Rewarding Smatch: Transition-Based AMR Parsing with Reinforcement Learning

Tahira Naseem♠ **Abhishek Shah**♢ **Hui Wan**♠
Radu Florian♠ **Salim Roukos**♠ **Miguel Ballesteros**♠
♠IBM Research, Yorktown Heights, NY, USA
♢IBM Watson, New York, NY, USA
tnaseem, hwan, raduf, roukos@us.ibm.com
abhishek.shah1, miguel.ballesteros@ibm.com

Abstract

Our work involves enriching the Stack-LSTM transition-based AMR parser (Ballesteros and Al-Onaizan, 2017) by augmenting training with Policy Learning and rewarding the Smatch score of sampled graphs. In addition, we also combined several AMR-to-text alignments with an attention mechanism and we supplemented the parser with pre-processed concept identification, named entities and contextualized embeddings. We achieve a highly competitive performance that is comparable to the best published results. We show an in-depth study ablating each of the new components of the parser.

1 Introduction

Abstract meaning representations (AMRs) (Banarescu et al., 2013) are rooted labeled directed acyclic graphs that represent a non inter-sentential abstraction of natural language with broad-coverage semantic representations. AMR parsing thus requires solving several natural language processing tasks; named entity recognition, word sense disambiguation and joint syntactic and semantic role labeling. AMR parsing has acquired a lot of attention (Wang et al., 2015a; Zhou et al., 2016; Wang et al., 2015b; Goodman et al., 2016; Guo and Lu, 2018; Lyu and Titov, 2018; Vilares and Gómez-Rodríguez, 2018; Zhang et al., 2019) in recent years.

We build upon a transition-based parser (Ballesteros and Al-Onaizan, 2017) that uses Stack-LSTMs (Dyer et al., 2015). We augment training with self-critical policy learning (Rennie et al., 2017) using *sentence-level* Smatch scores (Cai and Knight, 2013) as reward. This objective is particularly well suited for AMR parsing, since it overcomes the issues arising from the lack of *token-level* AMR-to-text alignments. In addition, we perform several modifications which are inspired from neural machine translation (Bahdanau et al., 2014) and by the recent trends on contextualized representations (Peters et al., 2018; Devlin et al., 2018).

Our contributions are: (1) combinations of different alignment methods: There has been significant research in that direction (Flanigan et al., 2014; Pourdamghani et al., 2014; Chen, 2015; Chu and Kurohashi, 2016; Chen and Palmer, 2017; Szubert et al., 2018; Liu et al., 2018). In this paper, we show that combination of different methods makes a positive impact. We also combine hard alignments with an attention mechanism (Bahdanau et al., 2014). (2) Preprocessing of named entities and concepts. (3) Incorporating contextualized vectors (with BERT) and compare their effectiveness with detailed ablation experiments. (4) Employing policy gradient training algorithm that uses Smatch as reward.

2 Stack-LSTM AMR Parser

We use the Stack-LSTM transition based AMR parser of Ballesteros and Al-Onaizan (2017) (henceforth, we refer to it as BO). BO follows the Stack-LSTM dependency parser by Dyer et al. (2015). This approach allows unbounded lookahead and makes use of greedy inference. BO also learns character-level word representations to capitalize on morphosyntactic regularities (Ballesteros et al., 2015). BO uses recurrent neural networks to represent the stack data structures that underlie many linear-time parsing algorithms. It follows transition-based parsing algorithms (Yamada and Matsumoto, 2003; Nivre, 2003, 2008); words are read from a buffer and they are incrementally combined, in a stack, with a set of actions towards producing the final parse. The input is a sentence and the output is a complete AMR graph without

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 4586–4592
Florence, Italy, July 28 - August 2, 2019.

any preprocessing required.[1] We use Dynet (Neubig et al., 2017) to implement the parser. In what follows, we present several additions to the original BO model that improved the results.

