Abstract

We report improved AMR parsing results by adding a new action to a transition-based AMR parser to infer abstract concepts and by incorporating richer features produced by auxiliary analyzers such as a semantic role labeler and a coreference resolver. We report final AMR parsing results that show an improvement of 7% absolute in $F_1$ score over the best previously reported result. Our parser is available at: https://github.com/Juicechuan/AMRParsing

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.

A competitive alternative to the MSCG approach is transition-based AMR parsing. Our previous work (Wang et al., 2015) describes a transition-based system 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, the transition-based AMR parser 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.

Figure 1: An example showing abstract concept have-org-role-91 for the sentence “Israel foreign minister visits South Korea.”

Unlike a dependency parse where each leaf node corresponds to a word in a sentence and there is an inherent alignment between the words in a sentence and the leaf nodes in the parse tree, the alignment between the word tokens in a sentence and the concepts in an AMR graph is non-trivial. Both JAMR and our transition-based parser rely on a heuristics based aligner that can align the words in a sentence and concepts in its AMR with a 90% $F_1$ score, but there are some concepts in the AMR that cannot be aligned to any word in a sentence.

This is illustrated in Figure 1 where the concept have-org-role-91 is not aligned to any word or word sequence. We refer to these concepts as abstract concepts, and existing AMR parsers do not have a systematic way of inferring such abstract concepts.
Current AMR parsers are in their early stages of development, and their features are not yet fully developed. For example, the AMR makes heavy use of the framesets and semantic role labels used in the Proposition Bank (Palmer et al., 2005), and it would seem that information produced by a semantic role labeling system trained on the Proposition Bank can be used as features to improve the AMR parsing accuracy. Similarly, since AMR represents limited within-sentence coreference, coreference information produced by an off-the-shelf coreference system should benefit the AMR parser as well.

In this paper, we describe an extension to our transition-based AMR parser (Wang et al., 2015) by adding a new action to infer the abstract concepts in an AMR, and new features derived from an off-the-shelf semantic role labeling system (Pradhan et al., 2004) and coreference system (Lee et al., 2013). We also experimented with adding Brown clusters as features to the AMR parser. Additionally, we experimented with using different syntactic parsers in the first stage. Following our previous work, we use the averaged perceptron algorithm (Collins, 2002) to train the parameters of the model and use the greedy parsing strategy during decoding to determine the best action sequence to apply for each training instance. Our results show that (i) the transition-based AMR parser is very stable across the different parsers used in the first stage, (ii) adding the new action significantly improves the parser performance, and (iii) semantic role information is beneficial to AMR parsing when used as features, while the Brown clusters do not make a difference and coreference information slightly hurts the AMR parsing performance.

The rest of the paper is organized as follows. In Section 2 we briefly describe the transition-based parser, and in Section 3 we describe our extensions. We report experimental results in Section 4 and conclude the paper in Section 5.

2 Transition-based AMR Parser

The transition-based parser first uses a dependency parser to parse an input sentence, and then performs a limited number of highly general actions to transform the dependency tree to an AMR graph. The transition actions are briefly described below but due to the limited space, we cannot describe the full details of these actions, and the reader is referred to our previous work (Wang et al., 2015) for detailed descriptions of these actions:

- **Next-Edge**-l<sub>r</sub> (ned): Assign the current edge with edge label l<sub>r</sub> and go to next edge.
- **Swap**-l<sub>r</sub> (sw): Swap the current edge, make the current dependent as the new head, and assign edge label l<sub>r</sub> to the swapped edge.
- **Reattach**<sub>k</sub>-l<sub>r</sub> (reat): Reattach current dependent to node k and assign edge label l<sub>r</sub>.
- **Replace-head** (rph): Replace current head node with current dependent node.
- **Reentrance**<sub>k</sub>-l<sub>r</sub> (reen): Add another head node k to current dependent and assign label l<sub>r</sub> 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, the parser performs the following 2 actions:

- **Next-Node**-l<sub>c</sub> (nnd): Assign the current node with concept label l<sub>c</sub> and go to next node.
- **Delete-Node** (dnd): Delete the current node and all edges associated with current node.

Crucially, none of these actions can infer the types of abstract concepts illustrated in Figure 1. And this serves as our baseline parser.

![Figure 2: Enhanced Span Graph for AMR in Figure 1, “Israel foreign minister visits South Korea.”](image)

s<sub>x,y</sub> corresponds to sentence span (x, y).

3 Parser Extensions

3.1 Inferring Abstract Concepts

We previously create the learning target by representing an AMR graph as a Span Graph, where each AMR concept is annotated with the text span...
of the word or the (contiguous) word sequence it is aligned to. However, abstract concepts that are not aligned to any word or word sequence are simply ignored and are unreachable during training. To address this, we construct the span graph by keeping the abstract concepts as they are in the AMR graph, as illustrated in Figure 2.

