Incorporating Graph Information in Transformer-based AMR Parsing

Pavlo Vasylenko[1] **Pere-Lluís Huguet Cabot**[1,2*]
Abelardo Carlos Martínez Lorenzo[1,2*] **Roberto Navigli**[1]

[1] Sapienza NLP Group, Sapienza University of Rome
[2] Babelscape, Rome

vasylen.pavlo@gmail.com
{martinez, huguetcabot}@babelscape.com
navigli@diag.uniroma1.it

Abstract

Abstract Meaning Representation (AMR) is a Semantic Parsing formalism that aims at providing a semantic graph abstraction representing a given text. Current approaches are based on autoregressive language models such as BART or T5, fine-tuned through Teacher Forcing to obtain a linearized version of the AMR graph from a sentence. In this paper, we present LeakDistill, a model and method that explores a modification to the Transformer architecture, using structural adapters to explicitly incorporate graph information into the learned representations and improve AMR parsing performance. Our experiments show how, by employing word-to-node alignment to embed graph structural information into the encoder at training time, we can obtain state-of-the-art AMR parsing through self-knowledge distillation, even without the use of additional data. We release the code at http://www.github.com/sapienzanlp/LeakDistill.

1 Introduction

Creating a machine-interpretable representation of meaning lies at the core of Natural Language Understanding and has been framed as the Semantic Parsing task. Multiple formalisms have been proposed over the years, e.g., Prague Czech-English Dependency Treebank (Hajič et al., 2012), Universal Conceptual Cognitive Annotation (Abend and Rappoport, 2013), BabelNet Meaning Representation (Navigli et al., 2022; Martínez Lorenzo et al., 2022); however, Abstract Meaning Representation (Banarescu et al., 2013, AMR) has received more attention thanks to the large corpus available and a well-defined structure. AMR captures text semantics in the form of a directed acyclic graph (DAG), with nodes representing concepts and edges representing semantic relationships between them (see Figure 1). Currently, AMR is widely employed
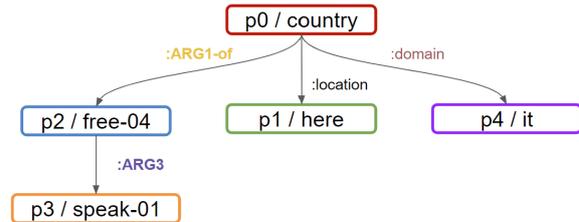
---
* Equal contributions.

Figure 1: Top: sentence. Middle: AMR graph. Bottom: Linearized graph. Alignment is represented by colours.

in a plethora of NLP domains, such as Information Extraction (Rao et al., 2017), Text Summarization (Hardy and Vlachos, 2018; Liao et al., 2018), Question Answering (Lim et al., 2020; Bonial et al., 2020b; Kapanipathi et al., 2021), Human-Robot Interaction (Bonial et al., 2020a), and Machine Translation (Song et al., 2019), among others.

Until a short while ago, autoregressive models proved to be the best approach for semantic parsing because of their outstanding performance without relying on sophisticated ad-hoc architectures (Bevilacqua et al., 2021). Then, more recently, several approaches have emerged to increase performance by including structural information in the model (Chen et al., 2022), adding extra Semantic Role Labeling tasks (Bai et al., 2022) or by ensembling strategies (Lam et al., 2021; Lee et al., 2022).

In this paper, following the effort of strengthening the model's learning phase by incorporating meaningful structural information, we investigate the use of structural adapters (Ribeiro et al., 2021a)

that are basically Graph Neural Networks (GNNs) embedded in the encoder of a Transformer Encoder-Decoder architecture. The structural information is derived from intrinsic concept-node alignments from which we build a word-based graph with a structure similar to the original AMR. Leveraging such a graph implies partial data leakage: the graph structure is revealed to a model during training. To overcome the lack of the leaked information at inference time, we explore Knowledge Distillation (KD), a technique that transfers knowledge from a teacher model to a student model (Hinton et al., 2015). The word-based graph is employed with the structural adapters to obtain soft targets (the teacher path), which are then used for self-distillation, transferring the knowledge to the student, which only has access to the text.

Our main contributions are: i) exploring how to add structural information to the AMR parsing model using structural adapters and self-knowledge distillation, ii) state-of-the-art results in AMR parsing for AMR 2.0 and AMR 3.0 datasets, and iii) competitive base models for AMR parsing.

2 Related Work

Over the years, multiple trends have appeared to parse AMR graphs: using statistical methods (Flanigan et al., 2014, 2016; Wang et al., 2015), neural-transition based parsers (Ballesteros and Al-Onaizan, 2017; Liu et al., 2018; Fernandez Astudillo et al., 2020; Zhou et al., 2021) or bidirectional Transformers (Lyu and Titov, 2018; Zhang et al., 2019; Cai and Lam, 2020) based on BERT (Devlin et al., 2019).

Recently, autoregressive models based on BART (Lewis et al., 2020) have emerged as a dominant approach for AMR parsing, since they obtained state-of-the-art performance without complex pipelines. One notable example is SPRING (Bevilacqua et al., 2021), which frames AMR parsing as a neural machine translation task, where text is translated into a linearized version of the graph. Subsequently, several works extended SPRING using a variety of different strategies. Procopio et al. (2021) leverages multitask learning to improve cross-lingual AMR parsing results. ATP (Chen et al., 2022) expands the dataset with extra auxiliary tasks such as Semantic Role Labeling and Dependency Parsing, with pseudo-AMR graphs constructed based on a particular task. AMRBART (Bai et al., 2022) uses a pre-training strat-

egy based on Masked Language Modeling where both text and graph need to be denoised, using 200k graphs generated by SPRING. However, despite their efforts to enhance SPRING's performance, all these systems rely on additional external data. Although Ancestor (Yu and Gildea, 2022), which modifies ancestor information during decoding, and BiBL (Cheng et al., 2022), that adds a secondary graph masking task while training, do not rely on extra data, their performance improvements remain relatively limited. Our proposed model effectively bridges the gap in performance between "with" and "without" extra data by integrating explicit structural information during the training phase.

3 Word-Aligned Graph

Our goal is to incorporate graph-structured information into the encoder of a Transformer-based parser. However, the model only has access to the input sentence at that stage, with no hidden representation of AMR-specific nodes and relations. Thus, we simplify the AMR structure to a word-based graph by exploiting a pre-existing alignment between spans in text and semantic units in the corresponding AMR graph (see Figure 1).

