Neural AMR: Sequence-to-Sequence Models for Parsing and Generation

Ioannis Konstas[†] **Srinivasan Iyer**[†] **Mark Yatskar**[†]
Yejin Choi[†] **Luke Zettlemoyer**[†‡]

[†]Paul G. Allen School of Computer Science & Engineering, Univ. of Washington, Seattle, WA
{ikonstas,sviyer,my89,yejin,lsz}@cs.washington.edu

[‡]Allen Institute for Artificial Intelligence, Seattle, WA
lukez@allenai.org

Abstract

Sequence-to-sequence models have shown strong performance across a broad range of applications. However, their application to parsing and generating text using Abstract Meaning Representation (AMR) has been limited, due to the relatively limited amount of labeled data and the non-sequential nature of the AMR graphs. We present a novel training procedure that can lift this limitation using millions of unlabeled sentences and careful preprocessing of the AMR graphs. For AMR parsing, our model achieves competitive results of 62.1 SMATCH, the current best score reported without significant use of external semantic resources. For AMR generation, our model establishes a new state-of-the-art performance of BLEU 33.8. We present extensive ablative and qualitative analysis including strong evidence that sequence-based AMR models are robust against ordering variations of graph-to-sequence conversions.

1 Introduction

Abstract Meaning Representation (AMR) is a semantic formalism to encode the meaning of natural language text. As shown in Figure 1, AMR represents the meaning using a directed graph while abstracting away the surface forms in text. AMR has been used as an intermediate meaning representation for several applications including machine translation (MT) (Jones et al., 2012), summarization (Liu et al., 2015), sentence compression (Takase et al., 2016), and event extraction (Huang et al., 2016). While AMR allows for rich semantic representation, annotating training data in AMR is expensive, which in turn limits the use of neural network models (Misra and Artzi, 2016; Peng et al., 2017; Barzdins and Gosko, 2016).

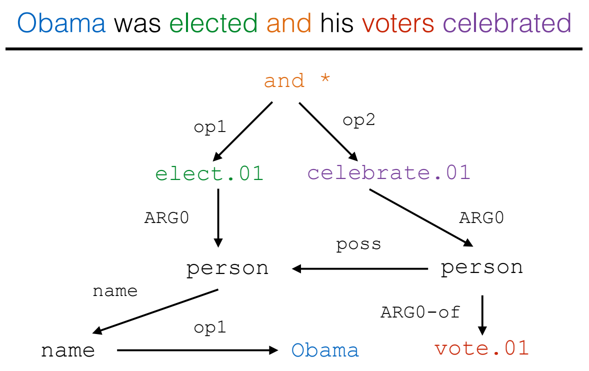


Figure 1: An example sentence and its corresponding Abstract Meaning Representation (AMR). AMR encodes semantic dependencies between entities mentioned in the sentence, such as "Obama" being the "arg0" of the verb "elected".

In this work, we present the first successful sequence-to-sequence (seq2seq) models that achieve strong results for both text-to-AMR parsing and AMR-to-text generation. Seq2seq models have been broadly successful in many other applications (Wu et al., 2016; Bahdanau et al., 2015; Luong et al., 2015; Vinyals et al., 2015). However, their application to AMR has been limited, in part because effective *linearization* (encoding graphs as linear sequences) and data sparsity were thought to pose significant challenges. We show that these challenges can be easily overcome, by demonstrating that seq2seq models can be trained using *any* graph-isomorphic linearization and that unlabeled text can be used to significantly reduce sparsity.

Our approach is two-fold. First, we introduce a novel paired training procedure that enhances both the text-to-AMR parser and AMR-to-text generator. More concretely, first we use self-training to

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 146–157
Vancouver, Canada, July 30 - August 4, 2017. ©2017 Association for Computational Linguistics
https://doi.org/10.18653/v1/P17-1014

bootstrap a high quality AMR parser from millions of unlabeled Gigaword sentences (Napoles et al., 2012) and then use the automatically parsed AMR graphs to pre-train an AMR generator. This paired training allows both the parser and generator to learn high quality representations of fluent English text from millions of weakly labeled examples, that are then fine-tuned using human annotated AMR data.

Second, we propose a preprocessing procedure for the AMR graphs, which includes anonymizing entities and dates, grouping entity categories, and encoding nesting information in concise ways, as illustrated in Figure 2(d). This preprocessing procedure helps overcoming the data sparsity while also substantially reducing the complexity of the AMR graphs. Under such a representation, we show that any depth first traversal of the AMR is an effective linearization, and it is even possible to use a different random order for each example.

Experiments on the LDC2015E86 AMR corpus (SemEval-2016 Task 8) demonstrate the effectiveness of the overall approach. For parsing, we are able to obtain competitive performance of 62.1 SMATCH without using any external annotated examples other than the output of a NER system, an improvement of over 10 points relative to neural models with a comparable setup. For generation, we substantially outperform previous best results, establishing a new state of the art of 33.8 BLEU. We also provide extensive ablative and qualitative analysis, quantifying the contributions that come from preprocessing and the paired training procedure.

2 Related Work

Alignment-based Parsing Flanigan et al. (2014) (JAMR) pipeline concept and relation identification with a graph-based algorithm. Zhou et al. (2016) extend JAMR by performing the concept and relation identification tasks jointly with an incremental model. Both systems rely on features based on a set of alignments produced using bi-lexical cues and hand-written rules. In contrast, our models train directly on parallel corpora, and make only minimal use of alignments to anonymize named entities.

