadarshi. 2017. SemEval-2017 task 9: Abstract Meaning Representation parsing and generation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 536–545, Vancouver, Canada. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Maja Popović. 2017. chrF++: words helping character n-grams. In *Proceedings of the Second Conference on Machine Translation*, pages 612–618, Copenhagen, Denmark. Association for Computational Linguistics.

Leonardo F. R. Ribeiro, Martin Schmitt, Hinrich Schütze, and Iryna Gurevych. 2021. Investigating pretrained language models for graph-to-text generation. In *Proceedings of the 3rd Workshop on Natural Language Processing for Conversational AI*, pages 211–227, Online. Association for Computational Linguistics.

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

Fan-Keng Sun, Cheng-Hao Ho, and Hung-Yi Lee. 2020. LAMOL: language modeling for lifelong language learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Rik van Noord and Johan Bos. 2017. Neural semantic parsing by character-based translation: Experiments with abstract meaning representations. *CoRR*, abs/1705.09980.

Tianming Wang, Xiaojun Wan, and Hanqi Jin. 2020a. AMR-to-text generation with graph transformer. *Transactions of the Association for Computational Linguistics*, 8:19–33.

Tianming Wang, Xiaojun Wan, and Shaowei Yao. 2020b. Better amr-to-text generation with graph structure reconstruction. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 3919–3925. International Joint Conferences on Artificial Intelligence Organization. Main track.

Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. 2020. Scalable zero-shot entity linking with dense entity retrieval. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6397–6407, Online. Association for Computational Linguistics.

Chen Yu and Daniel Gildea. 2022. Sequence-to-sequence AMR parsing with ancestor information. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL-22)*.

Sheng Zhang, Xutai Ma, Kevin Duh, and Benjamin Van Durme. 2019a. AMR parsing as sequence-to-graph transduction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 80–94, Florence, Italy. Association for Computational Linguistics.

Sheng Zhang, Xutai Ma, Kevin Duh, and Benjamin Van Durme. 2019b. Broad-coverage semantic parsing as transduction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3786–3798, Hong Kong, China. Association for Computational Linguistics.

Jiawei Zhou, Tahira Naseem, Ramón Fernandez Astudillo, Young-Suk Lee, Radu Florian, and Salim Roukos. 2021. Structure-aware fine-tuning of sequence-to-sequence transformers for transition-based AMR parsing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6279–6290, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Qiji Zhou, Yue Zhang, Donghong Ji, and Hao Tang. 2020. AMR parsing with latent structural information. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4306–4319, Online. Association for Computational Linguistics.

Jie Zhu, Junhui Li, Muhua Zhu, Longhua Qian, Min Zhang, and Guodong Zhou. 2019. Modeling graph structure in transformer for better AMR-to-text generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5459–5468, Hong Kong, China. Association for Computational Linguistics.

| | SMATCH | |
|---|---|---|
| | AMR 2.0 | AMR 3.0 |
| **BiBL** | | |
| - $\lambda_g = 0.00$ | <u>84.06</u> | 83.81 |
| - $\lambda_g = 0.15$ | 83.98 | 83.92 |
| - $\lambda_g = 0.25$ | 84.01 | **84.06** |
| - $\lambda_g = 0.50$ | 83.92 | 83.83 |
| - $\lambda_g = 1.00$ | **84.17** | <u>83.98</u> |

Table 10: Text-to-AMR parsing experiments on weight configurations of the generation auxiliary task. Bold denotes the best results. Underline denotes second-best results. Results are reported on the validation set.

| | BLEU | |
|---|---|---|
| | AMR 2.0 | AMR 3.0 |
| **BiBL** | | |
| - $\lambda_g = 0.00$ | 40.21 | 41.50 |
| - $\lambda_g = 0.15$ | **40.34** | **41.72** |
| - $\lambda_g = 0.25$ | 40.19 | 41.20 |
| - $\lambda_g = 0.50$ | 39.89 | 41.24 |
| - $\lambda_g = 1.00$ | <u>40.23</u> | <u>41.52</u> |

Table 11: AMR-to-Text generation experiments on weight configurations of the generation auxiliary task. Bold denotes the best results. Underline denotes second-best results. Results are reported on the validation set.

## A Empirical Study on Weight Configuration of Auxiliary Tasks

For the weight configuration of the auxiliary tasks proposed for Bidirectional Bayesian Learning (BiBL), we have experimented with several BiBL variants of different $\lambda_g$ and $\lambda_r$. All experimental models are trained without extra silver data in order to discriminate the direct effect of different parameters.

During experiments, we find that the magnitudes of transduction, generation, and reconstruction loss should be on the same level. Otherwise, the generation validity of BiBL will be greatly influenced by auxiliary tasks constructed using the concatenation of AMR graphs and texts. Therefore, reasonable ranges of $\lambda_g$ and $\lambda_r$ could be roughly determined. Moreover, we find that the weight sensitivity of the generation task is relatively high compared with the auxiliary reconstruction task. Therefore, it is necessary to conduct a grid search over reasonable ranges of $\lambda_g$. Since the weight sensitivity of the reconstruction task is relatively low according to our conclusion, $0.5$ is chosen for $\lambda_r$.

