

AMR-to-text Generation with Synchronous Node Replacement Grammar

Linfeng Song, Xiaochang Peng, Yue Zhang, Zhiguo Wang and Daniel Gildea

Department of Computer Science, University of Rochester, Rochester, NY 14627

IBM T.J. Watson Research Center, Yorktown Heights, NY 10598

Singapore University of Technology and Design

Abstract

This paper addresses the task of AMR-to-text generation by leveraging synchronous node replacement grammar. During training, graph-to-string rules are learned using a heuristic extraction algorithm. At test time, a graph transducer is applied to collapse input AMRs and generate output sentences. Evaluated on a standard benchmark, our method gives the state-of-the-art result.

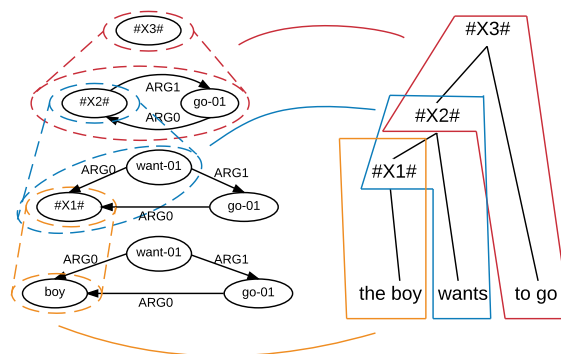


Figure 1: Graph-to-string derivation.

1 Introduction

Abstract Meaning Representation (AMR) (Banasescu et al., 2013) is a semantic formalism encoding the meaning of a sentence as a rooted, directed graph. AMR uses a graph to represent meaning, where nodes (such as “boy”, “want-01”) represent concepts, and edges (such as “ARG0”, “ARG1”) represent relations between concepts. Encoding many semantic phenomena into a graph structure, AMR is useful for NLP tasks such as machine translation (Jones et al., 2012; Tamchyna et al., 2015), question answering (Mitra and Baral, 2015), summarization (Takase et al., 2016) and event detection (Li et al., 2015).

AMR-to-text generation is challenging as function words and syntactic structures are abstracted away, making an AMR graph correspond to multiple realizations. Despite much literature so far on text-to-AMR parsing (Flanigan et al., 2014; Wang et al., 2015; Peng et al., 2015; Vanderwende et al., 2015; Pust et al., 2015; Artzi et al., 2015; Groschwitz et al., 2015; Goodman et al., 2016; Zhou et al., 2016; Peng et al., 2017), there has been little work on AMR-to-text generation (Flanigan et al., 2016; Song et al., 2016; Pourdamghani et al., 2016).

Flanigan et al. (2016) transform a given AMR graph into a spanning tree, before translating it to a sentence using a tree-to-string transducer. Their method leverages existing machine translation techniques, capturing hierarchical correspondences between the spanning tree and the surface string. However, it suffers from error propagation since the output is constrained given a spanning tree due to the projective correspondence between them. Information loss in the graph-to-tree transformation step cannot be recovered. Song et al. (2016) directly generate sentences using graph-fragment-to-string rules. They cast the task of finding a sequence of disjoint rules to transduce an AMR graph into a sentence as a traveling salesman problem, using local features and a language model to rank candidate sentences. However, their method does not learn hierarchical structural correspondences between AMR graphs and strings.

We propose to leverage the advantages of hierarchical rules without suffering from graph-to-tree errors by directly learning graph-to-string rules. As shown in Figure 1, we learn a synchronous node replacement grammar (NRG) from a corpus of aligned AMR and sentence pairs. At test time, we apply a graph transducer to collapse input

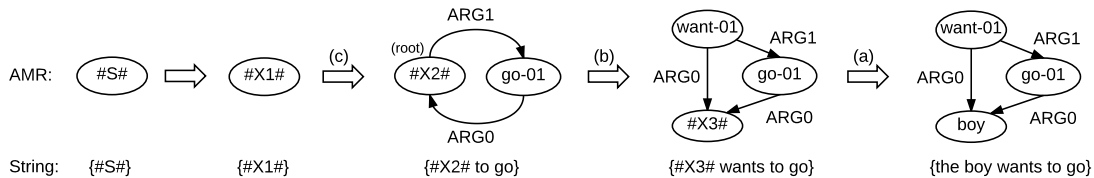


Figure 2: Example deduction procedure

ID.	F	E
(a)	(b / boy)	the boy
(b)	(w / want-01 :ARG0 (X / #X#))	#X# wants
(c)	(X / #X# :ARG1 (g / go-01 :ARG0 X))	#X# to go
(d)	(w / want-01 :ARG0 (b / boy))	the boy wants

Table 1: Example rule set

AMR graphs and generate output strings according to the learned grammar. Our system makes use of a log-linear model with real-valued features, tuned using MERT (Och, 2003), and beam search decoding. It gives a BLEU score of 25.62 on LDC2015E86, which is the state-of-the-art on this dataset.