Grammar-based Parsing Wang et al. (2016) (CAMR) perform a series of shift-reduce transformations on the output of an externally-trained dependency parser, similar to Damonte et al. (2017), Brandt et al. (2016), Puzikov et al. (2016), and Goodman et al. (2016). Artzi et al. (2015) use a grammar induction approach with Combinatory Categorical Grammar (CCG), which relies on pretrained CCGBank categories, like Bjerva et al. (2016). Pust et al. (2015) recast parsing as a string-to-tree Machine Translation problem, using unsupervised alignments (Pourdamghani et al., 2014), and employing several external semantic resources. Our neural approach is engineering lean, relying only on a large unannotated corpus of English and algorithms to find and canonicalize named entities.

Neural Parsing Recently there have been a few seq2seq systems for AMR parsing (Barzdins and Gosko, 2016; Peng et al., 2017). Similar to our approach, Peng et al. (2017) deal with sparsity by anonymizing named entities and typing low frequency words, resulting in a very compact vocabulary (2k tokens). However, we avoid reducing our vocabulary by introducing a large set of unlabeled sentences from an external corpus, therefore drastically lowering the out-of-vocabulary rate (see Section 6).

AMR Generation Flanigan et al. (2016) specify a number of tree-to-string transduction rules based on alignments and POS-based features that are used to drive a tree-based SMT system. Pourdamghani et al. (2016) also use an MT decoder; they learn a classifier that linearizes the input AMR graph in an order that follows the output sentence, effectively reducing the number of alignment crossings of the phrase-based decoder. Song et al. (2016) recast generation as a traveling salesman problem, after partitioning the graph into fragments and finding the best linearization order. Our models do not need to rely on a particular linearization of the input, attaining comparable performance even with a per example random traversal of the graph. Finally, all three systems intersect with a large language model trained on Gigaword. We show that our seq2seq model has the capacity to learn the same information as a language model, especially after pretraining on the external corpus.

Data Augmentation Our paired training procedure is largely inspired by Sennrich et al. (2016). They improve neural MT performance for low resource language pairs by using a back-translation MT system for a large monolingual corpus of the target language in order to create synthetic output, and mixing it with the human translations. We instead pre-train on the external corpus first, and then fine-tune on the original dataset.

3 Methods

In this section, we first provide the formal definition of AMR parsing and generation (section 3.1). Then we describe the sequence-to-sequence models we use (section 3.2), graph-to-sequence conversion (section 3.3), and our paired training procedure (section 3.4).

3.1 Tasks

We assume access to a training dataset D where each example pairs a natural language sentence s with an AMR a. The AMR is a rooted directed acylical graph. It contains nodes whose names correspond to sense-identified verbs, nouns, or AMR specific concepts, for example `elect.01`, `Obama`, and `person` in Figure 1. One of these nodes is a distinguished root, for example, the node `and` in Figure 1. Furthermore, the graph contains labeled edges, which correspond to PropBank-style (Palmer et al., 2005) semantic roles for verbs or other relations introduced for AMR, for example, `arg0` or `op1` in Figure 1. The set of node and edge names in an AMR graph is drawn from a set of tokens C, and every word in a sentence is drawn from a vocabulary W.

We study the task of training an **AMR parser**, i.e., finding a set of parameters θ_P for model f, that predicts an AMR graph $\hat{a}$, given a sentence s:

$$\hat{a} = \underset{a}{\operatorname{argmax}}\, f\big(a|s;\theta_P\big) \qquad (1)$$

We also consider the reverse task, training an **AMR generator** by finding a set of parameters θ_G, for a model f that predicts a sentence $\hat{s}$, given an AMR graph a:

$$\hat{s} = \underset{s}{\operatorname{argmax}}\, f\big(s|a;\theta_G\big) \qquad (2)$$

In both cases, we use the same family of predictors f, sequence-to-sequence models that use global attention, but the models have independent parameters, θ_P and θ_G.

3.2 Sequence-to-sequence Model

For both tasks, we use a stacked-LSTM sequence-to-sequence neural architecture employed in neural machine translation (Bahdanau et al., 2015; Wu et al., 2016).[1] Our model uses a global attention decoder and unknown word replacement with small modifications (Luong et al., 2015).

The model uses a stacked bidirectional-LSTM encoder to encode an input sequence and a stacked LSTM to decode from the hidden states produced by the encoder. We make two modifications to the encoder: (1) we concatenate the forward and backward hidden states at every level of the stack instead of at the top of the stack, and (2) introduce dropout in the first layer of the encoder. The decoder predicts an attention vector over the encoder hidden states using previous decoder states. The attention is used to weigh the hidden states of the encoder and then predict a token in the output sequence. The weighted hidden states, the decoded token, and an attention signal from the previous time step (input feeding) are then fed together as input to the next decoder state. The decoder can optionally choose to output an unknown word symbol, in which case the predicted attention is used to copy a token directly from the input sequence into the output sequence.

3.3 Linearization

Our seq2seq models require that both the input and target be presented as a linear sequence of tokens. We define a linearization order for an AMR graph as any sequence of its nodes and edges. A linearization is defined as (1) a linearization order and (2) a rendering function that generates any number of tokens when applied to an element in the linearization order (see Section 4.2 for implementation details). Furthermore, for parsing, a valid AMR graph must be recoverable from the linearization.

3.4 Paired Training

Obtaining a corpus of jointly annotated pairs of sentences and AMR graphs is expensive and current datasets only extend to thousands of examples. Neural sequence-to-sequence models suffer from sparsity with so few training pairs. To reduce the effect of sparsity, we use an external unannotated corpus of sentences S_e, and a procedure which pairs the training of the parser and generator.

Our procedure is described in Algorithm 1, and first trains a parser on the dataset D of pairs of sentences and AMR graphs. Then it uses self-training

[1]We extended the Harvard NLP seq2seq framework from `http://nlp.seas.harvard.edu/code`.