In order to predict these abstract concepts, we design an INFERENCE action that is applied in the following way: 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 3, after applying action INFERENCE-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 3: INFERENCE-have-org-role-91 action](image)

### 3.2 Feature Enrichment

In our previous work, we only use simple lexical features and structural features. We extend the feature set to include (i) features generated by a semantic role labeling system—ASSERT (Pradhan et al., 2004), including a frameset disambiguator trained using a word sense disambiguation system—IMS (Zhong and Ng, 2010) and a coreference system (Lee et al., 2013) and (ii) features generated using semi-supervised word clusters (Turian et al., 2010; Koo et al., 2008).

**Coreference features** Coreference is typically represented as a chain of mentions realized as noun phrases or pronouns. AMR, on the other hand, represents coreference as re-entrance and uses one concept to represent all co-referring entities. To use the coreference information to inform AMR parsing actions, we design the following two features: 1) SHARE_DEPENDENT. When applying REENTRANCE$_{l_c}$-action on edge $(a, b)$, we check whether the corresponding head node $k$ of a candidate concept has any dependent node that co-refers with current dependent $b$. 2) DEPENDENT_LABEL. If SHARE_DEPENDENT is true for head node $k$ and assuming $k$’s dependent $m$ co-refers with the current dependent, the value of this feature is set to the dependency label between $k$ and $m$.

For example, for the partial graph shown in Figure 4, when examining edge (wants, boy), we may consider REENTRANCE$_{believe}$-ARG1 as one of the candidate actions. The candidate head believe has dependent him which is co-referred with current dependent boy, therefore the value of feature SHARE_DEPENDENT is set to true for this candidate action. Also the value of feature DEPENDENT_LABEL is dobj given the dependency label between (believe, him).

**Semantic role labeling features** We use the following semantic role labeling features: 1) EQ_FRAMESET. For action that predicts the concept label (NEXT_NODE-$l_c$), we check whether the candidate concept label $l_c$ matches the frameset predicted by the semantic role labeler. For example, for partial graph in Figure 4, when examining 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 the semantic role labeler, the value of feature EQ_FRAMESET is set to true. 2) IS_ARGUMENT. For actions that predicts the edge label, we check whether the semantic role labeler predicts that the current dependent is an argument of the current head. Note that the arguments in semantic role labeler output are non-terminals which corresponds to a sentence span. Here we simply take the head word in the sentence span as the argument.

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

4 Experiments

We first tune and evaluate our system on the newswire section of LDC2013E117 dataset. Then we show our parser’s performance on the recent LDC2014T12 dataset.

4.1 Experiments on LDC2013E117

We first conduct our experiments on the newswire section of AMR annotation corpus (LDC2013E117). The train/dev/test split of dataset is 4.0K/2.1K/2.1K, which is identical to the settings of JAMR. We evaluate our parser with Smatch v2.0 (Cai and Knight, 2013) on all the experiments.

<table>
<thead>
<tr>
<th>System</th>
<th>P</th>
<th>R</th>
<th>F₁</th>
</tr>
</thead>
<tbody>
<tr>
<td>Charniak (ON)</td>
<td>.67</td>
<td>.64</td>
<td>.65</td>
</tr>
<tr>
<td>Charniak</td>
<td>.66</td>
<td>.62</td>
<td>.64</td>
</tr>
<tr>
<td>Stanford</td>
<td>.64</td>
<td>.62</td>
<td>.63</td>
</tr>
<tr>
<td>Malt</td>
<td>.65</td>
<td>.61</td>
<td>.63</td>
</tr>
<tr>
<td>Turbo</td>
<td>.65</td>
<td>.61</td>
<td>.63</td>
</tr>
</tbody>
</table>

Table 1: AMR parsing performance on development set using different syntactic parsers.

<table>
<thead>
<tr>
<th>System</th>
<th>P</th>
<th>R</th>
<th>F₁</th>
</tr>
</thead>
<tbody>
<tr>
<td>Charniak (ON)</td>
<td>.67</td>
<td>.64</td>
<td>.65</td>
</tr>
<tr>
<td>+INFER</td>
<td>.71</td>
<td>.67</td>
<td>.69</td>
</tr>
<tr>
<td>+INFER+BROWN</td>
<td>.71</td>
<td>.68</td>
<td>.69</td>
</tr>
<tr>
<td>+INFER+BROWN+SRL</td>
<td>.72</td>
<td>.69</td>
<td>.71</td>
</tr>
<tr>
<td>+INFER+BROWN+SRL+COREF</td>
<td>.72</td>
<td>.69</td>
<td>.70</td>
</tr>
</tbody>
</table>

Table 2: AMR parsing performance on the development set.

4.1.1 Impact of different syntactic parsers

We experimented with four different parsers: the Stanford parser (Manning et al., 2014), the Charniak parser (Charniak and Johnson, 2005) (Its phrase structure output is converted to dependency structure using the Stanford CoreNLP converter), the Malt Parser (Nivre et al., 2006), and the Turbo Parser (Martins et al., 2013). All the parsers we used are trained on the 02-22 sections of the Penn Treebank, except for CHARNIKA(ON), which is trained on the OntoNotes corpus (Hovy et al., 2006) on the training and development partitions used by Pradhan et al. (2013) after excluding a few documents that overlapped with the AMR corpus. All the different dependency trees are then used as input to our baseline system and we evaluate AMR parsing performance on the development set.

