CAMR at SemEval-2016 Task 8: An Extended Transition-based AMR Parser

Chuan Wang
Brandeis University
cwang24@brandeis.edu

Sameer Pradhan
cemantix.org
Boulder Learning Inc.
pradhan@cemantix.org

Nianwen Xue
Brandeis University
xuen@brandeis.edu

Xiaoman Pan
Rensselaer Polytechnic Institute
panx2@rpi.edu

Heng Ji
Rensselaer Polytechnic Institute
jih@rpi.edu

Abstract

This paper describes CAMR, the transition-based parser that we use in the SemEval-2016 Meaning Representation Parsing task. The main contribution of this paper is a description of the additional sources of information that we use as features in the parsing model to further boost its performance. We start with our existing AMR parser and experiment with three sets of new features: 1) rich named entities, 2) a verbalization list, 3) semantic role labels. We also use the RPI Wikifier to wikify the concepts in the AMR graph. Our parser achieves a Smatch F-score of 62% on the official blind test set.

1 Introduction

AMR parsing is the task of taking a sentence as input and producing as output an Abstract Meaning Representation (AMR) that is a rooted, directed, edge-labeled and leaf-labeled graph that is used to represent the meaning of a sentence (Banarescu et al., 2013). AMR parsing has drawn an increasing amount of attention recently. The first published AMR parser, JAMR (Flanigan et al., 2014), performs AMR parsing in two stages: concept identification and relation identification. Flanigan et al. (2014) treat concept identification as a sequence labeling task and utilize a semi-Markov model to map spans of words in a sentence to concept graph fragments. For relation identification, they adopt graph-based techniques similar to those used in dependency parsing (McDonald et al., 2005). Instead of finding maximum spanning trees (MST) over words, they propose an algorithm that finds the maximum spanning connected subgraph (MSCG) over concept fragments identified in the first stage.

Wang et al. (2015b) describes a transition-based parser that also involves two stages. In the first step, an input sentence is parsed into a dependency tree with a dependency parser. In the second step, it transforms the dependency tree into an AMR graph by performing a series of actions. Note that the dependency parser used in the first step can be any off-the-shelf dependency parser and does not have to trained on the same data set as used in the second step.

There are also approaches which utilize grammar induction to parse the AMR. Artzi et al. (2015) presents a model that first use Combinatory Categorial Grammar (CCG) to construct the lambda-calculus representations of the sentence, then further resolve non-compositional dependencies using a factor graph. Peng et al.(2015) and Pust et al.(2015) formalize parsing AMR as a machine translation problem by learning string-graph/string-tree rules from the annotated data.

Although the field of AMR parsing is growing and several systems (Wang et al., 2015a; Artzi et al., 2015; Pust et al., 2015; Flanigan et al., 2014) have substantially advanced the state of the art, the overall performance of existing AMR parsers is far less accurate than syntactic parsers (Charniak and Johnson, 2005). This makes it difficult to use in downstream NLP tasks. In this paper, we aim to boost the AMR parsing performance by introducing additional features. We mainly experiment with three sets of features derived from: 1) rich named entities, 2) a verbalization list provided by ISI, and 3) semantic role labels produced by an automatic SRL system.

The rest of the paper is organized as follows. In Section 2 we briefly describe CAMR, and in Section 3 we describe our extensions for the SemEval shared task. In Section 4 we describe the different AMR releases available with some salient characteristics. We report experimental results in Section 5 and conclude the paper in Section 6.

2 CAMR Overview

2.1 Basic Configuration

CAMR first uses a dependency parser to parse an input sentence, and then performs a small number of highly general actions to transform the resulting dependency tree to an AMR graph. The transition actions are briefly described below but due to the limited space, we cannot provide the full details of these actions here, and the reader is referred to previous work (Wang et al., 2015b) for a detailed description of these actions with illustrating examples. CAMR uses three types of actions: actions performed when an edge is visited, actions performed when a node is visited, and actions used to infer abstract concepts in AMR that does not correspond to any word or word sequence in the sentence.

CAMR performs one of the following six actions when an edge is visited:

- NEXT-EDGE-l_r (ned): Assign the current edge with edge label l_r and go to next edge.
- SWAP-l_r (sw): Swap the current edge, make the current dependent as the new head, and assign edge label l_r to the swapped edge.
- REATTACH$_k$-l_r (reat): Reattach current dependent to node k and assign edge label l_r.
- REPLACE-HEAD (rph): Replace current head node with current dependent node.
- REENTRANCE$_k$-l_r (reen): Add another head node k to current dependent and assign label l_r to edge between k and current dependent.
- MERGE (mrg): Merge two nodes connected by the edge into one node.

