Abstract

As Abstract Meaning Representation (AMR) implicitly involves compound semantic annotations, we hypothesize auxiliary tasks which are semantically or formally related can better enhance AMR parsing. With carefully designed control experiments, we find that 1) Semantic role labeling (SRL) and dependency parsing (DP), would bring much more significant performance gain than unrelated tasks in the text-to-AMR transition. 2) To make a better fit for AMR, data from auxiliary tasks should be properly “AMRized” to PseudoAMR before training. 3) Intermediate-task training paradigm outperforms multitask learning when introducing auxiliary tasks to AMR parsing. From an empirical perspective, we propose a principled method to choose, reform, and train auxiliary tasks to boost AMR parsing. Extensive experiments show that our method achieves new state-of-the-art performance on in-distribution, out-of-distribution, and few-shots benchmarks of AMR parsing.

1 Introduction

Abstract Meaning Representation (AMR) (Banarescu et al., 2013) parsing aims to translate a sentence to a directed acyclic graph, which represents the relations among abstract concepts as shown in Figure 1. AMR can be applied to many downstream tasks, such as information extraction (Rao et al., 2017; Wang et al., 2017; Zhang and Ji, 2021), text summarization (Liao et al., 2018; Hardy and Vlachos, 2018) question answering (Mitra and Baral, 2016; Sachan and Xing, 2016) and dialogue modeling (Bonial et al., 2020).

Recently, AMR Parsing with the sequence-to-sequence framework achieves most promising results (Xu et al., 2020; Bevilacqua et al., 2021). Comparing with transition-based or graph-based methods, sequence-to-sequence models do not require tedious data processing and is naturally compatible with auxiliary tasks (Xu et al., 2020) and powerful pretrained encoder-decoder models (Bevilacqua et al., 2021). Previous work (Xu et al., 2020; Wu et al., 2021) has shown that the performance of AMR parser can be effectively boosted through co-training with certain auxiliary tasks, e.g. Machine Translation or Dependency Parsing.

However, when introducing auxiliary tasks to enhance AMR parsing, we argue that three important issues still remain under-explored in the previous work. 1) How to choose auxiliary task? The task selection is important since loosely related tasks may even impede the AMR parsing according to Damonte and Monti (2021). However, in literature there are no principles or consensus on how to choose the proper auxiliary tasks for AMR parsing. Though previous work achieves noticeable performance gain through multi-task learning, they do not provide explainable insights on why certain task outperforms others or in which aspects the auxiliary tasks benefit the AMR parser. 2) How to bridge the gap between tasks? The form and semantic gaps between AMR parsing and auxiliary tasks are non-negligible. For example, Machine Translation generates text sequence while Dependency Parsing (DP) and Semantic Role Labeling
we introduce dependency parsing, a syntactic parsing task. Traditionally, SRL produces dependency trees and semantic role forests, respectively. The structural differences between DP, SRL and AMR are visualized in Figure 1. Many prior studies (Xu et al., 2020; Wu et al., 2021; Damonte and Monti, 2021) do not attach particular importance to the gap, which might lead the auxiliary tasks to be what is called an outlier-task (Zhang and Yang, 2021; Cai et al.) in the Multitask Learning, deteriorating the performance of AMR parsing. 3) How to introduce auxiliary tasks more effectively? After investigating different training paradigms to combine the auxiliary task training with the major objective (AMR parsing), we figure out that, although all baseline models (Xu et al., 2020; Wu et al., 2021; Damonte and Monti, 2021) choose to jointly train the auxiliary tasks and AMR parsing with Multi-task Learning (MTL), Intermediate-task Learning (ITL) is a more effective way to introduce the auxiliary tasks. Our observation is also consistent with (Pruksachatkun et al., 2020; Poth et al., 2021), which improve other NLP tasks with enhanced pretrained models.

In response to the above three issues, we summarize a principled method to select, transform and train the auxiliary tasks (Figure 2) to enhance AMR parsing from extensive experiments. 1) Auxiliary Task Selection. We choose auxiliary tasks by estimating their similarities with AMR from the semantics and formality perspectives. AMR is recognized as a deep semantic parsing task which encompasses multiple semantic annotations, e.g. semantic roles, name entities and co-references. As a direct semantic-level sub-task of AMR parsing, we select SRL as one auxiliary task. Traditionally, formal semantics views syntactic parsing a precursor to semantic parsing, leading to the mapping between syntactic and semantic relations. Hence we introduce dependency parsing, a syntactic parsing task as another auxiliary task. 2) AMRization. Despite highly related, the output formats of SRL, DP and AMR are distinct from each other. To this end, we introduce transformation rules to “AMRize” SRL and DP to PseudoAMR, intimating the feature of AMR. Specifically, through Reentrancy Restoration we transform the structure of SRL to a graph and restore the reentrancy within arguments, which mimics AMR structure. Through Redundant Relation Removal we conduct transformation in dependency trees and remove relations that are far from semantic relations in AMR graph. 3) Training Paradigm Selection. We find that ITL makes a better fit for AMR parsing than MTL since it allows model progressively transit to the target task instead of learning all tasks simultaneously, which benefits knowledge transfer (Zhang and Yang, 2021).