First, starting with the source AMR graph, we replace the labels of the AMR nodes and relations with the words of the corresponding sentence as provided by the alignment (Figure 2, left). Next, we convert each edge into a node and connect it to its original endpoints (see Figure 2, center). Moreover, following what Ribeiro et al. (2021b) did for AMR graphs, we split each multi-token node (e.g., `freedom` in Figure 2) into a parent node represented by the first token and children nodes connected to it which contain the remaining tokens. We name the resulting graph representation the Word-Aligned Graph (WAG).

We will leverage WAGs to enrich the encoder's hidden representations of words with the AMR graph's structural information. Unfortunately, a problem arises with non-aligned nodes (e.g., the `:location` relation in Figure 2), since they will not have associated hidden states. Therefore, we have two alternatives: i) remove nodes for which we do not have hidden states (*Contracted WAG*), or ii) create new hidden states for them (*Full WAG*).

Contracted WAG As a first option, we remove non-aligned nodes from the graph. However, deleting the nodes from the original graph would pro-
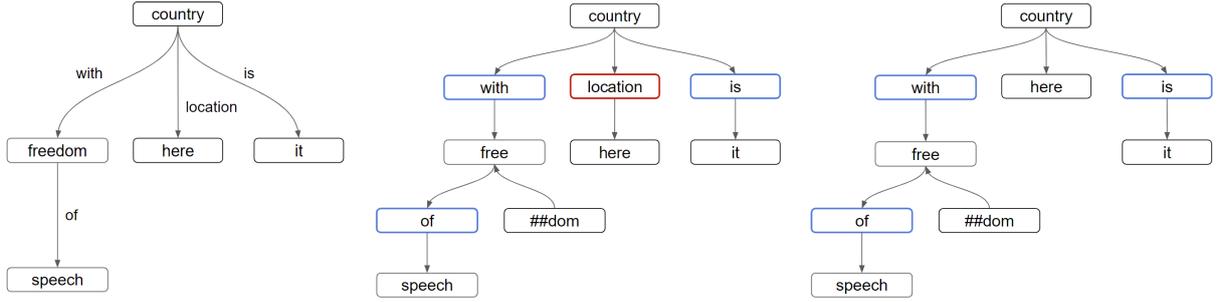
Figure 2: WAG construction of the sentence: "Here, it is a country with the freedom of speech". A graph where AMR concepts are replaced with words (left), a Full WAG (center) and a Contracted WAG (right). Blue lines indicate former AMR relations, and red lines indicate non-aligned nodes. Best seen in color.
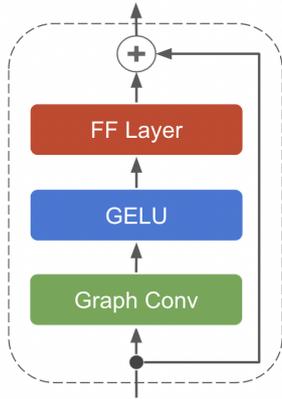

Figure 3: Structural adapter without layer normalization and with GELU activation.

duce a disconnected graph. To obtain a connected structure similar to the original graph, we contract nodes rather than removing them. A contracted WAG (*CWAG*) is a graph in which non-aligned nodes are merged with their closest parent node along with all their relations. Figure 2 (right) depicts a CWAG.

Full WAG Alternatively, we preserve the nodes without alignment (e.g., the node "location" in Figure 2 (center)). This type of graph is referred to as a Full WAG (FWAG), Figure 2 (center) shows an example of FWAG.

4 Structural Adapters for AMR parsing

In this section, we describe the main components of our structure-enhanced approach to AMR parsing.

4.1 Parsing with BART

AMR parsing can be defined as a sequence-to-sequence (seq2seq) problem where the input $x = (x_1, ..., x_n)$ is a sequence of n words (or subwords) and the output $g = (e_1, ..., e_m)$ is a linearized graph with m elements. Our goal is to learn a function that models the conditional probability:

$$p(g|x) = \prod_{t=1}^{m} p(e_t|e_{<t}, x), \tag{1}$$

where $e_{<t}$ are the tokens of the linearized graph g before step t.

Suppose we have a dataset D of size $|D|$ which consists of pairs (x^i, g^i), with each g^i having length m^i. Our objective is then to minimize a negative log-likelihood loss function:

$$L_{nll}^{D} = L_{nll}(D) = -\sum_{i=1}^{|D|} \log p(g^i|x^i) =$$
$$= -\sum_{i=1}^{|D|} \sum_{t=1}^{m^i} \log p(e_t^i|e_{<t}^i, x^i) \tag{2}$$

We use BART as our seq2seq model implementing the above formulation and, following Blloshmi et al. (2021, SPRING), add special tokens corresponding to i) AMR-related tokens, ii) variable names <R0>, <R1>, ... <Rn>, and iii) other tokens needed for the graph linearizations. Then, we fine-tune BART with the input x and the target g.

4.2 Structural adapters

To incorporate AMR structural information into the encoder, we embed the WAGs – obtained from AMR graphs as illustrated in Section 3 – into adapters that encode the graph structure imposed by them. Structural adapters, as introduced by Ribeiro et al. (2021b), are a modification of the Transformer architecture that improves pre-trained language models for modeling graph information. They consist of a Graph Convolutional (GraphConv) layer and a feed-forward layer, which are connected
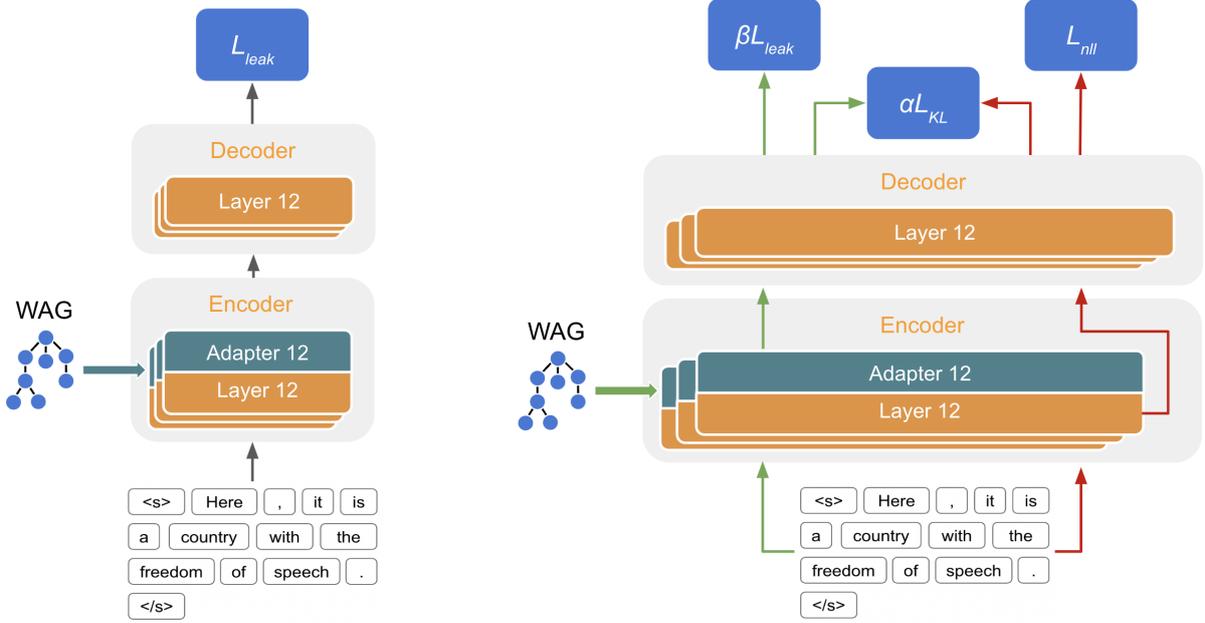
Figure 4: Left: Scheme of the Graph Leakage Model. Right: Scheme of the LeakDistill method with two forward paths: the green path incorporates WAG information via adapters; the red path omits adapters, and it is basically the outcome model for the problem. Consequently, the green path is engaged exclusively during the training phase to guide the red path, while during the inference process, only the red path is operative.

