Sheffield at SemEval-2017 Task 9: Transition-based language generation from AMR.

Gerasimos Lampouras
Department of Computer Science
University of Sheffield, UK
g.lampouras@sheffield.ac.uk

Andreas Vlachos
Department of Computer Science
University of Sheffield, UK
a.vlachos@sheffield.ac.uk

Abstract

This paper describes the submission by the University of Sheffield to the SemEval 2017 Abstract Meaning Representation Parsing and Generation task (SemEval 2017 Task 9, Subtask 2). We cast language generation from AMR as a sequence of actions (e.g., insert/remove/rename edges and nodes) that progressively transform the AMR graph into a dependency parse tree. This transition-based approach relies on the fact that an AMR graph can be considered structurally similar to a dependency tree, with a focus on content rather than function words. An added benefit to this approach is the greater amount of data we can take advantage of to train the parse-to-text linearizer. Our submitted run on the test data achieved a BLEU score of 3.32 and a Trueskill score of -2.204 on automatic and human evaluation respectively.

1 Introduction

Abstract meaning representation (AMR) is a formalism representing the meaning of a sentence (or multiple sentences) as a directed, acyclic graph, where each node represents a concept, and each edge represents a relation between concepts (Banarescu et al., 2013). Natural language generation (NLG) from AMRs introduces challenges, as AMR abstracts away from syntactic structure, function words, or inflections. Flanigan et al. (2016) were the first work to perform NLG from AMR; they used a weighted combination of a tree-to-string transducer and a language model to transform the AMR graph into English. Later work by Song et al. (2016) proposed segmenting the AMR graph into fragments and generating subphrases from them, using a set of subgraph-to-string rules. They then cast the problem of ordering these subphrases as a travelling salesman problem. Pourdamghani et al. (2016) suggested linearizing the AMR graph using a maximum entropy classifier. The linearization is then used as input to a phrase-based machine translation system, to produce the final sentence.

Our submission to SemEval task 9 on AMR-to-English Generation is based on inverting previous work on transition-based parsers (Goodman et al., 2016a,b), which was in turn based on the previous work of Wang et al. (2015). Beyond inverting the transition from AMR graph to dependency tree, our system also separates the transition in three passes. Briefly, during the first pass we convert the AMR concepts into content words, during the second pass the structure of the tree is modified (e.g. by inserting, deleting, and moving nodes and edges), while in the third pass missing function words are inserted, and existing words realized in their final form. To form a natural language sentence, the dependency tree needs only to be linearized; we note that this is not part of the transition, but should be considered a separate post-processing step. We train a separate classifier for each pass, to learn which action should be taken at each time-step.

2 System description

2.1 Pre-processing

During pre-processing the graph structure of the AMR is converted to a tree by identifying each node n with multiple incoming edges in the graph. Each additional incoming edge is redirected to a duplicate node n' (as shown in the transition between stage a and b in Figure 1). These duplicate nodes are inserted as leaves in the structure, and maintain no edges to the n's children. The system randomly determines which of the incoming edges will remain connected with n, and lets the transition system remove duplicate nodes, or move any

Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017), pages 586–591,
Vancouver, Canada, August 3 - 4, 2017.

of n's descendants as required.

During training, we employ the SpaCy dependency parser (Honnibal and Johnson, 2015) to construct the dependency tree of the training sentence and obtain part-of-speech tags; the dataset's sentences are already split into tokens. Heuristics are used to normalize all date occurrences and numeric expressions in both the sentence and dependency tree, to help our system handle temporal and numerical AMR concepts and structures. Additionally, we construct a simplified version of the dependency tree where articles, auxiliary words, and punctuation, are removed. This simplified tree is useful for the first and second phases of the transition where the focus is on content words.

2.2 Phase 1

Phase 1 is initialized with a stack σ containing all nodes in the AMR tree, with the leaf nodes first; in subsequent phases, σ is initialized with the nodes of the modified tree of the previous phase. A second stack β is initialized with the children of the top node in σ. At each time-step the transition system considers the current state, which consists of the aforementioned stacks, and the tree (which may be in any intermediate stage between an AMR tree and a dependency tree). Each phase concludes once σ is exhausted, with both σ and β stacks being reinitiated for the next phase as needed.

All transition actions are detailed in Table 1, separated according to which phase they may be applied. Some actions may appear in multiple phases (e.g. the NextNode and NextEdge actions, which are primarily used to traverse the σ and β stacks) but note that their outcome may slightly differ from phase to phase. Particularly, during phase 1 the action NextNode is also used to modify the labels of the graph, in effect transforming AMR concepts to content words. If a content word is determined to be a verb, noun, adjective or adverb, a parameter l_n is used which consists of the word's stem and the appropriate part-of-speech tag. The stem is obtained by applying Porter's stemmer to the AMR concept identifier.[1] The intuition here is that, while the stem and part-of-speech tag may be useful in structuring the dependency tree in phase 2, the inflected form of each word can more accurately be determined after the dependency tree is finalized (i.e. after phase 2).