We summarize our contributions as follows:

1. Semantically or formally related tasks, e.g., SRL and DP, are better auxiliary tasks for AMR parsing compared with distantly related tasks, e.g. machine translation and machine reading comprehension.

2. We propose task-specific rules to AMRize the structured data to PseudoAMR. SRL and DP with properly transformed output format further improve AMR parsing.

3. ITL outperforms classic MTL methods when introducing auxiliary tasks to AMR Parsing. We show that ITL derives a steadier and better converging process during training.

Extensively experiments show that our method (PseudoAMR + ITL) achieves the new state-of-the-art of single model on in-distribution (85.1 Smatch score on AMR 2.0, 83.9 on AMR 3.0), out-of-distribution and few-shots benchmarks. Specifically we observe that AMR parser gains larger improvement on the SRL(+3.3), Reentrancy(+3.1) and NER(+2.0) metrics\(^1\), due to higher resemblance with the selected auxiliary tasks.

2 Methodology

As shown in Figure 2, in this paper, we propose a principled method to select auxiliary tasks (Section 2.1), AMRize them into PseudoAMR (Section 2.2) and train PseudoAMR and AMR effectively (Section 2.3) to boost AMR parsing. We formulate both PseudoAMR and AMR parsing as the

\(^1\)Computed on AMR 2.0 and 3.0 dataset.
sequence-to-sequence generation problem. Given a sentence \( x = [x_1]_{1 \leq i \leq N} \), the model aims to generate a linearized PseudoAMR or AMR graph \( y = [y_i]_{1 \leq i \leq M} \) (the right part of Figure 3) with a product of conditional probability:

\[
P(y) = \prod_{i=1}^{M} p(y_i | (y_1, y_2, \ldots, y_{i-1}))
\]

2.1 Auxiliary Task Selection

When introducing auxiliary tasks for AMR parsing, the selected tasks should be formally or semantically related to AMR, thus the knowledge contained in them can be transferred to AMR parsing. Based on this principle of relevance, we choose semantic role labeling (SRL) and dependency parsing (DP) as our auxiliary tasks.

Semantic Role Labeling  SRL aims to recover the predicate-argument structure of a sentence, which can enhance AMR parsing, because: (1) Recovering the predicate-argument structure is also a sub-task of AMR parsing. As illustrated in Figure 3(a,b), both AMR and SRL locate the predicates (‘want’, ‘leave’) of the sentence and conduct word-sense disambiguation. Then they both capture the multiple arguments of center predicate. (2) SRL and AMR are known as shallow and deep semantic parsing, respectively. It is reasonable to think that the shallow level of semantic knowledge in SRL is useful for deep semantic parsing.

Dependency Parsing  DP aims to parse a sentence into a tree structure, which represents the dependency relation among tokens. The knowledge of DP is useful for AMR parsing, since: (1) Linguistically, DP (syntactic parsing task) can be the precursor task of AMR (semantic parsing). (2) The dependency relation of DP is also related to semantic relation of AMR, e.g., as illustrated in Figure 1(c), ‘NSUBJ’ in DP usually represents ‘ARG0’ in AMR. Actually, they both correspond to the agent-patient relations in the sentence. (3) DP is similar to AMR parsing from the perspective of edge prediction, because both of them need to capture the relation of nodes (tokens/concepts) in the sentence.

2.2 AMRization

Although SRL and DP are highly related to AMR parsing, there still exists gaps between them, e.g., SRL annotations may be disconnected, while AMR is always a connected graph. To bridge these gaps, we transform them into PseudoAMR, which we call AMRization.

2.2.1 Transform SRL to PseudoAMR

We summarize typical gaps between SRL and AMR as: (1) Connectivity. AMR is a connected directed graph while the structure of SRL is a forest. (2) Span-Concept Gap. Nodes in AMR graph represent concepts (e.g., ‘boy’) while that of SRL are token spans (e.g., ‘the boy’, ‘that boy’). Actually all the mentioned token spans correspond to the same concept. (3) Reentrancy. Reentrancy is an important feature of AMR as shown in Figure 3(a), the instance boy is referenced twice as ARG0. The feature can be applied to conduct coreference reso-
lution. However, there is no reentrancy in SRL. To bridge such gaps, we propose Connectivity Formation, Argument Reduction and Reentrancy Restoration to transform SRL into PseudoAMR.

Connectivity Formation To address the connectivity gap, we need to merge all SRL trees into a connective graph. As shown in Figure 3(b-1), we first add a virtual root node, then generating a directed edge from the virtual root to each root of SRL trees, thus the SRL annotation becomes a connected graph.

Argument Reduction To address the Span-Concept Gap, as shown in Figure 3(b-2), if the argument of current predicate is a span with more than one token, we will replace this span with its head token in its dependency structure. Thus token spans “the boy”, “that boy” will be transformed to “boy”, more similar to the corresponding concept. Similar methods have been to applied by (Zhang et al., 2021) to find the head of token spans of argument.

Reentrancy Restoration For the reentrancy gap, we design a heuristic algorithm based on DFS to restore reentrancy in SRL. As shown in Figure 3(b-3), the core idea of the restoration is that we create a variable when the algorithm first sees a node. Next time if meeting node with the same name, the destination of the edge will be referenced to the same variable we have created at first. Please refer to Appendix A for the pseudo code of the reentrancy restoration.