through a residual connection. Moreover, we remove layer normalization and set GELU as an activation function (see Figure 3).

Structural adapters are inserted after each encoder's layer (see Figure 4). For each hidden representation $\mathbf{h}_v^l \in \mathbb{R}^b$ from the encoder layer l and the set of edges $\mathcal{E}$ in the WAG, we define the Graph-Conv operation as:

$$\text{GraphConv}_l(\mathbf{h}_v^l, \mathcal{E}) = \sum_{u \in \mathcal{N}(v)} \frac{1}{\sqrt{d_u d_v}} \mathbf{W}_g^l \mathbf{h}_u^l \quad (3)$$

where $\mathcal{N}(v)$ is the set of node v's adjacent nodes in the WAG (including v itself), d_v is the degree of v, and $\mathbf{W}_g^l \in \mathbb{R}^{b \times b}$ is a parameter matrix. Then, the updated hidden states $\mathbf{z}_v^l$ are computed as:

$$\begin{aligned} \mathbf{g}_v^l &= \text{GraphConv}_l(\mathbf{h}_v^l, \mathcal{E}) \\ \mathbf{z}_v^l &= \mathbf{W}_a^l \sigma(\mathbf{g}_v^l) + \mathbf{h}_v^l, \end{aligned} \quad (4)$$

where σ is the GELU activation function and $\mathbf{W}_a^l \in \mathbb{R}^{b \times b}$ is the feed-forward layer parameter matrix.

5 Our Models

5.1 Graph Leakage Model

We bring together the two main components described in Section 4 by incorporating structural

adapters in each layer of the encoder of a BART-based AMR parsing model (see Figure 4 (left) and Algorithm 1). Here, a WAG, together with the hidden representations of tokens in the sentence, are input to the adapters. Since WAGs are constructed using gold AMR graphs, this constitutes a form of information leakage. We name this model the Graph Leakage Model (GLM), with the idea that it will serve as a study of the impact on performance when including WAGs (be they contracted or full, cf. Section 3).

To use FWAGs as input to the adapter, we need representations for non-aligned nodes that do not have an associated hidden state. Therefore, for nodes with labels corresponding to AMR special tokens (e.g., :location) we use their embedding. For other nodes, we tokenize the label and take the average embedding. Furthermore, these representations are concatenated after the hidden states in the first adapter layer. After each adapter block, we split representations into two groups: i) the updated hidden states for the original input tokens, which serve as inputs of the subsequent Transformer layer, ii) the updated hidden states for the non-aligned nodes, which are concatenated again in the next adapter block (see Algorithm 1).

Then, for both CWAG and FWAG, the input to each adapter layer l consists of a matrix of hidden

Algorithm 1 Modified BART Encoder

Input: $\mathcal{E}$ - set of WAG edges, S^0 - states for non-aligned nodes, H^0 - initial hidden states of the input sequence
for $l \in \{1, ..., 12\}$ **do**
    $H^l \leftarrow \text{BARTLayer}_l(\text{H}^{l-1})$
    if Leak Mode **then**
        if Full WAG **then**
            $G^l \leftarrow \text{Concat}(H^l, S^{l-1})$
        else
            $G^l \leftarrow H^l$
        end if
        $\widetilde{G}^l \leftarrow \text{StructAdapt}_l(G^l, \mathcal{E})$
        if Full WAG **then**
            $[\widetilde{H}^l; S^l] \leftarrow \text{Split}(\widetilde{G}^l)$
        else
            $\widetilde{H}^l \leftarrow \widetilde{G}^l$
        end if
    else
        $\widetilde{H}^l \leftarrow H^l$
    end if
    $H^l \leftarrow \widetilde{H}^l$
end for

states H^l and a set of edges $\mathcal{E}$. Note that the set of edges $\mathcal{E}$ does not change through layers. Finally, the loss function for GLM is:

$$L_{leak} = L_{nll}(\tilde{D}) = -\sum_{i=1}^{|\tilde{D}|} \log q(g^i|x^i, w^i), \quad (5)$$

where $\tilde{D}$ is the updated dataset consisting of pairs $((x^i, w^i), g^i)$, q is the probability for GLM, w^i is the WAG.

5.2 Knowledge Distillation

GLM leverages the alignment information to improve the model's understanding of the graph structure and enhance its (the model's) performance in AMR parsing. Unfortunately, as discussed in the previous section, this constitutes a form of leakage at inference time. Therefore, following the idea of Knowledge Distillation (Hinton et al., 2015, KD), we set the fine-tuned GLM as a teacher model, which receives both the sentence and WAG as inputs, and our plain BART parser as the student (see Section 4.1). Then, the knowledge acquired by the teacher model is transferred to the student model, which only has access to the sentence. This enables the utilization of WAGs during training while avoiding their use during inference. Hence, our

objective is to achieve the following:

$$p(g|x) = q(g|x, w) \quad (6)$$

where p and q are probabilities of the student and the teacher, respectively, and w is the WAG, used only at training time.

