Incorporating Graph Information in Transformer-based AMR Parsing

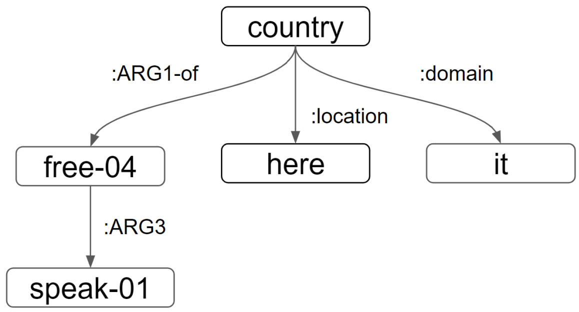
Anonymous ACL submission

Abstract

Abstract Meaning Representation (AMR) is a Semantic Parsing formalism that aims at providing a semantic graph abstraction representing a given text. Current approaches employ Transformer-based 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 explore a modification to the Transformer architecture, using structural adapters to explicitly incorporate graph structural information into the learned representations and improve AMR parsing performance. Our experiments show how, by employing word-to-node alignment, we can construct a graph to embed structural information using the hidden states through the Encoder. While employing the graph structure constitutes a data leak, we demonstrate how this information leads to a performance gain that can be preserved implicitly via self-knowledge distillation, providing a new State-of-the-Art (SotA) AMR parser, improving over previous ones even without the use of additional data. We release the code at availableafterreview.

1 Introduction

Creating a machine-interpretable representation of meaning lies at the core of Natural Language Understanding, which has been framed as Semantic Parsing. Even though multiple formalisms have been proposed throughout the years (e.g., Hajič et al. (2012); Abend and Rappoport (2013); White et al. (2016)), 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. As of now, AMR is widely employed 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 other areas. Figure 1 shows an example of an AMR graph.



1) Here, it is a country with the freedom of speech.
2) It is a country with the freedom of speech here.
3) Here it is a country with speech freedom.

Figure 1: Three sentences with the same AMR.

In recent years, autoregressive models proved to be the best approach for semantic parsing because of their outstanding performances without relying on sophisticated ad-hoc architectures (Xu et al., 2020; Bevilacqua et al., 2021; Procopio et al., 2021). Several approaches have recently emerged to increase performance by including structural information into the model (Chen et al., 2022), adding extra Semantic Role Labeling tasks (Bai et al., 2022) or by using 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) based on Graph Neural Networks (GNN) to boost performance through self-distillation. GNN is a type of neural network which deals with data that has structural dependencies; that is, it can be represented as a graph. However, graph information is not available at inference time, hence the need to transfer the knowledge of a model exploiting the graph structure. Knowledge Distillation (KD) is the technique that transfers the knowledge from a teacher model to a student model (Hinton et al., 2015). In our approach, we leverage concept-node alignments to generate a word-based graph with a structure similar to the original AMR. After that, such a graph is employed, with structural adapters, in the Encoder of a Transformer Encoder-Decoder architecture to obtain soft targets, which are then used for self-knowledge distillation, transferring the knowledge from the teacher path with the leaked graph structure to the student, which only has access to the text.

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

2 Related Work

Throughout the years, multiple trends have appeared to parse AMR graphs: using statistical methods (Flanigan et al., 2014, 2016a; Wang et al., 2015a), neural-transition based parsers (Ballesteros and Al-Onaizan, 2017; Liu et al., 2018; Fernandez Astudillo et al., 2020; Zhou et al., 2021a), bidirectional transformers (Lyu and Titov, 2018; Zhang et al., 2019; Cai and Lam, 2020) based on BERT (Devlin et al., 2019), sequence-to-sequence transformers (Xu et al., 2020; Bevilacqua et al., 2021; Procopio et al., 2021; Chen et al., 2022; Bai et al., 2022), or by ensemble models (Lam et al., 2021; Lee et al., 2022).

The interest in Transformer models based on BART (Lewis et al., 2020) has constantly increased over the last years since they obtained SotA performances without complex pipelines. In semantic parsing, these models face the task similarly to Neural Machine Translation, where the text is translated into a linearized version of the graphs. The earliest attempts (Xu et al., 2020; Bevilacqua et al., 2021) were trained with pairs of sentences and graphs, so the model automatically generates the representation of the sentences.

