Modeling Source Syntax and Semantics for Neural AMR Parsing

Donglai Ge[1] , **Junhui Li**[1*] , **Muhua Zhu**[2] and **Shoushan Li**[1]

[1]School of Computer Science and Technology, Soochow University, Suzhou, China
[2]Alibaba Group, Hangzhou, China

20175227014@stu.suda.edu.cn, {lijunhui, lishoushan}@suda.edu.cn,
muhua.zmh@alibaba-inc.com

Abstract

Sequence-to-sequence (seq2seq) approaches formalize Abstract Meaning Representation (AMR) parsing as a translation task from a source sentence to a target AMR graph. However, previous studies generally model a source sentence as a word sequence but ignore the inherent syntactic and semantic information in the sentence. In this paper, we propose two effective approaches to explicitly modeling source syntax and semantics into neural seq2seq AMR parsing.[1] The first approach linearizes source syntactic and semantic structure into a mixed sequence of words, syntactic labels, and semantic labels, while in the second approach we propose a syntactic and semantic structure-aware encoding scheme through a self-attentive model to explicitly capture syntactic and semantic relations between words. Experimental results on an English benchmark dataset show that our two approaches achieve significant improvement of 3.1% and 3.4% F1 scores over a strong seq2seq baseline.

1 Introduction

Recent studies in Abstract Meaning Representation (AMR) parsing regard it as a sequence-to-sequence (seq2seq) mapping problem, where AMR graphs are properly linearized into sequences [Peng *et al.*, 2017; Konstas *et al.*, 2017; van Noord and Bos, 2017]. Nevertheless, most previous works in this direction simply recast source sentences as word sequences, without considering the inherent syntactic and semantic information deemed to be useful for the construction of AMR graphs. In fact, recent progress in seq2seq modeling, such as neural machine translation [Shi *et al.*, 2016; Li *et al.*, 2017], shows that vanilla seq2seq models still fail to learn deep structural details from million-scale parallel corpora, let alone AMR corpora which are of much smaller sizes. As a result, in the absence of linguistic knowledge, seq2seq models for AMR parsing tend to produce results that do not well respect syntax and semantics of the input sentence.

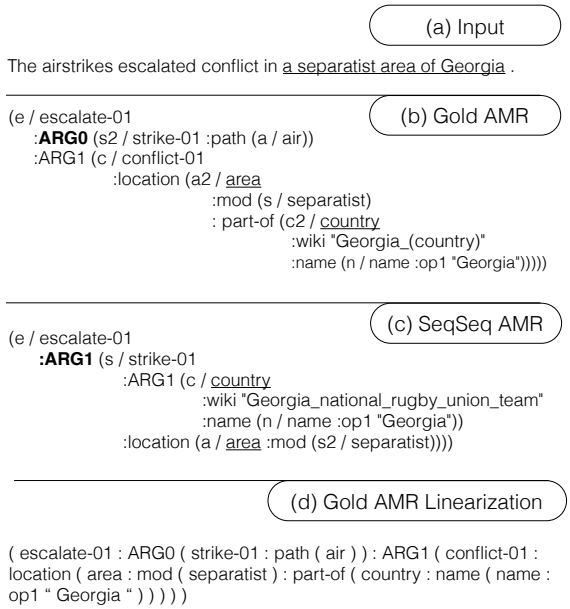

Figure 1: An example of seq2seq-based AMR parsing output (c) and AMR graph linearization (d).

As an example, Figure 1(c) shows the output of a state-of-the-art seq2seq baseline system, which indicates at least the following issues. First, the seq2seq model mistakenly recognizes the segment *a separatist area of Georgia* as two parallel concepts: *country* and *area*. Secondly, it deems both the concepts to be modifiers of the concept *strike*, which offenses the syntactic structure of the source sentence. Finally, it mistakenly views the concept *strike* as ARG1 (i.e., patient) of the concept *escalate*, which is inconsistent with semantic role labeling (SRL) output in which the segment *The airstrikes* acts as ARG0 of predicate *escalated*. By manually examining the AMR parsing outputs of 50 sentences from the development set, we find that 52% of automatic AMRs offense their corresponding source syntactic structures.

