Robust Sub-Graph Generation for Abstract Meaning Representation Parsing

Keenon Werling
Stanford University
keenon@stanford.edu

Gabor Angeli
Stanford University
gabor@stanford.edu

Chris Manning
Stanford University
manning@stanford.edu

Abstract

The Abstract Meaning Representation (AMR) is a representation for open-domain rich semantics Generating semantic sub-graphs from contiguous tokens is a crucial part of AMR parsing. We propose a small set of actions to *construct* a sub-graph at test time from a span of tokens, which allow us to greatly expand our generalization from training data. We show that our set of construction actions is a good approximation which we can learn with a simple classifier. This reduces the need for sparse dictionary lookups, which improves generalization on unknown words and allows us to exploit statistical efficiency on a small training set. We demonstrate that our approach improves on published state-of-the-art AMR parsing, from *0.58* smatch to *0.64* smatch on the LDC2013E117 dataset.

1 Introduction

The Abstract Meaning Representation (AMR) (Banarescu et al., 2013) is a rich language for expressing semantic understanding on both a broad domain and at a relatively deep level. AMR captures many useful pieces of semantic information in a single joint representation. These include (but are not limited to) named entity recognition, verb argument labeling, word sense disambiguation, time expression parsing, and coreference. There is an ongoing AMR data labeling effort that promises to produce a breakthrough resource in broad domain semantic parsing, for both its size and the AMR formalism's expressive richness.

As of this writing there is one published parser that targets the AMR formalism, JAMR (Flanigan et al., 2014), which reports very promising results. After experimentation with several different structured prediction algorithms, we find that JAMR's architecture is a very strong framework for further parser development. JAMR follows a two-stage approach to parse AMR, first generating a set of AMR sub-graphs from spans of the sentence, and then stitching those sub-graphs together with a constrained MST (using dual decomposition).

However, JAMR's architecture, while it has special case machinery for creating sensible chunks for named entities and time expressions, cannot generate AMR sub-graphs from unseen words in the general case.

For instance, the adjective *exhaustive* in "the *exhaustive* search" should be parsed as an AMR verb, *exhaust-01*, but JAMR is unable to recognize this because it has no instance of the adjective *exhaustive* used in the training data. Our system can generalize from training data to learn that *exhaustive* is

Instead of memorizing a bank of AMR sub-graphs to generate from spans of tokens, we propose a small set of 'generative actions' that our system can take to derive an AMR sub-graph from a span of tokens (see Figure 2). For example, we have an action *IDENT* that will generate a node with the same title as the token in the source sentence that receives the label, which allows us to capture unseen nouns like 'mother'. We show that having a set of actions as an intermediate to generating AMR sub-graphs greatly improves statistical efficiency when training a sub-graph generation system on the relatively small amount of available training data, and that end to end performance in-domain is improved from **0.58** smatch to **0.64** smatch when using JAMR's constrained MST to stitch together the resulting sub-graphs. This approach also enables us to use dense features to improve brittle out-of-domain performance significantly.

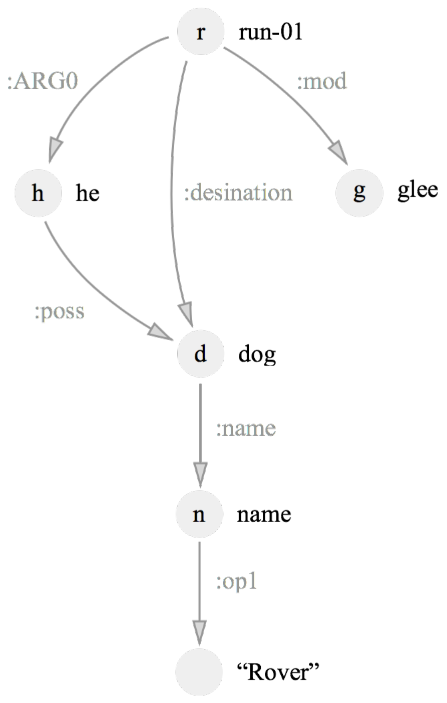

Figure 1: AMR graph for "He gleefully ran to his dog Rover". Nodes represent concepts, and arcs are relationships between concepts.