Lately, some works have extended sequence-to-sequence models to incorporate extra information useful for parsing. Procopio et al. (2021) leverages multitask learning to improve cross-lingual AMR parsing results. Chen et al. (2022, ATP) expand the dataset with extra auxiliary tasks such as Semantic Role Labeling and Dependency Parsing, with pseudo-AMR graphs constructed based on a particular task. During training, a special task tag is added at the beginning of the input sentence, and the ATP model predicts the output for such a task. Bai et al. (2022, AMRBART) pre-train the model on 200k graphs generated by SPRING where generated linearized graphs are modified with a masking strategy and used as input. Additionally, they use a unified strategy involving the concatenation of a masked graph and a masked text. The model needs to reconstruct the original sequence, similarly to Masked Language Modeling. In such a way, the model improves structure awareness of Pretrained Language Models (PLM) over AMR graphs. Moreover, recent research has shown the capabilities of the autoregressive models for extracting alignment information online while parsing (Huguet Cabot et al., 2022). Finally there is a recent surge of ensemble models. Lam et al. (2021) devised a new strategy to merge predicted graphs, and Lee et al. (2022) expanded on it by ensemble distillation. However we decide not to compare our work with ensemble strategies, as they rely on multiple parsers, such as our proposed one, rendering comparisons unfit.

3 Fundamentals

3.1 AMR 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:

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

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:

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

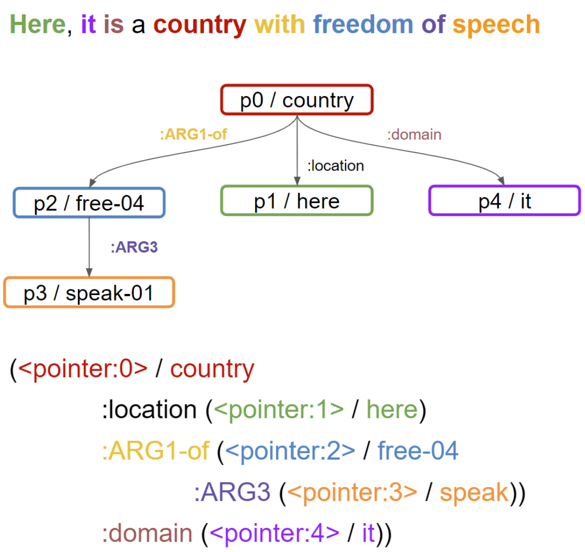


Figure 2: Top: sentence. Middle: AMR graph. Bottom: Linearized graph. Alignment is represented by colours.

In order to model the problem, one can exploit the transfer learning capabilities of BART. In Bevilacqua et al. (2021, SPRING), the vocabulary of BART is updated with tokens corresponding to i) AMR-related tokens, ii) variable names <R0>, <R1>, ... <Rn> and other tokens needed for the various graph linearizations. In addition, BART is fine-tuned with the input x and the target g. The approach described in this work is built on top of the SPRING model, which we consider our baseline system.

3.2 AMR alignment

Since the AMR graph represents the semantic meaning behind a sentence, there exists an alignment between the spans in text and semantic units in graphs. In Figure 2, we can find an example. Notice how most of the words are connected to a node in the graph but some, such as the preposition *a*, are not reflected or aligned to the graph. Indeed, some Semantic Parsers rely on alignment to be trained (Wang et al., 2015b; Flanigan et al., 2016b; Misra and Artzi, 2016; Damonte et al., 2017; Zhou et al., 2021b). Multiple alignment formalisms have been proposed through the years, such as JAMR (Flanigan et al., 2014), ISI (Pourdamghani et al., 2014) or LEAMR (Blodgett and Schneider, 2021). We will leverage alignment to construct a graph based on the words in the sentence and their representation.

3.3 Structural adapters

Ribeiro et al. (2021b) have shown how the Transformer architecture can be modified to improve PLM for modeling graph information. They introduced the Structural Adapter (StructAdapt), a residual neural network involving a Graph Convolutional (GraphConv) layer.