In principle, syntactic information provides a useful and promising avenue for AMR parsing, considering the fact that an AMR graph explicitly contains syntactic modifier-head relations between AMR concepts. The idea of incorporating syntactic and semantic information has already been shown

*Corresponding Author

[1]In this paper, we view the output of PropBank semantic role labeling (SRL) as source semantics.

effective in non-seq2seq AMR parsing [Wang *et al.*, 2015a; Wang *et al.*, 2015b; Damonte *et al.*, 2017; Peng *et al.*, 2018]. However, to the best of our knowledge, this idea has not been well explored for neural seq2seq AMR parsing.

In this paper, we focus on improving AMR parsing by modeling source syntax and semantics via a neural seq2seq modeling. Specifically, we exploit syntactic and semantic information for AMR parsing by explicitly taking advantage of linguistic knowledge derived from the outputs of syntactic parsing and SRL on source sentences. In our first approach, we linearize source syntax and/or semantics to obtain a mixed input sequence which consists of words, syntactic labels, and/or semantic labels. This approach is simple yet effective, even not requiring any modifications to existent seq2seq models. In our second approach, we propose syntactic (and/or semantic) structure-aware encoder to explicitly model syntactic (or semantic) structure between word pairs. Hereafter, we call the first approach as a sequence-aware linearization approach and the second approach as a structure-aware encoding approach. Experimental results on an English benchmark dataset show that both the two approaches significantly improve the performance over a strong baseline.

2 AMR Parsing as Neural Seq2Seq Learning

Our baseline system is built on *Transformer*, a state-of-the-art seq2seq model that is originally proposed for neural machine translation and syntactic parsing [Vaswani *et al.*, 2017]. To make the model applicable to AMR parsing, we linearize AMR graphs into sequences in pre-processing and recover AMR graphs from sequences in post-processing.

Sequence-to-Sequence Modeling. Specifically, the encoder in the *Transformer* consists of a stack of multiple identical layers, each of which has two sub-layers, one for 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 sub-layers as in the encoder layers as well as an additional sub-layer that performs multi-head attention to the output of the encoder stack. Experiments on the tasks of machine translation and syntactic parsing show that *Transformer* outperforms RNN-based seq2seq models. See [Vaswani *et al.*, 2017] for more details.

Pre-Processing: 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. For example, Figure 1(d) presents the linearization 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: Target Sequence to AMR Graph. There is no surprise that the obtained translation 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].[2] Note that there exist some other linearization methods proposed in related studies [Konstas *et al.*, 2017; Peng *et al.*, 2017], and our approach can also be applied to them.

3 Modeling Source Syntax and Semantics via Sequence-Aware Linearization

Seq2seq models treat a sentence as a sequence of words and ignore external knowledge, failing to effectively capture various kinds of inherent structure of the sentence. To leverage external knowledge of input sentence, specifically the syntax and semantics, we focus on the syntactic parse tree and semantic roles of a sentence and propose three different parsing models that explicitly consider the syntactic and semantic structures into encoding. Figure 2 shows an example of a source sentence with syntactic parse tree and semantic roles. Though these syntactic and semantic information is tree structured, we follow recent studies [Vinyals *et al.*, 2015; Choe and Charniak, 2016; Li *et al.*, 2017] to linearize the tree structure into a sequence, which can be viewed as an alternative to tree structure.

3.1 Linearizing Source Syntax

Syntactic parsing is a process of mapping a sentence into a syntactic parse tree. Our approach of incorporating syntactic structure information is straightforward. The basic idea here is to convert a source sentence into a syntactic sequence from its parse tree, as exemplified in the Figure 2. Note that there exist many variants in linearization of a syntactic parse tree (e.g., including symbols indicating syntactic ending boundaries or not [Vinyals *et al.*, 2015; Choe and Charniak, 2016; Li *et al.*, 2017]). However, our preliminary experiments showed that the performance gap between these methods is very small. Therefore, we follow [Li *et al.*, 2017] and simply use pre-order traversal to obtain the syntactic sequence, mixed with grammar labels, POS tags as well as words.

+Syntax in Figure 2 shows the syntactic sequence of the example sentence. The new sequence will directly be used as input of our AMR seq2seq system.