2.1 Label Separation

BO's actions are enriched with labels that may correspond to AMR nodes or labels that decorate the arcs of the graph. BO reported a total of 478 actions in the 2014 dataset. We tried splitting the prediction in two separate steps, first the action, then the label or concept. This reduces the number of actions to 10 and helps the model to drive the search better.

2.2 Hard Alignments and Soft Alignments

AMR annotations do not provide alignments between the nodes of an AMR graph and the tokens in the corresponding sentence. We need such alignments to generate action sequences with an oracle for training. The parser is then trained to generate these action sequences. The quality of word-to-graph alignments has a direct impact in the accuracy of the parser.

In previous work, both rule-based (Flanigan et al., 2014) and machine learning (Pourdamghani et al., 2014) methods have been used to produce word-to-graph alignments. Once generated, the alignments are often not updated during training (Flanigan et al., 2016; Damonte et al., 2016; Wang and Xue, 2017; Foland and Martin, 2017). More recently, Lyu and Titov (2018) learn these alignments as latent variables.

In this work, we combine pre-learned (hard) alignments with an attention mechanism. As shown in section 4, the combination has a synergistic effect. In the following, we first explain our method for producing hard alignments and then we elaborate on the attention mechanism.

Hard Alignments Generation: In order to produce word-to-graph alignments, we combine the outputs of the symmetrized Expectation Maximization approach (SEM) of Pourdamghani et al. (2014) with those of the rule-based algorithm (JAMR) of Flanigan et al. (2014). Pourdamghani et al. (2014) do not produce alignments for all concepts; for example, named-entity nodes, date-entity nodes and numerical-quantity nodes are left unaligned. We post-process the output to deterministically align these nodes based on the alignments of its children (if any). We then merge the output with JAMR alignments. Overall, the alignment process involves the following steps:

1. Produce initial alignments using SEM[2].
2. Fill in the unaligned nodes by upwards percolation of child node alignments.[3]
3. Use JAMR alignments[4] for any nodes still unaligned and fill in intermediate nodes again.

Soft Alignments via Attention: The parser state is represented by the STACK, BUFFER and a list with the history of actions (which are encoded as LSTMs, the first two being Stack-LSTMs (Dyer et al., 2015)). This forms the vector $\mathbf{s}_t$ that represents the state:

$$\mathbf{s}_t = \max\{\mathbf{0}, \mathbf{W}[\mathbf{st}_t; \mathbf{b}_t; \mathbf{a}_t] + \mathbf{d}\}. \quad (1)$$

This vector $\mathbf{s}_t$ is used to predict the best action (and concept to add, if applicable) to take, given the state with a softmax. We complement the state with an attention over the input sentence (Bahdanau et al., 2014). In particular, we use general attention (Luong et al., 2015). In order to do so, we add a bidirectional LSTM encoder to the BO parsing model and we run attention over it in each time step. More formally, the attention weights α_i (for position i) are calculated based on the actions predicted so far (represented as a_j), the encoder representation of the sentence (h_i) and a projection weight matrix W_a:

$$e_i = a_j^\top W_a h_i \quad (2)$$

$$\alpha_i = \frac{\exp(e_i)}{\sum_k \exp(e_k)}. \quad (3)$$

A vector representation (c_j) is computed by a weighted sum of the encoded sentence word representations and the α values.

$$c_j = \sum_i \alpha_i \cdot h_i. \quad (4)$$

[1] We refer interested readers to (Ballesteros and Al-Onaizan, 2017) for details.

[2] https://isi.edu/~damghani/papers/Aligner.zip

[3] When multiple child nodes are aligned, role labels are used to select best node for alignment percolation. Node role labels are preferred in the following order – :name (in general), *:unit* (for quantities), *:ARG2* (for have-org-role and rate-entities) and then any other labels except *:mod*.