We employ structural adapters to encode the graph structure imposed by a Word-Aligned Graph (see Section 4.1), leading to the construction of a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where each token of the input x is linked to a node $v \in \mathcal{V}$, and $\mathcal{E}$ is an unlabeled set of edges $\{(u, v)|u, v \in \mathcal{V}\}$. Moreover, we remove layer normalization and set GELU as an activation function (Figure 4). Then, for each hidden representation $\mathbf{h}_v^l \in \mathbb{R}^b$ from the encoder layer l and the set of edges $\mathcal{E}$, we compute the updated hidden states $\mathbf{z}_v^l$ as:

$$\begin{aligned} \mathbf{g}_v^l &= \mathrm{GraphConv}_l(\mathbf{h}_v^l, \mathcal{E}) \\ \mathbf{z}_v^l &= \mathbf{W}_a^l \sigma(\mathbf{g}_v^l) + \mathbf{h}_v^l, \end{aligned} \quad (3)$$

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

Likewise Ribeiro et al. (2021b), the adapters are inserted after each Encoder's layer having the same amount of adapters as encoder layers (Figure 5).

Graph Convolution In a similar manner to Ribeiro et al. (2021b), we use GraphConv proposed by Kipf and Welling (2017) and computed as:

$$\mathrm{GraphConv}_l(\mathbf{h}_v^l, \mathcal{E}) = \sum_{u \in \mathcal{N}(v)} \frac{1}{\sqrt{d_u d_v}} \mathbf{W}_g^l \mathbf{h}_u^l, \quad (4)$$

where $\mathcal{N}(v)$ is a set that contains v and its adjacent nodes, d_v is the degree of v, $\mathbf{W}_g^l \in \mathbb{R}^{b \times b}$ is a parameter.

4 Model

In the next section, we explain how we have extended the SPRING architecture in order to leverage AMR structure information from word-nodes alignments using structural adapters and self-knowledge distillation.

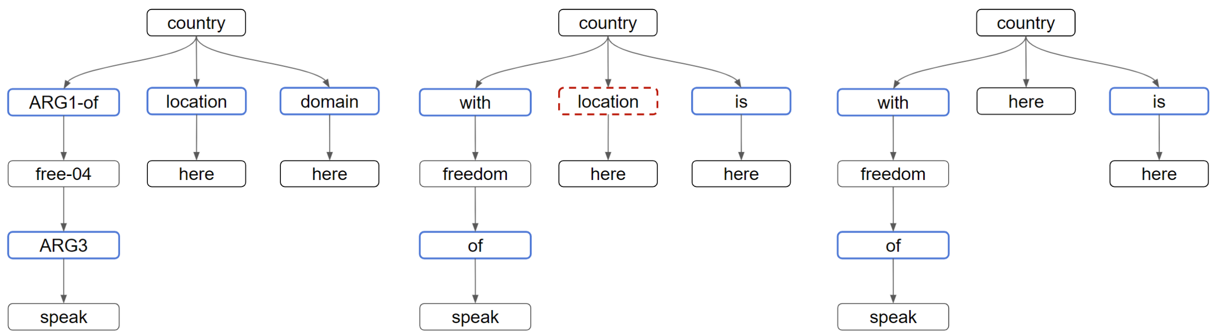


Figure 3: Different representations of the sentence: "Here, it is a country with the freedom of speech". An AMR with edges converted in nodes (left), a Full WAG (center) and a Contracted WAG (right).

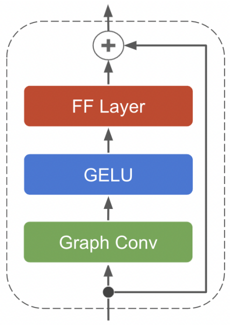


Figure 4: Structural adapter without layer normalization and with GELU activation.

4.1 Word-Aligned Graph

Using AMR alignment, we design a Word-Aligned Graph (WAG) representation where nodes and relations are actual words of the input sentence instead of the AMR concepts when it is possible. First, we convert the relations into nodes that connect two adjacent nodes in the original graph (see Figure 3, left). Then, we replace the nodes and relations with the aligned words of their respective sentence (Figure 3, center).

Unfortunately, a problem arises when dealing with the structural adapter since non-aligned nodes (e.g., the :location relation in Figure 3) do not have associated hidden states. Therefore, in order to use WAG in the structural adapter, we have two alternatives: i) remove nodes for which we do not have hidden states, i.e., contract non-aligned nodes (see Section 4.1.1), or ii) create new hidden states for them (see Section 4.1.2).

