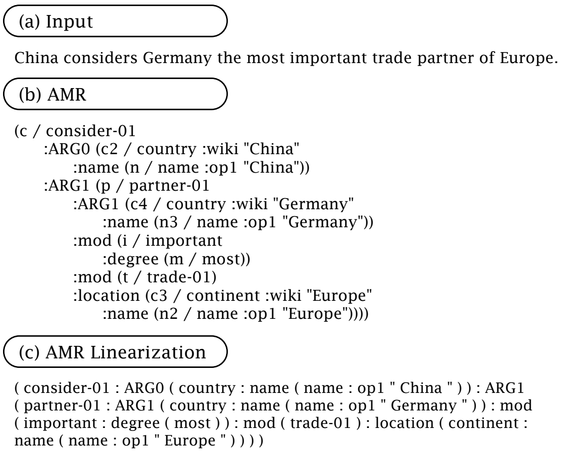
Improving AMR Parsing with Sequence-to-Sequence Pre-training

Dongqin Xu[1] **Junhui Li**[1*] **Muhua Zhu**[2] **Min Zhang**[1] **Guodong Zhou**[1]
[1]School of Computer Science and Technology, Soochow University, Suzhou, China
[2]Tencent News, Beijing, China
xdqck@live.com, {lijunhui, minzhang, gdzhou}@suda.edu.cn
muhuazhu@tencent.com

Abstract

In the literature, the research on abstract meaning representation (AMR) parsing is much restricted by the size of human-curated dataset which is critical to build an AMR parser with good performance. To alleviate such data size restriction, pre-trained models have been drawing more and more attention in AMR parsing. However, previous pre-trained models, like BERT, are implemented for general purpose which may not work as expected for the specific task of AMR parsing. In this paper, we focus on sequence-to-sequence (seq2seq) AMR parsing and propose a seq2seq pre-training approach to build pre-trained models in both single and joint way on three relevant tasks, i.e., machine translation, syntactic parsing, and AMR parsing itself. Moreover, we extend the vanilla fine-tuning method to a multi-task learning fine-tuning method that optimizes for the performance of AMR parsing while endeavors to preserve the response of pre-trained models. Extensive experimental results on two English benchmark datasets show that both the single and joint pre-trained models significantly improve the performance (e.g., from 71.5 to 80.2 on AMR 2.0), which reaches the state of the art. The result is very encouraging since we achieve this with seq2seq models rather than complex models. We make our code and model available at https://github.com/xdqkid/S2S-AMR-Parser.



(a) Input

China considers Germany the most important trade partner of Europe.

(b) AMR

```
(c / consider-01
      :ARG0 (c2 / country :wiki "China"
            :name (n / name :op1 "China"))
      :ARG1 (p / partner-01
            :ARG1 (c4 / country :wiki "Germany"
                  :name (n3 / name :op1 "Germany"))
            :mod (i / important
                  :degree (m / most))
            :mod (t / trade-01)
            :location (c3 / continent :wiki "Europe"
                  :name (n2 / name :op1 "Europe"))))
```

(c) AMR Linearization

```
( consider-01 : ARG0 ( country : name ( name : op1 " China " ) ) : ARG1
( partner-01 : ARG1 ( country : name ( name : op1 " Germany " ) ) : mod
( important : degree ( most ) ) : mod ( trade-01 ) : location ( continent :
name ( name : op1 " Europe " ) ) ) )
```

Figure 1: An example of seq2seq-based AMR parsing.

1 Introduction

Abstract meaning representation (AMR) parsing aims to translate a textual sentence into a directed and acyclic graph which consists of concept nodes and edges representing the semantic relations between the nodes (Banarescu et al., 2013). Previous studies focus on building diverse approaches to modeling the structure in AMR graphs, such as tree-based approaches (Wang et al., 2015b; Groschwitz et al., 2018), graph-based approaches (Flanigan et al., 2014; Werling et al., 2015; Cai and Lam, 2019), transition-based approaches (Damonte et al., 2017; Guo and Lu, 2018), sequence-to-sequence (seq2seq) approaches (Peng et al., 2017; van Noord and Bos, 2017; Konstas et al., 2017; Ge et al., 2019), and sequence-to-graph (seq2graph) approaches (Zhang et al., 2019a,b; Cai and Lam, 2020). Among these approaches, seq2seq-based approaches, which properly transform AMR graphs into sequences, have received much interest, due to the simplicity in implementation and the competitive performance.

Similar to many NLP tasks, the performance of AMR parsing is much restricted by the size of human-curated dataset. For example, even recent AMR 2.0 contains only 36.5K training AMRs. To alleviate the effect of such restriction, a previous attempt is to utilize large-scale unlabeled sentences with self-training (Konstas et al., 2017). Alternatively, a more recent feasible solution is to resort to pre-training which builds pre-trained models on large-scale (unlabeled) data. Linguistic knowledge captured in pre-trained models can then be properly

[*] Corresponding Author: Junhui Li.