[4] https://github.com/jflanigan/jamr

Given the sentence representation produced by the attention mechanism (c_j), we complement the parser state as follows:

$$g_j = \tanh(W^1_{Dec} d_j + W^1_{Att} c_j) \quad (5)$$

$$u_j = \tanh(g_j + W^2_{Dec} d_j + W^2_{Att} c_j) \quad (6)$$

$$\mathbf{s}_t = \max\left\{\mathbf{0}, \mathbf{W}[\mathbf{st}_t; \mathbf{b}_t; \mathbf{a}_t; \mathbf{u}_j] + \mathbf{d}\right\}, \quad (7)$$

where d_j is the concatenation of the output vector of the LSTM with the history of actions LSTM and the output vector of the LSTM that represents the stack. This new vector $\mathbf{s}_t$ replaces the one described in (1).

Those familiar with neural machine translation will recognize that we are using the concatenation of the output of the LSTMs that represent the stack and the action history as the decoder is used in the standard sequence to sequence with attention model (Bahdanau et al., 2014).

2.3 Preprocessed Nodes

We produce two types of pre-processed nodes: 1) Named Entity labels (NER) and 2) Concept labels (such as want-01, boy etc.). We use NER labels and preprocessed concepts the same way BO and Dyer et al. (2015) used part-of-speech tags – as another vector concatenated to the word representation and learned during training.

Concepts: AMR representation abstracts away from exact lexical forms. In the case of objects, the concepts are usually represented using the uninflected base forms; for events, the OntoNotes PropBank sense number is attached with the base form (such as *want-01*). We train a linear classifier that uses contextualized BERT embeddings (Devlin et al., 2018) of each word to predict the corresponding concept (which can be none). Each label is predicted in isolation with no regard to the surrounding labels. The tagger is trained using a combination of OntoNotes 5.0 (LDC2013T19) and LDC2017T10 AMR training data.

Named entities: We extracted named entities from the AMR dataset (there are more than 100 entity types in the AMR language) and we trained a neural network NER model (Ni et al., 2017) to predict NER labels for the AMR parser. In the NER model, the target word and its surrounding words and tags are used as features. We jackknifed (90/10) the training data, to train the AMR parser. The ten jackknifed models got an average NER F1 score of 79.48 on the NER dev set.

2.4 Contextualized Vectors

Recent work has shown that the use of pre-trained networks improves the performance of downstream tasks. BO uses pre-trained word embeddings by Ling et al. (2015) along with learned character embeddings. In this work, we explore the effect of using contextualized word vectors as pre-trained word embeddings. We experiment with recent context based embedding obtained with BERT (Devlin et al., 2018).

We use average of last 4 layers of BERT Large model with hidden representations of size 1024. We produce the word representation by mean pooling the representations of word piece tokens obtained using BERT. We only use the contextualized word vectors as input to our model, we do not back-propagate through the BERT layers.

2.5 Wikification

Given that BO does not produce Wikipedia nodes during prediction, we pre-process the AMR data removing all Wikipedia nodes. In order to produce Wikipedia entries in our AMR graphs, we run a wikification approach as post-processing. We combine the approach of Lyu and Titov (2018) with the entity linking technique of Sil et al. (2018).

First, we produce a dictionary of Wikipedia links for all the named entity nodes that appear with *:wiki* label in the training data. If a node appears with multiple Wikipedia links, the most frequent one is added to the dictionary. Separately, we also process the target sentence using the entity linking system of Sil et al. (2018). This system identifies the entities as well as links them.

During post processing, every node with *:name* label is looked up in the dictionary and if found, is assigned the corresponding Wikipedia link. This is very similar to the approach of Lyu and Titov (2018). If the node is not found in the dictionary, and the system of Sil et al. (2018) produces a Wikipedia link, we use that link.

2.6 Smatch Weighting

The upper bound for BO's oracle is only 93.3 F1 for the entire development set. We observed that the oracle produces a score close to perfect for most sentences, yet it loses some points in others. During training, we have the gold AMR graph available for every sentence. We compare it to the oracle graph and use the Smatch score as a weight for the training example. This is a way to down-weight the examples whose oracle actions sequence is incomplete or erroneous. This modification resulted in moderate gains (see row 14 in Table 1) and also lead to the training with exploration experiments described below.