2 Synchronous Node Replacement Grammar

2.1 Grammar Definition

A synchronous node replacement grammar (NRG) is a rewriting formalism: $G = \langle N, \Sigma, \Delta, P, S \rangle$, where N is a finite set of nonterminals, Σ and Δ are finite sets of terminal symbols for the source and target sides, respectively. $S \in N$ is the start symbol, and P is a finite set of productions. Each instance of P takes the form $X_i \rightarrow (\langle F, E \rangle, \sim)$, where $X_i \in N$ is a nonterminal node, F is a rooted, connected AMR fragment with edge labels over Σ and node labels over $N \cup \Sigma$, E is a corresponding target string over $N \cup \Delta$ and \sim denotes the alignment of nonterminal symbols between F and E . A classic NRG (Engelfriet and Rozenberg, 1997, Chapter 1) also defines C , which is an embedding mechanism defining how F is connected to the rest of the graph when replacing X_i with F on the graph. Here we omit defining C and allow arbitrary connections.¹ Following Chiang

¹This may over generate, but does not affect our case, as in our bottom-up decoding procedure (section 3) when F is replaced with X_i , nodes previously connected to F are re-connected to X_i

Data: training corpus C
Result: rule instances R

```

1  $R \leftarrow []$ ;
2 for ( $Sent, AMR, \sim$ ) in  $C$  do
3    $R_{cur} \leftarrow \text{FRAGMENTEXTRACT}(Sent, AMR, \sim)$ ;
4   for  $r_i$  in  $R_{cur}$  do
5      $R.APPEND(r_i)$ ;
6     for  $r_j$  in  $R_{cur}/\{r_i\}$  do
7       if  $r_i.CONTAINS(r_j)$  then
8          $r_{ij} \leftarrow r_i.COLLAPSE(r_j)$ ;
9          $R.APPEND(r_{ij})$ ;
10      end
11    end
12  end
13 end

```

Algorithm 1: Rule extraction

(2005), we use only one nonterminal X in addition to S , and use subscripts to distinguish different non-terminal instances.

Figure 2 shows an example derivation process for the sentence “the boy wants to go” given the rule set in Table 1. Given the start symbol S , which is first replaced with X_1 , rule (c) is applied to generate “ X_2 to go” and its AMR counterpart. Then rule (b) is used to generate “ X_3 wants” and its AMR counterpart from X_2 . Finally, rule (a) is used to generate “the boy” and its AMR counterpart from X_3 . Our graph-to-string rules are inspired by synchronous grammars for machine translation (Wu, 1997; Yamada and Knight, 2002; Gildea, 2003; Chiang, 2005; Huang et al., 2006; Liu et al., 2006; Shen et al., 2008; Xie et al., 2011; Meng et al., 2013).

2.2 Induced Rules

There are three types of rules in our system, namely *induced rules*, *concept rules* and *graph glue rules*. Here we first introduce induced rules, which are obtained by a two-step procedure on a training corpus. Shown in Algorithm 1, the first step is to extract a set of initial rules from training $\langle \text{sentence}, \text{AMR}, \sim \rangle^2$ pairs (Line 2) using the phrase-to-graph-fragment extraction algorithm of Peng et al. (2015) (Line 3). Here an *initial rule*

² \sim denotes alignment between words and AMR labels.

contains only terminal symbols in both F and E . As a next step, we match between pairs of initial rules r_i and r_j , and generate r_{ij} by collapsing r_i with r_j , if r_i contains r_j (Line 6-8). Here r_i contains r_j , if $r_j.F$ is a subgraph of $r_i.F$ and $r_j.E$ is a sub-phrase of $r_i.E$. When collapsing r_i with r_j , we replace the corresponding subgraph in $r_i.F$ with a new non-terminal node, and the sub-phrase in $r_i.E$ with the same non-terminal. For example, we obtain rule (b) by collapsing (d) with (a) in Table 1. All initial and generated rules are stored in a rule list R (Lines 5 and 9), which will be further normalized to obtain the final induced rule set.