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 2501–2511,
November 16–20, 2020. ©2020 Association for Computational Linguistics

Task	Dataset	Source	Target
machine translation	gold	sentence	sentence
syntactic parsing	silver	sentence	tree sequence
AMR parsing	silver	sentence	AMR sequence

Table 1: Three seq2seq learning tasks explored in this paper to obtain pre-trained models. Here *silver* dataset indicates that the sequences in the target-side are generated automatically .

incorporated into the training of an AMR parser. However, the widely used pre-trained models such as ELMO (Peters et al., 2017) and BERT (Devlin et al., 2019) may not work as expected for building a state-of-the-art seq2seq AMR parser. The reasons are two-fold. On the one hand, previous studies on both seq2seq-based AMR parsing and AMR-to-text generation demonstrate the necessity of a shared vocabulary for the source and target sides (Ge et al., 2019; Zhu et al., 2019). Using pre-trained models like BERT as pre-trained encoders for AMR parsing, however, will violate the rule of sharing a vocabulary. On the other hand, pre-trained models such as BERT are basically tuned for the purpose of representing sentences instead of generating target sequences. According to Zhu et al. (2020), by contrast to using BERT directly as the encoder, a more reasonable approach is to utilize BERT as an extra feature or view BERT as an extra encoder. See Section 5.1 for more detailed discussions on the effect of BERT on AMR parsing.

In this paper, we propose to pre-train seq2seq models that aim to capture different linguistic knowledge from input sentences. To build such pre-trained models, we explore three different yet relevant seq2seq tasks, as listed in Table 1. Here, machine translation acts as the most representative seq2seq task which takes a bilingual dataset as the training data. According to Shi et al. (2016) and Li et al. (2017), a machine translation system with good performance requires the model to well derive linguistic information from input sentences. The other two tasks require auto-parsed syntactic parse trees and AMR graphs as the training data, respectively. It is worth noting that the pre-training task of AMR parsing is in the similar spirit of self-training (Konstas et al., 2017).

In order to investigate whether various seq2seq pre-trained models are complementary to each other in the sense that they can be learned jointly to achieve better performance, we further explore joint learning of several pre-training tasks and evaluate its effect on AMR parsing. In addition, motivated by Li and Hoiem (2018), we extend the vanilla fine-tuning method to optimize for both the performance of AMR parsing and response preservation of the pre-trained models. Detailed experimentation on two widely used English benchmarks shows that our approach substantially improves the performance, which greatly advances the state-of-the-art. This is very encouraging since we achieve the state-of-the-art by simply making use of the generic seq2seq framework rather than designing sophisticated AMR parsing models.

2 Baseline: AMR Parsing as Seq2Seq Learning

Seq2Seq Modeling. The encoder in the *Transformer* (Vaswani et al., 2017) consists of a stack of multiple identical layers, each of which has two sub-layers: one implements the multi-head self-attention mechanism and the other is a position-wise fully connected feed-forward network. The decoder is also composed of a stack of multiple identical layers. Each layer in the decoder consists of the same sub-layers as in the encoder layers plus an additional sub-layer that performs multi-head attention to the output of the encoder stack. See Vaswani et al. (2017) for more details.

Pre-Processing: Linearize AMR Graph to Target Sequence. As in van Noord and Bos (2017), we obtain simplified AMRs by removing variables and wiki links. Variables in AMR graphs are only necessary to indicate co-referring nodes and they do not carry any semantic information by themselves. Therefore, AMR graphs are first converted into AMR trees by removing variables and duplicating the co-referring nodes. Then newlines present in an AMR tree are replaced by spaces to get a sequence. Figure 1(c) illustrates the linearization result of the AMR graph in Figure 1(b). Based on the data of sentences paired with linearized AMR graphs, we train a seq2seq model whose outputs are also linearized AMRs.

Post-Processing: Recover AMR Graph from Target Sequence. The output from *Transformer* is an AMR sequence without variables, wiki-links, and co-occurrent variables. Moreover, the output may contain brackets that do not match, resulting incomplete concepts. To recover its full graph, the post-processing should restore information removed in pre-processing by assigning a unique variable to each concept, pruning duplicated and redundant material, performing Wikification, and restoring co-referring nodes. Meanwhile, it should fix incomplete concepts.

We use the pre-processing and post-processing scripts provided by van Noord and Bos (2017). [1]

3 Seq2Seq Pre-training for AMR Parsing

In this section, we first present our single pre-training approach, followed by the joint pre-training approach on two or more pre-training tasks. Then we present our fine-tuning methods.

3.1 Single Pre-training

To be consistent with the seq2seq model for AMR parsing, the pre-trained models in this paper are all built on the Transformer. That is, for each pre-training task listed in Table 1, we learn a seq2seq model which will be used to initialize seq2seq model for AMR parsing in the fine-tuning phase. When building the pre-trained models, we merge all the source and target sides of the three pre-training tasks, and construct a shared vocabulary. Moreover, in all the models we share vocabulary embeddings for both the source and target sides.

PTM-MT is a seq2seq neural machine translation (NMT) model which is trained on a publicly available bilingual dataset. According to findings in Goldberg (2019) and Jawahar et al. (2019), the Transformer encoder is strong in capturing syntax and semantics from source sentences, which is helpful to AMR parsing.

PTM-SynPar is a seq2seq constituent parsing model. Building such a model requires a training dataset which consists of sentences paired with constituency parse trees. To construct a silver treebank, we parse the English sentences in the bilingual data for MT by using an off-the-shelf parser. Then we linearize the automatic parse trees to get syntax sequences, as illustrated in Figure 2. Note that in the linearization, we let the output contain the words from the source sentence. The motivation here is to regard parsing as a language generation problem, similar to the idea in Choe and Charniak (2016).

PTM-SemPar is a seq2seq AMR parsing model trained on a silver corpus of auto-parsed AMR graphs. To construct such a corpus, we apply the baseline system of AMR parsing to process the English sentences in the bilingual MT corpus. Then we adopt the linearization process illustrated in Figure 1 to obtain source-target pairs. Finally, we train a seq2seq-based AMR parsing model on the silver corpus that will be used as a pre-trained model.