4.1.1 Contracted WAG

For the first approach, we must remove non-aligned nodes from the graph. However, deleting the nodes from the original graph would produce a disconnected graph. To achieve a similar connected structure to the original graph, we contract nodes rather than remove them. A contracted WAG (CWAG) is a graph in which non-aligned nodes are compressed with the closest parent node, preserving all relations from both nodes. Figure 3 (right) depicts an AMR and its corresponding CWAG. Additionally, in multi-token nodes (e.g., "Romneycare"), we merge to the first token any of the subword tokens.

4.1.2 Full WAG

In the second representation, we preserve the nodes without alignment (the node "location" in Figure 3 (center)). First, if the node's label is in the new AMR special tokens which were added to the model's vocabulary (e.g., :location), we extract the embedding from the embedding matrix. Otherwise, we tokenize the node's label and take the average of their embedding tokens as the representation. Furthermore, the representations for the non-aligned nodes are added to the model in the first adapter layer by concatenating them with the hidden states of the encoder. After each adapter block, we split representations into two groups: i) the updated hidden states for the original input tokens, which are inputs for the next Transformer layer, ii) the updated hidden states for the non-aligned nodes, which are concatenated again before the next adapter block. This type of graph is referred to as a Full WAG (FWAG). Figure 3 (center) shows an example of FWAG.

4.2 Graph Leakage Model (GLM)

Through WAG, we explore whether information leakage has an impact on performance. In this manner, we can determine the model's upper bound performance with the enhanced Encoder, determining which graphs and adapter architectures are suitable for the Two-Path Model described in Section 4.4.

Thereby, we insert structural adapters in each

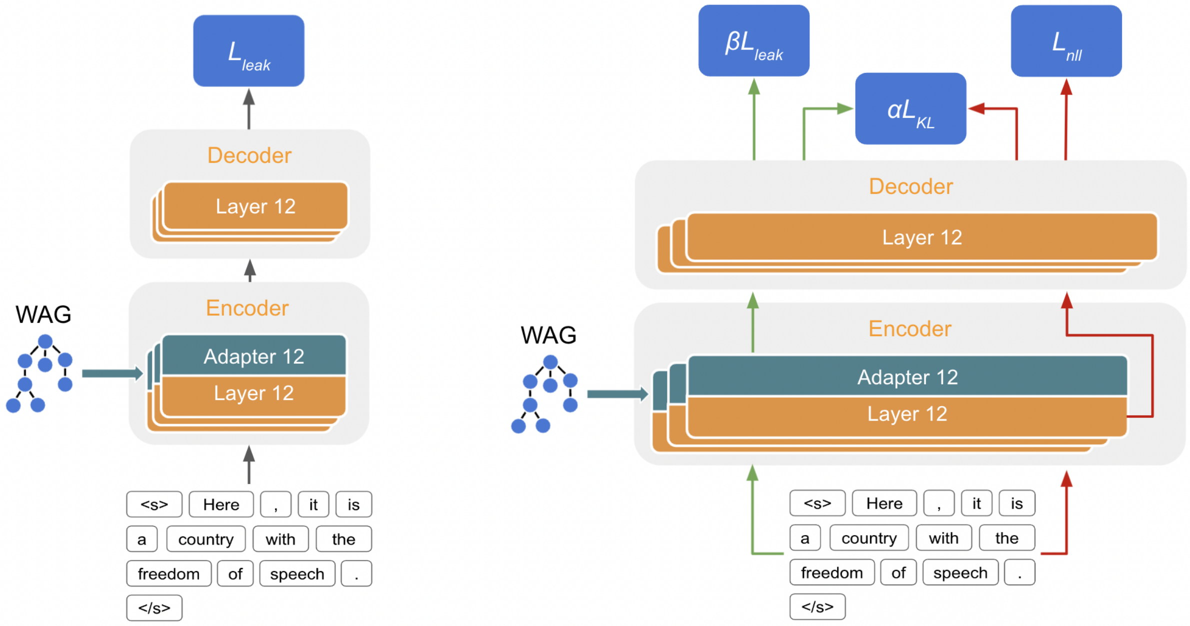


Figure 5: Left: Scheme of Graph Leakage Model. Right: Scheme of Two-Path Model, where parameters of the original BART Encoder and Decoder are shared partially among two paths: (green) path incorporating WAG information via adapters, (red) path omitting adapters which is basically the outcome model for the problem.

