UofR at SemEval-2016 Task 8: Learning Synchronous Hyperedge Replacement Grammar for AMR Parsing

Xiaochang Peng and **Daniel Gildea**
Department of Computer Science
University of Rochester
Rochester, NY 14627

Abstract

In this paper, we apply a synchronous-graph-grammar-based approach to SemEval-2016 Task 8, Meaning Representation Parsing. In particular, we learn Synchronous Hyperedge Replacement Grammar (SHRG) rules from aligned pairs of sentences and AMR graphs. Then we use Earley algorithm with cube-pruning for AMR parsing given new sentences and the learned SHRG. Experiments on the evaluation dataset demonstrate that competitive results can be achieved using a SHRG-based approach.

1 Introduction

Abstract Meaning Representation (AMR) (Banarescu et al., 2013) is a semantic formalism where the meaning of a sentence is encoded as a rooted, directed graph. Figure 1 shows an example of the edge-labeled representation of an AMR graph, where the edges are labeled while the nodes are not. AMR utilizes PropBank frames, non-core semantic roles, coreference, named entity annotations and other semantic phenomena to represent the semantic structure of a sentence and abstracts away its syntax form. These properties render AMR representation useful in applications like question answering and semantics-based machine translation.

SemEval-2016 Task 8 is the task of recovering this type of semantic formalism for plain text. A large corpus of annotated English/AMR pairs is provided to learn this mapping. Hyperedge replacement grammar (HRG) is a context-free rewriting formalism for generating graphs (Drewes et al., 1997). Its
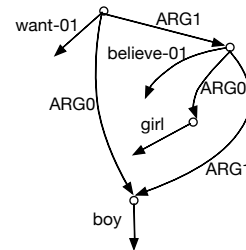

Figure 1: An example of AMR graph representing the meaning of: "The boy wants the girl to believe him"

synchronous counterpart, SHRG, can be used for transforming a graph from/to another structured representation such as a string or tree structure. Therefore, an SHRG-based approach can be used for AMR parsing. Previous approaches usually first map the components of the sentence to components of the graph. Then different supervised algorithms are used to assemble these graph components to generate a complete AMR graph (Flanigan et al., 2014; Wang et al., 2015b; Wang et al., 2015a).

Previously, we have developed a system that learns SHRG rules from sentence/AMR graph pairs (Peng et al., 2015), with automatic alignments extracted from JAMR (Flanigan et al., 2014). During the decoding procedure, we also use the concept identification results from Flanigan et al. (2014). The system is evaluated on the newswire section of LDC2013E117, which has around 4000 sentence-AMR pairs as training data.

In this paper, we extend this system by using the alignments from Ulf Hermjakob's automatic aligner and building a perceptron-based concept identifier where the boundary information of the mapped frag-
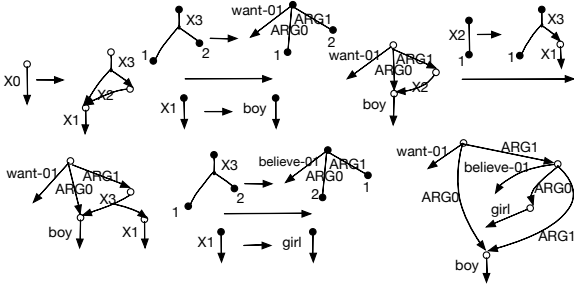
Figure 2: The series of HRG rules applied to derive the AMR graph of "The boy wants the girl to believe him". The first rule is directly shown. The other HRG rules are either above or below each right arrow. The white circle shows the root of each hyperedge. The indexes in each rule show the one-to-one mapping between the attachment nodes of l.h.s. nonterminal edges and the external nodes of the r.h.s. subgraph

ments is captured. We first introduce the overall pipeline of our parser. Then we describe the SHRG learning and the AMR parsing procedure in detail in Section 4 and Section 5. Finally we show the experimental results on the SemEval-2016 evaluation datasets.