2.2.2 Transform Dependency Structure to PseudoAMR
Firstly, we summarize gaps between Dependency Tree and AMR as: (1) Redundant Relation. Some relations in dependency parsing focuses on syntax, e.g., ‘:PUNCT’ and ‘:DET’, which are far from semantic relations in AMR. (2) Token-Concept Gap. The basic element of dependency structure is token while that of AMR is concept, which captures deeper syntax-independent semantics. Then, we use Redundant Relation Removal and Token Lemmatization to transform the dependency structure to PseudoAMR to handle these gaps.

Redundant Relation Removal For the Redundant Relation Gap, we remove some relations which are far from the sentence’s semantics most of the time, such as “PUNCT” and “DET”. As illustrated in Figure 3(c-1), by removing some relations of the dependence, the parsing result become more compact compared with original DP tree, forcing the model to ignore some semantics-unrelated tokens during seq2seq training.

Token Lemmatization As shown in Figure 3(c-2), for Token-Concept Gap, we conduct lemmatization on the node of dependency tree based on the observation that the affixes of single word do not affect the concept it corresponds to. Together with the smart-initialization (Bevilacqua et al., 2021) by setting the concept token’s embedding as the average of the subword constituents, the embedding vector of lemmatized token (‘want’) becomes closer to the vector concept (‘want-01’) in the embedding matrix, therefore requiring the model to capture deeper semantic when conducting DP task.

2.2.3 Linearization
After all AMRization steps, the graph structure of SRL/DP also should be linearized before doing seq2seq training. As depicted in the right part of Figure 3, we linearize the graph by the DFS-based travel, and use special tokens $<R0>$, ..., $<Rk>$ to indicate variables, and parentheses to mark the depth, which is the best AMR linearization method of Bevilacqua et al. (2021).

2.3 Training Paradigm Selection
After task selection and AMRization, we still need to choose an appropriate training paradigm to train PseudoAMR and AMR effectively. We explore three training paradigms as follows:

Multitask training Following Xu et al. (2020); Damonte and Monti (2021), we use classic schema in sequence-to-sequence multitask training by adding special task tag at the beginning of input sentence and training all tasks simultaneously. The validation of best model is conducted only on the AMR parsing sub-task.

Intermediate training Similar to Pruksachatkun et al. (2020), we first fine-tune the pretrained model on the intermediate task (PseudoAMR parsing), followed by fine-tuning on the target AMR parsing task under same training setting.

Multitask & Intermediate training We apply a joint paradigm to further explore how different paradigms affect AMR parsing. We first conduct multitask training, followed by fine-tuning on AMR parsing. Under this circumstance, Multitask training plays the role as the intermediate task.
We use the Smatch scores (Cai and Knight, 2013) to evaluate the performance. "Topological-Related" and "Topology-related" scores care more about concept centered prediction. We use NoWSD and Wikification are listed as isolated scores because NoWSD is a simplified version of Smatch by removing all word senses thus it’s highly correlated with Smatch score and wikification relies on external entity linker in our experiments, which doesn’t faithfully reflect the parsing models’ ability.

### 3.3 Experiment Setups

**Model Setting** We use current state-of-the-art Seq2Seq AMR Paring model SPRING (Bevilacqua et al., 2021) as our main baseline model and apply BART-Large (Lewis et al., 2020) as our pre-trained model. Blink (Li et al., 2020) is used to add wiki tags to the predicted AMR graphs. We do not apply re-category methods and other post-processing methods are the same with Bevilacqua et al. (2021) to restore AMR from token sequence. We use RAadam (Liu et al., 2019) as our optimizer, and the learning rate is $3e^{-5}$. Batch-size is set to 2048 tokens with 10 steps accumulation.

**AMRization Setting** For SRL, we explore four AMRization settings. 1) Trivial Linearization. Concept :multi-sentence and relation :snt are used to represent the virtual root and it’s edges to each of the SRL trees. 2) With Argument Reduction. We use dependency parser from Stanford CoreNLP Toolkit (Manning et al., 2014) to do the argument reduction. 3) With Reentrancy Restoration 4) All techniques.

For DP, we apply four AMRization settings 1) Trivial Linearization. Extra relations in dependency tree are added to the vocabulary of BART 2) With Lemmatization. We use NLTK (Bird, 2006) to conduct token lemmatization 3) With Redundant Relation Removal. We remove PRECT, DET and ROOT relations in the main experiments 4) All techniques.

---

**Table 1:** The SMATCH scores fine-grained F1 scores on the AMR 2.0 and 3.0. We report results of the model with all AMRization methods applied for both DP and SRL here. $^E$ denotes result given by the ensemble of one model from different checkpoints. Model with $^*$ denotes contemporary work.