Transformer layer of the Encoder (Figure 5 left). The adapter's input in a layer l consists of a matrix of hidden states H^l and the set of edges $\mathcal{E}$. In the case of CWAG, H^l corresponds to hidden states of Transformer layer l. Whilst, for FWAG, we add extra representations as described in Section 4.1.2. Note that the set of edges $\mathcal{E}$ does not change through layers. The loss function for GLM is:

$$L_{leak} = L_{nll}(\tilde{D}), \quad (5)$$

where $\tilde{D}$ is the updated dataset consisting of pairs $((x^i, a^i), g^i)$, where a^i is the WAG.

GLM performance may be considered as the upper model's bound in which the encoder learns graph information from alignments.

4.3 Knowledge Distillation

GLM leverages the alignment information to enhance the model's conception of the graph structure to improve the parsing performance of the model. Unfortunately, WAG cannot be employed at inference time since alignment information is not available when parsing text to predict a graph. Therefore, following the idea of knowledge distillation, we set the teacher to be the pre-trained GLM that employs the structural information using sentences and WAGs as input, and then we project such knowledge to the student model, which is just aware of the sentences, with no adapters. Therefore, our objective is to achieve the following:

$$f_{stud}(x) = f_{GLM}(x, a), \quad (6)$$

where a is the WAG.

For this purpose, we set SPRING as the student. The fundamental objective is to encourage the student to match the teacher's probability distribution by minimizing the following loss:

$$L_{KL}(p, q) = KL(p, q) = \sum_{i=0}^{C-1} p_i \log(\frac{p_i}{q_i}), \quad (7)$$

where q and p are probabilities of the teacher and the student respectively, KL is Kullback–Leibler divergence, C is the number of classes. Usually, the loss L_{nll} for the original task is added to the total loss:

$$L_{KD} = L_{nll} + \alpha L_{KL}, \quad (8)$$

where α is a hyper-parameter.

The architectural differences between the teacher and the student model belong to the encoder, not the decoder, since the teacher is the one with the structural adapters. Therefore, we copy the GLM decoder to the student model and fix the decoder and the language model head parameters.

4.4 Two-Path Model

We propose a Two-Path Model (TPM) for learning where, in contrast with the KD approach, a single model is trained via two paths simultaneously, one path with the structural adapters and the other without them. Then, we force the two paths to learn the same distribution by adding a Kullback–Leibler divergence loss on the output logits. As a result, the total loss is:

$$L_{TPM} = \alpha L_{KL} + \beta L_{leak} + L_{nll}, \quad (9)$$

where L_{leak} is the loss for the first path, with leaked information, L_{nll} is the loss for the second path, which is the original negative log-likelihood loss, and finally L_{KL} is the above-described Kullback–Leibler divergence loss. α, β are hyper-parameters to control each loss scale.

Notably, the Two-Path Model is a variant of knowledge distillation as described in Section 4.3, called self-knowledge distillation (Hahn and Choi, 2019). In this case, we project the knowledge via the adapter's path rather than computing soft target probabilities. Moreover, we calculate KL divergence for all classes to distill more knowledge from the first path. Finally, based on the assumption that there is not enough information to distill at the initiation of the training process, we train with scheduling the L_{leak} multiplier β, where β is gradually decreasing.

5 Experimental Setup

To demonstrate the benefits of incorporating structural information in AMR parsing, we devise a set of experiments to assess its performance with respect to State-of-the-Art models. Before delving into their details, we first provide thorough information regarding the dataset (Subsection 5.1) and model (Subsection 5.2) used in our experiments.

5.1 Datasets

We tested 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. Furthermore, we tested on the Little Prince and the Bio AMR out-of-distribution datasets.

Alignment Our approach directly relies on the structural information extracted between 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 anything in the sentence.

Silver Data Following Bevilacqua et al. (2021), we have explored the same strategy to generate a dataset with 140k silver sentence-graph pairs. The silver alignments were generated using the approach of Huguet Cabot et al. (2022), where they are extracted from the cross-attention of the model.

5.2 Models

We use SPRING (Bevilacqua et al., 2021) as our baseline model, an auto-regressive model based on BART (Lewis et al., 2020) for predicting linearized versions of AMR graphs. Our models have been built on top of SPRING, inheriting some of its hyper-parameters (see Table 7). Structural adapters leverage one graph convolutional layer and GELU activation. In the next paragraphs, we explain the specific setup per each model.