As is common in KD, we employ Kullback–Leibler divergence to match the student and the teacher probabilities:

$$L_{KL} = KL(p, q) = \sum_{k=0}^{C-1} p_k \log(\frac{p_k}{q_k}) \quad (7)$$

where C is the number of classes, i.e. our token vocabulary. Usually, the loss L_{nll}^D for the original task is added to the total loss, thus becoming:

$$L_{KD} = L_{nll}^D + \alpha L_{KL} =$$

$$= -\sum_{i=1}^{|D|} \sum_{t=1}^{m^k} \sum_{k=0}^{C-1} (\delta_t^i(k) \log p_{t,k}^i - \alpha\, p_{t,k}^i \log(\frac{p_{t,k}^i}{q_{t,k}^i})),$$

$$p_{t,k}^i = p(e_t^i{=}k \mid e_{<t}^i, x^i),$$

$$q_{t,k}^i = q(e_t^i{=}k \mid e_{<t}^i, x^i, w^i) \quad (8)$$

where $\delta_t^i(k)$ is 1 when k is a target class at step t and 0 otherwise; α is a hyperparameter.

There are only architectural differences between the teacher and the student model at the encoder, since the teacher additionally includes the structural adapters. Therefore, we copy the GLM decoder to the student model and freeze the decoder parameters.

5.3 LeakDistill

We anticipate that, in our experimentation, KD will have failed to properly transfer the structural information to the student model. Therefore, we propose a single model approach that can be trained by performing two forward passes at each training step, one with and one without the WAG structural information (see Figure 4 and Algorithm 2). We force the two passes to learn the same distribution by adding a Kullback–Leibler divergence loss to the output logits. As a result, the total loss becomes:

$$L_{LeakDistill} = L_{nll}^D + \beta L_{leak} + \alpha L_{KL} =$$

$$= -\sum_{i=1}^{|D|} \sum_{t=1}^{m^k} \sum_{k=0}^{C-1} (\delta_t^i(k) \log p_{t,k}^i + \beta \, \delta_t^i(k) \log q_{t,k}^i$$

$$-\alpha \, p_{t,k}^i \log(\frac{p_{t,k}^i}{q_{t,k}^i})),$$

$$(9)$$

where L_{leak} is the loss for the first pass (basically, GLM), with leaked information, L_{nll}^D is the loss for the second pass (basically, BART), which is the original negative log-likelihood loss, and finally L_{KL} is the above-described Kullback–Leibler divergence loss. α and β are hyperparameters to control each loss scale.

The above formulation implements what is called self-knowledge distillation (Hahn and Choi, 2019, SKD). Specifically, in our work we project the knowledge via leveraging data leakage in the first pass rather than computing soft target probabilities. Moreover, we calculate KL divergence for all classes to obtain more knowledge. Finally, based on the intuition that there is not enough information to distill at the beginning of training, we schedule a gradual decrease of L_{leak}'s multiplier β.

6 Experimental Setup

To demonstrate the benefits of incorporating structural information in AMR parsing, we devise a set of experiments to assess its performance in comparison to state-of-the-art models. Before delving into details, we provide information regarding the datasets (Section 6.1), the metrics (Section 6.2) and the model (Section 6.3) used in our experiments.

6.1 Datasets

We test on two AMR benchmark datasets: i) AMR 2.0, which has 36521, 1368, and 1371 sentence-AMR pairs in the training, validation, and test sets, respectively, and ii) AMR 3.0, which contains 55635, 1722, and 1898 sentence-AMR pairs in the training, validation, and test sets, respectively (see Appendix E). Furthermore, we test on The Little Prince (TLP) and the Bio AMR out-of-distribution datasets.

6.2 Metrics

We evaluate our models using the SMATCH metric (see Appendix D for more details). Additionally we

Model	AMR 3.0
SPRING (ours)	84.55
Contracted WAG	86.01
Full WAG	89.58

Table 1: GLM results for AMR 3.0 development set.

also perform evaluation with two additional metrics: S^2MATCH (Opitz et al., 2020) and WWLK (Opitz et al., 2021). For WWLK we use WWLK-k3e2n introduced in Opitz et al. (2021).

Alignment Our approach relies directly on the structural information extracted from the word-concept alignment. There are several alignment standards: first, Information Sciences Institute (ISI) provides extended AMR 2.0 and AMR 3.0 datasets with alignments of all the graph semantic units that are directly related to the sentences' spans (Pourdamghani et al., 2014). Second, Linguistically Enriched AMR (Blodgett and Schneider, 2021, LEAMR) achieves full graph-alignment coverage by aligning all the graph semantic units to a corresponding span in the sentence.

Silver Data Following Bevilacqua et al. (2021), we explore the same strategy to generate a dataset with 140k silver sentence-graph pairs. The silver LEAMR alignments are generated using the approach of Huguet Cabot et al. (2022).

6.3 Models

We use SPRING (Bevilacqua et al., 2021) as our baseline, and an auto-regressive model based on BART (Lewis et al., 2020) for predicting linearized versions of AMR graphs. Our models are built on top of this model, inheriting some hyperparameters (see Table 9).

In order to address the issue of overfitting, we implement a masking strategy which is used in conjunction with dropout and weight decay. For each batch, input tokens are masked with a varying probability p_{mask}, which is uniformly sampled from the specified masking range (see Appendix A for details). The strategy is used for all models including SPRING (ours). In the following paragraphs, we explain the specific setup per each model.

Graph Leakage Model We explore two different settings for GLM: i) Contracted WAG, and ii) Full WAG (see Section 3).

	Model	AMR 3.0
	SPRING (ours)	84.55
KD	Full WAG (89.58)	83.90
LeakDistill (Self-KD)	$L_{leak} + L_{nll}^D$	84.47
	$L_{leak} + L_{KL}$	85.03
	$L_{leak} + L_{nll}^D + L_{KL}$	**85.04**

Table 2: Knowledge Distillation results for the development set of AMR 3.0.

Knowledge Distillation We test KD on the GLM with the highest SMATCH among CWAG and FWAG (see Table 1).

LeakDistill As done for GLM, we first examine the difference in performance between Contracted WAG and Full WAG. Then, we test Full WAG with i) β scheduling, ii) the silver data, iii) the combination of the silver data and the β scheduling. In the case of the scheduling of β, we start from $\beta = 90$ and decrease it linearly at each iteration for 21k iterations in total until it reaches 10. The hyperparameter α is set to 20. The value of β for the case i) and other hyperparameters are listed in Table 9.

7 Results

In this section, we provide our experimental findings. All tables show single-run results.

Graph Leakage Model Table 1 shows results for the Graph Leakage Model. While this setup relies on information being leaked from the final graph structure, it sets an upper bound on how encoding such information can improve performance. Here, we observe an increase of around five SMATCH points when using FWAG, whereas CWAG improvements are much smaller. While the model is certainly taking advantage of the leaked information, this is provided through the hidden states of the encoder. Therefore, we need to explore whether some of this performance gain can be kept implicitly without any information leak. Moreover, it is necessary to investigate the persistence of any performance disparity between CWAG and FWAG. This information is intriguing, as CWAG and FWAG differ in the context of additional information availability. CWAG only possesses a structure akin to the original graph, while FWAG not only exhibits a greater degree of structural similarity but also includes the original labels for non-aligned nodes.

