The Meaning Factory at SemEval-2017 Task 9: Producing AMRs with Neural Semantic Parsing

Rik van Noord
CLCG
University of Groningen
r.i.k.van.noord@rug.nl

Johan Bos
CLCG
University of Groningen
johan.bos@rug.nl

Abstract

We evaluate a semantic parser based on a character-based sequence-to-sequence model in the context of the SemEval-2017 shared task on semantic parsing for AMRs. With data augmentation, super characters, and POS-tagging we gain major improvements in performance compared to a baseline character-level model. Although we improve on previous character-based neural semantic parsing models, the overall accuracy is still lower than a state-of-the-art AMR parser. An ensemble combining our neural semantic parser with an existing, traditional parser, yields a small gain in performance.

1 Introduction

Traditional open-domain semantic parsers often use statistical syntactic parsers to derive syntactic structure on which to build a meaning representation. Recently there have been interesting attempts to view semantic parsing as a translation task, mapping a source language (here: English) to a target language (a logical form of some kind). Dong and Lapata (2016) used sequence-to-sequence and sequence-to-tree neural translation models to produce logical forms from sentences, while Barzdins and Gosko (2016) and Peng et al. (2017) used a similar method to produce AMRs. From a purely engineering point of view, these are interesting attempts as complex models of the semantic parsing process can be avoided. Yet little is known about the performance and fine-tuning of such parsers, and whether they can reach performance of traditional semantic parsers, or whether they could contribute to performance in an ensemble setting.

In the context of SemEval-2017 Task 9 we aim to shed more light on these questions. In particular we participated in Subtask 1, Parsing Biomedical Data, and work with parallel English-AMR training data comprising extracts of scientific articles about cancer pathway discovery.

More specifically, our objectives are (1) try to reproduce the results of Barzdins and Gosko (2016), who used character-level models for neural semantic parsing; (2) improve on their results by employing several novel techniques; and (3) combine the resulting neural semantic parser with an existing off-the-shelf AMR parser to reach state-of-the-art results.

2 Neural Semantic Parsing

2.1 Datasets

Our training set consists of the second LDC AMR release (LDC2016E25) containing 39,620 AMRs, as well as the training set of the bio AMR corpus that contains 5,452 AMRs. As development and test set we use the designated development and test partition of the bio AMR corpus, both containing 500 AMRs. HTML-tags are removed from the sentences.

2.2 Basic Translation Model

We use a seq2seq neural translation model to *translate* English sentences into AMRs. This is a bi-LSTM model with added attention mechanism, as described in Bahdanau et al. (2014). Similar to Barzdins and Gosko (2016), but contrasting with Peng et al. (2017), we train the model only on character-level input. Model specifics are shown in Table 1.

In a preprocessing step, we remove all variables from the AMR and duplicate co-referring nodes. An example of this is shown in Figure 1. The variables and co-referring nodes are restored after testing, using the restoring script from

Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017), pages 929–933,
Vancouver, Canada, August 3 - 4, 2017. ©2017 Association for Computational Linguistics

Parameter	Value
Layers	1
Nodes	400
Buckets	(510,510)
Epochs	25–35
Vocabulary	150–200
Learning rate	0.5
Decay factor	0.99
Gradient norm	5

Table 1: Model specifics for the seq2seq model.

Barzdins and Gosko (2016).[1] Wikipedia links are also removed from the training set, but get restored in a separate Wikification post-processing step.

```
(require-01
   :ARG0 (induce-01
      :ARG1 (cell)
      :ARG2 (migrate-01
         :ARG0 cell))
   :ARG1 (bind-01
      :ARG1 (protein
         :name (name :op1 "Crk"))
      :ARG2 (protein
         :name (name :op1 "CAS"))))
```

Figure 1: *"Crk binding to CAS is required for the induction of cell migration"* - seq2seq tree representation.

2.3 Improvements

In this section we describe the methods used to improve the neural semantic parser.

Augmentation AMRs, as introduced by Banarescu et al. (2013), are rooted, directed, labeled graphs, in which the different nodes and triples are unordered by definition. However, in our tree representation of AMRs (see Figure 1), there is an order of branches. This means that we are able to permute this order into a more intuitive representation of the sentence, by matching the word order using the AMR-sentence alignments. An example of this method is shown in Figure 2.

This approach can also be used to augment the training data, since we are now able generate "new" AMR-sentence pairs that can be added to our training set. However, due to the exponential growth, there are often more than 1,000 different AMR permutations for long sentences. We ran multiple experiments to find the best way to use this oversupply of data. Ultimately, we found that it is most beneficial to "double" the training data by adding the best matching AMR (based on word order) to the existing data set.

```
(require-01
   :ARG1 (bind-01
      :ARG1 (protein
         :name (name :op1 "Crk"))
      :ARG2 (protein
         :name (name :op1 "CAS"))))
   :ARG0 (induce-01
      :ARG1 (cell)
      :ARG2 (migrate-01
         :ARG0 cell))
```

Figure 2: *"Crk binding to CAS is required for the induction of cell migration"* - seq2seq representation that best matches the word order.