Graph Leakage Model We explore two different settings for GLM: i) *Contracted WAG*, Section 4.1.1 - Figure 3 right; and ii) *Full WAG*, Section 4.1.2 - Figure 3 center.

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

Two-Path Model Likewise 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 hyper-parameter α is set to 20.

6 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

Model	AMR 3.0
SPRING	84.10
Contracted WAG	86.01
Full WAG	89.58
Leaked Path of TPM	86.09

Table 1: GLM results for AMR 3.0 development set.

	Model	AMR 3.0
	SPRING	84.10
KD	Full WAG (89.58)	83.90
Two-Path	$L_{leak} + L_{nll}$	84.47
	$L_{leak} + L_{nll} + L_{KL}$	**85.04**

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

Model	AMR 3.0
SPRING	84.10
Contracted WAG	84.90
Full WAG	85.04
+ β scheduling	85.08
+ Silver	**85.34**
+ Silver + β scheduling	85.28

Table 3: Performance of Two-Path models on the development set of AMR 3.0.

we observe an increase of around five SMATCH points when including concept labels and token masking. While the model is certainly taking advantage of the leaked information, this is encoded 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.

Knowledge Distillation and TPM Table 2 compares the results between applying KD with GLM as the teacher versus the self-KD approach, TPM, explained in Section 4.4. We see how KD alone falls short of taking full advantage of the performance gains of GLM. On the other hand, TPM, especially when including the KL loss, leads to over one SMATCH point increase on the development set. Hence we focus on TPM as our main approach. Table 3 shows a breakdown of the experiments with TPM, such as scheduling the KL loss or adding a silver data pretraining phase.

Model	TLP	BioAMR
SPRING	81.3	61.6
ATP	79.0	55.2
AMRBART	82.3	63.4
Ours	**82.6**	**64.5**

Table 4: Out of distribution results. ATP, 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
Ours	**84.7**	**83.5**

Table 5: BART-base versions performance.

Main Results Table 6 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 the need of extra data. 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 SotA 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.

BART base Our SotA system relies on BART-large, which has 400M parameters. While it shows great 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 5 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 AMRBART and, noticeably, surpassing SPRING-large performance by half a point. We believe it is crucial to have close to SotA performance base models, closing the gap from 2 points to 1 when compared to its large counterparts.

	Model	Extra Data	Smatch	Unlab.	NoWSD	Conc.	Wiki	NER	Reent.	Neg.	SRL
AMR 2.0	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
	Ours	0K	85.7^{s}	88.6	86.2	91.0	83.9	91.1	74.2	76.8	81.8
	Ours	140K	**86.1**^{s,a}	**88.8**	**86.5**	**91.4**	83.9	91.6	**75.1**	**76.6**	82.4
AMR 3.0	SPRING	0K	83.0	85.4	83.5	89.5	81.2	87.1	71.3	71.7	79.1
	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}	87.1	84.6	90.2	78.9	88.5	72.4	72.1	80.3
	Ours	0K	84.5^{s,a}	87.5	84.9	90.5	80.7	88.5	73.1	73.7	80.7
	Ours	140K	**84.6**^{s,a}	87.5	84.9	**90.7**	**81.3**	87.8	73.4	73.0	80.9

Table 6: Results and comparisons with previous systems. Bold indicates best performance per set, underline in case of a tie. Breakdown extra scores after vertical line. Upperscript indicates result is significantly better using an approximate randomization test (Riezler and Maxwell, 2005) at $p < 0.05$. $s = SPRING$, $a = ATP$. Ours is the only system significantly better than ATP.

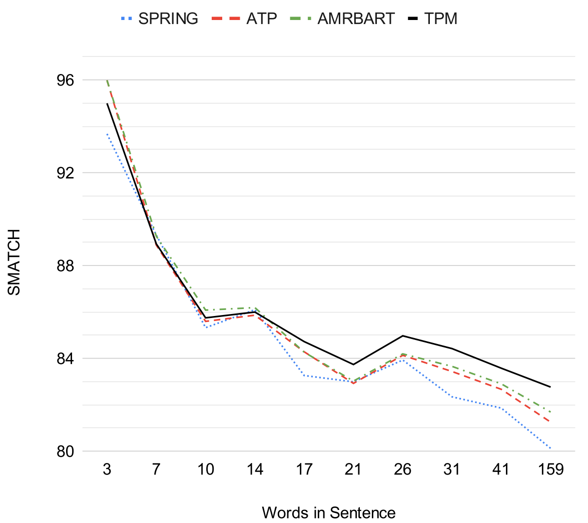


Figure 6: SMATCH score for buckets of 200 instances. X axis shows max. number of words per sentence.