Algorithm 1 Paired Training Procedure

```
Input: Training set of sentences and AMR graphs $(s, a) \in \mathcal{D}$, an unannotated external corpus of sentences $S_e$, a number of self training iterations, $N$, and an initial sample size $k$.
Output: Model parameters for AMR parser $\theta_P$ and AMR generator $\theta_G$.
 1: $\theta_P \leftarrow$ Train parser on D
    ▷ Self-train AMR parser.
 2: $S_e^1 \leftarrow$ sample $k$ sentences from $S_e$
 3: for $i = 1$ to $N$ do
 4:     $A_e^i \leftarrow$ Parse $S_e^i$ using parameters $\theta_P$
    ▷ Pre-train AMR parser.
 5:     $\theta_P \leftarrow$ Train parser on $(A_e^i, S_e^i)$
    ▷ Fine tune AMR parser.
 6:     $\theta_P \leftarrow$ Train parser on D with initial parameters $\theta_P$
 7:     $S_e^{i+1} \leftarrow$ sample $k \cdot 10^i$ new sentences from $S_e$
 8: end for
 9: $S_e^N \leftarrow$ sample $k \cdot 10^N$ new sentences from $S_e$
    ▷ Pre-train AMR generator.
10: $A_e \leftarrow$ Parse $S_e^N$ using parameters $\theta_P$
11: $\theta_G \leftarrow$ Train generator on $(A_e^N, S_e^N)$
    ▷ Fine tune AMR generator.
12: $\theta_G \leftarrow$ Train generator on $D$ using initial parameters $\theta_G$
13: return $\theta_P$, $\theta_G$
```

to improve the initial parser. Every iteration of self-training has three phases: (1) parsing samples from a large, unlabeled corpus S_e, (2) creating a new set of parameters by training on S_e, and (3) fine-tuning those parameters on the original paired data. After each iteration, we increase the size of the sample from S_e by an order of magnitude. After we have the best parser from self-training, we use it to label AMRs for S_e and pre-train the generator. The final step of the procedure fine-tunes the generator on the original dataset D.

4 AMR Preprocessing

We use a series of preprocessing steps, including AMR linerization, anonymization, and other modifications we make to sentence-graph pairs. Our methods have two goals: (1) reduce the complexity of the linearized sequences to make learning easier while maintaining enough original information, and (2) address sparsity from certain open class vocabulary entries, such as named entities (NEs) and quantities. Figure 2(d) contains example inputs and outputs with all of our preprocessing techniques.

Graph Simplification In order to reduce the overall length of the linearized graph, we first remove variable names and the `instance-of` relation (/) before every concept. In case of re-entrant nodes we replace the variable mention with its co-referring concept. Even though this replacement incurs loss of information, often the surrounding context helps recover the correct realization, e.g., the possessive role `:poss` in the example of Figure 1 is strongly correlated with the surface form *his*. Following Pourdamghani et al. (2016) we also remove senses from all concepts for AMR generation only. Figure 2(a) contains an example output after this stage.

4.1 Anonymization of Named Entities

Open-class types including NEs, dates, and numbers account for 9.6% of tokens in the sentences of the training corpus, and 31.2% of vocabulary W. 83.4% of them occur fewer than 5 times in the dataset. In order to reduce sparsity and be able to account for new unseen entities, we perform extensive anonymization.

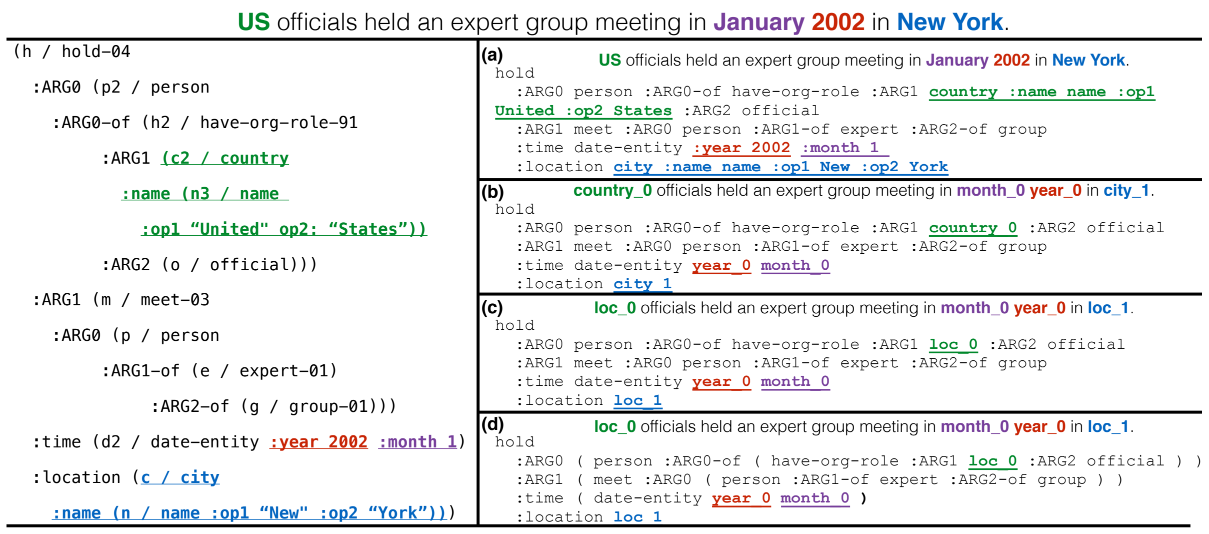
First, we anonymize sub-graphs headed by one of AMR's over 140 fine-grained entity types that contain a `:name` role. This captures structures referring to entities such as `person`, `country`, miscellaneous entities marked with `*-enitity`, and typed numerical values, `*-quantity`. We exclude `date` entities (see the next section). We then replace these sub-graphs with a token indicating fine-grained type and an index, i, indicating it is the ith occurrence of that type.[2] For example, in Figure 2 the sub-graph headed by `country` gets replaced with `country_0`.