3.2 Linearizing Source Semantics

Semantic role labeling [Gildea and Jurafsky, 2002], sometimes also called shallow semantic parsing, is a process that assigns labels to phrases in a sentence that indicate their semantic roles in the sentence, such as that of agent, patient, time, or location, and so on. Given a source sentence, a predicate in it may have multiple semantic roles which usually map to syntactic nodes. There also exist many variants in

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

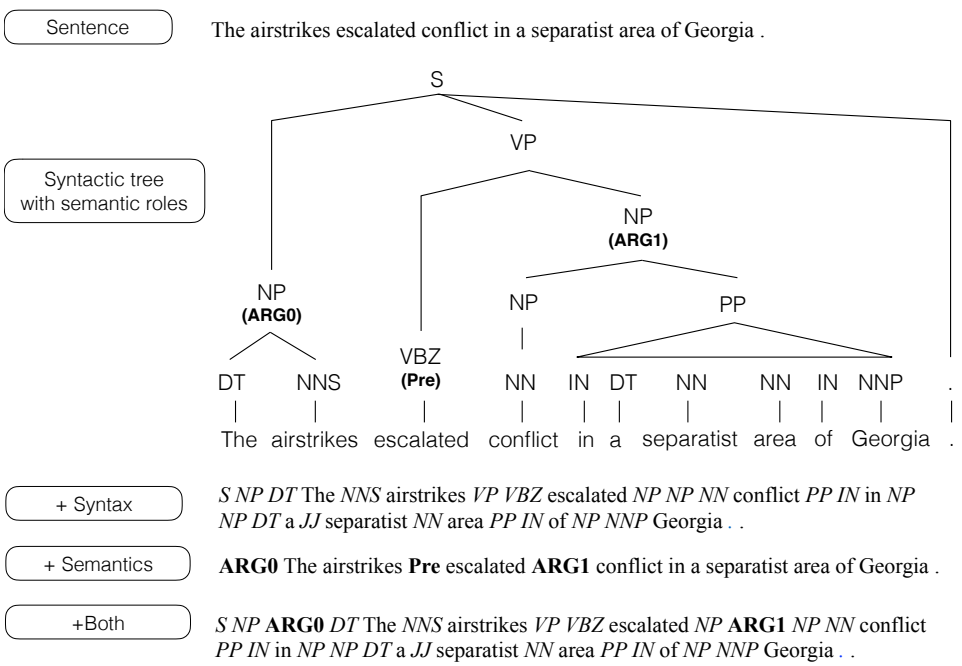

Figure 2: An example of a source sentence with syntactic and semantic structure. To save space, we do not present the detailed syntactic structure for prepositional phrase PP(in a separatist area of Georgia). In the mixed sequences, the syntactic labels are in italic font while the semantic labels are in bold font.

linearization of semantic role labels. To be consistent with the method of linearizing syntactic structure, we insert semantic role labels at the corresponding segment's beginning position. For example, segment *The airstrikes* with *ARG0* semantic role label will be linearized as *ARG0 The airstrikes*. Note that a sentence may have multiple predicates, thus it is frequent that a segment plays same or different semantic roles to two or more predicates. In this case, we linearize the segment with multiple semantic labels. For example, if segment *The airstrikes* plays roles of *ARG0*, *ARG0* and *ARG1* for three left-to-right predicates in one sentence, it will be linearzed as *ARG0 ARG0 ARG1 The airstrikes*.

+*Semantics* in Figure 2 shows the semantic sequence of the example sentence. We also include symbol *Pre* to indicate the position of predicates. Again the new sequence will directly be used as input of our AMR seq2seq system.

3.3 Linearizing Both Source Syntax and Semantics

In order to obtain a sequence with both syntactic and semantic information, we also run the pre-order traversal on the syntactic tree with semantic roles to obtain the linearized version: if a syntactic node is augmented with semantic roles, then it is sequenced as its syntactic label, followed by the semantic role labels. For example, as the syntactic node *NP* in Figure 2 is associated with *ARG0* semantic role, it is linearized as *NP ARG0*. Note that we discard predicate label Pre in the linearization due to its redundancy with verbal POS tags.