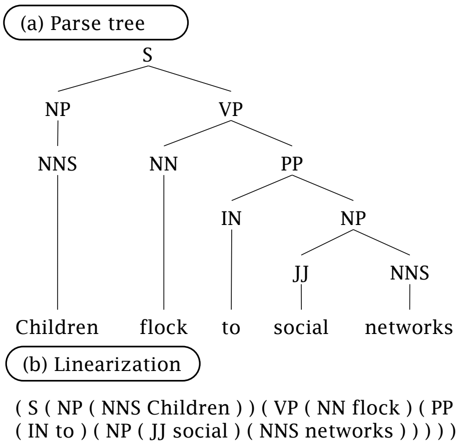

(a) Parse tree
S
NP
VP
NNS
NN
PP
IN
NP
JJ
NNS
Children
flock
to
social
networks
(b) Linearization
(S (NP (NNS Children)) (VP (NN flock) (PP
(IN to) (NP (JJ social) (NNS networks)))))

Figure 2: A linearization example of the parse tree for the sentence of *Children flock to social networks*.

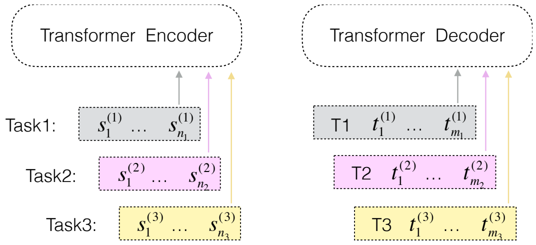

Transformer Encoder
Transformer Decoder
Task1:
$s_1^{(1)} \ldots s_{n_1}^{(1)}$
T1 $t_1^{(1)} \ldots t_{m_1}^{(1)}$
Task2:
$s_1^{(2)} \ldots s_{n_2}^{(2)}$
T2 $t_1^{(2)} \ldots t_{m_2}^{(2)}$
Task3:
$s_1^{(3)} \ldots s_{n_3}^{(3)}$
T3 $t_1^{(3)} \ldots t_{m_3}^{(3)}$

Figure 3: Illustration of the joint pre-training approach.

3.2 Joint Pre-training

Intuitively, the above described single pre-trained models can capture linguistic features from different perspectives. One question is whether these models are complementary when they are properly used to initialize a seq2seq-based AMR parser. To empirically answer this question, we propose to build pre-trained models through jointly learning multiple pre-training tasks. Inspired by the zero-shot approach proposed for multi-lingual neural machine translation (Johnson et al., 2017), we add a unique preceding tag to the target side of training data to distinguish the task of each training instance, as illustrated in Figure 3.

With such tagged training instances, multi-task learning is actually quite straightforward. We simply combine the training data of all the pre-training tasks that we are focusing on and then feed the

[1] https://github.com/RikVN/AMR

combined training data to the Transformer model. The training process interleaves training data from each task. For example, we update parameters on a batch of training instances from task1 and then update parameters on a batch of training instances from task2, and the process iterates. With such a joint training strategy, we obtain four joint pre-trained models, i.e., PTM-MT-SynPar, PTM-MT-SemPar, PTM-SynPar-SemPar, and PTM-MT-SynPar-SemPar. Names of the models can tell what pre-training tasks are learned jointly.

3.3 Fine-tuning Methods

Given a pre-trained model, we can directly fine-tune it on a gold AMR corpus to train an AMR parser. For this purpose we use two different fine-tuning methods. In the following we first present the vanilla fine-tuning method, and then extend it under the framework of multi-task learning. For simplicity, we refer to the latter method as Multi-Task Learning (MTL) fine-tuning hereafter.

Vanilla Fine-Tuning optimizes the parameters of an existing pre-trained seq2seq models to train AMR parsing on a gold AMR corpus. Fine-tuning adapts the shared parameters to make them more discriminative for AMR parsing, and the low learning rate is an indirect mechanism to preserve some of the representational structure captured in the pre-training models.

MTL Fine-Tuning is designed to attack the potential drawback of the vanilla fine-tuning method. In vanilla fine-tuning, optimizing model parameters to train AMR parsing presents a potential risk of overfitting. Inspired by Li and Hoiem (2018), we propose to optimize for high accuracy of AMR parsing while preserving the performance on the pre-training tasks. Preservation of the performance on the pre-training tasks can be regarded as a regularizer for the training of AMR parsing. To implement such MTL fine-tuning, we once again adopt the generic multi-task learning framework depicted in Figure 3.

Now the question left behind is how to obtain fine-tuning instances for pre-training tasks. To this end, we use the pre-trained model focused and input sentences of gold AMR corpus to generate fine-tuning instances for pre-training tasks. Formally speaking, given an instance $\{s, t^{(0)}\}$ of the fine-tuning task , and a pre-trained model learned from k pre-training tasks, we first feed the pre-trained model with input s and obtain its k outputs, i.e. $t^1, \cdots, t^k$ for the k pre-training tasks, respectively. Therefore, each input s in the fine-tuning task is now equipped with $k+1$ outputs, one for the fine-tuning task while the other k for the k pre-training tasks. Meanwhile, each output is associated with a unique preceding tag which indicates the corresponding task.

Please also note that we do not apply MTL fine-tuning to the pre-training task of AMR parsing. This is because the fine-tuning task is the same as the pre-training task. For example, for the pre-trained model PTM-MT-SynPar-SemPar, in MTL fine-tuning we only keep the pre-training tasks of MT and syntactic parsing.