3 Reinforcement Learning

BO relies on the oracle action sequences. The training objective is to maximize the likelihood of oracle actions. This strategy has two drawbacks. First, inaccurate/incomplete alignments between the tokens and the graph nodes introduce noise into oracle action sequences.(As mentioned above, the oracle upper bound is only 93.3 F1. With the enhanced alignments, BO reported 89.5 F1 in the LDC2014 development set). Second, even for the perfectly aligned sentences, the oracle action sequence is not the only or the best action sequence that can lead to the gold graph; there could be shorter sequences that are easier to learn. Therefore, strictly binding the training objective to the oracle action sequences can lead to suboptimal performance, as evidenced in (Daumé III and Marcu, 2005; Daumé III et al., 2009; Goldberg and Nivre, 2012, 2013; Ballesteros et al., 2016) among others.

To circumvent these issues, we resort to a Reinforcement Learning (RL) objective where the Smatch score of the predicted graph for a given sentence is used as reward. This alleviates the strong dependency on hard alignment and leaves room to training with exploration of the action space. This line of work is also motivated by Goodman et al. (2016), who used imitation learning to build AMR parses from dependency trees.

We use the self-critical policy gradient training algorithm by Rennie et al. (2017) which is a special case of the REINFORCE algorithm of Williams (1992) with a baseline. This method allows the use of an external evaluation measure as reward (Paulus et al., 2017). In particular, we want to maximize the expected Smatch reward,

$$L_{RL} = E_{g^s \sim p_\theta}[r(g^s)] \tag{8}$$

where p_θ is the policy specified by the parser parameters θ and g^s is a graph sampled from p_θ. The gradient of this objective can be approximated using a single sample from p_θ. For each sentence, we produce two graphs using the current model parameters. A greedy best graph $\hat{g}$ and a graph g^s produced by sampling from action space. The gradient of 8 is approximated as in (Rennie et al., 2017),

$$\nabla_\theta L_{RL} = (r(g^s) - r(\hat{g}))\nabla_\theta \log(p_\theta(g^s)) \tag{9}$$

where $r(g)$ is the Smatch score of graph g with respect to the ground truth. The Smatch of the greedy graph $r(\hat{g})$ serves as a baseline that can reduce the variance in the gradient estimate (Williams, 1992).

With ϵ probability, we flatten the sampling distribution by calculating the square root of the probabilities. In our experiments, ϵ is set to 0.05. We first train our full model with the maximum-likelihood objective of BO that achieves an F-score 72.8 without beam search when evaluated in the development set. The RL training is then initialized with the parameters of this trained model. For RL training, we use a batch-size of 40.

4 Experiments and Results

We start by reimplementing BO[5] and we train models with the most recent dataset (LDC2017T10)[6]. We include label separation in our reimplementation (Experiments 1..16) which separates the prediction of actions and labels in two different softmax layers. All our experiments use beam 10 for decoding and they are the best (when evaluated in the development set) of 5 different random seeds. Word, input and hidden representations have 100 dimensions (with BERT, input dimensions are 1024), action and label embeddings are of size 20. Our results are presented in Table 1.

We achieve the best results ever reported in some of the metrics. Unlabeled Smatch (16) by 1 point and SRL by 2 points. These two metrics represent the structure and semantic parsing task. For all the remaining metrics, our parser consistently achieves the second best results. Also, our best single model (16) achieves more than 9 Smatch points on top of BO (0). Guo and Lu (2018)'s parser is a reimplementation of BO with a refined search space (which we did not attempt) and we beat their performance by 5 points.

[5] BO reported results on the 2014 dataset.

[6] LDC2016E25 and LDC2017T10 contain the same AMR annotations as of March 2016. LDC2017T10 is the general release while LDC2016E25 was released for Semeval 2016 participants (May, 2016).