The NextEdge action is used to traverse the β stack, alternating with MergeNode actions. The latter are applied when two AMR concepts should be combined to form a single content word, e.g. the negation concept "-" and concept "security" combining to form the word "insecurity". Additionally, during phase 1, certain AMR fragments with typified structure (e.g., `name`, `date-entity`, `time`) are collapsed into single nodes, and occurrences of `wiki` relations are removed. Consult stage c of Figure 1 for an example.

2.3 Phase 2

In phase 2, the transition actions aim to transform the structure of the tree. They are based on the actions used by Goodman et al. (2016a,b), with some alterations due to σ being initialized by traversing the tree from leaves to root, namely the Insert action is allowed to add a parent node above the current one, but limited to adding only leaf nodes as children. Similarly to Wang et al. (2015), but unlike Goodman et al. (2016a,b), the Reattach, SwapEdge, InsertParent, and InsertLeaf actions are parameterized with an edge label l_e. The NextNode action in phase 2 simply traverses σ.

To improve runtime, the Reattach, InsertParent, and InsertLeaf actions are allowed only to σ_0 nodes they were applied to in the training data, and will not be considered otherwise. Similarly, labels l_n, l_e are limited to those observed during training.

Finally, all actions preserve full connectivity of the tree, and any Reattach actions that would introduce a cycle are not considered. To avoid conflicts between actions, the following restrictions are enforced: a DeleteLeaf or ReplaceHead action cannot delete a previously inserted node, and vice-versa; a SwapEdge action cannot swap a previously swapped edge; a Reattach action cannot move a previously reattached or inserted node.

Stages d, e, f, and g of Figure 1 show the intermediate trees produced in phase 2. The double-bordered nodes denote nodes already visited via NextNode actions and thus no longer in σ.

2.4 Phase 3

During phase 3, InsertParent and InsertLeaf actions are used to add any closed-set function words (e.g. auxiliary verbs and articles) and punctuation that are missing from the dependency tree. The ModifyNode action is a variant of the NextNode action, which modifies (rather than replaces) any temporary labels with a suffix operation (i.e. "-s", "-ing") that, when combined with the stem of

[1] https://tartarus.org/martin/PorterStemmer/

Action name	β status	Parameters	Action outcome
Phase 1: Convert AMR concepts to content words.			
NextNode	empty	l_n	Set label of node σ_0 to l_n. Pop σ_0, and initialize β.
MergeNode	non-empty	l_n	Set label of node σ_0 to l_n. Pop β_0, and remove it from the tree. The children of β_0 are attached as children to σ_0.
NextEdge	non-empty	-	Pop β_0.
Phase 2: Modify the structure of the tree.			
InsertParent	-	l_n, l_e	Insert new node δ with label l_n as the parent of σ_0, via dependency label l_e. Insert δ into σ.
InsertLeaf	empty	l_n, l_e	Insert new node δ with label l_n as a child of **leaf node** σ_0, via dependency label l_e. Insert δ into σ.
NextEdge	non-empty	l_e	Set label of edge (σ_0, β_0) to l_e. Pop β_0.
SwapEdge	non-empty	l_e	Reverse edge (σ_0, β_0) to (β_0, σ_0), and set its label to l_e. β_0 becomes the parent of σ_0 and its subgraph, while the previous parent of σ_0 becomes the parent of β_0. Pop β_0.
ReplaceHead	non-empty	-	Pop σ_0, delete it from the graph. Attach β_0 as a child to the previous parent of σ_0. All other children of σ_0 become children of β_0. Insert β_0 at the head of σ, initialize β.
DeleteLeaf	empty	-	Pop **leaf node** σ_0, delete it from the graph. Initialize β.
Reattach	non-empty	p, l_e	Change edge (σ_0, β_0) to (p, β_0), where p is an existing node in the graph. p becomes the parent of β_0 and its subgraph. Pop β_0, and insert p to σ.
NextNode	empty	-	Pop σ_0, and initialize β.
Phase 3: Insert function words, punctuation, and determine the proper inflection of content words.			
InsertParent	-	l_n, l_e	Insert new node δ with label l_n as the parent of σ_0, via dependency label l_e. Insert δ into σ.
InsertLeaf	-	l_n, l_e	Insert new node δ with label l_n as a child of **leaf node** σ_0, via dependency label l_e. Insert δ into σ.
ModifyNode	-	m_n	Modify label of node σ_0 by m_n. Pop σ_0, and initialize β.