2.3 Concept Rules and Glue Rules

In addition to induced rules, we adopt concept rules (Song et al., 2016) and graph glue rules to ensure existence of derivations. For a concept rule, F is a single node in the input AMR graph, and E is a morphological string of the node concept. A concept rule is used in case no induced rule can cover the node. We refer to the verbalization list³ and AMR guidelines⁴ for creating more complex concept rules. For example, one concept rule created from the verbalization list is “(k / keep-01 :ARG1 (p / peace)) ||| peacekeeping”.

Inspired by Chiang (2005), we define graph glue rules to concatenate non-terminal nodes connected with an edge, when no induced rules can be applied. Three glue rules are defined for each type of edge label. Taking the edge label “ARG0” as an example, we create the following glue rules:

ID.	F	E
r_1	(X1 / #X1# :ARG0 (X2 / #X2#))	#X1# #X2#
r_2	(X1 / #X1# :ARG0 (X2 / #X2#))	#X2# #X1#
r_3	(X1 / #X1# :ARG0 X1)	#X1#

where for both r_1 and r_2 , F contains two non-terminal nodes with a directed edge connecting them, and E is the concatenation the two non-terminals in either the monotonic or the inverse order. For r_3 , F contains one non-terminal node with a self-pointing edge, and E is the non-terminal. With concept rules and glue rules in our final rule set, it is easily guaranteed that there are legal derivations for any input AMR graph.

3 Model

We adopt a log-linear model for scoring search hypotheses. Given an input AMR graph, we find

³<http://amr.isi.edu/download/lists/verbalization-list-1.06.txt>

⁴<https://github.com/amrisi/amr-guidelines>

the highest scored derivation t^* from all possible derivations t :

$$t^* = \operatorname{argmax}_t \exp \sum_i w_i f_i(g, t), \quad (1)$$

where g denotes the input AMR, $f_i(\cdot, \cdot)$ and w_i represent a feature and the corresponding weight, respectively. The feature set that we adopt includes phrase-to-graph and graph-to-phrase translation probabilities and their corresponding lexicalized translation probabilities (section 3.1), language model score, word count, rule count, re-ordering model score (section 3.2) and moving distance (section 3.3). The language model score, word count and phrase count features are adopted from SMT (Koehn et al., 2003; Chiang, 2005).

We perform bottom-up search to transduce input AMRs to surface strings. Each hypothesis contains the current AMR graph, translations of collapsed subgraphs, the feature vector and the current model score. Beam search is adopted, where hypotheses with the same number of collapsed edges and nodes are put into the same beam.

3.1 Translation Probabilities

Production rules serve as a basis for scoring hypotheses. We associate each synchronous NRG rule $n \rightarrow (\langle F, E \rangle, \sim)$ with a set of probabilities. First, phrase-to-fragment translation probabilities are defined based on maximum likelihood estimation (MLE), as shown in Equation 2, where $c_{\langle F, E \rangle}$ is the fractional count of $\langle F, E \rangle$.

$$p(F|E) = \frac{c_{\langle F, E \rangle}}{\sum_{F'} c_{\langle F', E \rangle}} \quad (2)$$

In addition, lexicalized translation probabilities are defined as:

$$p_w(F|E) = \prod_{l \in F} \sum_{w \in E} p(l|w) \quad (3)$$

Here l is a label (including both edge labels such as “ARG0” and concept labels such as “want-01”) in the AMR fragment F , and w is a word in the phrase E . Equation 3 can be regarded as a “soft” version of the lexicalized translation probabilities adopted by SMT, which picks the alignment yielding the maximum lexicalized probability for each translation rule. In addition to $p(F|E)$ and $p_w(F|E)$, we use features in the reverse direction, namely $p(E|F)$ and $p_w(E|F)$, the definitions of which are omitted as they are consistent with

Equations 2 and 3, respectively. The probabilities associated with concept rules and glue rules are manually set to 0.0001.