According to the results presented in Tables 10, 11, 12, and 13, for Text-to-AMR parsing, it is obvious that BiBL variant with $\lambda_g = 1.00$ is the best choice. However, for AMR-to-Text generation, the effect of $\lambda_g$ varies across datasets. Focusing on achieving the best performance on in-distribution datasets, BiBL variants with $\lambda_g = 0.15$ and $\lambda_g = 1.00$ are the best models. However, according to Table 14, after inspecting the combination effect of generation and reconstruction tasks on AMR-to-Text generation, BiBL variant with $\lambda_g = 0.15$ and $\lambda_r = 0.5$ greatly outperforms the model with $\lambda_g = 1.00$ and $\lambda_r = 0.5$.

| | SMATCH | | |
|---|---|---|---|
| | AMR 2.0 | AMR 3.0 | TLP |
| **BiBL** | | | |
| - $\lambda_g = 0.00$ | 83.80 | <u>83.00</u> | 77.30 |
| - $\lambda_g = 0.15$ | **84.28** | 82.78 | <u>77.91</u> |
| - $\lambda_g = 0.25$ | 83.97 | 82.93 | 77.88 |
| - $\lambda_g = 0.50$ | 83.86 | **83.30** | 77.62 |
| - $\lambda_g = 1.00$ | <u>84.23</u> | <u>83.00</u> | **77.94** |

Table 12: Text-to-AMR parsing experiments on weight configurations of the generation auxiliary task. Bold denotes the best results. Underline denotes second-best results. Results are reported on the test set.

Therefore, according to the above analysis, respectively for Text-to-AMR parsing and AMR-to-Text generation, BiBL variants with $\lambda_g = 1.0$ and $\lambda_r = 0.5$ and $\lambda_g = 0.15$ and $\lambda_r = 0.5$ are chosen due to their excellent performances.

## B Case Study

In section 5, we analyzed the performance of BiBL according to chosen metrics. Here we present four cases to further inspect the improvements of BiBL. First, we inspect two representative cases for Text-to-AMR parsing. Figure 3 shows a case where BiBL correctly labels the named entity *OCD* as a disease. However, in the output AMR graph of the SPRING baseline, the named entity is falsely labeled as a medical condition, which is too general. This proves that BiBL could better recognize and understand the named entities included in the text sentences, which leads to a large performance boost of BiBL on the fine-grained NER metric. Figure 4 shows a case where BiBL correctly recognizes the reentrancy that occurred in the text. Although the short sentence provides limited semantic informa-

| | BLEU | | |
|---|---|---|---|
| | AMR 2.0 | AMR 3.0 | TLP |
| **BiBL** | | | |
| - $\lambda_g = 0.00$ | **44.77** | 44.90 | 24.37 |
| - $\lambda_g = 0.15$ | <u>44.65</u> | 45.18 | 23.83 |
| - $\lambda_g = 0.25$ | 43.61 | **45.50** | 24.17 |
| - $\lambda_g = 0.50$ | 44.30 | 45.35 | **25.11** |
| - $\lambda_g = 1.00$ | 44.62 | <u>45.47</u> | <u>24.78</u> |

Table 13: AMR-to-Text generation experiments on weight configurations of the generation auxiliary task. Bold denotes the best results. Underline denotes second-best results. Results are reported on the test set.

| Model | BLEU |
|---|---|
| **BiBL** | |
| - $\lambda_g = 0.15 + \lambda_r = 0.5$ | **46.95** |
| - $\lambda_g = 1.00 + \lambda_r = 0.5$ | 46.65 |

Table 14: Experiments on the combination effect of generation and reconstruction tasks. AMR 2.0 is chosen as the dataset.

tion and the speaker information is indicated by a colon separation, BiBL still fully understands the meaning of the sentence and generated the correct reentrancy relations in the AMR graph result. However, in SPRING-generated AMR graphs, the reentrancy information is missing. This proves that BiBL could still understand the structures of text sentences even with limited context provided.

Then, we inspect two representative cases for AMR-to-Text generation. Figure 5 shows a case where BiBL could understand the complex semantic meaning indicated by the AMR graph. While the SPRING baseline misrepresents the semantic relation between *sight* and *hope*, BiBL could restore the correct meaning in the generated text by using the same grammar structure in the gold label. This proves that BiBL could accurately grasp the underlying semantic meaning of texts and correctly represent the information using proper grammar structures. Figure 6 shows a case where a complex AMR graph is involved. The relations between the nouns in the AMR graphs are complicated, and there are obvious overlaps between several nodes. We could see that the text generated by SPRING fails to reconstruct the word "Japan delegation", misrepresents the relation between *head* and *Tokyo Handicapped Integrated Sports Center*, and misses the concept of *director*. However, BiBL could correctly generate all the concepts involved in the complex graph and maintain the correct relations. This proves that BiBL could deal with situations where graph information is rather complicated.