From Table 1, we can see that the performance of the baseline transition-based system remains very stable when different dependency parsers used are trained on same data set. However, the Charniak parser that is trained on a much larger and more diverse dataset (CHARNIKA(ON)) yields the best overall AMR parsing performance. Subsequent experiments are all based on this version of the Charniak parser.

4.1.2 Impact of parser extensions

In Table 2 we present results from extending the transition-based AMR parser. All experiments are conducted on the development set. From Table 2, we can see that the INFER action yields a 4 point improvement in F₁ score over the CHARNIKA(ON) system. Adding Brown clusters improves the recall by 1 point, but the F₁ score remains unchanged. Adding semantic role features on top of the Brown clusters leads to an improvement of another 2 points in F₁ score, and gives us the best result. Adding coreference features actually slightly hurts the performance.

4.1.3 Final Result on Test Set

We evaluate the best model we get from §4.1 on the test set, as shown in Table 3. For comparison purposes, we also include results of all published parsers on the same dataset: the updated version of JAMR, the old version of JAMR (Flanigan et al., 2014), the Stanford AMR parser (Werling et al., 2015), the SHRG-based AMR parser (Peng et al., 2015) and our baseline parser (Wang et al., 2015).

Footnote:

1Documents in the AMR corpus have some overlap with the documents in the OntoNotes corpus. We excluded these documents (which are primarily from Xinhua newswire) from the training data while retraining the Charniak parser, ASSERT semantic role labeler, and IMS frameset disambiguation tool). The full list of overlapping documents is available at http://cemantix.org/ontonotes/ontonotes-amr-document-overlap.txt
Table 3: AMR parsing performance on the news wire test set of LDC2013E117.

<table>
<thead>
<tr>
<th>System</th>
<th>P</th>
<th>R</th>
<th>F₁</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our system</td>
<td>.71</td>
<td>.69</td>
<td>.70</td>
</tr>
<tr>
<td>JAMR (GitHub)</td>
<td>.69</td>
<td>.58</td>
<td>.63</td>
</tr>
<tr>
<td>JAMR (Flanigan et al., 2014)</td>
<td>.66</td>
<td>.52</td>
<td>.58</td>
</tr>
<tr>
<td>Stanford</td>
<td>.66</td>
<td>.59</td>
<td>.62</td>
</tr>
<tr>
<td>SHRG-based</td>
<td>.59</td>
<td>.58</td>
<td>.58</td>
</tr>
<tr>
<td>Wang et al. (2015)</td>
<td>.64</td>
<td>.62</td>
<td>.63</td>
</tr>
</tbody>
</table>

From Table 3 we can see that our parser has significant improvement over all the other parsers and outperforms the previous best parser by 7% points in Smatch score.

4.2 Experiments on LDC2014T12

In this section, we conduct experiments on the AMR annotation release 1.0 (LDC2014T12), which contains 13,051 AMRs from newswire, weblogs and web discussion forums. 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 §4.1, and test the parser on the test set. The result is shown in Table 4. For comparison purposes, we also include the results of the updated version of JAMR and our baseline parser in (Wang et al., 2015) which are also trained on the same dataset. There is a significant drop-off in performance compared with the results on the LDC2013E117 test set for all the parsers, but our parser outperforms the other parsers by a similar margin on both test sets.

<table>
<thead>
<tr>
<th>System</th>
<th>P</th>
<th>R</th>
<th>F₁</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our system</td>
<td>.70</td>
<td>.62</td>
<td>.66</td>
</tr>
<tr>
<td>Wang et al. (2015)</td>
<td>.63</td>
<td>.56</td>
<td>.59</td>
</tr>
<tr>
<td>JAMR (GitHub)</td>
<td>.64</td>
<td>.53</td>
<td>.58</td>
</tr>
</tbody>
</table>

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

We also evaluate our parser on the newswire section of LDC2014T12 dataset. Table 5 compares the performance of JAMR, the Stanford AMR parser and our system on the same dataset.

<table>
<thead>
<tr>
<th>System</th>
<th>P</th>
<th>R</th>
<th>F₁</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our system</td>
<td>.72</td>
<td>.67</td>
<td>.70</td>
</tr>
<tr>
<td>Stanford</td>
<td>.67</td>
<td>.58</td>
<td>.62</td>
</tr>
<tr>
<td>JAMR (GitHub)</td>
<td>.67</td>
<td>.53</td>
<td>.59</td>
</tr>
</tbody>
</table>

Table 5: AMR parsing performance on newswire section of LDC2014T12 test set

And our system still outperforms the other parsers by a similar margin.

5 Conclusion

We presented extensions to a transition-based AMR parser that leads to an improvement of 7% in absolute F₁ score over the best previously published results. The extensions include designing a new action to infer abstract concepts and training the parser with additional semantic role labeling and coreference based features.

Acknowledgments

We want to thank the anonymous reviewers for their suggestions. This work was partially supported by the National Science Foundation via Grant No.0910532 entitled Richer Representations for Machine Translation. All views expressed in this paper are those of the authors and do not necessarily represent the view of the National Science Foundation.

References