From each node in the dependency tree, CAMR performs the following two actions:

- NEXT-NODE-l_c (nnd): Assign the current node with concept label l_c and go to next node.
- DELETE-NODE (dnd): Delete the current node and all edges associated with current node.

Finally CAMR infers **abstract concepts** that are not aligned to any tokens in sentence with an INFER-l_c action. The INFER-l_c action works as follows: when the parser visits an node in dependency tree, it inserts an abstract node with concept label l_c right between the current node and its parent. For example in Figure 1, after applying action INFER-`have-org-role-91` on node *minister*, the abstract concept is recovered and subsequent actions can be applied to transform the subgraph to its correct AMR.

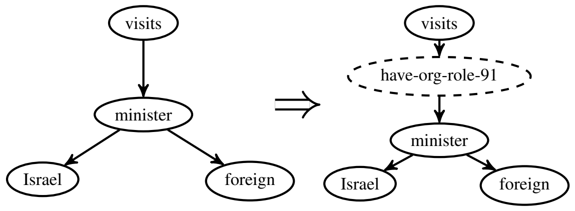


Figure 1: INFER-`have-org-role-91` action

3 CAMR Extensions

3.1 Feature Enrichment

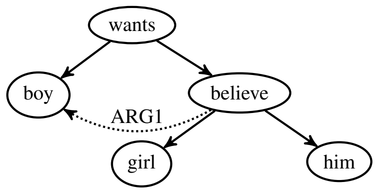
Rich named entity tags Since named entity types in AMR are much more fine-grained than the named entity types defined in a typical named entity tagging system, we assume that using a richer named entity tagger could improve concept identification in parsing. Here we use the 18 named entity types defined in the OntoNotes v5.0 Corpus (Weischedel et al., 2011; Pradhan et al., 2013).

The ISI verbalization list A large proportion of AMR concepts are "normalized" English words. This typically involves cases where the verb form of a noun or an adjective is used as the AMR concept. For example, the AMR concept "attract-01" is used for the adjective "attractive". Similarly, the noun "globalization" would invoke the AMR concept "globalize-01". To help CAMR produce these AMR concepts correctly, we use the verbalization-list provided by ISI[1] to improve the word-to-AMR-concepts alignment. If any alignment is missed by the JAMR aligner and left un-aligned, we simply add

[1]`http://amr.isi.edu/download/lists/verbalization-list-v1.01.txt`

an alignment to map the unaligned concept to its corresponding word token if the word token in the input sentence is in the verbalization list.

semantic role labeling:
wants, want-01,
ARG0: the boy, ARG1: the girl to believe him



For action NEXT-NODE-`want-01`
EQ_FRAMESET: `true`

Figure 2: An example of semantic role labeling feature in partial parsing graph of sentence,"The boy wants the girl to believe him."

Semantic role labeling features We use the following semantic role labeling features: 1) EQ_FRAMESET. For actions that predict the concept label (NEXT-NODE-l_c), we check whether the candidate concept label l_c matches the frameset predicted by ASSERT (Pradhan et al., 2004). For example, in the partial graph in Figure 2, when we examine node $wants$, one of the candidate actions would be NEXT-NODE-`want-01`. Since the candidate concept label `want-01` is equal to node $wants$'s frameset `want-01` as predicted by ASSERT, the value of feature EQ_FRAMESET is set to `true`. 2) IS_ARGUMENT. For actions that predict the edge label, we check whether ASSERT predicts that the current dependent is an argument of the current head. Note that arguments output by the semantic role labeler are typically constituents in a syntactic tree. We find the head of the argument and match it against the dependent. If the argument predicted by ASSERT matches the dependent, the value of the IS_ARGUMENT is set to `true`.

Word Clusters For the semi-supervised word cluster feature, we use Brown clusters, more specifically, the 1000-class word clusters trained by Turian et al. (2010). We use prefixes of lengths 4, 6, 10 and 20 of the word's bit-string as features.

3.2 Wikification

We apply an AMR based wikification system (Pan et al., 2015) which utilizes AMR to represent semantic information about entity mentions expressed in their textual context. Given an entity mention m, this system first constructs a Knowledge Graph $g(m)$ with m at the hub and leaf nodes obtained from entity mentions reachable by AMR graph traversal from m. A subset of the leaf nodes are selected as collaborators of m. Mentions connected by AMR conjunction relations are grouped into sets of coherent mentions. For each entity mention m, an initial ranked list of entity candidates $E = (e_1, \ldots, e_n)$ is generated based on a salience measure (Medelyan and Witten, 2008). Then a Knowledge Graph $g(e_i)$ is generated for each entity candidate e_i in m's entity candidate list E. The entity candidates are then re-ranked according to Jaccard Similarity, which computes the similarity between $g(m)$ and $g(e_i)$: $J(g(m), g(e_i)) = \frac{|g(m) \cap g(e_i)|}{|g(m) \cup g(e_i)|}$. Finally, the entity candidate with the highest score is selected as the appropriate entity for wikifying m. Moreover, the Knowledge Graphs of coherent mentions will be merged and wikified collectively.