KD and LeakDistill Table 2 compares the results between applying KD with GLM as the teacher versus the LeakDistill approach, explained in Section 5.3.We see how KD alone falls short of taking full advantage of the performance gains of GLM. On the other hand, LeakDistill, especially when including the KL loss, leads to about a 0.5 SMATCH point increase on the development set. Hence, we focus on LeakDistill as our main approach. Table 5 shows a breakdown of the experiments with LeakDistill, such as scheduling the KL loss or adding a silver data pretraining phase. It is evident that the performance difference between CWAG and FWAG remains, paving the way for more in-depth research into the types of information that prove advantageous for LeakDistill. Additionally, the final row of Table 5 presents the outcome when the adaptors are active (the green path). It is noticeable that, despite the green path essentially being the GLM, it fails to match the performance level of 89.58.

Main results Tables 3 and 4 shows results for our proposed model, based on BART-large. Our system performs better than any previous single model parser, and, most notably, does so even without extra data, i.e. silver sentence-graph pairs. For AMR 2.0, we see up to 0.7 SMATCH increase over AMRBART and 0.4 on AMR 3.0. The use of extra data only leads to a small improvement, showing the efficiency of our approach, which is able to outperform previous state-of-the-art systems that relied on up to 200K extra samples. In the breakdown performance, we see how our system performs worse than ATP on Reentrancies, Negation and notably SRL. We believe this is due to the multitask nature of ATP, where SRL is explicitly included as a task. This opens the door to future work exploring the interaction between our approach and the inclusion of auxiliary tasks.

It is worth noting that our system relies on alignment information which is openly discussed at various stages in the paper. We do not consider this information as extra data since it is generated based on the existing data.

Out-of-distribution evaluation Table 6 shows the Out-of-Distribution of LeakDistill. We see a smaller improvement on TLP, 0.3 over AMRBART. On the harder BioAMR, performance increased by over a point, showing how the model is able to generalize well on different domains.

Model	Extra Data	Smatch	Unlab.	NoWSD	Conc.	Wiki	NER	Reent.	Neg.	SRL
SPRING (ours)	✗	84.4	87.4	84.8	90.4	<u>84.1</u>	90.9	71.6	73.5	80.1
BiBL	✗	84.6	87.8	85.1	90.3	83.6	92.5	74.4	73.9	83.1
Ancestor	✗	84.8	88.1	85.3	90.5	<u>84.1</u>	91.8	<u>75.1</u>	74.0	83.4
LeakDistill	✗	$85.7^{s,o}$	88.6	86.2	91.0	83.9	91.1	74.2	**76.8**	81.8
SPRING	200K	84.3	86.7	84.8	90.8	83.1	90.5	72.4	73.6	80.5
ATP	40K	85.2^{s}	88.3	85.6	90.7	83.3	**93.1**	74.7	74.9	**83.3**
AMRBART	200K	85.4^{s}	88.3	85.8	91.2	81.4	91.5	73.5	74.0	81.5
LeakDistill	140K	$\mathbf{86.1}^{s,o,b,a}$	**88.8**	**86.5**	**91.4**	83.9	91.6	<u>75.1</u>	76.6	82.4

Table 3: AMR 2.0 results and comparisons with previous systems. Bold indicates best performance per set, underline in case of a tie. Breakdown extra scores after vertical line. Superscript indicates the result is significantly better using an approximate randomization test (Riezler and Maxwell, 2005) at $p < 0.05$ with respect to $s = SPRING$, $o = SPRING(ours)$, $b = BiBL$, $a = ATP$. We are unable to test Ancestor due to no public checkpoint. Appendix D contains the descriptions for the columns.

Model	Extra Data	Smatch	Unlab.	NoWSD	Conc.	Wiki	NER	Reent.	Neg.	SRL
SPRING	✗	83.0	85.4	83.5	89.5	81.2	87.1	71.3	71.7	79.1
SPRING (ours)	✗	83.8	86.7	84.3	89.9	81.5	87.2	71.4	71.5	79.8
Ancestor	✗	83.5	86.6	84.0	89.5	81.5	88.9	**74.2**	72.6	82.2
BiBL	✗	83.9^{s}	87.2	84.3	89.8	**83.7**	**93.2**	73.8	68.1	81.9
LeakDistill	✗	$84.5^{s,o,a}$	<u>87.5</u>	<u>84.9</u>	90.5	80.7	88.5	73.1	73.7	80.7
ATP	40K	83.9^{s}	87.0	84.3	89.7	81.0	88.4	73.9	**73.9**	82.5
AMRBART	200K	$84.2^{s,o,a}$	87.1	84.6	90.2	78.9	88.5	72.4	72.1	80.3
LeakDistill	140K	$\mathbf{84.6}^{s,o,b,a}$	<u>87.5</u>	<u>84.9</u>	**90.7**	81.3	87.8	73.4	73.0	80.9

Table 4: AMR 3.0 results and comparisons with previous systems. Bold indicates best performance per set, underline in case of a tie. Breakdown extra scores after vertical line. Superscript indicates the result is significantly better using an approximate randomization test (Riezler and Maxwell, 2005) at $p < 0.05$ with respect to $s = SPRING$, $o = SPRING(ours)$, $b = BiBL$, $a = ATP$. We are unable to test Ancestor due to no public checkpoint. Appendix D contains the descriptions for the columns.

Model	AMR 3.0
SPRING (ours)	84.55
Contracted WAG	84.90
Full WAG	85.04
+ β scheduling	85.08
+ Silver	**85.34**
+ Silver + β scheduling	85.28
The green path (Figure 4)	
FWAG + Silver	86.09

Table 5: Performance of LeakDistill models on the development set of AMR 3.0.

BART base Our state-of-the-art system relies on BART-large, which has 400M parameters. While it shows very strong performance, it has a big computational footprint, especially at inference time due to its auto-regressive generative nature. This makes the need for lighter, more compute efficient models an important step towards better Semantic Parsers. Table 7 shows the performance of our approach when trained on top of BART-base, which has 140M parameters, achieving 83.5 SMATCH points on AMR 3.0, 1 point higher than AMR-BART and, noticeably, surpassing SPRING-large performance by half a point. We believe it is crucial to have close to state-of-the-art performance base models, closing the gap from 2 points to 1 when compared to their large counterparts.