+*Both* in Figure 2 shows the sequence with syntax and semantics for the example sentence.

4 Modeling Source Syntax and Semantics via Structure-Aware Encoding

Transformer uses multi-head self-attention which enables the encoder to learn sentence-wide context for every source word. Therefore, to make the encoder be aware of sentence structure, we extend the relation-aware self-attention proposed in [Shaw *et al.*, 2018] to encode structural information.

4.1 Structure-Aware Self-Attention

The conventional self-attention in *Transformer* uses *Scaled Dot-Product Attention* which operates on an input sequence, $x = (x_1, \cdots, x_n)$ of n elements where $x_i \in R^{d_x}$ and computes a new sequence $z = (z_1, \cdots, z_n)$ with the same length:

$$z = Attention\,(x) \tag{1}$$

where $z \in R^{n \times d_z}$. Each output element z_i is calculated as a weighted sum of a linearly transformed input elements:

$$z_i = \sum_{j=1}^{n} \alpha_{ij} \left(x_j W^V\right) \tag{2}$$

where $W^V \in R^{d_x \times d_z}$ is a parameter matrix, and

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{n} exp(e_{ik})} \tag{3}$$

$$e_{ij} = \frac{\left(x_i W^Q\right)\left(x_j W^K\right)^T}{\sqrt{d_z}} \tag{4}$$

where the weight vector $\alpha_i = (\alpha_{i1}, \cdots, \alpha_{in})$ over input vectors is obtained by self-attention model, which captures the

correspondences between element x_i and others, and e_{ij} is an alignment model which scores how well the input elements x_i and x_j match. Here $W^Q, W^K \in R^{d_x \times d_z}$ are parameter matrices. Motivated by [Shaw *et al.*, 2018], our main innovation over the conventional self-attention architecture is that we encode the structure relationships between two element pair (x_i, x_j) in the alignment model as shown in Eq. 4. We generalize this to an arbitrary number of features that represent the structure relationships between (x_i, x_j):

$$e_{ij} = \frac{\left(x_i W^Q\right)\left(x_j W^K + \sum_{l=1}^{|F|} f_{ij_l} W^{K_l}\right)^T}{\sqrt{d_z}} \quad (5)$$

where $|F|$ is the number of features, and $f_{ij_l} \in R^{d_f}$ is the l-th feature embedding for (x_i, x_j) pair, $W^{K_l} \in R^{d_f \times d_z}$ is a parameter matrix. Then, we update Eq. 2 accordingly to propagate structure information to the sublayer output by:

$$z_i = \sum_{j=1}^{n} \alpha_{ij} \left(x_j W^V + \sum_{l=1}^{|F|} f_{ij_l} W^{V_l} \right) \quad (6)$$

where $W^{V_l} \in R^{d_f \times d_z}$ is a parameter matrix. For simplicity, we set feature embedding size d_f as d_z.

Note that to enable an efficient implementation, we follow [Shaw *et al.*, 2018] and use simple addition to incorporate structure representations in Eq. 5 and Eq. 6.

4.2 Syntactic and Semantic Features

Our generalized self-attention model supports an arbitrary number of input features to represent the structure relationships between word pairs. Next, we will focus on a few of well known syntactic and semantic features extracted from parse tree augmented with semantic information.

Syntactic Path (SynP). We use syntactic path, the combination of syntactic labels along the path from x_i to x_j to indicate the syntactic structure of x_i and x_j.[3] For example, the syntactic path from *airstrikes* to *conflict* is *→NP↑S↓VP↓NP*. Note that we use a positional tag → or ← to indicate if x_i is on the left or right side of x_j.

Syntactic Distance (SynD). We use syntactic distance, the numbers of nodes up-along and down-along the syntactic path, to indicate how far the two words x_i and x_j are from syntax perspective. For example, the syntactic distance from *airstrikes* to *conflict* is →1↑2↓. Similarly, positional tag → indicates that *airstrikes* is on the left side of *conflict*. Note that this feature is a coarse version of syntactic path.