<table>
<thead>
<tr>
<th>Model</th>
<th>Extra Data</th>
<th>SMATCH</th>
<th>NoWSD</th>
<th>Wiki</th>
<th>Conc.</th>
<th>NER</th>
<th>Neg.</th>
<th>Unll.</th>
<th>Reen.</th>
<th>SRL</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMR 2.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cai and Lam (2020)</td>
<td>N</td>
<td>78.7</td>
<td>79.2</td>
<td>81.3</td>
<td>88.1</td>
<td>87.1</td>
<td>66.1</td>
<td>81.5</td>
<td>63.8</td>
<td>74.5</td>
</tr>
<tr>
<td>Fernandez Astudillo et al. (2020)</td>
<td>N</td>
<td>80.2</td>
<td>80.7</td>
<td>78.8</td>
<td>88.1</td>
<td>87.5</td>
<td>64.5</td>
<td>84.2</td>
<td>70.3</td>
<td>78.2</td>
</tr>
<tr>
<td>Zhou et al. (2021a)</td>
<td>70k</td>
<td>81.8</td>
<td>82.3</td>
<td>78.8</td>
<td>88.7</td>
<td>86.5</td>
<td>69.7</td>
<td>85.5</td>
<td>71.1</td>
<td>80.8</td>
</tr>
<tr>
<td>SPRING (Bevilacqua et al., 2021)</td>
<td>N</td>
<td>83.8</td>
<td>84.4</td>
<td>84.3</td>
<td>90.2</td>
<td>90.6</td>
<td>74.4</td>
<td>86.1</td>
<td>70.8</td>
<td>79.6</td>
</tr>
<tr>
<td>Ours (w/ DP)</td>
<td>200k</td>
<td>84.3</td>
<td>84.8</td>
<td>83.1</td>
<td>90.8</td>
<td>90.5</td>
<td>73.6</td>
<td>86.7</td>
<td>72.4</td>
<td>80.5</td>
</tr>
<tr>
<td>Ours (w/ SRL)</td>
<td>40k</td>
<td>85.0</td>
<td>85.4</td>
<td>84.1</td>
<td>90.4</td>
<td>92.5</td>
<td>74.7</td>
<td>88.2</td>
<td>74.7</td>
<td>83.1</td>
</tr>
<tr>
<td>*SPRING $^E$ (Lam et al., 2021)</td>
<td>200k</td>
<td>84.2</td>
<td>84.7</td>
<td>82.8</td>
<td>90.0</td>
<td>90.8</td>
<td>72.7</td>
<td>87.4</td>
<td>74.3</td>
<td>82.9</td>
</tr>
<tr>
<td>*Graphene 4S</td>
<td>200k</td>
<td>84.8</td>
<td>85.3</td>
<td>83.9</td>
<td>90.6</td>
<td>92.2</td>
<td>75.2</td>
<td>88.0</td>
<td>71.4</td>
<td>83.5</td>
</tr>
<tr>
<td>*Structure-aware$^E$ (Zhou et al., 2021b)</td>
<td>47k</td>
<td>84.9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ours (w/ SRL)$^E$</td>
<td>40k</td>
<td>85.3</td>
<td>85.7</td>
<td>83.9</td>
<td>90.7</td>
<td>92.2</td>
<td>75.0</td>
<td>88.4</td>
<td>75.0</td>
<td>83.6</td>
</tr>
</tbody>
</table>

| AMR 3.0                |            |        |       |      |       |     |      |       |       |     |
| Bevilacqua et al. (2021) (w/ silver) | 200k       | 83.0   | 83.5  | 82.7 | 89.8  | 87.2| 73.0 | 85.4  | 70.4  | 78.9|
| Ours (w/ DP)           | 40k        | 83.9   | 84.3  | 81.6 | 89.7  | 89.2| 73.0 | 87.0  | 73.7  | 82.3|
| Ours (w/ SRL)          | 40k        | 83.9   | 84.3  | 81.0 | 89.7  | 88.4| 73.9 | 87.0  | 73.9  | 82.5|
| *SPRING $^E$ (Lam et al., 2021) | 200k       | 83.2   | 83.7  | 81.2 | 89.4  | 87.2| 72.9 | 86.4  | 73.3  | 82.0|
| *Graphene 4S           | 200k       | 83.8   | 84.2  | 81.9 | 90.1  | 88.3| 74.6 | 86.9  | 70.2  | 82.5|
| *Structure-aware$^E$ (Zhou et al., 2021b)  | 47k        | 83.1   |       |      |       |     |      |       |       |     |
| Ours (w/ SRL)$^E$      | 40k        | 84.0   | 84.5  | 80.7 | 90.0  | 88.9| 73.1 | 87.1  | 73.9  | 82.6|

We conduct experiments on two AMR dataset, AMR 2.0 and AMR 3.0. AMR 2.0 contains 36521, 1368 and 1371 sentence-AMR pairs in training, validation and testing sets, respectively. AMR 3.0 has 55635, 1722 and 1898 sentence-AMR pairs for training validation and testing set, respectively. We also conducted experiments in out-of-distribution and few-shots setting.

### 3.2 Evaluation Metrics

We use the Smatch scores (Cai and Knight, 2013) and further the break down scores (Damonte et al., 2017) to evaluate the performance.