Other metrics Recent studies have shown that achieving a higher SMATCH score does not necessarily result in better performance of an AMR parser, as demonstrated by Opitz and Frank (2022). To address this issue, we use two additional evaluation metrics, namely S^2MATCH and WWLK-k3e2n (WWLK), which measure graded concept similarity and edge label importance, respectively. Our experiments reveal that S^2MATCH correlates well with SMATCH, as expected for monolingual

Model	TLP	BioAMR
SPRING	81.3	61.6
BiBL	78.6	61.1
ATP	78.9	61.2
AMRBART	82.3	63.4
LeakDistill	**82.6**	**64.5**

Table 6: Out of distribution results. AMRBART and SPRING are taken from Lee et al. (2022).

Model	AMR 2.0	AMR 3.0
SPRING	82.8	-
AMRBART	83.6	82.5
LeakDistill	**84.7**	**83.5**

Table 7: BART-base versions performance.

Model	SMATCH	S^{2}MATCH	WWLK
SPRING	83.0	84.2	84.8
BiBL	83.9	84.6	82.3
ATP	83.9	84.7	85.7
AMRBART	84.2	85.1	83.9
LeakDistill	**84.6**	**85.5**	**85.9**

Table 8: Performance on AMR 3.0 for different metrics. S^{2}MATCH is taken from Opitz et al. (2020). We use WWLK-k3e2n as proposed in Opitz and Frank (2022).

parsers. Conversely, WWLK is specifically designed for monolingual AMR parsing and emphasizes edge labels. Interestingly, our findings suggest that ATP performs well, second only to our proposed system, LeakDistill. This may be due to the fact that both systems place greater emphasis on edges, with ATP leveraging semantic role labeling data and LeakDistill utilizing structural information such as edges in the FWAGs. In contrast, AMRBART and BiBL exhibit a significant drop in performance compared to the SPRING baseline, possibly due to their use of masking as an additional signal, as their masking strategies may not be beneficial for edge labels.

8 Performance Analysis

Seq2seq parsers show decreased performance for longer sentences since a single error at decoding time in an early step can lead to compound errors and suffer from exposure bias. We explore how this affects our model compared to SPRING, ATP and AMRBART. Figure 5 shows the performance
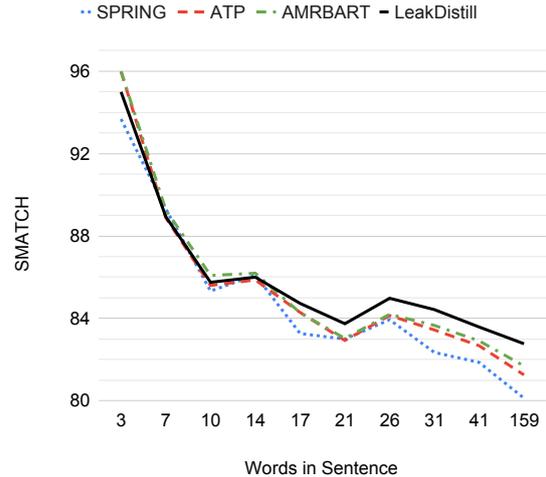

Figure 5: SMATCH score for buckets of 200 instances. X axis shows max. number of words per sentence.

on AMR 3.0 test set for buckets of 200 sentences split by the number of words. While performance is similar on shorter sentences, with AMRBART showing slightly better performance, in longer sentences of over 14 words LeakDistill fares better, especially compared to the baseline, which drops to 80 SMATCH points. This experiment also shows how performance is relatively stable for medium-length sentences (10-30 words, oscillating around 85 points), while it starts deteriorating for longer ones. The high performance on short sentences is likely due to easy-to-parse structures, such as single date sentences.

9 Conclusion

We presented a new approach to training the Transformer architecture where partial information of the target sequence can be learned via self-knowledge distillation: the information can be leaked in the encoder implicitly through Transformer adapters which improve training but are switched off during inference. By employing this approach in AMR parsing, we achieved state-of-the-art results among non-ensemble methods. Moreover, we produced a lightweight AMR parser that outperforms SPRING while having four times fewer parameters. We also showed that, for all methods, performance degrades as the number of words increases.

Interestingly, our approach can potentially be used in other tasks, such as Relation Extraction, where alignments between input and target sequence elements exist, or structural information is unavailable at inference time.

10 Limitations

Our approach for training the Transformer architecture using self-knowledge distillation is promising, but there are still some limitations that need to be addressed in future work. One limitation is that our approach is only tested on the task of AMR parsing, and more evaluations are needed to see if it generalizes well to other tasks, such as Relation Extraction. Additionally, our approach, as is also the case for other current methods, exhibits performance degradation as the number of words in the sentence increases. This may be an indication of the current methods' limitation or lack of robustness to longer sentences.

Another limitation is the added complexity and extra parameters required by the use of Transformer adapters, which increases the overall complexity of the architecture and training time. Even though our approach still achieves state-of-the-art results and it is as lightweight as previous systems at inference time, this fact should be considered by researchers if they should decide to adopt it for other tasks.

In summary, our approach presents an innovative way to train the Transformer architecture and achieve state-of-the-art results in AMR parsing. However, more work is needed to further improve the performance of the model and to apply it to other tasks as well.

11 Ethical considerations

In considering the ethical and social implications of our proposed approach to AMR parsing, we acknowledge that there are several important considerations to take into account.

One significant concern is the potential for bias in the training data and models, which can result in unfair or discriminatory outcomes for certain groups of individuals. Additionally, the training and test data may not be representative of the population that the model will be applied to, potentially leading to poor performance in specific domains.

Furthermore, our approach relies on the use of Transformer-based models, which have been shown to perpetuate societal biases present in the data used for training. It is, therefore, crucial to ensure that the data used for training is diverse and unbiased.

Moreover, the use of techniques such as self-knowledge distillation may lead to data leakage, where the model overfits the training data and performs poorly on new data, which could have negative impacts on the predictions.

In conclusion, even if we consider our approach does not have negative implications, it is important to note that bias and fairness are complex issues that require ongoing attention and improvement.

References

Omri Abend and Ari Rappoport. 2013. Universal Conceptual Cognitive Annotation (UCCA). In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 228–238, Sofia, Bulgaria. 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)*, pages 6001–6015, Dublin, Ireland. Association for Computational Linguistics.

Miguel Ballesteros and Yaser Al-Onaizan. 2017. AMR parsing using stack-LSTMs. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1269–1275, Copenhagen, Denmark. 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.

Michele Bevilacqua, Rexhina Blloshmi, and Roberto Navigli. 2021. One SPRING to Rule Them Both: Symmetric AMR semantic Parsing and Generation without a Complex Pipeline. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(14):12564–12573.