Id	Experiment	Smatch	Unlabeled	No WSD	Named Entities	Wikification	Negations	Concepts	Rentrancies	SRL
0	BO (JAMR)	65.9	71	66	80	0	45	82	46	59
1	BO + Label (JAMR)	67.0	72	68	81	79	46	82	48	64
2	BO + Label	68.3	73	69	79	78	62	82	51	66
3	2 + POS	69.0	74	70	80	79	62	83	51	67
4	3 + DEP	69.4	75	70	81	79	65	83	52	67
5	4 + NER	69.8	75	70	83	79	62	83	52	67
6	5 + Concepts	70.9	76	71	83	79	66	84	54	69
7	6 + BERT	72.9	78	73	83	78	67	84	58	**72**
8	1 + Attention	69.8	75	70	80	78	63	83	53	68
9	8 + POS	70.4	75	71	80	79	64	83	53	68
10	9 + DEP	70.7	75	71	80	79	62	83	53	68
11	10 + NER	70.8	76	71	83	79	64	8	53	68
12	11 + Concepts	71.8	77	72	82	78	66	84	56	70
13	12 + BERT[11]	73.1	78	74	82	79	66	84	58	**72**
14	13 + Smatch	73.6	78	74	84	79	64	85	59	**72**
15	8 + BERT	73.4	78	74	83	79	64	84	57	71
16	14 + RL	75.5	**80**	76	83	80	67	**86**	56	**72**
	Zhang et al. (2019)	**76.3**	79	**77**	78	**86**	**75**	85	**60**	70
	Lyu and Titov (2018)	74.4	77	76	**86**	76	58	**86**	52	70
	van Noord and Bos (2017)	71.0	74	72	79	65	62	82	52	66
	Guo and Lu (2018)	69.8	74	72	78	71	57	84	49	64

Table 1: Results, including comparison with the best systems, in the LDC2017T10 test set (aka AMR 2.0). Results highlighted in bold are the best in each metric. BO is (Ballesteros and Al-Onaizan, 2017) (which did not produce wikification). (JAMR) means that the model uses JAMR alignments, the rest use our alignments. Metrics by Cai and Knight (2013) and Damonte et al. (2016).

The hard alignments proposed in this paper present a clear advantage over the JAMR alignments. BO ignores nodes that are not aligned to tokens in the sentence, and it benefits from a more recall oriented alignment method. Adding attention on top of that adds a point, while preprocessing named entities improve the NER metric. Adding concepts preprocessed with our BERT based tagger adds more than a point. Smatch weighting lead to half a point on top of (14).

BERT contextualized vectors provide more than a point on top of the best model with traditional word embeddings (without attention, the difference is of 2 points). Combining BERT with a model that only sees words (15), we achieve the best results surpassed only by models that also use contextualized vectors and reinforcement learning objective, However, we added Smatch weighting (14) and Reinforcement Learning (16) on top of 13. This was decided based on development data results, where 13 performed better than the BERT only model (15) by about a point.

Finally, training with exploration via reinforcement learning gives further gains of about 2 points and achieves one of the best results ever reported on the task and state of the art in some of the metrics.

5 Conclusions

We report modifications in a competitive AMR parser achieving one of the best results in the task. Our main contribution augments training with Policy Learning by priming samples that are more suitable for the evaluation metric. We perform an in-depth ablation experiment that shows the impact of each of our contributions. Our unlabeled Smatch score (achieving the best graph structure) suggests that a new strategy to predict labels may reach even higher numbers.

Acknowledgments

We thank Ramón Astudillo, Avi Sil, Young-Suk Lee, Vittorio Castelli and Todd Ward for useful comments and support.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Miguel Ballesteros and Yaser Al-Onaizan. 2017. Amr parsing using stack-lstms. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1269–1275. Association for Computational Linguistics.

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved transition-based parsing by modeling characters instead of words with LSTMs. pages 349–359.

Miguel Ballesteros, Yoav Goldberg, Chris Dyer, and Noah A. Smith. 2016. Training with exploration improves a greedy stack lstm parser. In *Proceedings of* *the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2005–2010. 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.

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

Wei-Te Chen. 2015. Learning to map dependency parses to abstract meaning representations. In *ACL*.