Semantic Relation (SemR). Deriving semantic relation for a word pair is not straight forward since semantic structure is not strictly of tree style. Given a sentence, every predicate has its own semantic structure which connects the predicate and its arguments. For example, the semantic structure for predicate *escalated* in Figure 2 is *ARG0 (The airstrikes) + Pre (escalated) + ARG1 (conflict .... Georgia)*. We use BIO tags to assign each word in a sentence with a semantic label per predicate. For predicate *escalated*, the semantic label for *The* is *B-ARG0*, similarly *I-ARG0* for *airstrikes*, *B-ARG1* for *conflict*, *O* for ., the sentence period. As a result, each word will have n semantic labels, where n is the number of predicates in this sentence. For word pair x_i and x_j, we first choose an appropriate predicate and then combine the two words' semantic labels w.r.t the chosen predicate. Here an appropriate predicate is the one (if exists) when the two words are covered by the semantic structure of the predicate. Otherwise, we use *NONE* to represent their semantic relation. Finally, we add the positional tag → or ← to indicate if x_i is on the left or the right side of x_j. For example, the semantic relation for *airstrikes* and *conflict* is *→I-ARG0*B-ARG1*.

5 Experimentation

5.1 Experimental Settings

For evaluation of our approach, we use the sentences annotated with AMRs from the LDC release LDC2017T10. The dataset consists of 36,521 training AMRs, 1,368 development AMRs and 1,371 testing AMRs. To assign syntactic parses and semantic roles to the sentences in the whole dataset, we utilize the toolkit AllenNLP [Gardner *et al.*, 2017] which achieves an F1 of 94.1% on Penn Treebank for syntactic parsing and 84.9% on English Ontonotes 5.0 for PropBank SRL. The results are currently state of the art of the tasks. For evaluation purpose, we use the AMR-evaluation tools to evaluate parsing performance in Smatch and other fine-grained metrics [Cai and Knight, 2013; Damonte *et al.*, 2017].

We use *tensor2tensor* as the implementation of *Transformer* seq2seq model.[4] In parameter setting, we set the number of layers in both the encoder and decoder to 6. For optimization we use Adam with $\beta 1 = 0.1$ [Kingma and Ba, 2015]. The number of heads is set to 8. In addition, we set the hidden size to 512 and the batch token-size to 4096. In beam searching, we increase the extra length as 100 from default 50. We also set Google NMT length penalty parameter $\alpha = 1.0$ to encourage longer generation. In all experiments, we train the models for 250K steps on a single K40 GPU.

Moreover, we share vocabulary for the input and the output. For better translating rare words, we segment words into word pieces by byte pair encoding (BPE) [Sennrich *et al.*, 2016]) with 20K operations. Our preliminary experiment showed that sharing vocabulary and using BPE substantially improve the performance on the development set from 64.8 in F1 to 71.4, revealing that they are two effective ways to address the issue of data sparsity [Peng *et al.*, 2017].

In the sequence-aware linearization approach, it will unavoidably increase the length of an input sentence. Statistics on training data shows that the averaged length for input sequence increases from 17, to 48 with syntactic labels, 27 with semantic labels, and 52 with both types of labels. Fortunately, it does not bring extra parameters since though the input becomes longer, we set the same number of BPE coding operation as baseline, resulting negligible changes in vocabulary.

In the structure-aware encoding approach, we choose the 40K most frequent SynP features, and map all other features

[3] If there exist two or more identical labels along the path, we use one as to alleviate path sparsity.

[4] https://github.com/tensorflow/tensor2tensor

Metric	Baseline	Sequence-Aware Linearizing			Structure-Aware Encoding			
		+Syntax	+Semantics	+Both	+SynP	+SynD	+SemR	+All
Smatch	70.9	73.2	72.4	74.1	73.8	73.6	72.7	74.3
Unlabeled	74.3	75.9	75.4	77.3	76.5	76.3	75.5	77.3
No WSD	71.6	73.4	72.5	74.7	74.0	73.8	72.8	74.8
Reentrancy	54.5	56.9	56.1	57.4	56.7	56.8	55.7	58.3
Concepts	82.1	82.5	82.2	84.2	83.0	82.4	82.1	84.2
NER	80.4	80.9	81.2	83.2	81.3	82.2	80.8	82.4
Wiki	69.2	71.7	71.2	72.7	72.0	72.6	72.4	71.3
Negations	60.2	65.0	63.3	64.2	63.5	64.1	61.3	64.0
SRL	66.8	68.9	68.0	70.3	69.2	69.0	68.7	70.4