Rexhina Blloshmi, Michele Bevilacqua, Edoardo Fabiano, Valentina Caruso, and Roberto Navigli. 2021. SPRING Goes Online: End-to-End AMR Parsing and Generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language*

Processing: System Demonstrations, pages 134–142, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Austin Blodgett and Nathan Schneider. 2021. Probabilistic, structure-aware algorithms for improved variety, accuracy, and coverage of AMR alignments. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3310–3321, Online. Association for Computational Linguistics.

Claire Bonial, Lucia Donatelli, Mitchell Abrams, Stephanie M. Lukin, Stephen Tratz, Matthew Marge, Ron Artstein, David Traum, and Clare Voss. 2020a. Dialogue-AMR: Abstract Meaning Representation for dialogue. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 684–695, Marseille, France. European Language Resources Association.

Claire Bonial, Stephanie M. Lukin, David Doughty, Steven Hill, and Clare Voss. 2020b. InfoForager: Leveraging semantic search with AMR for COVID-19 research. In *Proceedings of the Second International Workshop on Designing Meaning Representations*, pages 67–77, Barcelona Spain (online). Association for Computational Linguistics.

Deng Cai and Wai Lam. 2020. 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.

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, editors. 2022. *ATP: AMRize Then Parse! Enhancing AMR Parsing with PseudoAMRs*. Association for Computational Linguistics.

Ziming Cheng, Zuchao Li, and Hai Zhao. 2022. BiBL: AMR parsing and generation with bidirectional Bayesian learning. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 5461–5475, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Marco Damonte, Shay B. Cohen, and Giorgio Satta. 2017. An incremental parser for Abstract Meaning Representation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 536–546, Valencia, Spain. Association for Computational Linguistics.

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.

Ramón Fernandez Astudillo, Miguel Ballesteros, Tahira Naseem, Austin Blodgett, and Radu Florian. 2020. Transition-based parsing with stack-transformers. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1001–1007, Online. Association for Computational Linguistics.

Jeffrey Flanigan, Chris Dyer, Noah A. Smith, and Jaime Carbonell. 2016. CMU at SemEval-2016 task 8: Graph-based AMR parsing with infinite ramp loss. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1202–1206, San Diego, California. Association for Computational Linguistics.

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. 2014. A discriminative graph-based parser for the Abstract Meaning Representation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1426–1436, Baltimore, Maryland. Association for Computational Linguistics.

Sangchul Hahn and Heeyoul Choi. 2019. Self-knowledge distillation in natural language processing. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pages 423–430, Varna, Bulgaria. INCOMA Ltd.

Jan Hajič, Eva Hajičová, Jarmila Panevová, Petr Sgall, Ondřej Bojar, Silvie Cinková, Eva Fučíková, Marie Mikulová, Petr Pajas, Jan Popelka, Jiří Semecký, Jana Šindlerová, Jan Štěpánek, Josef Toman, Zdeňka Urešová, and Zdeněk Žabokrtský. 2012. Announcing Prague Czech-English Dependency Treebank 2.0. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 3153–3160, Istanbul, Turkey. European Language Resources Association (ELRA).

Hardy Hardy and Andreas Vlachos. 2018. Guided neural language generation for abstractive summarization using Abstract Meaning Representation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 768–773, Brussels, Belgium. Association for Computational Linguistics.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *ArXiv*, abs/1503.02531.

Pere-Lluís Huguet Cabot, Abelardo Carlos Martínez Lorenzo, and Roberto Navigli. 2022. AMR Alignment: Paying Attention to Cross-Attention. *ArXiv*, abs/2206.07587.

Pavan Kapanipathi, Ibrahim Abdelaziz, Srinivas Ravishankar, Salim Roukos, Alexander Gray, Ramón Fernandez Astudillo, Maria Chang, Cristina Cornelio, Saswati Dana, Achille Fokoue, Dinesh Garg, Alfio Gliozzo, Sairam Gurajada, Hima Karanam, Naweed Khan, Dinesh Khandelwal, Young-Suk Lee, Yunyao Li, Francois Luus, Ndivhuwo Makondo, Nandana Mihindukulasooriya, Tahira Naseem, Sumit Neelam, Lucian Popa, Revanth Gangi Reddy, Ryan Riegel, Gaetano Rossiello, Udit Sharma, G P Shrivatsa Bhargav, and Mo Yu. 2021. Leveraging Abstract Meaning Representation for knowledge base question answering. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3884–3894, Online. Association for Computational Linguistics.

Laura Baranescu Claire Bonial Madalina Bardocz Kira Griffitt Ulf Hermjakob Daniel Marcu Martha Palmer Tim O'Gorman Nathan Schneider Kevin Knight, Bianca Badarau. 2020. Abstract meaning representation (amr) annotation release 3.0.

Hoang Thanh Lam, Gabriele Picco, Yufang Hou, Young-Suk Lee, Lam M. Nguyen, Dzung T. Phan, Vanessa López, and Ramon Fernandez Astudillo. 2021. Ensembling Graph Predictions for AMR Parsing.

Young-Suk Lee, Ramón Astudillo, Hoang Thanh Lam, Tahira Naseem, Radu Florian, and Salim Roukos. 2022. Maximum Bayes Smatch ensemble distillation for AMR parsing. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5379–5392, Seattle, United States. 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.

Kexin Liao, Logan Lebanoff, and Fei Liu. 2018. Abstract Meaning Representation for multi-document summarization. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1178–1190, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Jungwoo Lim, Dongsuk Oh, Yoonna Jang, Kisu Yang, and Heuiseok Lim. 2020. I know what you asked: Graph path learning using AMR for commonsense reasoning. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages

2459–2471, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Yijia Liu, Wanxiang Che, Bo Zheng, Bing Qin, and Ting Liu. 2018. An AMR aligner tuned by transition-based parser. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2422–2430, Brussels, Belgium. Association for Computational Linguistics.

Chunchuan Lyu and Ivan Titov. 2018. AMR parsing as graph prediction with latent alignment. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 397–407, Melbourne, Australia. Association for Computational Linguistics.

Abelardo Carlos Martínez Lorenzo, Marco Maru, and Roberto Navigli. 2022. Fully-Semantic Parsing and Generation: the BabelNet Meaning Representation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1727–1741, Dublin, Ireland. Association for Computational Linguistics.

Roberto Navigli, Rexhina Blloshmi, and Abelardo Carlos Martinez Lorenzo. 2022. BabelNet Meaning Representation: A Fully Semantic Formalism to Overcome Language Barriers. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36.

Juri Opitz, Angel Daza, and Anette Frank. 2021. Weisfeiler-leman in the bamboo: Novel AMR graph metrics and a benchmark for AMR graph similarity. *Transactions of the Association for Computational Linguistics*, 9:1425–1441.