Table 1: Available actions per phase, for transition-based transformation of AMR graphs to parse trees.

the label, can properly inflect the word (e.g. modify "make_VB" with "-s" to construct "makes"). Stage h of Figure 1 shows the outcome of phase 3.

2.5 Expert policy

During training, an expert policy (also known as oracle) is constructed to determine which action should be performed given a particular state. By consulting the alignments between the concepts of each AMR graph and the words of the corresponding sentence in the training data, the expert policy detects any unaligned AMR concepts to be deleted by appropriate actions, as well as any unaligned words in the dependency tree to be inserted; other actions can be similarly inferred. During phases 1 and 2 we consider the simplified dependency tree, where function words have been removed, and in phase 3 we consider the full tree. The alignments were provided in the dataset using the system of Pourdamghani et al. (2014).

2.6 Post-processing

In post-processing, the dependency tree constructed by the transition needs to be linearized into a sentence. Tree linearization has most commonly been addressed by overgenerating word sequences and ranking (e.g. according to a trigram language model); however there has been a lot of recent research studying this topic (Filippova and Strube, 2009; He et al., 2009; Belz et al., 2011; Bohnet et al., 2011; Zhang, 2013; Futrell and Gibson, 2015). Our approach in this paper is to simply order the nodes in each subtree using a classifier, in effect creating ordered subphrases of the tree. The subtrees are thus incrementally ordered, in a bottom-up approach, and subsequently formed into a natural language sentence. Any date occurrences or numerical expressions that were normalized during pre-processing, are restored to their original form. It is also important to note again, that the structure of this approach allows it to take

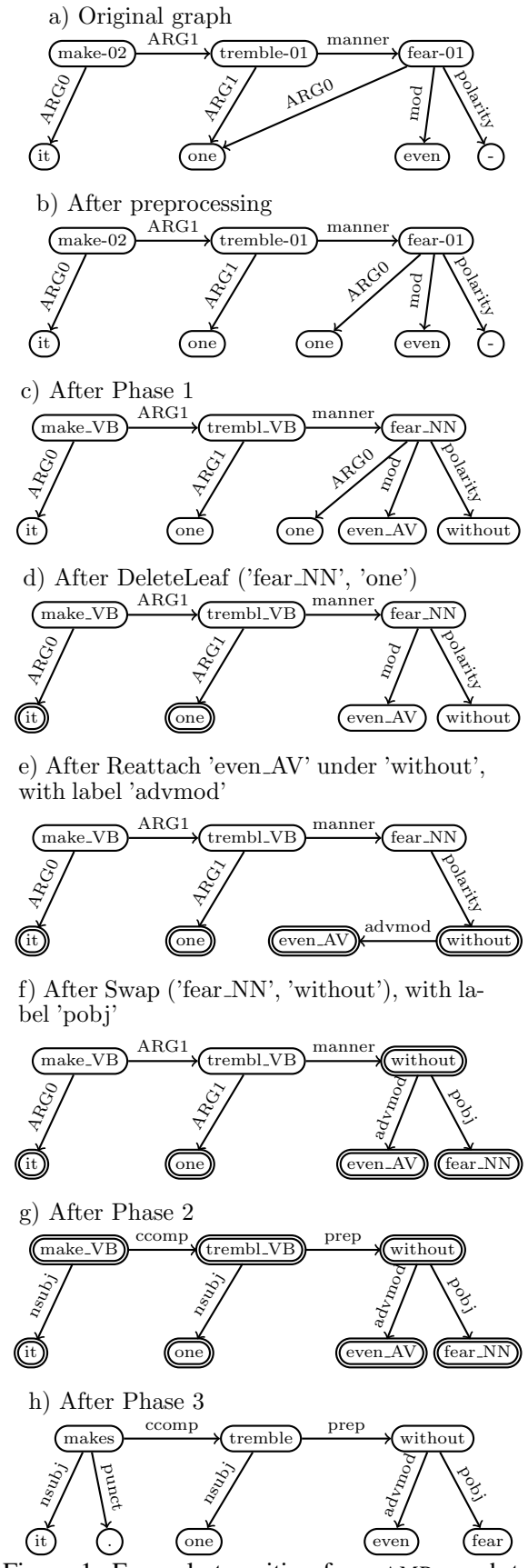

Figure 1: Example transition from AMR graph to dependency tree of the sentence "It makes one tremble even without fear."

advantage of additional parse-tree datasets to augment the training of the post-precessing step.

3 Results

We use the adaptive regularization of weight vectors (AROW) algorithm (Crammer et al., 2013) for all aforementioned classifiers. All the features we use are boolean indicators and similar to those proposed by Goodman et al. (2016a,b) and Wang et al. (2015). All classifiers were trained on the same corpus of AMRs released by LDC, and created as part of tehe DARPA DEFT program (LDC2016E25); we hope to augment the linearization's training with other dependency parse datasets in future work. To provide further speed improvement in testing time, we filter actions (conditioned on specific parameters) that appear infrequently in the training set.