To fully understand the aspects where auxiliary tasks improve AMR parsing, we divide the fine-grained scores to two categories: **Concept-Related** including Concept, NER and Negation scores, which care more about concept centered prediction. **Topology-Related** including Unlabeled, Reentrancy and SRL scores, which focus on edge and relation prediction. Note that NoWSD and Wikification are listed as isolated scores because NoWSD is a simplified version of Smatch by removing all word senses thus it’s highly correlated with Smatch score and wikification relies on external entity linker in our experiments, which doesn’t faithfully reflect the parsing models’ ability.
3.4 Main Results

We report the result (ITL + All AMRization Techniques) on benchmark AMR 2.0 and 3.0 in Table 1. On AMR 2.0, our models with DP or SRL as intermediate task gains consistent improvement over the SPRING model by a large margin (1.3 Smatch) and reach new state-of-the-art for single model (85.1 Smatch). Compared with SPRING with 200k extra data, our models achieve higher performance with much less extra data (40k v.s. 200k), suggesting the effectiveness of our intermediate tasks. We also compare our models with contemporary work\(^2\) (Lam et al., 2021; Zhou et al., 2021b). It turns out that our ensemble model beats its counterpart with less extra data, reaching a higher performance (85.3 Smatch). In fact, even without ensembling, our model still performs better than those ensembling models, showing the effectiveness of our methods.

On AMR 3.0, Our models consistently outperform other models under both single model (83.9 Smatch) and ensembling setting (84.0 Smatch). Same as AMR 2.0, our single model reaches higher Smatch score than those ensembling models, revealing the effectiveness of our proposed methods.

**Fine-grained Performance** To better analyse what skills the AMR parser gains from the intermediate training and how different intermediate task affect the models’ performance. We report the fine-grained score as shown in Table 1. We can tell that by incorporating intermediate tasks, our models provide better predictions on most sub-metrics especially on the Topology-related terms. On both AMR 2.0 and 3.0 our single model with SRL as intermediate task reaches highest score in Unlabeled, Reentrancy and SRL, suggesting that SRL intermediate task improves our parser’s capability in Coreference and SRL.

DP leads to consistent improvement in topology-related terms. It also gives best result on NER subtask (92.5 on AMR 2.0, 89.2 on AMR 3.0), we suppose that the “:nn” relation which annotates multiword names in dependency parsing helps the AMR parser recognize multiword name-entities.

Overall, above from the Smatch scores, AMR parser gains large improvement in Topology-related subtasks and NER by incorporating our intermediate tasks.

---
\(^2\)We do not report score of Graphene All (Lam et al., 2021), since it aggregates 7 models from different architectures, it could reach higher performance if involving our model.

<table>
<thead>
<tr>
<th>Model</th>
<th>Extra</th>
<th>SMATCH</th>
<th>Conc.</th>
<th>Topo.</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPRING</td>
<td>N</td>
<td>83.8</td>
<td>85.1</td>
<td>78.8</td>
</tr>
<tr>
<td>Ours (w/ NLG)</td>
<td>N</td>
<td>85.0</td>
<td>85.9</td>
<td>82.0</td>
</tr>
<tr>
<td>- w/ DialogSum 13k</td>
<td></td>
<td>84.5</td>
<td>85.5</td>
<td>81.7</td>
</tr>
<tr>
<td>- w/ CNNDM 40k</td>
<td></td>
<td>84.4</td>
<td>84.7</td>
<td>81.5</td>
</tr>
<tr>
<td>- w/ DE-EN 40k</td>
<td></td>
<td>84.0</td>
<td>85.0</td>
<td>80.8</td>
</tr>
<tr>
<td>- w/ EN-DE 40k</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ours (w/ Parsing)</td>
<td>N</td>
<td>85.1</td>
<td>85.8</td>
<td>82.2</td>
</tr>
<tr>
<td>- w/ DP 40k</td>
<td></td>
<td>85.0</td>
<td>85.9</td>
<td>82.0</td>
</tr>
<tr>
<td>- w/ SRL 40k</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Result of Task Selection. We also report the average scores of Concept-related scores (Conc.) and Topology-related scores (Topo.)

![Figure 4: The distance distribution of sentences representation. SRL and DP consistently provide more similar sentence representation to AMR than Translation. The computation is illustrated in Figure 6 in appendix.](image)

4 Analysis

4.1 Exploration in Auxiliary Task Selection

Apart from DP and SRL, we also explore how different tasks affect AMR parsing. We involve two classic conditional NLG tasks, Summarization and Translation for comparison. The result is shown in Table 2. Xu et al. (2020) show that compared with syntax parsing tasks, Machine Translation is a better pretraining task for AMR parsing since it captures more semantic information rather than syntax. However, according to our research, compared to DP and SRL, the benefit of Machine Translation faded under the intermediate setting with equal number of data. We argue that the volume of gold data might be the reason for MT to outperform in Xu et al. (2020)’s work. The EN-DE task even leads to a negative result in Concept-related terms. It indicates that the intermediate task should be close to AMR parsing in form or it might lead to a sub-optimal initialization for target task.

To better understand how different auxiliary tasks affect AMR parsing, we collect the sentences’ representation from different tasks’ trained encoders. We use the average hidden state of the encoder’s output as the sentence representation. We
Table 3: We report the average scores of Concept-related scores and Topology-related scores. The full scores are listed in Table 7. The improvement of involving all techniques against trivial linearization is significant with p < 0.005 for both SRL and DP.