Table 1: F1 scores of our two approaches on Smatch and other fine-grained metrics. Significant test by bootstrap resampling [Koehn, 2004] shows that *+Both* is significant over *+Semantics* at $p = 0.01$ while it is not over *+Syntax*. Similarly, *+All* is significant over *+SemR* at $p = 0.05$ and is not over *+SynP* and *+SynD*.

into either $\rightarrow$*UNK* or $\leftarrow$*UNK*. For SynD and SemR features, we do not do feature selection since the two sets of features are limited in size.

5.2 Experimental Results

Table 1 shows the performance of our two approaches on the test set. Overall, modeling source syntax or semantics in our two approaches benefits AMR parsing in Smatch and all other fine-grained metrics.

Comparing the systems of linearizing approach against the baseline, we see that modeling source syntax (+Syntax) outperforms the baseline by 2.3 in F1 scores while modeling semantics (+Semantics) outperforms the baseline by 1.5 in F1 scores. Besides, in the presence of source syntax, integrating semantics results in further improvement of 0.9 in F1 scores (e.g., 73.2 *vs.* 74.1). This observation suggests that there exist both complementarity and overlapping between syntactic and semantic information. The gain achieved by modeling semantics is limited in the presence of syntax. Interestingly, this performance trend is similar to that of modeling source syntax and semantics for machine translation [Li *et al.*, 2014].

Similar conclusion could be drawn when comparing the systems of structure-aware approach against the baseline. With all the three types of syntactic and semantic features, it achieves 3.4 F1 score improvement over the baseline.

Comparing the two proposed approaches, we see that the sequence-aware linearization approach achieves slightly lower performance than the structure-aware encoding approach. This suggests that even without manually designed features, the linearization approach is capable of implicitly learning useful structure information from linearized syntactic and semantic sequence. This is very encouraging since this approach does not require extra parameters and modification to the seq2seq model itself. The structure-aware encoding approach, on the other hand, is extensible to incorporate more word pair-related features.

Interestingly, modeling semantic structure from SRL output helps for SRL subtask in AMR parsing (see the last line in Table 1). This is probably due to the fact that the standalone SRL labeler (i.e., the AllenNLP toolkit) achieves more accurate semantic analysis than the AMR parsers.

5.3 Comparison with Other Systems

We also compare our performance with related studies on LDC2017T10, as presented in Table 2. From the results we can see that by modeling source syntax and semantics, our parsers outperform all other seq2seq models, even with more silver data [van Noord and Bos, 2017]. This indicates the effectiveness of our approaches in modeling source syntax and semantics. Moreover, our parsers achieve better performance than non-seq2seq models, except the graph prediction parser in [Lyu and Titov, 2018].

Table 3 compares the detailed performance. We obtain relatively high results for *Reentrancies*, *Negations*, and *SRL*, most likely due to their close relevance to syntactic and semantic structure. Compare to the state-of-the-art performance in [Lyu and Titov, 2018], we achieve lower performance for *concepts*, *NER*, and *Wikification*. This is probably due to that seq2seq models tend to stop early for translating long sentences, resulting in some source concept words not translated.

6 Related Work

We first group recent studies in AMR parsing into seq2seq and non-seq2seq parsing. Then we discuss related studies that employ either syntactic or semantic knowledge.

Seq2seq parsing. Seq2seq models generally require less features and build the AMR graph in an end-to-end way. However, previous models usually suffer from data sparsity issue [Peng *et al.*, 2017]. In order to address this issue, [Barzdins and Gosko, 2016] and [van Noord and Bos, 2017] translate char-based plain English sentences into char-based AMR linearization while [Konstas *et al.*, 2017] and [van Noord and Bos, 2017] utilize external training data. In this paper, our baseline achieves comparable results to non-seq2seq models, suggesting that segmenting words into word pieces and sharing vocabulary on two sides are two effective ways to handle data sparsity issue. Based on the strong baseline, this paper presents two different approaches to explore syntactic and semantic knowledge for seq2seq AMR parsing.