4 Data

Research on AMR parsing so far has reported on two releases of the annotated data. This SemEval evaluation adds another release. The main difference between the SemEval release and the previous two releases is that the SemEval release contains wikification information which was absent from the previous two releases. Since the Smatch scorer uses this information as part of its scoring algorithm, we cannot make comparisons between results on the SemEval release and results previously reported by other systems. We summarize the characteristics of these three releases below.

a. **LDC2013E117**—This is a non-public release that was used to report the very first results on AMR parsing. The first results were reported only on a subset of the test partition—The newswire proxy section. We do not report performance on this release in this paper.
b. **LDC2014T12**—This was the first public release of the AMR data through LDC. Comparable numbers have been reported on this release

for the same newswire proxy section as in (a.) as well as the entire test set.

c. **LDC2015E86**—This is the release specifically made available for the SemEval evaluation. One main difference between this version and the two previous versions is the addition of wikification information. Thus, the performance numbers on the full test set of this release are not directly comparable with previously published results on either of the other two releases.

In the following sections, we will report experiments primarily on the Semeval release (c.). In Section 5.3 we also use the full test set of release (b.) to evaluate the performance improvement made to CAMR as part of the SemEval evaluations against previously reported performance.

Syntactic Parser	P	R	F_1
Charniak (ON)	70.76	60.57	65.27
Charniak (WSJ)	69.88	60.24	64.70

Table 1: AMR parsing performance on the SemEval development set (LDC2015E86) across two Charniak parser models

5 Experiments

We use the official release dataset and standard train/dev/test split of SemEval Task 8 for experiments. All the sentences are preprocessed using Stanford CoreNLP (Manning et al., 2014) to get tokenization, lemma, named entity tag, POS tag. And we use the aligner that comes with JAMR (Flanigan et al., 2014) to align the sentence with its AMR graph. We then parse the tokenized sentences using Charniak parser (Charniak and Johnson, 2005)(Its phrase structure output is converted to dependency structure using a slightly modified version of the Stanford CoreNLP converter). Rich named entity tags are generated using Stanford named entity tagger. The semantic role labels are generated using ASSERT—a semantic role labeler (Pradhan et al., 2005), including a frameset disambiguator trained using a word sense disambiguation system—IMS (Zhong and Ng, 2010). All these components viz., the Charniak parser, Stanford named entity tagger, ASSERT, and IMS word sense disambiguator were retrained on the OntoNotes v5.0 training data[2] (Pradhan et al., 2013)[3]. We use the version of CAMR described in (Wang et al., 2015a) (without the feature extensions) as the baseline. We evaluate our parser with Smatch v2.0.2 (Cai and Knight, 2013) on all the experiments. It should be noted that all the rows in Table 2 except for the last one get implicitly penalized by the scorer for lack of wikification information.

5.1 SemEval Development Set

As discussed in (Wang et al., 2015a), the performance of the syntactic parser in the first stage has a high impact on the AMR parsing accuracy. We first do a sanity check to choose the best first stage parser. Here we only consider two scenarios: the Charniak parser trained on WSJ and OntoNotes, as shown in Table 1. As using the Charniak parser trained on OntoNotes yields slightly better AMR parsing result, we will use this set-up for the following experiments.

In Table 2 we present results from extending CAMR. All experiments are conducted on the SemEval development set. we can see that the three major improvements are given by adding the verbalization list, semantic role labels and wikification separately. Rich named entities also yield a 0.4% point improvement, indicating that the more fine-grained named entity tagger is helpful to concept identification. In contrast, Brown cluster features actually hurt the overall performance. Therefore we did not use them in the configuration used for the official run.

5.2 SemEval Test Set and Blind Test Set

We evaluate our parser on the SemEval test set and also report the evaluation result on the SemEval blind test set with the best configuration obtained from §5.1, as shown in Table 3.