**TEXT**

*To me that just sounds like you being overly anxious, as opposed to having really bad OCD.*

**GOLD AMR**

```
(s / sound-01
    [OMIT]
    :ARG2 (a / anxious
        [OMIT]
        :ARG1-of (i2 / instead-of-91
            :ARG2 (h / have-03
                :ARG0 y
                :ARG1 (d / disease
                        :name (n / name :op1 "OCD")
                        [OMIT])))))
    [OMIT])
```

**SPRING AMR**

```
(z0 / sound-01
    [OMIT]
    :ARG2 (z2 / anxious
        [OMIT]
        :ARG1-of (z5 / instead-of-91
            :ARG2 (z6 / have-03
                :ARG0 z3
                :ARG1 (z7 / medical-condition
                        :name (z8 / name :op1 "OCD")
                        [OMIT])))))
    [OMIT])
```

**BiBL AMR**

```
(z0 / sound-01
    [OMIT]
    :ARG2 (z2 / anxious
        [OMIT]
        :ARG1-of (z5 / instead-of-91
            :ARG2 (z6 / have-03
                :ARG0 z3
                :ARG1 (z7 / disease
                        :name (z8 / name :op1 "OCD")
                        [OMIT])))))
    [OMIT])
```

Figure 3: Text-to-AMR parsing case study on NER. The input text sentence is italicized. Wikification labels are omitted on gold, SPRING, and BiBL generated AMR graphs. The [OMIT] sign represents the omission of other graph nodes that are the same across gold, SPRING, and BiBL generated AMR graphs. The underlined parts highlight the label generated for the "OCD" named entity.

**TEXT**

*TMT: I don't understand your point.*

**GOLD AMR**

```
(s / say-01
    :ARG0 (i / i)
    :ARG1 (u / understand-01 :polarity -
            :ARG0 i
            :ARG1 (p2 / point-04
                    :ARG0 p))
    :ARG2 (p / person :wiki -
            :name (t / name
                    :op1 "TMT")))
```

**SPRING AMR**

```
(z0 / say-01
    :ARG1 (z1 / understand-01 :polarity -
            :ARG0 (z2 / i)
            :ARG1 (z3 / point-04
                    :ARG0 (z4 / you)))
    :ARG2 (z5 / person :wiki -
            :name (z6 / name
                    :op1 "TMT")))
```

**BiBL AMR**

```
(z0 / say-01
    :ARG1 (z1 / understand-01 :polarity -
            :ARG0 (z2 / i)
            :ARG1 (z3 / point-04
                    :ARG0 (z4 / person :wiki -
                            :name (z5 / name
                                    :op1 "TMT"))))
    :ARG2 z4)
```

Figure 4: Text-to-AMR parsing case study on Reentrancies. The input text sentence is italicized. The underline parts highlight the occurrence of a graph reentrancy.

**AMR**

```
(s / sight-01
    :ARG1 (h / hopeful-03
        :ARG0 (t / they)
        :ARG1 (e / enter-01
            :ARG0 t
            :ARG1 (h2 / heat)))
    :time (a / already)
    :condition (a2 / accident :polarity -))
```

**GOLD TEXT**

*If accidents do not occur, their hopes of entering the heats are already insight.*

**SPRING TEXT**

*If there is no accident, there is already a sight of them hoping to enter the heat.*

**BiBL TEXT**

*Without the accident, their hopes of entering the heat were already in sight.*

Figure 5: AMR-to-Text generation case study on semantic understanding. The text sentence output is italicized. The underline parts highlight the main differences between gold, SPRING, and BiBL generated sentences.

**AMR**

```
(h / hand-over-02
    :ARG0 (p3 / person :wiki -
        :name (n / name :op1 "Souya"))
    :ARG1 (f2 / flag
        :poss (d / delegation
            :source (c2 / country :wiki "Japan"
                :name (n2 / name :op1 "Japan")))
        :mod (d2 / delegation))
    :ARG2 (p / person :wiki -
        :name (n3 / name :op1 "Banminyan")
        :ARG0-of (h2 / head-01
            :ARG1 d)
        :ARG0-of (h3 / have-org-role-91
            :ARG1 (s2 / sports-facility :wiki -
                :name (n4 / name :op1 "Tokyo"
                    :op2 "Handicapped"
                    :op3 "Integrated"
                    :op4 "Sports" :op5 "Center"))
            :ARG2 (d3 / director))))
```

**GOLD TEXT**

*Souya handed over the delegation flag of the Japanese delegation to the head of the delegation and Director of the Tokyo Handicapped Integrated Sports Center, Banminyan .*

**SPRING TEXT**

*Souya handed over the delegation flag of Japan to the head of the Tokyo Handicapped Integrated Sports Center Banminyan .*

**BiBL TEXT**

*Souya handed over the delegation flag of the Japanese delegation to Banminyan, head of the delegation and director of the Tokyo Handicapped Integrated Sports Center .*

Figure 6: AMR-to-Text generation case study on complex graph structures. The text sentence output is italicized. The underline parts highlight the main differences between gold, SPRING, and BiBL generated sentences.