3.2 Reordering Model

Although the word order is defined for induced rules, it is not the case for glue rules. We learn a reordering model that helps to decide whether the translations of the nodes should be monotonic or inverse given the directed connecting edge label. The probabilistic model using smoothed counts is defined as:

$$p(M|h, l, t) = \frac{1.0 + \sum_h \sum_t c(h, l, t, M)}{2.0 + \sum_{o \in \{M, I\}} \sum_h \sum_t c(h, l, t, o)} \quad (4)$$

$c(h, l, t, M)$ is the count of monotonic translations of head h and tail t , connected by edge l .

3.3 Moving Distance

The moving distance feature captures the distances between the subgraph roots of two consecutive rule matches in the decoding process, which controls a bias towards collapsing nearby subgraphs consecutively.

4 Experiments

4.1 Setup

We use LDC2015E86 as our experimental dataset, which contains 16833 training, 1368 dev and 1371 test instances. Each instance contains a sentence, an AMR graph and the alignment generated by a heuristic aligner. Rules are extracted from the training data, and model parameters are tuned on the dev set. For tuning and testing, we filter out sentences with more than 30 words, resulting in 1103 dev instances and 1055 test instances. We train a 4-gram language model (LM) on gigaword (LDC2011T07), and use BLEU (Papineni et al., 2002) as the evaluation metric. MERT is used (Och, 2003) to tune model parameters on k -best outputs on the devset, where k is set 50.

We investigate the effectiveness of rules and features by ablation tests: “NoInducedRule” does not adopt induced rules, “NoConceptRule” does not adopt concept rules, “NoMovingDistance” does not adopt the moving distance feature, and “NoReorderModel” disables the reordering model. Given an AMR graph, if *NoConceptRule* cannot produce a legal derivation, we concatenate

System	Dev	Test
TSP-gen	21.12	22.44
JAMR-gen	23.00	23.00
All	25.24	25.62
NoInducedRule	16.75	17.43
NoConceptRule	23.99	24.86
NoMovingDistance	23.48	24.06
NoReorderModel	25.09	25.43

Table 2: Main results.

existing translation fragments into a final translation, and if a subgraph can not be translated, the empty string is used as the output. We also compare our method with previous works, in particular JAMR-gen (Flanigan et al., 2016) and TSP-gen (Song et al., 2016), on the same dataset.

4.2 Main results

The results are shown in Table 2. First, *All* outperforms all baselines. *NoInducedRule* leads to the greatest performance drop compared with *All*, demonstrating that induced rules play a very important role in our system. On the other hand, *NoConceptRule* does not lead to much performance drop. This observation is consistent with the observation of Song et al. (2016) for their TSP-based system. *NoMovingDistance* leads to a significant performance drop, empirically verifying the fact that the translations of nearby subgraphs are also close. Finally, *NoReorderingModel* does not affect the performance significantly, which can be because the most important reordering patterns are already covered by the hierarchical induced rules. Compared with *TSP-gen* and *JAMR-gen*, our final model *All* improves the BLEU from 22.44 and 23.00 to 25.62, showing the advantage of our model. To our knowledge, this is the best result reported so far on the task.

4.3 Grammar analysis

We have shown the effectiveness of our synchronous node replacement grammar (SNRG) on the AMR-to-text generation task. Here we further analyze our grammar as it is relatively less studied than the hyperedge replacement grammar (HRG) (Drewes et al., 1997).

Statistics on the whole rule set

We first categorize our rule set by the number of terminals and nonterminals in the AMR fragment F , and show the percentages of each type in Figure 3. Each rule contains at most 1 nonterminal, as we collapse each initial rule only once. First

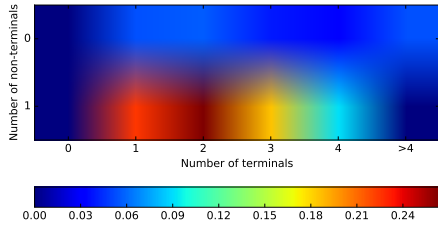


Figure 3: Statistics on the right-hand side.

	Glue	Nonterminal	Terminal
1-best	30.0%	30.1%	39.9%

Table 3: Rules used for decoding.

of all, the percentage of rules containing nonterminals are much more than those without nonterminals, as we collapse each pair of initial rules (in Algorithm 1) and the results can be quadratic the number of initial rules. In addition, most rules are small containing 1 to 3 terminals, meaning that they represent small pieces of meaning and are easier to matched on a new AMR graph. Finally, there are a few large rules, which represent complex meaning.