4 Experimentation

In this section, we report the performance of our seq2seq pre-training approach to AMR parsing.

4.1 Experimental Settings

Pre-training Dataset and Pre-trained Models For pre-trained models, we use the WMT14 English-to-German dataset[2] which consists of about 3.9M training sentence pairs after filtering out long and imbalanced pairs. To obtain syntactic parse trees for the source sentences, we utilize toolkit AllenNLP (Gardner et al., 2017) which is trained on Penn Treebank (Marcus et al., 1993). To obtain AMR graphs for the source sentences, we utilize our baseline AMR parsing system. Then we merge English/German sentences and linearized parse trees, and AMR graphs together and segment all the tokens into subwords by byte pair encoding (BPE) (Sennrich et al., 2016) with 20K operations.

We implement above pre-trained models based on *OpenNMT-py* (Klein et al., 2017).[3] For simplicity, we unify parameters of these models as the Transformer-base model in Vaswani et al. (2017). The number of layers in encoder and decoder is 6 while the number of heads is 8. Both the embedding size and the hidden size are 512 while the size of feedforward network is 2048. Moreover, we use Adam optimizer (Kingma and Ba, 2015) with β_1 of 0.9 and β_2 of 0.998. Warm_up step, learning rate, dropout rate and label smoothing epsilon are 16000, 2.0, 0.1 and 0.1 respectively. In addition, we set the batch token-size to 8,192. We train the models for 300K steps and choose the model

[2] https://www.statmt.org/wmt14/translation-task.html

[3] https://github.com/OpenNMT/OpenNMT-py

with the best performance on WMT2014 English-to-German development set as the final pre-trained model.

AMR Parsing Benchmarks We evaluate AMR performance on AMR 1.0 (LDC2015E86) and AMR 2.0 (LDC2017T10). The two datasets contain 16,833 and 36,521 training AMRs, respectively, and share 1,368 development AMRs and 1,371 testing AMRs. All the source sentences and linearized AMRs are segmented into subwords by using the BPE trained for the pre-trained models.

To fine-tune the pre-trained models for AMR parsing, we follow the settings of hyper-parameters used for training pre-trained models.

Evaluation Metrics For evaluation purpose, we use the AMR-evaluation toolkit to evaluate parsing performance in Smatch and other fine-grained metrics (Cai and Knight, 2013; Damonte et al., 2017). We report results of single models that are tuned on the development set.

4.2 Experimental Results

Table 2 presents the comparison of our approach and related studies on the test sets of AMR 1.0 and AMR 2.0. From the results, we have the following observations:

- Pre-trained models on a single task (i.e., from #2 to #6) significantly improve the performance of AMR parsing, indicating seq2seq pre-training is helpful for seq2seq-based AMR parsing. We also note that the pre-trained model of NMT achieves the best performance, followed by the pre-trained models on AMR parsing and on syntactic parsing. This indicates that seq2seq AMR parsing benefits more from pre-training tasks that require the encoder be able to capture the semantics from source sentences.

- Joint pre-trained models on two or more pre-training tasks further improve the performance of AMR parsing. However, in the presence of NMT pre-training task, the benefits from joint pre-training with either AMR parsing, syntactic parsing or both are shrunk.

- MTL fine-tuning consistently outperforms the vanilla fine-tuning method. For example, on single pre-training tasks, MTL outperforms vanilla fine-tuning by 1.5 $\sim$ 2.0 Smatch F1 scores while on joint pre-training tasks, the improvements of MTL over vanilla fine-tuning instead decrease.

- With twice training sentences in AMR 2.0, overall the performance on AMR 2.0 is higher than that on AMR 1.0. However, the gap between the performance on AMR 2.0 and AMR 1.0 gets smaller when we move from single pre-training models to joint pre-training models. For example, based on PTM-MT-SynPar-SemPar, the performance gap is 1.1 in Smatch F1 scores, much less than the performance gap 6.9 between their corresponding baselines.

- Finally, our approach achieves the best reported performance on AMR 1.0 and the performance on AMR 2.0 is higher than or close to that achieved by previous studies which use BERT. This is very encouraging taking into consideration the fact that our seq2seq model is much simper than the graph-based models proposed in related studies (Zhang et al., 2019a,b; Naseem et al., 2019; Cai and Lam, 2020).

Table 3 compares the performance of our best system and the systems reported recently with fine-grained metrics. We obtain the best performance for Reentrancies, NER, and SRL. Compared to the systems of Z'19a, Z'19b, and C'20, we achieve lower performance for Wiki and Negations. One possible reason for our relatively lower performance on Wiki and Negations is that unlike above three systems, in this paper we do not anonymize named entities and do not use an extra algorithm to add polarity attributes.

5 Analysis and Discussion

In this section, we conduct more analysis on AMR parsing with pre-trained models. In the following all the results are obtained on AMR 2.0.

5.1 Effect of BERT on Seq2Seq AMR Parsing

To explore the effect of BERT on seq2seq AMR parsing, motivated by Zhu et al. (2020), we use BERT in various ways to boost the performance of AMR parsing.