On the training set, we use alignments obtained using the JAMR aligner (Flanigan et al., 2014) and the unsupervised aligner of Pourdamghani et al. (2014) in order to find mappings of anonymized subgraphs to spans of text and replace mapped text with the anonymized token that we inserted into the AMR graph. We record this mapping for use during testing of generation models. If a generation model predicts an anonymization token, we find the corresponding token in the AMR graph and replace the model's output with the most frequent mapping observed during training for the entity name. If the entity was never observed, we copy its name directly from the AMR graph.

Anonymizing Dates For dates in AMR graphs, we use separate anonymization tokens for year, month-number, month-name, day-number and day-name, indicating whether the date is mentioned by word or by number.[3] In AMR gener-

[2]In practice we only used three groups of ids: a different one for NEs, dates and constants/numbers.

[3]We also use three date format markers that appear in the text as: *YYYYMMDD*, *YYMMDD*, and *YYYY-MM-DD*.


US officials held an expert group meeting in January 2002 in New York.

```
(h / hold-04
  :ARG0 (p2 / person
    :ARG0-of (h2 / have-org-role-91
        :ARG1 (c2 / country
          :name (n3 / name
            :op1 "United" op2: "States"))
        :ARG2 (o / official)))
  :ARG1 (m / meet-03
    :ARG0 (p / person
        :ARG1-of (e / expert-01)
            :ARG2-of (g / group-01)))
  :time (d2 / date-entity :year 2002 :month 1)
  :location (c / city
    :name (n / name :op1 "New" :op2 "York")))
```

(a) US officials held an expert group meeting in January 2002 in New York.

```
hold
  :ARG0 person :ARG0-of have-org-role :ARG1 country :name name :op1
United :op2 States :ARG2 official
  :ARG1 meet :ARG0 person :ARG1-of expert :ARG2-of group
  :time date-entity :year 2002 :month 1
  :location city :name name :op1 New :op2 York
```

(b) country_0 officials held an expert group meeting in month_0 year_0 in city_1.

```
hold
  :ARG0 person :ARG0-of have-org-role :ARG1 country_0 :ARG2 official
  :ARG1 meet :ARG0 person :ARG1-of expert :ARG2-of group
  :time date-entity year_0 month_0
  :location city_1
```

(c) loc_0 officials held an expert group meeting in month_0 year_0 in loc_1.

```
hold
  :ARG0 person :ARG0-of have-org-role :ARG1 loc_0 :ARG2 official
  :ARG1 meet :ARG0 person :ARG1-of expert :ARG2-of group
  :time date-entity year_0 month_0
  :location loc_1
```

(d) loc_0 officials held an expert group meeting in month_0 year_0 in loc_1.

```
hold
  :ARG0 ( person :ARG0-of ( have-org-role :ARG1 loc_0 :ARG2 official ) )
  :ARG1 ( meet :ARG0 ( person :ARG1-of expert :ARG2-of group ) )
  :time ( date-entity year_0 month_0 )
  :location loc_1
```

Figure 2: Preprocessing methods applied to sentence (top row) - AMR graph (left column) pairs. Sentence-graph pairs after (a) graph simplification, (b) named entity anonymization, (c) named entity clustering, and (d) insertion of scope markers.

ation, we render the corresponding format when predicted. Figure 2(b) contains an example of all preprocessing up to this stage.

Named Entity Clusters When performing AMR generation, each of the AMR fine-grained entity types is manually mapped to one of the four coarse entity types used in the Stanford NER system (Finkel et al., 2005): person, location, organization and misc. This reduces the sparsity associated with many rarely occurring entity types. Figure 2 (c) contains an example with named entity clusters.

NER for Parsing When parsing, we must normalize test sentences to match our anonymized training data. To produce fine-grained named entities, we run the Stanford NER system and first try to replace any identified span with a fine-grained category based on alignments observed during training. If this fails, we anonymize the sentence using the coarse categories predicted by the NER system, which are also categories in AMR. After parsing, we deterministically generate AMR for anonymizations using the corresponding text span.

4.2 Linearization

Linearization Order Our linearization order is defined by the order of nodes visited by depth first search, including backward traversing steps. For example, in Figure 2, starting at `meet` the order contains `meet`, `:ARG0`, `person`, `:ARG1-of`, `expert`, `:ARG2-of`, `group`, `:ARG2-of`, `:ARG1-of`, `:ARG0`.[4] The order traverses children in the sequence they are presented in the AMR. We consider alternative orderings of children in Section 7 but always follow the pattern demonstrated above.

Rendering Function Our rendering function marks scope, and generates tokens following the pre-order traversal of the graph: (1) if the element is a node, it emits the type of the node. (2) if the element is an edge, it emits the type of the edge and then recursively emits a bracketed string for the (concept) node immediately after it. In case the node has only one child we omit the scope markers (denoted with left "(", and right ")" parentheses), thus significantly reducing the number of generated tokens. Figure 2(d) contains an example showing all of the preprocessing techniques and scope markers that we use in our full model.

5 Experimental Setup

We conduct all experiments on the AMR corpus used in SemEval-2016 Task 8 (LDC2015E86), which contains 16,833/1,368/1,371 train/dev/test examples. For the paired training procedure of Algorithm 1, we use Gigaword as our external corpus and sample sentences that only contain words from the AMR corpus vocabulary W. We subsampled the original sentence to ensure there is no overlap with the AMR training or test sets. Table 2

[4]Sense, `instance-of` and variable information has been removed at the point of linearization.