Statistics on the rules used for decoding

In addition, we collect the rules that our well-tuned system used for generating the 1-best output on the testset, and categorize them into 3 types: (1) glue rules, (2) nonterminal rules, which are not glue rules but contain nonterminals on the right-hand side and (3) terminal rules, whose right-hand side only contain terminals. Over the rules used on the 1-best result, more than 30% are non-terminal rules, showing that the induced rules play an important role. On the other hand, 30% are glue rules. The reason is that the data sparsity for graph grammars is more severe than string-based grammars (such as CFG), as the graph structures are more complex than strings. Finally, terminal rules take the largest percentage, while most are *induced rules*, but not *concept rules*.

Rule examples

Finally, we show some rules in Table 4, where F and E are the right-hand-side AMR fragment and phrase, respectively. For the first rule, the root of F is a verb (“give-01”) whose subject is a nonterminal and object is a AMR fragment “(p / person :ARG0-of (u / use-01))”, which means “user”. So it is easy to see that the corresponding phrase E conveys the same meaning. For the second rule, “(s3 / stay-01 :accompanier (i / i))” means “stay

	(g / give-01
F :	:ARG0 (X1 / #X1#)
	:ARG2 (p / person
	:ARG0-of (u / use-01)))
E :	#X1# has given users an
	(X1 / #X1#
F :	:ARG2 (s3 / stay-01 :ARG1 X1
	:accompanier (i / i)))
E :	#X1# staying with me

Table 4: Example rules.

(u / understand-01
:ARG0 (y / you)
:ARG1 (t2 / thing
:ARG1-of (f2 / feel-01
:ARG0 (p2 / person
:example (p / person :wiki -
:name (t / name :op1 “TMT”)
:location (c / city :wiki “Fairfax, Virginia”
:name (f / name :op1 “Fairfax”))))))
:time (n / now))
Trans: now, you have to understand that people feel about
such as tmt fairfax
Ref: now you understand how people like tmt in fairfax
feel .

Table 5: Generation example.

with me”, which is also covered by its phrase.

4.4 Generation example

Finally, we show an example in Table 5, where the top is the input AMR graph, and the bottom is the generation result. Generally, most of the meaning of the input AMR are correctly translated, such as “:example”, which means “such as”, and “thing”, which is an abstract concept and should not be translated, while there are a few errors, such as “that” in the result should be “what”, and there should be an “in” between “tmt” and “fairfax”.

5 Conclusion

We showed that synchronous node replacement grammar is useful for AMR-to-text generation by developing a system that learns a synchronous NRG in the training time, and applies a graph transducer to collapse input AMR graphs and generate output strings according to the learned grammar at test time. Our method performs better than the previous systems, empirically proving the advantages of our graph-to-string rules.

Acknowledgement

This work was funded by a Google Faculty Research Award. Yue Zhang is funded by NSFC61572245 and T2MOE201301 from Singapore Ministry of Education.