2 Overall System Description

Our system is divided into two major components: SHRG learning and AMR parsing. Given English/AMR pairs and the automatic alignments from Ulf Hermjakob, we build a derivation forest representation of possible derivations. Markov Chain Monte Carlo (MCMC) algorithms are applied to sample a series of SHRG rules that generate each sentence/AMR pair.

Given the extracted SHRG and new sentences, we first identify the component spans on the string side. Then we use a perceptron classifier to find the graph fragment aligned to each of these spans. Finally, we use a decoder similar to those used for Synchronous Context-Free Grammar (SCFG) in machine translation, where the graph-side derivation is composed using HRG derivation instead of CFG, to get the AMR graphs for these sentences.

3 Hyperedge Replacement Grammar

HRG is similar to CFG in that it rewrites nonterminals independently. While CFG generates natural language strings by successively rewriting nonter-

minal tokens, the nonterminals in HRG are hyperedges, and each rewriting step in HRG replaces a hyperedge nonterminal with a subgraph instead of a span of a string.

The rewriting mechanism replaces a nonterminal hyperedge with the graph fragment specified by a production's righthand side (r.h.s.), attaching each external node of the r.h.s. to the corresponding attachment node of the lefthand side. Take Figure 2 as an example. Starting from our initial hypergraph with one edge labeled with the start symbol "$X0$", we select one edge with a nonterminal label in our current hypergraph, and rewrite it using a rule in our HRG. The first rule rewrites the start symbol with a subgraph shown on the r.h.s. We continue the rewriting steps until there are no more nonterminal-labeled edges.

We use the synchronous counterpart of HRG where the source side is a CFG and the target side is an HRG. Given such a synchronous grammar and a string as input, we can parse the string with the CFG side and then derive the counterpart graph by deduction from the derivation. The benefit of parsing with SHRG is that the complexity is bounded by that of CFG parsing. Table 2 shows the rule format of our SHRG. For each nonterminal $Xi\text{-}b_1 \cdots b_i$, i defines the type of the nonterminal, while each b_i indicates whether the i-th external node will have a concept edge in the rewriting result. This design guarantees that there is exactly one concept edge going out of each node.

4 Learning Synchronous Hyperedge Replacement Grammar

The **fragment decomposition forest** represents all possible ways to decompose the sentence-AMR pairs into component-level alignments. It also encodes a compact representation of possible SHRG derivations that are consistent with the alignments.

Therefore, starting with the component-level alignment pairs, we keep composing large span/fragment pairs from bottom up to build the derivation forest. Then we use Gibbs sampling to sample one derivation from the forest representation.

	JAMR	Ulf Hermjakob
string side	span, can be multiple	single token
graph side	concept fragments	single concept/relation
prepositions	usually not aligned	relation edges
mapping type	one-to-one	multiple-to-multiple

Table 1: Comparisons of English-AMR alignments from JAMR and Ulf Hermjakob
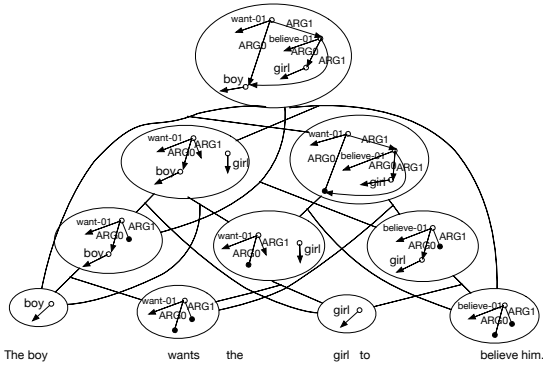

Figure 3: The fragment decomposition forest for the (sentence, AMR graph) pair for "The boy wants the girl to believe him"