Given an input sentence $x = (x_1, \cdots, x_n)$ with n words, the BERT tokenizer segments it into a subword sequence $x' = (x'_1, \cdots, x'_m)$ with m

#	**Pre-trained Model**	**Fine-Tune**	**AMR 1.0**			**AMR 2.0**		
			P.	**R.**	**F1**	**P.**	R.	**F1**
1	None	None	69.8	60.2	64.6	75.8	67.7	71.5
2	PTM-MT	Vanilla	78.8	69.5	73.8	80.0	74.3	77.1
3		MTL	81.1	72.2	76.4	81.3	77.1	79.1
4	PTM-SynPar	Vanilla	74.3	65.8	69.8	76.2	71.5	73.8
5		MTL	76.7	68.1	72.2	78.0	72.8	75.3
6	PTM-SemPar	Vanilla	80.8	73.5	77.0	80.8	75.2	77.9
7	PTM-MT-SynPar	Vanilla	79.1	70.5	74.6	79.5	75.0	77.1
8		MTL	81.2	74.0	77.5	81.5	77.6	79.5
9	PTM-MT-SemPar	Vanilla	82.3	75.4	78.7	**82.4**	77.3	79.7
10		MTL	82.4	74.6	78.3	82.3	78.0	80.1
11	PTM-SynPar-SemPar	Vanilla	81.6	74.0	77.6	81.1	76.3	78.6
12		MTL	81.8	74.0	77.7	81.3	76.8	79.0
13	PTM-MT-SynPar-SemPar	Vanilla	82.4	75.4	78.7	82.1	77.6	79.8
14		MTL	**82.6**	**75.9**	**79.1**	82.3	**78.3**	**80.2**
Previous work without extra resources								
	Graph Prediction(Lyu and Titov, 2018)		-	-	-	-	-	74.4
	Prediction(Guo and Lu, 2018)		-	-	-	-	-	69.8
	Prediction(Groschwitz et al., 2018)		-	-	-	-	-	71.0
	Seq2Seq(Ge et al., 2019)		-	-	-	74.0	68.1	70.9
	Seq2Seq(Cai and Lam, 2019)		-	-	-	-	-	73.2
	Graph(Cai and Lam, 2020)		-	-	71.2	-	-	77.3
Previous work with extra resources								
	Seq2Graph(Zhang et al., 2019a)†		-	-	70.2	-	-	76.3
	Seq2Graph(Zhang et al., 2019b)†		-	-	71.3	-	-	77.0
	RL(Naseem et al., 2019)†		-	-	-	-	-	75.5
	Seq2Seq(Ge et al., 2019)∗		-	-	-	77.7	71.1	74.3
	Graph(Cai and Lam, 2020)†		-	-	75.4	-	-	**80.2**

Table 2: Smatch scores on the test sets of AMR 1.0 and AMR 2.0. † is for using BERT as extra resource while ∗ for using other resources.

Metric	**C'19**	**G'19**	**N'19**	**Z'19a**	**Z'19b**	**C'20**	**Our**
Smatch	73.2	74.3	75.5	76.3	77	**80.2**	**80.2**
Unlabeled	77.0	77.3	80	79.0	80	82.8	**83.7**
No WSD	74.2	74.8	76	76.8	78	**80.8**	**80.8**
Reentrancy	55.3	58.3	56	60.0	61	64.6	**66.5**
Concepts	84.4	84.2	86	84.8	86	**88.1**	87.4
NER	82.0	82.4	83	77.9	79	81.1	**85.4**
Wiki	73.2	71.3	80	85.8	86	**86.3**	75.1
Negations	62.9	64.0	67	75.2	77	**78.9**	71.5
SRL	66.7	70.4	72	69.7	71	74.2	**78.9**

Table 3: Detailed F1 scores on AMR 2.0 test set. Here, C'19 is for Cai and Lam (2019), G'19 for Ge et al. (2019), N'19 for Naseem et al. (2019), Z'19 for Zhang et al. (2019a), Z'19b for Zhang et al. (2019b), C'20 for Cai and Lam (2020)

subwords. Then BERT returns a hidden state sequence $b = (b_1, \cdots, b_m)$ in shape $\mathbb{R}^{m\times d_{BERT}}$, where d_{BERT} is the size of BERT hidden states (e.g., d_{BERT}=768 in our experiment). Figure 4 illustrates the process of obtaining BERT hidden states for an input sentence. Next we use the following methods to properly incorporate BERT hidden states b into Transformer-based AMR parsing.

- BERT as embedding, which uses $f\left(bW^B\right)$ as input of the the Transformer encoder, where $W^B \in \mathbb{R}^{d_{BERT}\times d}$ are model parameters to be learned, d is the model size for seq2seq AMR parsing, and f is the activation function *ReLu*.

- BERT as encoder, which uses $f\left(bW^B\right)$ as the output of the Transformer encoder. That is to say, we replace the Transformer encoder with BERT.

- BERT as extra feature, which views b as extra features for an input sentence x'. The input of the Transformer encoder is defined as $f\left([b, (Emb\left(x'\right) + Pos\left(x'\right))]W^E\right)$, where $[\cdot,\cdot]$ represents the operation of concatenation, $Emb\left(x'\right)$ and $Pos\left(x'\right)$ return the word embeddings and position embeddings of x' respectively, and $W^E \in \mathbb{R}^{(d+d_{BERT})\times d}$ are model parameters to be learned.

- BERT as extra encoder, which adds a sub-layer, i.e, BERT-context-attention sub-layer, in the Transformer decoder after the masked-self-attention sub-layer and the context-attention sub-layer. The BERT-context-attention sub-layer works in a similar way as the context-attention sub-layer by attending to BERT hidden states $f\left(bW^B\right)$.

#	Methods	P.	R.	F1
1	None	73.5	66.9	70.0
2	BERT as embedding	78.1	72.2	75.1
3	BERT as encoder	75.5	68.0	71.5
4	BERT as extra feature	79.2	71.5	75.2
5	BERT as extra encoder	75.1	68.2	71.5

Table 4: Smatch scores on AMR 2.0 when incorporate BERT in various methods.