2 A Crash-Course in AMR

AMR is a language for expressing semantic understanding that represents meaning as a directed graph, where nodes represent concepts and arcs are relationships between concepts. AMR makes no effort to have a one-to-one correspondence between nodes in a graph and tokens in the sentence whose semantics is being represented. Thus AMR is not a "semantic dependency" representation. AMR represents the relationships between objects referred to by the surface text, not merely the relationships between the words themselves. In fact, AMR will often expand single tokens into large sub-graph elements, or ignore tokens completely.

To introduce AMR and its notation, we'll unpack the translation of the sentence "he gleefully ran to his dog Rover". We show in Figure 1 the interpretation of this sentence as an AMR graph.

Note that the root node of the graph is labeled "run-01". This is the name of a verb sense definition drawn from PropBank [citation needed] for the sense of the verb "ran" in this sentence. This distinguishes this use of the verb "run" from senses like those expressed in "he *ran* the business" or "he gave him a *run* for his money".

"run-01" has an outgoing "ARG0" arc to a node "he", with semantics (drawn from the PropBank frame) that roughly correspond to "he" being the doer of the "run-01" action. The "run-01" has an outgoing "mod" to "glee," which has the catch-all semantics that "run-01" is somehow modified by the concept "glee." "run-01" also has a "destination" arc to "dog," which draws its semantics from Vivek Srikumar's thesis chapter on preposition sense tagging [citation needed], and means that the destination of the "run-01" action is "dog". Then we have a section of the graph that is best interpreted as a unit, where all of the children of "dog" effectively mean that "dog" has the name "Rover."

TODO: Discuss nasty nominalizations and NER

3 Previous Work

At the time of this writing, the JAMR parser (Flanigan et al., 2014) is the only published AMR parser. It uses a two-stage approach to parsing AMR. In the first stage, a sequence model is used to generate small AMR sub-chunks. Then in the second stage these chunks are stitched together by a variation of a maximum spanning tree algorithm with dual decomposition to impose linguistically motivated constraints.

To more carefully define what is meant by AMR sub-chunks,

4 Methods

Our method relies on a very simple insight about AMR: a vast majority of the AMR nodes can be explained by a very small set of derivation types. We do a sequence labeling of each token in the sentence with one of the following labels:

- **VERB**: Look for the most similar PropBank frame, make that the title of the corresponding node.
- **IDENTITY**: Take the lowercased version of the token to be the title of the corresponding node.
- **VALUE**: Parse the token to an integer value, and use that as the node.
- **LEMMA**: Take the lemma of the token to be the title of the corresponding node.
- **NONE**: Ignore this token in the final output.