4.1 Constructing Derivation Forests

We use the alignments from Ulf Hermjakob's aligner, which are provided for the training data. Some differences between this alignment and the alignment from JAMR (Flanigan et al., 2014) are summarized in Table 1. As the alignment is between a single token and a single concept/relation, we first heuristically identify the mappings of name entities by tracing the :*op* and :*name* relations. For each of the other aligned tokens, we compose its aligned concept or fragment.[1] There are also situations where multiple tokens are aligned to the same concept or relation. If they are adjacent, we compose them on the string side to form a larger span. Otherwise we delete the alignment of the shorter span to make the alignment one-to-one. We additionally use lemma information to retrieve some mappings from unaligned tokens to unaligned concepts.[2]

After we have extracted these alignments, we build the fragment decomposition forest from bottom up and left to right, gradually composing larger

---

[1]Our parser currently does not support multi-root graph fragments, so for alignments that align one token to multiple disconnected fragments, we only choose the largest fragment as the aligned fragment.

[2]Following Peng et al. (2015), we attach :*ARG*s and :*op*s to their head concepts and :*ARGx-of*s to their tail concepts.

(span, fragment) pairs from smaller ones until we have reached the root of the forest: the whole sentence-AMR pair. We maintain an ordered list of external nodes for each fragment, which will be used to track the order of the external nodes during the rule extraction phase. Figure 3 shows an example of a fragment decomposition forest. The forest construction procedure is described in detail in Peng et al. (2015).

Each node in the forest represents a span-fragment pair. We extract all span-fragment pairs of length smaller than 7 to construct a span to fragments alignment table Φ, which will be used during the concept identification procedure in Section 5.1

4.2 SHRG Learning Using MCMC

The fragment decomposition forest represents possible derivations of the sentence/AMR pairs in terms of minimal rules extracted from the alignments. We use an MCMC algorithm to learn a grammar of larger rules, by sampling both which minimal rules are used, and how minimal rules combine to form larger composed rules.

We sample two types of variables: an edge variable e_n representing which incoming hyperedge is chosen at a given node n in the forest (allowing us to sample one tree from a forest) and a cut variable z_n representing whether node n in the forest is a boundary between two SHRG rules or is internal to an SHRG rule (allowing us to sample rules/fragments from a tree). We use an MCMC algorithm to sample from top-down and one variable at a time (Peng and Gildea, 2014). Sampling tree fragments from forests is described in detail in Chung et al. (2014). Table 2 shows some examples of the sampled SHRG rules.

5 AMR parsing

In Section 4 we have described how we extract the SHRG from the training data. Now given a new sentence, we first use the perceptron algorithm to identify the graph fragments each span in the sentence aligns to. Then we use the Earley algorithm to decode the sentence and recover its AMR graph.

5.1 Concept Identification

First we identify the segmentation of the sentence. We use the Illinois Named Entity Tagger (NER) (Ratinov and Roth, 2009) to identify all the

[A1-1] → ordinary people | (. :p/person :mod (. :o/ordinary))
[A0-1] → [A1-1,1] say that [A1-1,2] |
(. :s/say-01 :ARG0 (. :[A1-1,1]) :ARG1 (. :[A1-1,2]))
[A0-1] → Do [A1-1,1] ? | (. :[A1-1,1] :mode (. :interrogative))

Table 2: Some example of learned rules

	Precision	Recall	F-score
Dev	0.57	0.55	0.56
Test	0.56	0.55	0.55
Task 8 eval	-	-	0.50

Table 3: Parsing results on dev and test LDC2015E86 and the evaluation data of SemEval-2016 Task 8

named entities. We also use heuristic rules to identify dates, time expressions etc. We heuristically build the AMR fragments for these spans and add the mappings to Φ.[3] The other spans are of length 1. Then we use the perceptron algorithm to predict the AMR fragment for each span:

$$\hat{g}[e] = \arg\max_{g \in \Phi(e)} w^T f(e, g, c) \quad (1)$$

where w is the weights, and f is the feature function. e is the current span, g is a candidate graph fragment from the span to fragment table Φ or matched using regular expressions. c is the context of span e. We use the following context features:

1. 3 words, pos tags, lemmas before and after the current span
2. 3 word, pos tag or lemma bigrams before and after the current span
3. the words, pos tags and lemmas in the current span
4. Length of the span