Meanwhile, we also provide another Transformer-based baseline in which we segment input sentences into subwords with the BERT tokenizer. For all above experiments, the source-side vocabulary is the set of subwords in training sentences segmented by the BERT tokenizer while the target-side vocabulary is the set of subwords in training AMRs segmented by BPE mentioned in Section 4.1.

Table 4 compares the performance of AMR parsing when incorporating BERT in various methods. By comparing the performance of #1 in Table 4 against the baseline #1 in Table 2, we observe a drop of Smatch F1 score from 71.5 to 70.0, indicating that it is important to share vocabulary for seq2seq AMR parsing. Based on the baseline of not sharing vocabulary, the four different methods of incorporating BERT result in very different performance ranging from 71.5 to 75.2 in Smatch F1 score. Among them, incorporating BERT as embedding or extra feature achieves similar performance, which is much higher than the performance of incorporating BERT as either encoder or extra encoder. This suggests that rather than straightly feeding BERT hidden states into a decoder, it is important to feed them into an encoder first. However, our pre-trained seq2seq models, even on a single pre-training task (i.e., #3, #5, #6) outperform using BERT, indicating the effectiveness of pre-trained seq2seq models for AMR parsing.

5.2 Effect of Training Data Sizes on Pre-training Models

In this section we investigate the impact of the size of pre-training data to check whether AMR parsing benefits more from pre-trained models that are trained on larger datasets. To this end, we randomly use 20%, 40%, 60%, and 80% of the full pre-training instances to train the pre-trained models, respectively.

As shown in Figure 5, except syntactic parsing (i.e., PTM-SynPar), the pre-training models on the other three kinds of pre-training tasks achieve higher AMR parsing performance with the increasing of training data sizes. Based on the learning curve, we suspect there still exists much room for further improvements if we enlarge the training data of pre-training tasks.

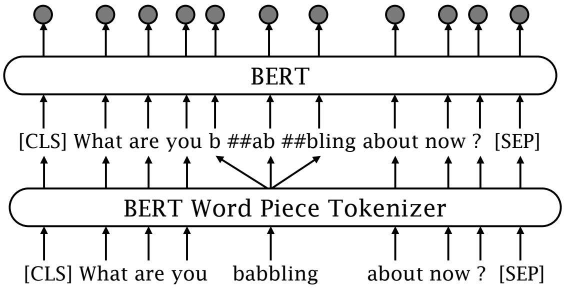

BERT
[CLS] What are you b ##ab ##bling about now ? [SEP]
BERT Word Piece Tokenizer
[CLS] What are you babbling about now ? [SEP]

Figure 4: Illustration of obtaining BERT hidden states for an given sentence.

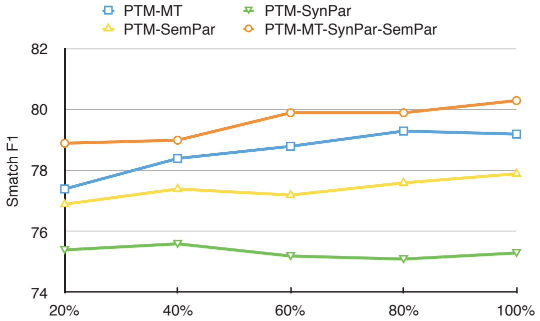


Figure 5: Learning curve over the number of training sentences in pre-training datasets.

5.3 Effect of Different Pre-Training Components on Seq2Seq AMR Parsing

When adapt a pre-trained model to AMR parsing, we initialize the whole seq2seq Transformer model of AMR parsing with the counterpart of the pre-trained model. However, it is unveiled what part of initialization contributes most. To this end, we decompose the whole seq2seq model into three components, i.e., (shared) word embedding, encoder and decoder. The three components take account of 31.1%, 29.5% and 39.4% of parameters, respectively. Then we do ablation study by accumulating the initialization using the pre-trained model while the other components will be randomly initialized.

We use the PTM-MT-SynPar-SemPar pre-trained model as representative (i.e., #14 in Table 2). Table 5 presents the performance. From the table, we observe that with well-learned word em-

Pre-trained Initialization	P.	R.	F1
None	75.8	67.7	71.5
Embedding	80.7	76.3	78.4
Embedding + Encoder	81.3	77.2	79.2
Embedding + Decoder	80.7	76.5	78.5
All	**82.3**	**78.3**	**80.2**

Table 5: Smatch F1 scores on the test sets of AMR2.0 when initialize different components of seq2seq model with a pre-trained model. Here we use MTL as fine-tuning method.

bedding, we substantially boost the performance from 71.5 in Smatch F1 score to 78.4 while initializing the other two components with the pre-trained model leads to another 1.8 improvement in Smatch F1 score (i.e., from 78.4 to 80.2).

5.4 Effect of Pre-trained Models Trained on Different Datasets

As shown in Table 2, the pre-trained model of PTM-SynPar (or PTM-SemPar) significantly improves the performance AMR parsing from 71.5 to 75.3 (or 77.9) in Smatch F1 score. However, in the presence of PTM-MT, joint pre-training with either PTM-SynPar, PTM-SemPar, or both gives another up to 1.0 improvement, suggesting that complementarity among the pre-trained models does exist but is relatively limited. We suspect that the overlapping is mainly due to the fact that we pre-train these models on the same source-side dataset. We conjecture that more improvement is potentially reachable if the pre-training tasks are trained on different datasets.