Non-seq2seq parsing. Most other AMR models can further categorized into tree-based, graph-based, and transition-based. Tree-based AMR models incrementally convert a dependency tree into its corresponding AMR graph [Wang *et*

Type	Parser	P	R	F1
Non-Seq2Seq	Graph Prediction [Lyu and Titov, 2018]	-	-	74.4
	Transition [Guo and Lu, 2018]	-	-	69.8
	Tree [Groschwitz *et al.*, 2018]	-	-	71.0
Seq2Seq	Neural-Pointer [Buys and Blunsom, 2017]	-	-	61.9
	charSeq [van Noord and Bos, 2017]	-	-	64.0
	charSeq + 100K silver data [van Noord and Bos, 2017]	76.0	67.0	71.0
	Baseline (This paper)	74.0	68.1	70.9
	Sequence-Aware Linearization (This paper)	76.8	71.7	74.1
	Structure-Aware Encoding (This paper)	77.7	71.1	74.3

Table 2: Comparison of our parser with other parsers on LDC2017T10.

Metric	vN'17	L'18	G'18a	G'18b	Our1	Our2	Our3
Smatch	71	**74.4**	71	70	70.9	74.1	74.3
Unlabeled	74	77.1	74	73	74.3	**77.3**	**77.3**
No WSD	72	**75.5**	72	71	71.6	74.7	74.8
Reentrancy	52	52.3	49	49	54.5	57.4	**58.3**
Concepts	82	**85.9**	**86**	84	82.1	84.2	84.2
NER	79	**86.0**	78	80	80.4	83.2	82.4
Wiki	65	**75.7**	71	70	69.2	72.7	71.3
Negations	62	58.4	57	48	60.2	**64.2**	64.0
SRL	66	69.8	64	63	66.8	70.3	**70.4**

Table 3: Detailed F1 scores on the LDC2017T10 test set. Here, vN'17 is for [van Noord and Bos, 2017] with 100K silver dataset, L'18 for [Lyu and Titov, 2018], G'18a for [Groschwitz *et al.*, 2018], G'18b for [Guo and Lu, 2018]. Our1 is our baseline while Our2 and Our3 are our two parsers of sequence-aware linearization approach and structure-aware encoding approach.

al., 2015a]. Graph-based models calculate scores of edges and then use a maximum spanning connected subgraph algorithm to select edges that will constitute the graph [Werling *et al.*, 2015; Flanigan *et al.*, 2014]. Besides, [Foland and Martin, 2017] adopt a pipeline approach, in which the concept identification requires 5 different LSTMs based on carefully designed features. Then they connect these components into a single graph. [Lyu and Titov, 2018] treat the alignments as latent variables in a joint probabilistic model. Transition-based models converts the plain text sentence into its corresponding graph via steps of transition [Damonte *et al.*, 2017; Ballesteros and Al-Onaizan, 2017; Peng *et al.*, 2018; Vilares and Gómez-Rodríguez, 2018; Guo and Lu, 2018]. Besides the above categories, there exits other AMR models that uses combinatory categorical grammar (CCG), or translation grammar [Artzi *et al.*, 2015; Pust *et al.*, 2015].

Syntax and Semantics for AMR parsing. Most AMR models incorporate additional features such as POS tags, dependency trees, named entities, non-lexical role labels, etc. Specifically, tree-based AMR models directly adopt dependency tree as inputs. In translation-based models, different features are exploited for better transition predicting, include POS and syntactic features [Damonte *et al.*, 2017; Ballesteros and Al-Onaizan, 2017; Vilares and Gómez-Rodríguez, 2018; Peng *et al.*, 2018] and semantic features [Wang *et al.*, 2015a; Damonte *et al.*, 2017]. In graph-based models, syntactic information, including POS, NER and dependency structures, are widely used for concept identification [Flanigan *et al.*, 2014; Werling *et al.*, 2015; Foland and Martin, 2017; Lyu and Titov, 2018]. In seq2seq-based models, [van Noord and Bos, 2017] and [Buys and Blunsom, 2017] incorporate POS tags for encoding source sentences. To the best of our knowledge, this paper is the first to explore syntactic and semantic structures for seq2seq AMR parsing.