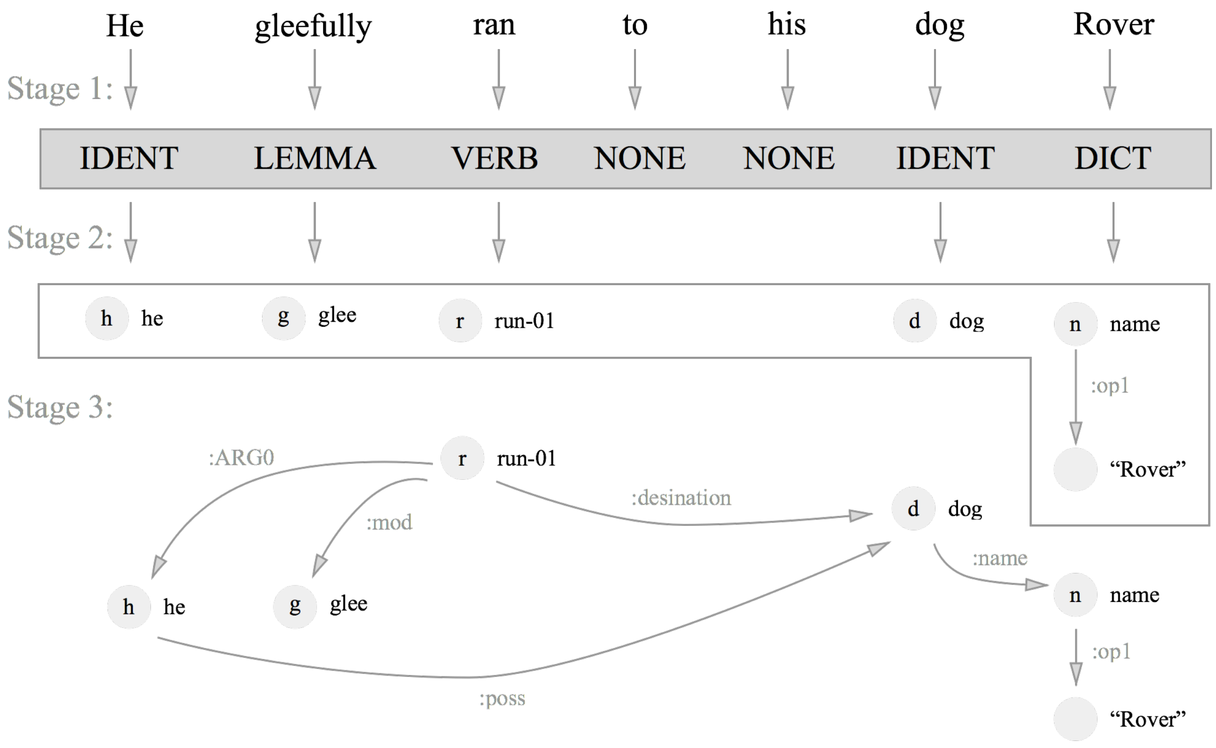

Figure 2: Derivation process for "He gleefully ran to his dog Rover". First the tokens in the sentence are labeled with derivation actions, then those actions are used to generate AMR sub-graphs, and then those sub-graphs are stitched together to form a coherent whole.

- **DICT**: Look up the most probable chunk associate with this lexical span. This functions as a back off if no other actions are appropriate.

This has two primary benefits: We're able to more effectively generalize from training data to unseen examples, and we're able to exploit much greater statistical efficiency over our training data.

The only class that doesn't give us an immediate win if we correctly classify it is **DICT**, for which we still have to resort to the JAMR expedient of looking up a span in a dictionary. This class functions as a catch-all. To see how rarely we need to resort to **DICT**, we turn to the distribution of different tag types on the training data, by token.

5 Preprocessing

AMR training data is in the form of bi-text, where we are given a set of (sentence,graph) pairs, with no explicit alignments between them. **TODO:** alignments **TODO:** sequence data gen **TODO:** dictionary data gen

TAG	**# Tokens**	**% Total**
NONE	28405	0.405
DICT	15861	0.226
VERB	11017	0.157
IDENTITY	10890	0.155
LEMMA	2581	0.036
VALUE	710	0.01
COREF	552	0.0078

Table 1: Distribution of tag types in the training data for LDC2013E117 dataset, generated from automatically aligned data.

6 Results

Our end to end results are reported by plugging the output of our subgraph generation system into the constrained-MST component of JAMR, which is able to produce final AMR graphs. AMR parsing accuracy is measured with a metric called smatch [citation needed], which stands for "s(emantic) match". We trained and tested on the **LDC2013E117** dataset, for which the last published result was a smatch score of 0.58 on the test set by JAMR (Flanigan et al., 2014). We report

0.64 on the same dataset, by substituting our subgraph generation system.

TODO: Report on generalization

7 Future Work

TODO: Better alignments

TODO: Etymological approach to generation

Acknowledgments

The acknowledgments should go immediately before the references. Do not number the acknowledgments section. Do not include this section when submitting your paper for review.

References

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, Noah A. Smith 2014. *ACL 14*, volume 1.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. *Proc. of the Linguistic Annotation Workshop and Iteroperability with Discourse*, volume 1.

Alfred V. Aho and Jeffrey D. Ullman. 1972. *The Theory of Parsing, Translation and Compiling*, volume 1. Prentice-Hall, Englewood Cliffs, NJ.

American Psychological Association. 1983. *Publications Manual*. American Psychological Association, Washington, DC.

Association for Computing Machinery. 1983. *Computing Reviews*, 24(11):503–512.

Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. 1981. Alternation. *Journal of the Association for Computing Machinery*, 28(1):114–133.

Dan Gusfield. 1997. *Algorithms on Strings, Trees and Sequences*. Cambridge University Press, Cambridge, UK.