Table 2 shows the ablation results of our system on the test set of the task. For Phase 1 we calculate the precision of the labels in the output tree compared to the labels of the dependency parse, while on phases 2 and 3 we calculate the unlabeled and labeled attachment scores. We also include the BLEU (Papineni et al., 2002) and Trueskill (Sakaguchi et al., 2014) scores achieved by our submitted run on the task's test data. In future work, we would like to examine the effect of error propagation from phase to phase.

	Precision	
Phase 1	0.45	
	UAS	LAS
Phase 2	0.16	0.11
Phase 3	0.08	0.06
	BLEU	Trueskill
Realization	3.32	−2.204

Table 2: Ablation results on the testing set.

4 Conclusion

We proposed a three-phase transition-based system for transforming an AMR graph into a dependency tree; the final sentence can then be acquired via a tree linearizer. Our results suggest there is much room for improvement; we hope to continuously refine the proposed action space and expert policy, and develop and apply a more complex linearizer to the constructed parse trees. Finally, we believe that by using imitation learning algorithms, the transition sequences could be improved to generalize better to unseen data.

Acknowledgements

This research is supported by the EPSRC grant Diligent (EP/M005429/1).

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*. Association for Computational Linguistics, Sofia, Bulgaria, pages 178–186.

Anja Belz, Michael White, Dominic Espinosa, Eric Kow, Deirdre Hogan, and Amanda Stent. 2011. The first surface realisation shared task: Overview and evaluation results. In *Proceedings of the 13th European Workshop on Natural Language Generation*. Association for Computational Linguistics, Stroudsburg, PA, USA, ENLG '11, pages 217–226.

Bernd Bohnet, Simon Mille, Benoît Favre, and Leo Wanner. 2011. StuMaBa : From deep representation to surface. In *ENLG 2011 - Proceedings of the 13th European Workshop on Natural Language Generation, 28-30 September 2011, Nancy, France*. pages 232–235.

Koby Crammer, Alex Kulesza, and Mark Dredze. 2013. Adaptive regularization of weight vectors. *Machine Learning* 91:155–187.

Katja Filippova and Michael Strube. 2009. Tree linearization in english: Improving language model based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*. Association for Computational Linguistics, Stroudsburg, PA, USA, NAACL-Short '09, pages 225–228.

Jeffrey Flanigan, Chris Dyer, Noah A. Smith, and Jaime G. Carbonell. 2016. Generation from abstract meaning representation using tree transducers. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*. pages 731–739.

Richard Futrell and Edward Gibson. 2015. Experiments with generative models for dependency tree linearization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*. pages 1978–1983.

James Goodman, Andreas Vlachos, and Jason Naradowsky. 2016a. 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 (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 1–11.

James Goodman, Andreas Vlachos, and Jason Naradowsky. 2016b. Ucl+sheffield at semeval-2016 task 8: Imitation learning for amr parsing with an alpha-bound. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. Association for Computational Linguistics, San Diego, California, pages 1167–1172.

Wei He, Haifeng Wang, Yuqing Guo, and Ting Liu. 2009. Dependency based chinese sentence realization. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL '09, pages 809–816.

Matthew Honnibal and Mark Johnson. 2015. An improved non-monotonic transition system for dependency parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1373–1378.

K. Papineni, S. Roukos, T. Ward, and W. J. Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of ACL*. Philadelphia, PA, pages 311–318.

Nima Pourdamghani, Yang Gao, Ulf Hermjakob, and Kevin Knight. 2014. Aligning english strings with abstract meaning representation graphs. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 425–429.

Nima Pourdamghani, Kevin Knight, and Ulf Hermjakob. 2016. Generating english from abstract meaning representations. In *INLG 2016 - Proceedings of the Ninth International Natural Language Generation Conference, September 5-8, 2016, Edinburgh, UK*. pages 21–25.

Keisuke Sakaguchi, Matt Post, and Benjamin Van Durme. 2014. Efficient elicitation of annotations for human evaluation of machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*. Association for Computational Linguistics, Baltimore, Maryland, USA, pages 1–11.

Linfeng Song, Yue Zhang, Xiaochang Peng, Zhiguo Wang, and Daniel Gildea. 2016. Amr-to-text generation as a traveling salesman problem. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*. pages 2084–2089.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015. A transition-based algorithm for AMR parsing. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*. pages 366–375.

Yue Zhang. 2013. Partial-tree linearization: Generalized word ordering for text synthesis. In *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*. pages 2232–2238.