	Dev			Test		
Model	Prec	Rec	F1	Prec	Rec	F1
SBMT (Pust et al., 2015)	-	-	69.0	-	-	67.1
CAMR (Wang et al., 2016)	72.3	61.4	66.6	70.4	63.1	66.5
CCG* (Artzi et al., 2015)	67.2	65.1	66.1	66.8	65.7	66.3
JAMR (Flanigan et al., 2014)	-	-	-	64.0	53.0	58.0
GIGA-20M	62.2	66.0	64.4	59.7	64.7	62.1
GIGA-2M	61.9	64.8	63.3	60.2	63.6	61.9
GIGA-200k	59.7	62.9	61.3	57.8	60.9	59.3
AMR-ONLY	54.9	60.0	57.4	53.1	58.1	55.5
SEQ2SEQ (Peng et al., 2017)	-	-	-	55.0	50.0	52.0
CHAR-LSTM (Barzdins and Gosko, 2016)	-	-	-	-	-	43.0

Table 1: SMATCH scores for AMR Parsing. *Reported numbers are on the newswire portion of a previous release of the corpus (LDC2014T12).

summarizes statistics about the original dataset and the extracted portions of Gigaword. We evaluate AMR parsing with SMATCH (Cai and Knight, 2013), and AMR generation using BLEU (Papineni et al., 2002)[5].

We validated word embedding sizes and RNN hidden representation sizes by maximizing AMR development set performance (Algorithm 1 – line 1). We searched over the set {128, 256, 500, 1024} for the best combinations of sizes and set both to 500. Models were trained by optimizing cross-entropy loss with stochastic gradient descent, using a batch size of 100 and dropout rate of 0.5. Across all models when performance does not improve on the AMR dev set, we decay the learning rate by 0.8.

For the initial parser trained on the AMR corpus, (Algorithm 1 – line 1), we use a single stack version of our model, set initial learning rate to 0.5 and train for 60 epochs, taking the best performing model on the development set. All subsequent models benefited from increased depth and we used 2-layer stacked versions, maintaining the same embedding sizes. We set the initial Gigaword sample size to $k = 200,000$ and executed a maximum of 3 iterations of self-training. For pretraining the parser and generator, (Algorithm 1 – lines 4 and 9), we used an initial learning rate of 1.0, and ran for 20 epochs. We attempt to fine-tune the parser and generator, respectively, after every epoch of pre-training, setting the initial learning rate to 0.1. We select the best performing model on the development set among all of these fine-tuning attempts. During prediction we perform decoding using beam search and set the beam size to 5 both for parsing and generation.

[5]We use the multi-BLEU script from the MOSES decoder suite (Koehn et al., 2007).

Corpus	Examples	OOV@1	OOV@5
AMR	16833	44.7	74.9
GIGA-200k	200k	17.5	35.3
GIGA-2M	2M	11.2	19.1
GIGA-20M	20M	8.0	12.7

Table 2: LDC2015E86 AMR training set, GIGA-200k, GIGA-2M and GIGA-20M statistics; OOV@1 and OOV@5 are the out-of-vocabulary rates on the NL side with thresholds of 1 and 5, respectively. Vocabulary sizes are 13027 tokens for the AMR side, and 17319 tokens for the NL side.

Model	Dev	Test
GIGA-20M	33.1	33.8
GIGA-2M	31.8	32.3
GIGA-200k	27.2	27.4
AMR-ONLY	21.7	22.0
PBMT* (Pourdamghani et al., 2016)	27.2	26.9
TSP (Song et al., 2016)	21.1	22.4
TREETOSTR (Flanigan et al., 2016)	23.0	23.0

Table 3: BLEU results for AMR Generation. *Model has been trained on a previous release of the corpus (LDC2014T12).

6 Results

Parsing Results Table 1 summarizes our development results for different rounds of self-training and test results for our final system, self-trained on 200k, 2M and 20M unlabeled Gigaword sentences. Through every round of self-training, our parser improves. Our final parser outperforms comparable seq2seq and character LSTM models by over 10 points. While much of this improvement comes from self-training, our model without Gigaword data outperforms these approaches by 3.5 points on F1. We attribute this increase in performance to different handling of preprocessing and more careful hyper-parameter tuning. All other models that we compare against use semantic resources, such as WordNet, dependency parsers or CCG parsers (models marked with * were trained with less data, but only evaluate on newswire text; the rest evaluate on the full test set, containing text from blogs). Our full models outperform JAMR, a graph-based model but still lags behind other parser-dependent systems (CAMR[6]), and resource heavy approaches (SBMT).

Generation Results Table 3 summarizes our AMR generation results on the development and test set. We outperform all previous state-of-the-art systems by the first round of self-training and further improve with the next rounds. Our final model trained on GIGA-20M outperforms TSP and TREETOSTR trained on LDC2015E86, by over 9 BLEU points.[7] Overall, our model incorporates less data than previous approaches as all reported methods train language models on the whole Gigaword corpus. We leave scaling our models to all of Gigaword for future work.

Sparsity Reduction Even after anonymization of open class vocabulary entries, we still encounter a great deal of sparsity in vocabulary given the small size of the AMR corpus, as shown in Table 2. By incorporating sentences from Gigaword we are able to reduce vocabulary sparsity dramatically, as we increase the size of sampled sentences: the out-of-vocabulary rate with a threshold of 5 reduces almost 5 times for GIGA-20M.

Preprocessing Ablation Study We consider the contribution of each main component of our preprocessing stages while keeping our linearization order identical. Figure 2 contains examples for each setting of the ablations we evaluate on. First we evaluate using linearized graphs without parentheses for indicating scope, Figure 2(c), then without named entity clusters, Figure 2(b), and additionally without any anonymization, Figure 2(a).