Out-of-distribution evaluation Table 4 shows the Out-of-Distribution of TPM. We see a smaller improvement on TLP, 0.3 over AMRBART. On the harder BioAMR, performance increased over a point, showing how the model is able to generalize well on different domains.

7 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 6 shows the SMATCH performance on AMR 3.0 test set for buckets of 200 sentences divided by the number of words. While the performance is similar on shorter sentences, with AMRBART showing slightly better performance, with longer sentences of over 14 words TPM shows better performance, especially compared to the baseline, which drops to 80 SMATCH points for longer sentences. 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 sentence is likely due to expressing easy-to-parse structures such as single date sentences.

8 Conclusion

We presented a new approach to training the Transformer architecture where partial information of the target sequence can be learned via multi-tasking and 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 SotA results among non-ensemble methods. Moreover, we produced a lightweight AMR parser that outperforms SPRING having four times fewer parameters. We also showed that, for all methods, including ours, performance degrades as the number of words increases, which raises a question of limitation of the current methods based on BART.

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

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.

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.

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. 2016a. 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, Chris Dyer, Noah A. Smith, and Jaime Carbonell. 2016b. Generation from Abstract Meaning Representation using tree transducers. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 731–739, 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.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*.

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.

Martin Josifoski Sebastian Riedel Luke Zettlemoyer Ledell Wu, Fabio Petroni. 2020. Zero-shot entity linking with dense entity retrieval. In *EMNLP*.

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.

Dipendra Kumar Misra and Yoav Artzi. 2016. Neural shift-reduce CCG semantic parsing. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1775–1786, Austin, Texas. Association for Computational Linguistics.

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. 2015a. 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.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015b. A transition-based algorithm for AMR parsing. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 366–375, Denver, Colorado. Association for Computational Linguistics.

Aaron Steven White, Drew Reisinger, Keisuke Sakaguchi, Tim Vieira, Sheng Zhang, Rachel Rudinger, Kyle Rawlins, and Benjamin Van Durme. 2016. Universal decompositional semantics on Universal Dependencies. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1713–1723, Austin, Texas. Association for Computational Linguistics.

Dongqin Xu, Junhui Li, Muhua Zhu, Min Zhang, and Guodong Zhou. 2020. Improving AMR parsing with sequence-to-sequence pre-training. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2501–2511, Online. 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. 2021a. 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.

Jiawei Zhou, Tahira Naseem, Ramón Fernandez Astudillo, Young-Suk Lee, Radu Florian, and Salim Roukos. 2021b. 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.

Appendices

A Model Hyper-Parameters

Table 7 lists hyperparameters and search space for the experiments:

- LR sched. - learning rate scheduler
- KL temp. - KL temperature
- AMR 3 aligns. - a type of alignments for AMR 3.0
- Mask. range - masking range. For each batch, we mask the input tokens with probability p, the value for which is sampled uniformly from the masking range.

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
Adapter	Encoder layers	1-12
	Activation	GELU
	Dropout	**0.01**, 0.1
GLM	LR	0.00005, **0.0001**
	LR sched.	const., **linear**
KD	LR	0.00005, **0.0001**
	LR sched.	const., **linear**
	Weight decay	0.004, **0.0001**
Two-Path	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 7: Final hyperparameters and search space for the experiments

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 TPM 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 Table 6 use BLINK (Ledell Wu, 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.

E Data

The AMR 3.0 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 Limitations

At train time, our system requires alignment between graph and sentence. We obtain them for the silver data with an external system which overcomes this limitation, but other systems do not rely on alignment. Since we have two pathways inside the TPM architecture, the model requires two forward paths. Along with the fact that we have three losses, the model is considered computationally heavier than its competitors from Table 6 at training time. However, the number of parameters and computational cost/time remains the same at inference time.