[2]Details of the training data and the trained models can be found at `http://cemantix.org/data/ontonotes.html` and `https://github.com/ontonotes/conll-formatted-ontonotes-5.0/releases/tag/v12`

[3]We excluded documents from the OntoNotes v5.0 training and development partitions that overlapped with the SemEval AMR data. List of overlapping document IDs is available at `http://cemantix.org/ontonotes/ontonotes-amr-document-overlap.txt`

Feature configuration	P	R	F_1
Baseline	70.76	60.57	65.27
+VERB	71.52	60.96	65.82
+VERB+BROWN	71.85	60.38	65.62
+VERB+RNE	71.89	61.02	66.01
+VERB+RNE+SRL	72.33	61.40	66.56
+VERB+RNE+SRL+WIKI	71.17	63.89	67.33

Table 2: AMR parsing performance on the official SemEval development set (LDC2015E86).VERB: ISI verbalization list. BROWN: Brown cluster features.RNE: Rich (OntoNotes) named entities.SRL: semantic role labeling features. WIKI: Addition of wikification of named entities in AMR.

Dataset	P	R	F_1
Test Set	70.36	63.12	66.54
Blind Test Set	67.44	57.39	62.01

Table 3: AMR parsing performance on full SemEval Test Set and the Blind Test Set

From Table 3 we can see that our parser remain relatively stable on the SemEval test set. However, the evaluation result on blind test set dropped by around 4 points, indicating the blind test set is much harder and we plan to do further error analysis to gain more insight on the difference.

5.3 Previous Release—LDC2014T12

Since the official dataset of SemEval is annotated with wiki relations that previous releases of the AMR Corpus do not have, we conduct additional experiments on the AMR annotation release 1.0 (LDC2014T12) to gain a clear understanding of the impact of the additional feature. We use the training/development/test split recommended in the release: 10,312 sentences for training, 1,368 sentences for development and 1,371 sentences for testing. We re-train the parser on the LDC2014T12 training set with the best parser configuration given in §5.1 — except for the wikification pass— and test the parser on the full test set. The result is shown in Table 4. For comparison, we include the result of our parser in (Wang et al., 2015a) which are also trained on the same dataset. The results show that the new features yield a modest improvement over our (Wang et al., 2015a) parser.

CAMR version	P	R	F_1
This paper	**71.35**	**62.29**	**66.51**
Wang et al. (2015a)	70.29	62.01	65.89

Table 4: CAMR parsing performance on the full test set of release LDC2014T12.

6 Conclusion

We build our system by taking our existing AMR parser and enriching it with three sets of features: 1) rich named entities, 2) a verbalization list, and 3) semantic role labels. We also use a wikifier to resolve the wiki relation in AMR graph. Our results show that the additional features are helpful to the AMR parsing task and a well-designed wikifier could be a helpful post-processing step to AMR parsing.

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*, pages 1699–1710, Lisbon, Portugal, September. Association for Computational Linguistics.

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. Association for Computational Linguistics.

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 (Volume 2: Short Papers)*, pages 748–752. Association for Computational Linguistics.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 173–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

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 (Volume 1: Long Papers)*, pages 1426–1436, Baltimore, Maryland, June. Association for Computational Linguistics.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 523–530, Stroudsburg, PA, USA. Association for Computational Linguistics.

Olena Medelyan and Ian H Witten. 2008. Domain-independent automatic keyphrase indexing with small training sets. *Journal of the American Society for Information Science and Technology*, 59(7):1026–1040.

Xiaoman Pan, Taylor Cassidy, Ulf Hermjakob, Heng Ji, and Kevin Knight. 2015. Unsupervised entity linking with abstract meaning representation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1130–1139, Denver, Colorado, May–June. Association for Computational Linguistics.

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*.

Sameer Pradhan, Wayne Ward, Kadri Hacioglu, James Martin, and Dan Jurafsky. 2004. Shallow semantic parsing using support vector machines. In *Proceedings of the Human Language Technology Conference/North American chapter of the Association of Computational Linguistics (HLT/NAACL)*, Boston, MA, May.

Sameer Pradhan, Kadri Hacioglu, Valerie Krugler, Wayne Ward, James Martin, and Dan Jurafsky. 2005. Support vector learning for semantic argument classification. *Machine Learning Journal*, 60(1):11–39.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using OntoNotes. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 143–152, Sofia, Bulgaria, August.

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*, pages 1143–1154, Lisbon, Portugal, September. Association for Computational Linguistics.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. 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.

Ralph Weischedel, Eduard Hovy, Mitchell Marcus, Martha Palmer, Robert Belvin, Sameer Pradhan, Lance Ramshaw, and Nianwen Xue. 2011. OntoNotes: A large training corpus for enhanced processing. Springer.

Zhi Zhong and Hwee Tou Ng. 2010. It makes sense: A wide-coverage word sense disambiguation system for free text. In *Proceedings of the ACL 2010 System Demonstrations*, pages 78–83, Uppsala, Sweden.