Our concept identification result is different from Flanigan et al. (2014) in that we are predicting not only the graph fragments but also the external nodes of each fragment. Therefore, each identified mapping is essentially a lexical SHRG rule at the leaf-level of the derivation tree. We add these identified lexical rules to the extracted SHRG locally before decoding each sentence. For lexical rules in Φ that cover more than 1 span of the sentence on the string side, we also add them to the SHRG locally.[4]

5.2 Decoding

Given the learned SHRG, we are simply parsing on the string side. We use the Earley algorithm with cube-pruning (Chiang, 2007) for the string-to-AMR

parsing. For each synchronous rule with N nonterminals on its l.h.s., we build an $N + 1$ dimensional cube and generate the top K candidates. Out of all the hypotheses generated by all satisfied rules within each span (i, j), we keep at most K candidates for this span. Our glue rules generate a pseudo *m/multisentence* concept and use *ARG* relations to connect disconnected components to make the result graph connected. The features used are described in Peng et al. (2015).

6 Experiments

We evaluate our parser on the LDC2015E86 dataset, which includes 16833 training, 1368 dev, and 1371 test sentences.

During the sampling procedure, all the cut variables in the derivation forest are initialized as 1 and an incoming hyperedge is sampled uniformly for each node. We run the sampler for 160 iterations and combine the grammar dumped every 10th iteration.

The performance of our SHRG-based parser is evaluated using Smatch v2.0.2 (Cai and Knight, 2013), which evaluates the precision, recall, and $F1$ of the concepts and relations all together. Table 3 shows the results on the dev and test set. We also report smatch score on the shared task evaluation data, which includes 1053 sentences. The smatch score on the evaluation data is 0.50. This score is much lower than the performance on the dev and test data. The reason might be that the evaluation data is much harder and includes more noise, which can break down the structure of the learned grammar.

The results show that SHRG-based parsing can be a viable approach for AMR parsing. Currently our system only uses a few simple features and all the weights are tuned by hand. The performance could be improved by using more complicated features and tuning their weights automatically. It would also be helpful to use external resources such as the com-

---

[3]As the named entity labels from Illinois NER is only a few, we predict their entity label from an entity label set built on the training data.

[4]We filter some rules that have non-content words on the left or right because such rules introduce a lot of errors from the alignments.

mon organizational roles, relational roles and the verbalization lists [5] and use fallback techniques to deal with unknown words.

7 Conclusion

In this paper, we have presented our SHRG-based AMR parsing system for SemEval-2016 Task 8. Our system extends from the work of Peng et al. (2015) and has shown some competitive results for a graph-grammar-based approach for AMR parsing. Currently our system only uses local features for decoding; it would be interesting to extend this system to incorporate a language model on the graph side and use discriminative models to incorporate more global features and tune the weights automatically.

Acknowledgments

References

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*.

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *ACL (2)*, pages 748–752.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Tagyoung Chung, Licheng Fang, Daniel Gildea, and Daniel Štefankovič. 2014. Sampling tree fragments from forests. *Computational Linguistics*, 40(1):203–229.

Frank Drewes, Hans-Jörg Kreowski, and Annegret Habel. 1997. Hyperedge replacement, graph grammars. In *Handbook of Graph Grammars*, volume 1, pages 95–162. World Scientific, Singapore.

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

Xiaochang Peng and Daniel Gildea. 2014. Type-based MCMC for sampling tree fragments from forests. In *Conference on Empirical Methods in Natural Language Processing (EMNLP-14)*.

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*, pages 32–41, Beijing, China.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 147–155, Boulder, Colorado, June. Association for Computational Linguistics.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015a. Boosting transition-based AMR parsing with refined actions and auxiliary analyzers. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 857–862, Beijing, China, July. Association for Computational Linguistics.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015b. A transition-based algorithm for AMR parsing. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 366–375, Denver, Colorado, May–June. Association for Computational Linguistics.

---

[5]http://amr.isi.edu/download.html