References

- Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage CCG semantic parsing with AMR. In *Conference on Empirical Methods in Natural Language Processing (EMNLP-15)*. pages 1699–1710.
- Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*. pages 178–186.
- David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL-05)*. Ann Arbor, Michigan, pages 263–270.
- Frank Drewes, Hans-Jörg Kreowski, and Annegret Habel. 1997. Hyperedge replacement, graph grammars. *Handbook of Graph Grammars* 1:95–162.
- J. Engelfriet and G. Rozenberg. 1997. Node replacement graph grammars. In Grzegorz Rozenberg, editor, *Handbook of Graph Grammars and Computing by Graph Transformation*, World Scientific Publishing Co., Inc., River Edge, NJ, USA, pages 1–94.
- Jeffrey Flanigan, Chris Dyer, Noah A. Smith, and Jaime Carbonell. 2016. Generation from abstract meaning representation using tree transducers. In *Proceedings of the 2016 Meeting of the North American chapter of the Association for Computational Linguistics (NAACL-16)*. pages 731–739.
- 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 (ACL-14)*. pages 1426–1436.
- Daniel Gildea. 2003. Loosely tree-based alignment for machine translation. In *Proceedings of the 41th Annual Conference of the Association for Computational Linguistics (ACL-03)*. Sapporo, Japan, pages 80–87.
- James Goodman, Andreas Vlachos, and Jason Naradowsky. 2016. Noise reduction and targeted exploration in imitation learning for abstract meaning representation parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL-16)*. Berlin, Germany, pages 1–11.
- Jonas Groschwitz, Alexander Koller, and Christoph Teichmann. 2015. Graph parsing with s-graph grammars. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL-15)*. Beijing, China, pages 1481–1490.
- Liang Huang, Kevin Knight, and Aravind Joshi. 2006. Statistical syntax-directed translation with extended domain of locality. In *Proceedings of Association for Machine Translation in the Americas (AMTA-2006)*. pages 66–73.
- 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 International Conference on Computational Linguistics (COLING-12)*. pages 1359–1376.
- Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Meeting of the North American chapter of the Association for Computational Linguistics (NAACL-03)*. pages 48–54.
- Xiang Li, Thien Huu Nguyen, Kai Cao, and Ralph Grishman. 2015. Improving event detection with abstract meaning representation. In *Proceedings of the First Workshop on Computing News Storylines*. Beijing, China, pages 11–15.
- Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics (ACL-06)*. Sydney, Australia, pages 609–616.
- Fandong Meng, Jun Xie, Linfeng Song, Yajuan Lü, and Qun Liu. 2013. Translation with source constituency and dependency trees. In *Conference on Empirical Methods in Natural Language Processing (EMNLP-13)*. Seattle, Washington, USA, pages 1066–1076.
- Arindam Mitra and Chitta Baral. 2015. Addressing a question answering challenge by combining statistical methods with inductive rule learning and reasoning. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-16)*.
- Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL-03)*. Sapporo, Japan, pages 160–167.
- Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-02)*. pages 311–318.
- Xiaochang Peng, Linfeng Song, and Daniel Gildea. 2015. A synchronous hyperedge replacement grammar based approach for AMR parsing. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning (CoNLL-15)*. pages 731–739.
- 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 (EACL-17). Valencia, Spain, pages 366–375.
- Nima Pourdamghani, Kevin Knight, and Ulf Hermjakob. 2016. Generating English from abstract meaning representations. In *International Conference on Natural Language Generation (INLG-16)*. Edinburgh, UK, pages 21–25.
- Michael Pust, Ulf Hermjakob, Kevin Knight, Daniel Marcu, and Jonathan May. 2015. Parsing English into abstract meaning representation using syntax-based machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP-15)*. pages 1143–1154.
- Libin Shen, Jinxi Xu, and Ralph Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL-08)*. Columbus, Ohio, pages 577–585.
- Linfeng Song, Yue Zhang, Xiaochang Peng, Zhiguo Wang, and Daniel Gildea. 2016. AMR-to-text generation as a traveling salesman problem. In *Conference on Empirical Methods in Natural Language Processing (EMNLP-16)*. Austin, Texas, pages 2084–2089.
- Sho Takase, Jun Suzuki, Naoaki Okazaki, Tsutomu Hirao, and Masaaki Nagata. 2016. Neural headline generation on abstract meaning representation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP-16)*. Austin, Texas, pages 1054–1059.
- Aleš Tamchyna, Chris Quirk, and Michel Galley. 2015. A discriminative model for semantics-to-string translation. In *Proceedings of the 1st Workshop on Semantics-Driven Statistical Machine Translation (S2MT 2015)*. Beijing, China, pages 30–36.
- Lucy Vanderwende, Arul Menezes, and Chris Quirk. 2015. An AMR parser for English, French, German, Spanish and Japanese and a new AMR-annotated corpus. In *Proceedings of the 2015 Meeting of the North American chapter of the Association for Computational Linguistics (NAACL-15)*. pages 26–30.
- Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015. A transition-based algorithm for AMR parsing. In *Proceedings of the 2015 Meeting of the North American chapter of the Association for Computational Linguistics (NAACL-15)*. pages 366–375.
- Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational linguistics* 23(3):377–403.
- Jun Xie, Haitao Mi, and Qun Liu. 2011. A novel dependency-to-string model for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP-11)*. Edinburgh, Scotland, UK., pages 216–226.
- Kenji Yamada and Kevin Knight. 2002. A decoder for syntax-based statistical MT. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-02)*. Philadelphia, Pennsylvania, USA, pages 303–310.
- Junsheng Zhou, Feiyu Xu, Hans Uszkoreit, Weiguang QU, Ran Li, and Yanhui Gu. 2016. AMR parsing with an incremental joint model. In *Conference on Empirical Methods in Natural Language Processing (EMNLP-16)*. Austin, Texas, pages 680–689.