To test the conjecture, we construct another silver dataset for both syntactic parsing and AMR parsing that is in the same size (i.e., 3.9M) as before. This is done by randomly selecting 3.9M English sentences from WMT14 English monolingual language model training data.[4] Table 6 compares the Smatch F1 scores. From it, we observe consistent improvement if the pre-trained models are jointly trained on different datasets. For example, by replacing the pre-training dataset of AMR parsing with the new constructed dataset, we improve AMR parsing from 80.1 in Smatch F1 score to 81.4. This suggests that assigning different pre-training tasks with different datasets improves the performance of AMR parsing.

#	Pre-trained Model	F1
1	PTM-MT (WMT14B)	79.1
2	PTM-MT(WMT14B)-SemPar(WMT14B)	80.1
3	PTM-MT(WMT14B)-SemPar(WMT14M)	**81.4**
4	PTM-MT(WMT14B)-SynPar(WMT14B)	79.5
5	PTM-MT(WMT14B)-SynPar(WMT14M)	79.9

Table 6: Smatch F1 scores on the test set of AMR 2.0 when the pre-training tasks are trained on different datasets. Here WMT14B is for WMT14 English-to-German dataset while WMT14M is for WMT14 English monolingual dataset.

#	Pre-trained Model		F1
1	PTM-MT on EN-DE	Vanilla	77.1
2		MTL	79.1
3	PTM-MT on EN-FR	Vanilla	77.5
4		MTL	**79.4**

Table 7: Smatch F1 scores on the test set of AMR 2.0 when the pre-training tasks are trained on different bilingual dataset.

5.5 Effect of Different Bilingual Datasets

For the pre-training task of machine translation, we have chosen English-to-German (EN-DE) with 3.9M sentence pairs. However, it is still unclear whether it is critical to choose the right language pair. To this end, we move to WMT14 Englilsh-to-French (EN-FR) translation and randomly select 3.9M sentence pairs from its training dataset, as the same size of EN-DE translation. Table 7 compares the Smatch F1 scores when the pre-trained models are trained on different bilingual datasets. From it, we observe that pre-training on EN-FR dataset achieves even slight higher performance than that on EN-DE dataset. This further confirms our finding that AMR parsing can greatly benefit from machine translation.

6 Related Work

We describe related work from two perspectives: pre-training and AMR parsing.

Pre-training. Pre-training a universal model and then fine-tuning the model on a downstream task have recently become a popular strategy in the field of natural language processing. Previous works on pre-training can be roughly grouped into two categories. One category of approaches is to learn static word embeddings such as word2vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014) while the other group builds dynamic pre-trained models that would also be used in downstream tasks. Representative examples in the latter group in-

[4] http://statmt.org/wmt14/training-monolingual-news-crawl/news.2008.en.shuffled.gz

clude Dai and Le (2015), CoVe (McCann et al., 2017), ELMo (Peters et al., 2017; Edunov et al., 2019), OpenAI GPT (Radford et al., 2018), and BERT (Devlin et al., 2019). Besides the aforementioned encoder-only (e.g., BERT) or decoder-only (e.g., GPT) pre-training approaches, recent studies also propose approaches to pre-training seq2seq models, such as MASS (Song et al., 2019), PoDA (Wang et al., 2019), PEGASUS (Zhang et al., 2020), BART (Lewis et al., 2020), and T5 (Raffel et al., 2020).

AMR Parsing. As a semantic parsing task that translates texts into AMR graphs, AMR parsing has received much attention in recent years. Diverse approaches have been applied to the task. Flanigan et al. (2014) pioneer the research work on AMR parsing by using a a two-stage approach: node identification followed by relation recognition. Werling et al. (2015) improve the first stage in the parser of Flanigan et al. (2014) by generating subgraph aligned to lexical items. To avoid conducting AMR parsing from scratch, Wang et al. (2015b) propose to obtain AMR graphs from dependency trees by using a transition-based method. Wang et al. (2015a) extend their previous work by introducing a new transition action to get better performance. Damonte et al. (2017) propose a complete transition-based approach that parses sentences left-to-right in linear time. The recent neural AMR parsing could be roughly grouped into two categories. On the one hand, the generic seq2seq-based approaches have been widely used for AMR parsing which show competitive performance (Peng et al., 2017; van Noord and Bos, 2017; Konstas et al., 2017; Ge et al., 2019). On the other hand, to better model the graph structure on the target side, graph-based models are well studies for AMR parsing which achieve the state-of-the-art-performance (Lyu and Titov, 2018; Guo and Lu, 2018; Groschwitz et al., 2018; Zhang et al., 2019a,b; Cai and Lam, 2020).

7 Conclusion

In this paper we proposed a seq2seq-based pre-training approach to improving the performance of seq2seq-based AMR parsing. To this end, we designed three relevant seq2seq learning tasks, including machine translation, syntactic parsing, and AMR parsing itself. Then we built seq2seq pre-trained models through either single or joint pre-training tasks. Detail experimentation shows that both the single and joint pre-trained models substantially improve our baseline and the performance reaches the state of the art. The accomplishment is encouraging since we achieve this simply by using the generic seq2seq framework rather than complex models.

Acknowledgments

We thank the anonymous reviewers for their insightful comments and suggestions. This work is supported by the National Natural Science Foundation of China (Grant No. 61876120).

References

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.

Deng Cai and Wai Lam. 2019. Core semantic first: A top-down approach for AMR parsing. In *Proceedings of EMNLP*, page 3799–3809.

Deng Cai and Wai Lam. 2020. AMR parsing via graph⇌sequence iterative inference. In *Proceedings of ACL*, pages 1290–1301.