Model	BLEU
FULL	21.8
FULL - SCOPE	19.7
FULL - SCOPE - NE	19.5
FULL - SCOPE - NE - ANON	18.7

Table 4: BLEU scores for AMR generation ablations on preprocessing (DEV set).

Model	Prec	Rec	F1
FULL	54.9	60.0	57.4
FULL - ANON	22.7	54.2	32.0

Table 5: SMATCH scores for AMR parsing ablations on preprocessing (DEV set).

Tables 4 summarizes our evaluation on the AMR generation. Each components is required, and scope markers and anonymization contribute the most to overall performance. We suspect without scope markers our seq2seq models are not as effective at capturing long range semantic relationships between elements of the AMR graph. We also evaluated the contribution of anonymization to AMR parsing (Table 5). Following previous work, we find that seq2seq-based AMR parsing is largely ineffective without anonymization (Peng et al., 2017).

7 Linearization Evaluation

In this section we evaluate three strategies for converting AMR graphs into sequences in the context of AMR generation and show that our models are largely agnostic to linearization orders. Our results argue, unlike SMT-based AMR generation methods (Pourdamghani et al., 2016), that seq2seq models can learn to ignore artifacts of the conversion of graphs to linear sequences.

7.1 Linearization Orders

All linearizations we consider use the pattern described in Section 4.2, but differ on the order in which children are visited. Each linearization generates anonymized, scope-marked output (see Section 4), of the form shown in Figure 2(d).

Human The proposal traverses children in the order presented by human authored AMR annotations exactly as shown in Figure 2(d).

[6]Since we are currently not using any Wikipedia resources for the prediction of named entities, we compare against the no-wikification version of the CAMR system.

[7]We also trained our generator on GIGA-2M and fine-tuned on LDC2014T12 in order to have a direct comparison with PBMT, and achieved a BLEU score of 29.7, i.e., 2.8 points of improvement.

Linearization Order	BLEU
HUMAN	21.7
GLOBAL-RANDOM	20.8
RANDOM	20.3

Table 6: BLEU scores for AMR generation for different linearization orders (DEV set).

Global-Random We construct a random global ordering of all edge types appearing in AMR graphs and re-use it for every example in the dataset. We traverse children based on the position in the global ordering of the edge leading to a child.

Random For each example in the dataset we traverse children following a different random order of edge types.

7.2 Results

We present AMR generation results for the three proposed linearization orders in Table 6. Random linearization order performs somewhat worse than traversing the graph according to Human linearization order. Surprisingly, a per example random linearization order performs nearly identically to a global random order, arguing seq2seq models can learn to ignore artifacts of the conversion of graphs to linear sequences.

Human-authored AMR leaks information The small difference between random and global-random linearizations argues that our models are largely agnostic to variation in linearization order. On the other hand, the model that follows the human order performs better, which leads us to suspect it carries extra information not apparent in the graphical structure of the AMR.

To further investigate, we compared the relative ordering of edge pairs under the same parent to the relative position of children nodes derived from those edges in a sentence, as reported by JAMR alignments. We found that the majority of pairs of AMR edges (57.6%) always occurred in the same relative order, therefore revealing no extra generation order information.[8] Of the examples corresponding to edge pairs that showed variation, 70.3% appeared in an order consistent with the order they were realized in the sentence. The relative ordering of some pairs of AMR edges was particularly indicative of generation order. For example, the relative ordering of edges with types `location` and `time`, was 17% more indicative of the generation order than the majority of generated locations before `time`.[9]

To compare to previous work we still report results using human orderings. However, we note that any practical application requiring a system to generate an AMR representation with the intention to realize it later on, e.g., a dialog agent, will need to be trained either using consistent, or random-derived linearization orders. Arguably, our models are agnostic to this choice.

[8]This is consistent with constraints encoded in the annotation tool used to collect AMR. For example, `:ARG0` edges are always ordered before `:ARG1` edges.

Error Type	%
Coverage	29
Disfluency	23
Anonymization	14
Sparsity	13
Attachment	12
Other	10

Table 7: Error analysis for AMR generation on a sample of 50 examples from the development set.

8 Qualitative Results

Figure 3 shows example outputs of our full system. The generated text for the first graph is nearly perfect with only a small grammatical error due to anonymization. The second example is more challenging, with a deep right-branching structure, and a coordination of the verbs `stabilize` and `push` in the subordinate clause headed by `state`. The model omits some information from the graph, namely the concepts `terrorist` and `virus`. In the third example there are greater parts of the graph that are missing, such as the whole sub-graph headed by `expert`. Also the model makes wrong attachment decisions in the last two sub-graphs (it is the `evidence` that is *unimpeachable* and *irrefutable*, and not the `equipment`), mostly due to insufficient annotation (`thing`) thus making their generation harder.

Finally, Table 7 summarizes the proportions of error types we identified on 50 randomly selected examples from the development set. We found that the generator mostly suffers from coverage issues,

[9]Consider the sentences *"She went to school in New York two years ago"*, and *"Two years ago, she went to school in New York"*, where *"two year ago"* is the time modifying constituent for the verb *went* and *"New York"* is the location modifying constituent of *went*.

an inability to mention all tokens in the input, followed by fluency mistakes, as illustrated above. Attachment errors are less frequent, which supports our claim that the model is robust to graph linearization, and can successfully encode long range dependency information between concepts.

9 Conclusions

We applied sequence-to-sequence models to the tasks of AMR parsing and AMR generation, by carefully preprocessing the graph representation and scaling our models via pretraining on millions of unlabeled sentences sourced from Gigaword corpus. Crucially, we avoid relying on resources such as knowledge bases and externally trained parsers. We achieve competitive results for the parsing task (SMATCH 62.1) and state-of-the-art performance for generation (BLEU 33.8).