Table 4: Analysis on Training Paradigms. Intermediate-task training is more suitable for AMR parsing than Multitask training.

Table 5: The loss curve on development set of AMR 2.0 for different training paradigms.

4.2 Ablation Study on AMRization Methods

As shown in Table 3, we conduct ablation study on how different AMRization methods affect the performance AMR parsing. For both SRL and DP, jointly adopting our AMRization techniques can further improve the performance of AMR parsing significantly, comparing to trivial linearization.

As shown in Table 7, compared with jointly using the two techniques, it is worth noting that model with solely Reentrancy Restoration reaches highest fine-grained scores in concept-related and topology-related terms especially on Reentrancy and SRL scores. We analyse the number of restored reentrancy, the result shows that about 10k more reentrancies are added when Argument Reduction (AR) is previously executed. It’s expected since AR replaces the token span to the root token. Compared with token span, single token is more likely to be recognized as the reference variable according to the Reentrancy Restoration (RR) algorithm, thus generating more reentrancy, which might include bias to the model. This explains why solely using RR can lead to better results on SRL and Reen.

4.3 ITL Outweighs MTL

We report the result of different fine-tuning paradigms in Table 4. It justifies our assumption that classic multitask learning with task tag as previously applied in Xu et al. (2020); Damonte and Monti (2021) does not compare with intermediate training paradigm for AMR Parsing task.

As shown in Figure 5, Intermediate-task training provides a faster and better converging process than MTL. We claim that this is because there is big gap from AMR parsing to other tasks both in difficulty and form, which harms the process of MTL. The process of optimizing all auxiliary tasks simultaneously will bias our target task AMR Parsing.

4.4 Exploration in Out-of-Distribution Generalization

Following Bevilacqua et al. (2021); Lam et al. (2021), we assess the performance of our models when trained on out-of-distribution (OOD) data. The models trained solely on AMR 2.0 training data are used to evaluate out-of-distribution performance on the BIO, the TLP and the News3 dataset.
which has 36521 training samples, the number of Appendix C. Compared with the AMR2.0 dataset we can achieve better results on unseen domains. (Banarescu et al., 2013) (e.g., semantic relations, (about 25 Smatch) in the BOLT dataset which is Table 6 reports the result. Surprisingly, our model surpasses the SPRING model by a real large margin (about 25 Smatch) in the BOLT dataset which is the most insufficient in data and gains a consistent improvement on all datasets, suggesting that our method is effective under low resources conditions.

4.5 Exploration in Few-Shots Learning

Since the annotation of AMR is both time and labor consuming, it raises our interests if we can improve the few-shots learning ability of AMR Parser. We propose three Few-Shots Learning benchmarks BOLT, LORELEI, DFA for AMR parsing based on the different sufficient degree of training examples. Detail of the datasets is described in Appendix C. Compared with the AMR2.0 dataset which has 36521 training samples, the number of training samples in BOLT, LORELEI, DFA are 2.9%, 12.2% and 17.7% of the number of AMR2.0. Table 6 reports the result. Surprisingly, our model surpasses the SPRING model by a real large margin (about 25 Smatch) in the BOLT dataset which is the most insufficient in data and gains a consistent improvement on all datasets, suggesting that our method is effective under low resources conditions.

5 Related Work

AMR Parsing AMR parsing is a challenging task, since AMR is a deep semantic representation and consists of many separate annotations (Banarescu et al., 2013) (e.g., semantic relations, named entities, co-reference and so on). There are four major methods to do AMR Parsing currently, sequence-to-sequence approaches (Ge et al., 2019; Xu et al., 2020; Bevilacqua et al., 2021), tree-based approaches (Zhang et al., 2019a), graph-based approaches (Lyu and Titov, 2018; Cai and Lam, 2020) and transition-based approaches (Naseem et al., 2019; Lee et al., 2020; Zhou et al., 2021a). There are many work introducing auxiliary task to AMR Parsing. (Goodman et al., 2016) builds AMR graph from dependency trees. Xu et al. (2020) introduces Machine Translation, Constituency Parsing as pretraining tasks for Seq2Seq AMR parsing. Wu et al. (2021) introduces Dependency Parsing for transition-based AMR parsing. However all of them do not take care of the semantic and formal gap between the auxiliary tasks and AMR parsing.

Among the AMR parsing approaches, currently sequence-to-sequence methods achieve most promising result in AMR parsing (Bevilacqua et al., 2021), and have better generalization in OOD data, since they do not need a complex, content-specific pre- and post-process pipelines. Therefore, in this paper, our exploration focuses on enhancing the sequence-to-sequence AMR parser.

Multitask & Intermediate-task Learning Multi-task Learning (MTL) (Caruana, 1997) aims to jointly train multiple related tasks to improve the performance of all tasks. Different from MTL, Intermediate-task Learning (ITL) is proposed to enhance pretrained models eg. BERT by training on intermediate task before fine-tuning on the target task. Recent studies (Praksachatkun et al., 2020; Poth et al., 2021) on ITL expose that choosing right intermediate tasks is important for the target task. Tasks that don’t match might even bring negative effect to the target. However, there is no consensus on how to choose right intermediate task and ITL’s influence on pretrained Seq2Seq models is still under-explored.