7 Conclusion

We propose two approaches to incorporate syntactic and semantic information into seq2seq-based AMR parsing. The first approach linearizes source syntax and semantic structure into a mixed sequence which stitchingly consists of words, syntactic labels, and semantic labels while the second approach uses structure-aware encoder to explicitly model syntactic and semantic relations for word pairs. Experimental results show that the two approaches are capable of modeling source syntax and semantics and achieve improvement of 3.1% and 3.4% F1 scores on the benchmark dataset over a strong seq2seq baseline.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Grant No. 61751206, 61876120), and the Priority Academic Program Development of Jiangsu Higher Education Institutions.

References

[Artzi *et al.*, 2015] Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. Broad-coverage ccg semantic parsing with amr. In *Proceedings of EMNLP*, pages 1699–1710, 2015.

[Ballesteros and Al-Onaizan, 2017] Miguel Ballesteros and Yaser Al-Onaizan. Amr parsing using stack-lstms. In *Proceedings of EMNLP*, page 1269–1275, 2017.

[Barzdins and Gosko, 2016] Guntis Barzdins and Didzis Gosko. RIGA at SemEval-2016 task 8: Impact of smatch extension and character-level neural translation on amr parsing accuracy. In *Proceedings of SemEval*, page 1143–1147, 2016.

[Buys and Blunsom, 2017] Jan Buys and Phil Blunsom. Oxford at semeval2017 task 9: Neural AMR parsing with pointer-augmented attention. In *Proceedings of SemEval*, page 914–919, 2017.

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

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

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

[Flanigan *et al.*, 2014] Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. A discriminative graph-based parser for the abstract meaning representation. In *Proceedings of ACL*, pages 1426–1436, 2014.

[Foland and Martin, 2017] William Foland and James H. Martin. Abstract meaning representation parsing using lstm recurrent neural networks. In *Proceedings of ACL*, pages 463–472, 2017.

[Gardner *et al.*, 2017] Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. AllenNLP: A deep semantic natural language processing platform. In *Proceedings of ACL Workshop for NLP Open Source Software*, pages 1–6, 2017.

[Gildea and Jurafsky, 2002] Daniel Gildea and Daniel Jurafsky. Automatic labeling of semantic roles. *Computational Linguistics*, 28:245–288, 2002.

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

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

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

[Koehn, 2004] Philipp Koehn. Statistical significance tests for machine translation evaluation. In *Proceedings of EMNLP*, pages 388–395, 2004.

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

[Li *et al.*, 2014] Junhui Li, Yuval Marton, Philip Resnik, and Hal Daumé III. A unified model for soft linguistic reordering constraints in statistical machine translation. In *Proceedings of ACL*, pages 1123–1133, 2014.

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

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

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

[Peng *et al.*, 2018] Xiaochang Peng, Daniel Gildea, and Giorgio Satta. Amr parsing with cache transition systems. In *Proceedings of AAAI*, page 4897–4904, 2018.

[Pust *et al.*, 2015] Michael Pust, Ulf Hermjakob, Kevin Knight, Daniel Marcu, and Jonathan May. Parsing english into abstract meaning representation using syntax-based machine translation. In *Proceedings of EMNLP*, pages 1143–1154, 2015.

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

[Shaw *et al.*, 2018] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. In *Proceedings of NAACL*, pages 464–468, 2018.

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

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

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

[Vilares and Gómez-Rodríguez, 2018] David Vilares and Carlos Gómez-Rodríguez. A transition-based algorithm for unrestricted amr parsing. In *Proceedings of NAACL*, pages 142–149, 2018.

[Vinyals *et al.*, 2015] Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. Grammar as a foreign language. In *Proceedings of NIPS*, pages 2773–2781, 2015.

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

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

[Werling *et al.*, 2015] Keenon Werling, Gabor Angeli, and Christopher D. Manning. Robust subgraph generation improves abstract meaning representation parsing. In *Proceedings of ACL*, page 982–991, 2015.