For future work, we would like to extend our work to different meaning representations such as the Minimal Recursion Semantics (MRS; Copestake et al. (2005)). This formalism tackles certain linguistic phenomena differently from AMR (e.g., negation, and co-reference), contains explicit annotation on concepts for number, tense and case, and finally handles multiple languages[10] (Bender, 2014). Taking a step further, we would like to apply our models on Semantics-Based Machine Translation using MRS as an intermediate representation between pairs of languages, and investigate the added benefit compared to directly translating the surface strings, especially in the case of distant language pairs such as English and Japanese (Siegel, 2000).

Acknowledgments

The research was supported in part by DARPA under the DEFT program through AFRL (FA8750-13-2-0019) and the CwC program through ARO (W911NF-15-1-0543), the ARO (W911NF-16-1-0121), the NSF (IIS-1252835, IIS-1562364, IIS-1524371), an Allen Distinguished Investigator Award, and gifts by Google and Facebook. The authors thank Rik Koncel-Kedziorski and the UW NLP group for helpful discussions, and the anonymous reviewers for their thorough and helpful comments.

[10] A list of actively maintained languages can be found here: `http://moin.delph-in.net/GrammarCatalogue`

```
limit
 :arg0 ( treaty :arg0-of ( control :arg1 arms ) )
 :arg1 ( number
 :arg1 ( weapon :mod conventional
  :arg1-of ( deploy
   :arg2 ( relative-pos :op1 loc_0 :dir west )
  :arg1-of possible ) ) )
```

REF: the arms control treaty limits the number of conventional weapons that can be deployed west of the Ural Mountains .

SYS: the arms control treaty limits the number of conventional weapons that can be deployed west **of** **Ural Mountains** .

COMMENT: **disfluency**

```
state
 :arg0 report
 :arg1 ( obligate :arg1 ( government-organization
   :arg0-of ( govern :arg1 loc_0 ) )
   :arg2 ( help :arg1 ( and
    :op1 ( stabilize :arg1 ( state :mod weak ) )
    :op2 ( push :arg1 ( regulate
      :mod international :arg0-of ( stop
       :arg1 terrorist
       :arg2 ( use
        :arg1 ( information
         :arg2-of ( available :arg3-of free ))
        :arg2 ( and
         :op1 ( create :arg1 ( form
          :domain ( warfare
           :mod biology :example ( version
             :arg1-of modify :poss other_1 ) )
           :mod new ) )
         :op2 ( unleash :arg1 form  )
      ) ) ) ) ) ) ) )
```

REF: the report stated British government must help to stabilize weak states and push for international regulations that would stop **terrorists** using freely available information to create and unleash new forms of biological warfare such as a modified version of the influenza **virus** .

SYS: the report stated that the **Britain government** must help stabilize **the weak** states and push international regulations to stop the use of freely available information to create a form of **new** biological warfare such as **the modified** version of the influenza .

COMMENT: **coverage** , **disfluency**, **attachment**

```
state
 :arg0 ( person
  :arg0-of ( have-org-role
   :arg1 ( committee :mod technical )
   :arg3 ( expert
    :arg1 person
    :arg2 missile
    :mod loc_0 ) ) )
 :arg1 ( evidence
  :arg0 equipment
  :arg1 ( plan :arg1 ( transfer :arg1 ( contrast
   :arg1 ( missile :mod ( just :polarity - ) )
    :arg2 ( capable
     :arg1 thing
     :arg2 ( make :arg1 missile ) ) ) ) )
  :mod ( impeach :polarity - :arg1 thing )
  :mod ( refute :polarity - :arg1 thing ) )
```

REF: a technical committee **of Indian missile** experts stated that the equipment was unimpeachable and irrefutable **evidence of a plan to transfer not just missiles but missile-making capability**.

SYS: **a technical committee expert on the technical committee** stated that the equipment is **not impeach** , **but it is not refutes** .

COMMENT: **coverage** , **disfluency**, **attachment**

Figure 3: Linearized AMR after preprocessing, reference sentence, and output of the generator. We mark with colors common error types: disfluency, coverage (missing information from the input graph), and attachment (implying a semantic relation from the AMR between incorrect entities).

References

Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage CCG semantic parsing with AMR. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1699–1710. http://aclweb.org/anthology/D15-1198.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 2015 International Conference on Learning Representations*. CBLS, San Diego, California. http://arxiv.org/abs/1409.0473.

Guntis Barzdins and Didzis Gosko. 2016. RIGA at SemEval-2016 Task 8: Impact of Smatch extensions and character-level neural translation on AMR parsing accuracy. In *Proceedings of the 10th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, San Diego, California, pages 1143–1147. http://www.aclweb.org/anthology/S16-1176.

Emily M. Bender. 2014. Language CoLLAGE: Grammatical description with the LinGO grammar matrix. In *Proceedings of the 9th International Conference on Language Resources and Evaluation*. Reykjavik, Iceland, pages 2447–2451.

Johannes Bjerva, Johan Bos, and Hessel Haagsma. 2016. The Meaning Factory at SemEval-2016 Task 8: Producing AMRs with Boxer. In *Proceedings of the 10th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, San Diego, California, pages 1179–1184. http://www.aclweb.org/anthology/S16-1182.

Lauritz Brandt, David Grimm, Mengfei Zhou, and Yannick Versley. 2016. ICL-HD at SemEval-2016 Task 8: Meaning representation parsing - augmenting AMR parsing with a preposition semantic role labeling neural network. In *Proceedings of the 10th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, San Diego, California, pages 1160–1166. http://www.aclweb.org/anthology/S16-1179.

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Sofia, Bulgaria, pages 748–752. http://www.aclweb.org/anthology/P13-2131.

Ann Copestake, Dan Flickinger, Carl Pollard, and Ivan A. Sag. 2005. Minimal Recursion Semantics: An introduction. *Research on Language and Computation* 3(2):281–332. https://doi.org/10.1007/s11168-006-6327-9.