Juri Opitz and Anette Frank. 2022. Better Smatch = better parser? AMR evaluation is not so simple anymore. In *Proceedings of the 3rd Workshop on Evaluation and Comparison of NLP Systems*, pages 32–43, Online. Association for Computational Linguistics.

Juri Opitz, Letitia Parcalabescu, and Anette Frank. 2020. AMR similarity metrics from principles. *Transactions of the Association for Computational Linguistics*, 8:522–538.

Nima Pourdamghani, Yang Gao, Ulf Hermjakob, and Kevin Knight. 2014. Aligning English strings with Abstract Meaning Representation graphs. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 425–429, Doha, Qatar. Association for Computational Linguistics.

Luigi Procopio, Rocco Tripodi, and Roberto Navigli. 2021. SGL: Speaking the graph languages of semantic parsing via multilingual translation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 325–337, Online. Association for Computational Linguistics.

Sudha Rao, Daniel Marcu, Kevin Knight, and Hal Daumé III. 2017. Biomedical event extraction using Abstract Meaning Representation. In *BioNLP 2017*, pages 126–135, Vancouver, Canada,. Association for Computational Linguistics.

Leonardo F. R. Ribeiro, Yue Zhang, and Iryna Gurevych. 2021a. Structural adapters in pretrained language models for AMR-to-Text generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4269–4282, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Leonardo F. R. Ribeiro, Yue Zhang, and Iryna Gurevych. 2021b. Structural adapters in pretrained language models for amr-to-text generation.

Stefan Riezler and John T. Maxwell. 2005. On some pitfalls in automatic evaluation and significance testing for MT. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 57–64, Ann Arbor, Michigan. Association for Computational Linguistics.

Linfeng Song, Daniel Gildea, Yue Zhang, Zhiguo Wang, and Jinsong Su. 2019. Semantic neural machine translation using AMR. *Transactions of the Association for Computational Linguistics*, 7:19–31.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015. Boosting transition-based AMR parsing with refined actions and auxiliary analyzers. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 857–862, Beijing, China. Association for Computational Linguistics.

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 (Volume 2: Short Papers)*, pages 571–577, Dublin, Ireland. Association for Computational Linguistics.

Sheng Zhang, Xutai Ma, Kevin Duh, and Benjamin Van Durme. 2019. 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.

Jiawei Zhou, Tahira Naseem, Ramón Fernandez Astudillo, and Radu Florian. 2021. AMR parsing with action-pointer transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5585–5598, Online. Association for Computational Linguistics.

Appendices

A Model Hyperparameters

Table 9 lists hyperparameters and search space for the experiments:

- LR sched. - learning rate scheduler

- KL temp. - Kullback–Leibler divergence temperature

- AMR 3 aligns. - type of alignments for AMR 3.0

- Mask. range - masking range. For each batch, we mask the input tokens with probability p_{mask}, the value for which is sampled uniformly from the masking range. For instance, the [0; 0.15] range means $p_{mask} \sim U(0, 0.15)$

The LeakDistill experiments detailed in Table 2 were performed utilizing the final set of hyperparameters listed in Table 9. However, it should be noted that the experiment that did not involve KL loss did not necessitate the use of the variable α.

B Hardware and size of the model

We performed experiments on a single NVIDIA 3090 GPU with 64GB of RAM and Intel® Core™ i9-10900KF CPU. The total number of trainable parameters of LeakDistill is 434,883,596. Training the model on the silver data took 33 hours, whereas further fine-tuning took 16 hours.

C BLINK

All systems from Tables 3 and 4 use BLINK (Wu et al., 2020) for wikification. For this purpose, we used the $blinkify.py$ script from the SPRING repository.

D Metric

We evaluate AMR parsing using the SMATCH metric (Cai and Knight, 2013) and extra scores of Damonte et al. (2017): i) Unlabel, compute on the predicted graphs after removing all edge labels, ii) No WSD, compute while ignoring Propbank senses (e.g., duck-01 vs duck-02), iii) Wikification, F-score on the wikification (:wiki roles), iv) NER, F-score on the named entity recognition (:name roles), v) Negations, F-score on the negation detection (:polarity roles), vi) Concepts, F-score on the concept identification task, vii) Reentrancy, computed on reentrant edges only, viii) Semantic Role Labeling (SRL), computed on :ARG-i roles only.

Group	Parameter	Values
Inherited (SPRING)	Optimizer	RAdam
	Batch size	500
	Dropout	0.25
	Attent. dropout	0
	Grad. accum.	10
	Weight decay	0.004
	LR	0.00005
	Beamsize	5
SPRING (ours)	LR sched.	const., **linear**
	Mask. range	[0; {0, **0.15**}]
	Beamsize	5, **10**
Adapter	Encoder layers	1-12
	Activation	GELU
	Dropout	**0.01**, 0.1
GLM	LR	0.00005, **0.0001**
	LR sched.	const., **linear**
	Mask. range	[0; 0.15]
KD	α	10
	LR	0.00005, **0.0001**
	LR sched.	const., **linear**
	Weight decay	0.004, **0.0001**
	Decoder	train, **freeze**
	Mask. range	[0; 0.15]
LeakDistill	LR sched.	const., **linear**
	KL temp.	**1**, 2
	α	1, 5, 10, **20**
	β	1, 5, 10, **sched.**
	AMR 3 aligns.	ISI, **LeAMR**
	Mask. range	[0; {0, 0.1, **0.15**}]
	Beamsize	5, **10**

Table 9: Final hyperparameters and search space for the experiments. All groups have the same parameters as original SPRING if they are not overwritten. For instance, SPRING (ours) and for LeakDistill have the same learning rate of 0.00005.

E Data

The AMR 3.0 (Kevin Knight, 2020) data used in this paper is licensed under the *LDC User Agreement for Non-Members* for LDC subscribers, which can be found here. The *The Little Prince* Corpus can be found here from the Information Science Institute of the University of Southern California.

F Algorithms

Algorithm 2 shows one training step of the LeakDistill model.

Algorithm 2 One training step of the LeakDistill model

    Input: X - batch of input sequences and WAGs,
    Y - batch of target graphs
    Set Model to Normal Mode
    $L_{nll}^{D}, Probs_1 \leftarrow Model(X, Y)$
    Set Model to Leak Mode
    $L_{leak}, Probs_2 \leftarrow Model(X, Y)$
    $L_{KL} \leftarrow$ KLDiv $(Probs_1, Probs_2)$
    $L \leftarrow \alpha L_{KL} + \beta L_{leak} + L_{nll}^{D}$
    Optimization step of L