Wei-Te Chen and Martha Palmer. 2017. Unsupervised amr-dependency parse alignment. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 558–567. Association for Computational Linguistics.

Chenhui Chu and Sadao Kurohashi. 2016. Supervised syntax-based alignment between english sentences and abstract meaning representation graphs. *arXiv preprint arXiv:1606.02126*.

Marco Damonte, Shay B Cohen, and Giorgio Satta. 2016. An incremental parser for abstract meaning representation. *arXiv preprint arXiv:1608.06111*.

Hal Daumé III, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Machine Learning*, 75:297–325.

Hal Daumé III and Daniel Marcu. 2005. Learning as search optimization: Approximate large margin methods for structured prediction. In *Proc. of ICML*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. pages 334–343.

Jeffrey Flanigan, Chris Dyer, Noah A Smith, and Jaime Carbonell. 2016. Cmu at semeval-2016 task 8: Graph-based amr parsing with infinite ramp loss. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1202–1206.

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)*, volume 1, pages 1426–1436.

William Foland and James H Martin. 2017. Abstract meaning representation parsing using lstm recurrent neural networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 463–472.

Yoav Goldberg and Joakim Nivre. 2012. A dynamic oracle for arc-eager dependency parsing. pages 959–976.

Yoav Goldberg and Joakim Nivre. 2013. Training deterministic parsers with non-deterministic oracles. *Transactions of the Association for Computational Linguistics*, 1:403–414.

James Goodman, Andreas Vlachos, and Jason Naradowsky. 2016. 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)*.

Zhijiang Guo and Wei Lu. 2018. Better transition-based amr parsing with a refined search space. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Wang Ling, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Two/too simple adaptations of word2vec for syntax problems. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1299–1304.

Yijia Liu, Wanxiang Che, Bo Zheng, Bing Qin, and Ting Liu. 2018. An amr aligner tuned by transition-based parser. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2422–2430. Association for Computational Linguistics.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1412–1421.

Chunchuan Lyu and Ivan Titov. 2018. Amr parsing as graph prediction with latent alignment. *arXiv preprint arXiv:1805.05286*.

Jonathan May. 2016. Semeval-2016 task 8: Meaning representation parsing. In *Proceedings of the 10th international workshop on semantic evaluation (semeval-2016)*, pages 1063–1073.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha

Swayamdipta, and Pengcheng Yin. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*.

Jian Ni, Georgiana Dinu, and Radu Florian. 2017. Weakly supervised cross-lingual named entity recognition via effective annotation and representation projection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1470–1480, Vancouver, Canada. Association for Computational Linguistics.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. pages 149–160.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. 34:513–553.

Rik van Noord and Johan Bos. 2017. Neural semantic parsing by character-based translation: Experiments with abstract meaning representations. *CoRR*, 1705.09980.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

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)*, pages 425–429.

Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7008–7024.

Avirup Sil, Gourab Kundu, Radu Florian, and Wael Hamza. 2018. Neural cross-lingual entity linking. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Ida Szubert, Adam Lopez, and Nathan Schneider. 2018. A structured syntax-semantics interface for english-amr alignment. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1169–1180. Association for Computational Linguistics.

David Vilares and Carlos Gómez-Rodríguez. 2018. A transition-based algorithm for unrestricted amr parsing. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*.

Chuan Wang and Nianwen Xue. 2017. Getting the most out of amr parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1257–1268.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015a. Boosting transition-based AMR parsing with refined actions and auxiliary analyzers. In *Proc. of ACL 2015*, pages 857–862.

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. Association for Computational Linguistics.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. pages 195–206.

Sheng Zhang, Xutai Ma, Kevin Duh, and Benjamin Van Durme. 2019. Amr parsing as sequence-to-graph transduction. *arXiv preprint arXiv:1905.08704*.

Junsheng Zhou, Feiyu Xu, Hans Uszkoreit, Weiguang QU, Ran Li, and Yanhui Gu. 2016. Amr parsing with an incremental joint model. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 680–689, Austin, Texas. Association for Computational Linguistics.