Marco Damonte, Shay B. Cohen, and Giorgio Satta. 2017. An incremental parser for abstract meaning representation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Valencia, Spain, pages 536–546. http://www.aclweb.org/anthology/E17-1051.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, Ann Arbor, Michigan, pages 363–370. https://doi.org/10.3115/1219840.1219885.

Jeffrey Flanigan, Chris Dyer, Noah A. Smith, and Jaime Carbonell. 2016. Generation from abstract meaning representation using tree transducers. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, San Diego, California, pages 731–739. http://www.aclweb.org/anthology/N16-1087.

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. 2014. A discriminative graph-based parser for the abstract meaning representation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Baltimore, Maryland, pages 1426–1436. http://www.aclweb.org/anthology/P14-1134.

James Goodman, Andreas Vlachos, and Jason Naradowsky. 2016. UCL+Sheffield at SemEval-2016 Task 8: Imitation learning for AMR parsing with an alpha-bound. In *Proceedings of the 10th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, San Diego, California, pages 1167–1172. http://www.aclweb.org/anthology/S16-1180.

Lifu Huang, Taylor Cassidy, Xiaocheng Feng, Heng Ji, Clare R. Voss, Jiawei Han, and Avirup Sil. 2016. Liberal event extraction and event schema induction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Berlin, Germany, pages 258–268. http://www.aclweb.org/anthology/P16-1025.

Bevan Jones, Jacob Andreas, Daniel Bauer, Karl Moritz Hermann, and Kevin Knight. 2012. Semantics-Based Machine Translation with Hyperedge Replacement Grammars. In *Proceedings of the 2012 International Conference on Computational Linguistics*. Bombay, India, pages 1359–1376. http://www.aclweb.org/anthology/C12-1083.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Prague, Czech Republic, pages 177–180. http://dl.acm.org/citation.cfm?id=1557769.1557821.

Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman Sadeh, and Noah A. Smith. 2015. Toward abstractive summarization using semantic representations. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Denver, Colorado, pages 1077–1086. http://www.aclweb.org/anthology/N15-1114.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1412–1421. http://aclweb.org/anthology/D15-1166.

Dipendra Kumar Misra and Yoav Artzi. 2016. Neural shift-reduce CCG semantic parsing. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 1775–1786. https://aclweb.org/anthology/D16-1183.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated Gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*. Association for Computational Linguistics, Montréal, Canada, pages 95–100. http://www.aclweb.org/anthology/W12-3018.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics* 31(1):71–106. http://www.cs.rochester.edu/ gildea/palmer-propbank-cl.pdf.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Philadelphia, Pennsylvania, USA, pages 311–318. https://doi.org/10.3115/1073083.1073135.

Xiaochang Peng, Chuan Wang, Daniel Gildea, and Nianwen Xue. 2017. Addressing the data sparsity issue in neural AMR parsing. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Valencia, Spain, pages 366–375. http://www.aclweb.org/anthology/E17-1035.

Nima Pourdamghani, Yang Gao, Ulf Hermjakob, and Kevin Knight. 2014. Aligning English strings with abstract meaning representation graphs. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Doha, Qatar, pages 425–429. http://www.aclweb.org/anthology/D14-1048.

Nima Pourdamghani, Kevin Knight, and Ulf Hermjakob. 2016. Generating English from abstract meaning representations. In *Proceedings of the 9th International Natural Language Generation conference*. Association for Computational Linguistics, Edinburgh, UK, pages 21–25. http://anthology.aclweb.org/W16-6603.

Michael Pust, Ulf Hermjakob, Kevin Knight, Daniel Marcu, and Jonathan May. 2015. Parsing english into abstract meaning representation using syntax-based machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1143–1154. https://aclweb.org/anthology/D/D15/D15-1136.

Yevgeniy Puzikov, Daisuke Kawahara, and Sadao Kurohashi. 2016. M2L at SemEval-2016 Task 8: AMR parsing with neural networks. In *Proceedings of the 10th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, San Diego, California, pages 1154–1159. http://www.aclweb.org/anthology/S16-1178.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Berlin, Germany, pages 86–96. http://www.aclweb.org/anthology/P16-1009.

Melanie Siegel. 2000. *HPSG Analysis of Japanese*, Springer Berlin Heidelberg, pages 264–279.

Linfeng Song, Yue Zhang, Xiaochang Peng, Zhiguo Wang, and Daniel Gildea. 2016. AMR-to-text generation as a traveling salesman problem. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 2084–2089. https://aclweb.org/anthology/D16-1224.

Sho Takase, Jun Suzuki, Naoaki Okazaki, Tsutomu Hirao, and Masaaki Nagata. 2016. Neural headline generation on abstract meaning representation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 1054–1059. https://aclweb.org/anthology/D16-1112.

Oriol Vinyals, Ł ukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Proceedings of the*

28th International Conference on Neural Information Processing Systems, MIT Press, pages 2773–2781. http://papers.nips.cc/paper/5635-grammar-as-a-foreign-language.pdf.

Chuan Wang, Sameer Pradhan, Xiaoman Pan, Heng Ji, and Nianwen Xue. 2016. CAMR at SemEval-2016 Task 8: An extended transition-based AMR parser. In *Proceedings of the 10th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, San Diego, California, pages 1173–1178. http://www.aclweb.org/anthology/S16-1181.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR* abs/1609.08144. http://arxiv.org/abs/1609.08144.

Junsheng Zhou, Feiyu Xu, Hans Uszkoreit, Weiguang QU, Ran Li, and Yanhui Gu. 2016. AMR parsing with an incremental joint model. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 680–689. https://aclweb.org/anthology/D16-1065.