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structure. In *Proceedings of EACL*, pages 748–752.

Do Kook Choe and Eugene Charniak. 2016. Parsing as language modeling. In *Proceedings of EMNLP*, page 2331–2336.

Andrew M. Dai and Quoc V. Le. 2015. Semi-supervised sequence learning. In *Proceedings of NeurIPS*, pages 3079–3087.

Marco Damonte, Shay B. Cohen, and Giorgio Satta. 2017. An incremental parser for abstract meaning representation. In *Proceedings of EACL*, pages 536–546.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL*, pages 4171–4186.

Sergey Edunov, Alexei Baevski, and Michael Auli. 2019. Pre-trained language model representations for language generation. In *Proceedings of NAACL*, page 4052–4059.

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 ACL*, pages 1426–1436.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2017. AllenNLP: A deep semantic natural language processing platform. In *Proceedings of ACL Workshop for Natural Language Processing Open Source Software*.

Donglai Ge, Junhui Li, Muhua Zhu, and Shoushan Li. 2019. Modeling source syntax and semantics for neural AMR parsing. In *Proceedings of IJCAI*, pages 4975–4981.

Yoav Goldberg. 2019. Assessing BERT's syntactic abilities. In *Computing Research Repository, arXiv:1901.05287. Version 1*.

Jonas Groschwitz, Matthias Lindemann, Meaghan Fowlie, Mark Johnson, and Alexander Koller. 2018. AMR dependency parsing with a typed semantic algebra. In *Proceedings of ACL*, pages 1831–1841.

Zhijiang Guo and Wei Lu. 2018. Better transition-based AMR parsing with a refined search space. In *Proceedings of EMNLP*, pages 1712–1722.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proceedings of ACL*, pages 3651–3657.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, and Fernanda Viégas. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *TACL*, 5:339–351.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR*.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL, System Demonstrations*, page 67–72.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural AMR: Sequence-to-sequence models for parsing and generation. In *Proceedings of ACL*, pages 146–157.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of ACL*, page 7871–7880.

Junhui Li, Deyi Xiong, Zhaopeng Tu, Muhua Zhu, Min Zhang, and Guodong Zhou. 2017. Modeling source syntax for neural machine translation. In *Proceedings of ACL*, pages 688–697.

Zhizhong Li and Derek Hoiem. 2018. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947.

Chunchuan Lyu and Ivan Titov. 2018. AMR parsing as graph prediction with latent alignment. In *Proceedings of ACL*, page 397–407.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, 19(2):313–330.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Proceedings of NeurIPS*, pages 6297–6308.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of NeurIPS*, pages 3111–3119.

Tahira Naseem, Abhishek Shah, Hui Wan, Radu Florian, Salim Roukos, and Miguel Ballesteros. 2019. Rewarding smatch: Transition-based AMR parsing with reinforcement learning. In *Proceedings of ACL*, pages 4586–4592.

Rik van Noord and Johan Bos. 2017. Neural semantic parsing by character-based translation: Experiments with abstract meaning representation. *Computational Linguistics in the Netherlands Journal*, 7:93–108.

Xiaochang Peng, Chuang Wang, Daniel Gildea, and Nianwen Xue. 2017. Addressing the data sparsity issue in neural AMR parsing. In *Proceedings of EACL*, pages 366–375.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2017. Deep contextualized word representations. In *Proceedings of NAACL-HLT*, pages 2227–2237.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. In *URL https://s3-us-west-2.amazonaws. com/openai-assets/researchcovers/languageunsupervised/ languageunderstandingpaper.pdf*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of ACL*, pages 1715–1725.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural MT learn source syntax? In *Proceedings of EMNLP*, pages 1526–1534.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. MASS: Masked sequence to sequence pre-training for language generation. In *Proceedings of ICML*, pages 5926–5936.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N.Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of NIPS*, pages 5998–6008.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015a. Boosting transition-based AMR parsing with refined actions and auxiliary analyzers. In *Proceedings of ACL*, pages 857–862.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015b. A transition-based algorithm for AMR parsing. In *Proceedings of NAACL*, pages 366–375.

Liang Wang, Wei Zhao, Ruoyu Jia, Sujian Li, and Jingming Liu. 2019. Denoising based sequence-to-sequence pre-training for text generation. In *Proceedings of EMNLP*, page 4003–4015.

Keenon Werling, Gabor Angeli, and Christoerpher D. Manning. 2015. Robust subgraph generation improves abstract meaning representation parsing. In *Proceedings of ACL*, pages 982–991.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. 2020. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *Proceedings of ICML*.

Sheng Zhang, Xutai Ma, Kevin Duh, and Benjamin Van Durme. 2019a. AMR parsing as sequence-to-graph transduction. In *Proceedings of ACL*, page 80–94.

Sheng Zhang, Xutai Ma, Kevin Duh, and Benjamin Van Durme. 2019b. Broad-coverage semantic parsing as transduction. In *Proceedings of EMNLP-IJCNLP*, pages 3786–3798.

Jie Zhu, Junhui Li, Muhua Zhu, Longhua Qian, Min Zhang, and Guodong Zhou. 2019. Modeling graph structure in transformer for better amr-to-text generation. In *Proceedings of EMNLP-IJCNLP*, pages 5459–5468.

Jinhua Zhu, Yingce Xia, Lijun Wu, Di He, Tao Qin, Wengang Zhou, Houqiang Li, and Tie-Yan Liu. 2020. Incorporating BERT into neural machine translation. In *Proceedings of ICLR*.