6 Conclusion

In this paper, We find that semantically or formally related tasks, e.g. SRL and DP are better auxiliary tasks for AMR parsing and can further improve the performance by proper AMRization methods to bridge the gap between tasks. And Intermediate-task Learning is more effective in introducing auxiliary tasks compared with Multitask Learning. Extensive experiments and analyses show the effectiveness and priority of our proposed methods.
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 interopera- 
tion with discourse, pages 178–186.

Michele Bevilacqua, Rethna Blloshmi, and Roberto Navigli. 2021. One spring to rule them both: Sym-
metric amr semantic parsing and generation without a complex pipeline. In Proceedings of the Thirty-
Fifth AAAI Conference on Artificial Intelligence.

Interactive Presentation Sessions, pages 69–72, Sydney, Australia. Association for Computational Lin-
guistics.


Deng Cai and Wai Lam. 2020. AMR parsing via graph-
sequence iterative inference. In Proceedings of the 58th Annual Meeting of the Association for Compu-

Shu Cai and Kevin Knight. 2013. Smatch: an evalu-
ation metric for semantic feature structures. In Pro-
cedings of the 51st Annual Meeting of the Associa-
tion for Computational Linguistics (Volume 2: Short 
Papers), pages 748–752, Sofia, Bulgaria. Association 
for Computational Linguistics.


Learning, 28(1):41–75.

summarization dataset. In Findings of the Association for Computational Linguistics: ACL-IJCNLP 

Marco Damonte, Shay B. Cohen, and Giorgio Satta. 2017. An incremental parser for Abstract Mean-
ing Representation. In Proceedings of the 15th Con-
ference of the European Chapter of the Association 
for Computational Linguistics: Volume 1, Long Pa-
ers, pages 536–546, Valencia, Spain. Association 
for Computational Linguistics.

Marco Damonte and Emilio Monti. 2021. One semantic 
parsor to parse them all: Sequence to sequence 
multi-task learning on semantic parsing datasets. In Proceeding of SEM 2021: The Tenth Joint Con-
ference on Lexical and Computational Semantics, 
pages 173–184, Online. Association for Computa-
tional Linguistics.

Ramón Fernandez Astudillo, Miguel Ballesteros, 
Tahira Naseem, Austin Blodgett, and Radu Flor-
ian. 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.

DongLai Ge, Junhui Li, Muhua Zhu, and Shoushan Li. 2019. Modeling source syntax and semantics for 
amr amr parsing. In IJCAI, pages 4975–4981.

James Goodman, Andreas Vlachos, and Jason Narad-
owsky. 2016. Noise reduction and targeted explo-
ration in imitation learning for Abstract Meaning 
Representation parsing. In Proceedings of the 54th Annual Meeting of the Association for Compu-
tational Linguistics (Volume 1: Long Papers), pages 1– 
11, Berlin, Germany. Association for Computational Linguistics.

Hardy Hardy and Andreas Vlachos. 2018. Guided neu-
ral language generation for abstractive summariza-
tion using abstract meaning representation. In Pro-
cedings of the 2018 Conference on Empirical Meth-
ods in Natural Language Processing, pages 768– 
773.

Karl Moritz Hermann, Tomáš Kociský, Edward Greffen-
stette, Lasse Espeholt, Will Kay, Mustafa Suleyman, 
and Phil Blunsom. 2015. Teaching machines to read 
and comprehend. In NIPS.

Hoang Thanh Lam, Gabriele Picco, Yufang Hou, 
Young-Suk Lee, Lam M. Nguyen, Dzung T. Phan, 
Vanessa Lópe, and Ramon Fernandez Astudillo. 2021. Ensemble graph predictions for amr pars-
ing.

Young-Suk Lee, Ramón Fernandez Astudillo, Tahira 
Naseem, Revanth Gangi Reddy, Radu Florian, and 
Salim Roukos. 2020. Pushing the limits of amr pars-
Conference on Empirical Methods in Natural Lan-
guage Processing: Findings, pages 3208–3214.

Mike Lewis, Yinhan Liu, Naman Goyal, Mar-
jan Ghazvininejad, Abdelrahman Mohamed, Omer 
2020. BART: Denoising sequence-to-sequence pre-
training for natural language generation, translation, 
and comprehension. In Proceedings of the 58th An-
nual Meeting of the Association for Computational 
Linguistics, pages 7871–7880, Online. Association 
for Computational Linguistics.

Belinda Z. Li, Sewon Min, Srinivasan Iyer, Yashar 
Mehdad, and Wen-tao Yih. 2020. Efficient one-pass 
end-to-end entity linking for questions. In Pro-
cedings of the 2020 Conference on Empirical Methods
in Natural Language Processing (EMNLP), pages 6433–6441, Online. Association for Computational Linguistics.


A Algorithms

**Algorithm 1** Reentrancy Restoration for SRL

**Input:** Treenode:T  
**Output:** Graph:G

**Description:** T is root node of the original SRL after node ROOT is added to form tree structure. G is the output graph with possible reentrancy restored.

**Global Variables:** Dict: V={}. Here Dict is the official data structure of Python’s dictionary.

```python
for predicate in T.sons do
    for son in predicate.sons() do
        if son.name in V.keys() then
            son = V[son.name]
        else
            V[son.name] = son
return T
```

B Auxiliary Datasets Description

B.1 Summarization

**CNN/DM** (Hermann et al., 2015) The CNN / DailyMail Dataset is an English-language dataset containing news articles as written by journalists at CNN and the Daily Mail. The dataset is widely accepted as benchmark to test models’ performance of summarizing. We select 40k training data for fair comparison.

**DIALOGS** (Chen et al., 2021) The Real-Life Scenario Dialogue Summarization (DIALOGS), is a large-scale summarization dataset for dialogues. Unlike CNN/DM which focuses on monologue news summarization, DIALOGS covers a wide range of daily-life topics in the form of spoken dialogue. We use all the training data (13k) to conduct the intermediate training.

B.2 Translation

**WMT14 EN-DE** We select 40k training examples from WMT14 EN-DE training set to form EN-DE and DE-EN translation intermediate task.

B.3 Dependency Parsing

**PENN TREEBANK** (Marcus et al., 1999) The Penn Treebank (PTB) project selected 2,499 stories from a three year Wall Street Journal (WSJ) collection of 98,732 stories for syntactic annotation. We only utilize the dependency structure annotations to form our intermediate dependency parsing task. There are 39,832 (~40k) sentences for dependency parsing.

B.4 Semantic Role Labeling

**ONTONOTES** (Weischedel et al., 2017) The OntoNotes project is built on two resources, following the **PENN TREEBANK** (Marcus et al., 1999) for syntax and the **PENN PropBank** for predicate-argument structure. We select 40k sentences with SRL annotations to form intermediate task.

C Few Shots Datasets Description

We propose three Few-Shots Learning benchmark for AMR parsing:

1. **BOLT** Using only the BOLT split of AMR data of AMR2.0 dataset. The training, validation and test data each has 1061, 133 and 133 amrs respectively.

2. **LORELEI** Using only the LORELEI split of AMR data of AMR3.0 dataset. The training, validation and test data each has 4441, 354 and 527 amrs respectively.

3. **DFA** Using only the DFA split of AMR data of AMR2.0 dataset. The training, validation and test data each has 6455, 210 and 229 amrs respectively.

Compared with the AMR2.0 dataset which has 36521 training samples, the number of training samples in BOLT, LORELEI, DFA are 2.9%, 12.2% and 17.7% of the number of AMR2.0.
The boy wants to leave.

Figure 6: Illustration of how to compute sentence representation distance of different tasks. The sentences used for evaluate are never seen in the training of AMR Parsing and other auxiliary tasks. Cosine Similarity is computed the same way. We collect all sentences’ distance of one encoder to draw the Gaussian distribution curve.

<table>
<thead>
<tr>
<th>Model</th>
<th>Extra Data</th>
<th>SMATCH</th>
<th>NoWSD</th>
<th>Wiki</th>
<th>Concept-related</th>
<th>Topology-related</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Conc.</td>
<td>NER</td>
<td>Neg.</td>
<td>Unit.</td>
<td>Reen.</td>
</tr>
<tr>
<td>SPRING (w/ silver) (Boviaacqua et al., 2021)</td>
<td>200k</td>
<td>84.3</td>
<td>84.8</td>
<td>83.1</td>
<td>90.8</td>
<td>90.5</td>
</tr>
<tr>
<td>Ours (w/ Semantic Role Labeling)</td>
<td>40k</td>
<td>84.5</td>
<td>84.9</td>
<td>84.0</td>
<td>90.2</td>
<td>91.8</td>
</tr>
<tr>
<td>- w/ Arg. Reduction (AR)</td>
<td>40k</td>
<td>84.8</td>
<td>85.2</td>
<td>83.9</td>
<td>90.4</td>
<td>92.2</td>
</tr>
<tr>
<td>- w/ Reen. Restoration (RR)</td>
<td>40k</td>
<td>85.0</td>
<td>85.4</td>
<td>83.5</td>
<td>90.6</td>
<td>92.1</td>
</tr>
<tr>
<td>Ours (w/ Dependency Parsing)</td>
<td>40k</td>
<td>85.1</td>
<td>85.6</td>
<td>83.6</td>
<td>90.4</td>
<td>91.4</td>
</tr>
<tr>
<td>- w/ Redundant Relation Removal (RRR)</td>
<td>40k</td>
<td>84.4</td>
<td>84.9</td>
<td>82.9</td>
<td>90.1</td>
<td>90.5</td>
</tr>
<tr>
<td>- w/ Lemmatization (Lemma)</td>
<td>40k</td>
<td>84.5</td>
<td>85.0</td>
<td>83.5</td>
<td>90.2</td>
<td>91.2</td>
</tr>
<tr>
<td>- w/ RRR + Lemma</td>
<td>40k</td>
<td>84.7</td>
<td>85.2</td>
<td>83.8</td>
<td>90.2</td>
<td>91.2</td>
</tr>
</tbody>
</table>

Table 7: Full scores of ablation on AMRization methods.