Super characters We do not necessarily have to restrict ourselves to using only individual characters as input. For example, the AMR relations (e.g. `:ARG0`, `:domain`, `:mod`) can be seen as single entities instead of a collection of characters. This decreases the input length of the AMRs in feature space, but increases the total vocabulary. We refer to these entities as *super characters*. This way, we essentially create a model that is a combination of character and word level input.

POS-tagging Character-level models might still be able to benefit from syntactic information, even when this is added directly to the input structure. Especially POS-tags can easily be added as features to the input data, while also providing valuable information. For example, proper nouns in a sentence often occur with the `:name` relation in the corresponding AMR, while adjectives correlate with the `:mod` relation. We append the corresponding POS-tag to each word in the sentence (using the C&C POS-tagger by Clark et al. (2003)), creating a new super character for each unique tag.

Post-processing In a post-processing step, first the variables and co-referring nodes are restored. We try to fix invalidly produced AMRs by applying a few simple heuristics, such as inserting special characters (e.g. parentheses, quotes) or removing unfinished edges. If the AMR is still invalid, we output a smart default AMR.[2]

We also remove all double nodes, i.e., relation-concept pairs that occur more than once in a branch of the AMR. This form of duplicate output is a common problem with deep learning models,

[1] https://github.com/didzis/tensorflowAMR

[2] This was not necessary for the evaluation data.

since the model does not keep track of what it has already output. We refer to this method as *pruning*.

Wikification Our Wikification method is based on Bjerva et al. (2016), using Spotlight (Daiber et al., 2013). They initially removed wiki links from the input and then tried to restore them in the output by simply adding a wiki link to the AMR if it matches with the name in a `:name` relation. Even though this approach worked well for the LDC data, it did not work for the biomedical data.

This is mainly due to the fact that `:name` nodes are not consistently annotated with a wiki link in the gold biomedical data. 138 unique names that had a corresponding wiki link at least once in the gold data did not have this wiki link 100% of the time. For example *DNA* occurred 86 times as a `:name` in the gold data and was only annotated with a wiki link in 69 cases, while *ERK* occurred 228 times with only 3 annotated wiki links. For this reason we opt for a safe Wikification approach: we only add wiki links to names that were annotated with the same wiki link more than 50% of the time in the gold data. Following our previous example, this means that *DNA* does still get a wiki link, but *ERK* does not.

2.4 CAMR ensemble

As we know that our neural semantic parser is unlikely to outperform a state-of-the-art AMR parser, but is likely to complement it, our strategy is to use an ensemble-based approach. The ensemble comprises the off-the-shelf parser CAMR (Wang et al., 2015) and our neural semantic parser. The implementation of this ensemble is similar to Barzdins and Gosko (2016), choosing the AMR that obtains the highest pairwise Smatch (Cai and Knight, 2013) score when compared to the other AMRs generated for a sentence. This method is designed to ultimately choose the AMR with the most prevalent relations and concepts.

We train CAMR models based on the biomedical data only, the LDC data only and the combination of both data sets. Since CAMR is nondeterministic, we can also train multiple models on the same data set. Ultimately, the best ensemble on the test data consisted of three bio-only models, two bio + LDC models and one LDC-only model. This ensemble was used to parse the evaluation set.

3 Results and Discussion

3.1 Results on Test Set

Table 2 shows the results of all improvement methods tested in isolation on the test set of the biomedical data. Augmenting the data only helps very slightly, while the super characters are responsible for the largest increase in performance. This shows that we do not necessarily have to use only character or word level input in our models, but that a combination of the two might be optimal. The best result was obtained by combining the different methods. This model was then used to parse the evaluation data. Table 3 shows the results of retraining CAMR on different data sets, as well as an ensemble of those models. Adding our seq2seq model to the ensemble only yielded a very small gain in performance.

Feature	**F-score**	**Increase**
Baseline	0.422	
Pruning	0.425	0.7%
Wikification	0.423	0.2%
Augmentation	0.424	0.5%
Super characters	0.481	14.0%
POS-tagging	0.436	3.3%
All combined	0.504	19.4%

Table 2: Results of the different seq2seq models on the test set of the biomedical data.

	F-score
CAMR retrained on LDC	0.399
CAMR retrained on bio	0.585
CAMR retrained on LDC + bio	0.582
Ensemble CAMR	0.588
Ensemble CAMR + seq2seq	0.589

Table 3: Results of retraining CAMR and results of best ensemble models, tested on the biomedical test data.

3.2 Official Results

In Table 4 we see the detailed results of the best seq2seq model and best ensemble on the evaluation data, using the scripts from Damonte et al. (2017).[3] While CAMR has similar scores on the

[3] Unofficial score for seq2seq negation; due to a mistake, all `:polarity` nodes were removed in the official submission. This had no influence on the final F-score.

test data, the score of the seq2seq model decreases by 0.04. It is interesting to note that seq2seq scores equally well without word sense disambiguation, while there is no separate module that handles this.

Setting	seq2seq	Ensemble
Smatch	**0.460**	**0.576**
Unlabeled	0.504	0.623
No WSD	0.463	0.579
Named Entities	0.512	0.576
Wikification	0.458	0.396
Negation	0.141	0.244
Concepts	0.630	0.759
Reentrancies	0.290	0.352
SRL	0.427	0.543

Table 4: Official results on the evaluation set for both the ensemble and the seq-to-seq neural semantic parser.

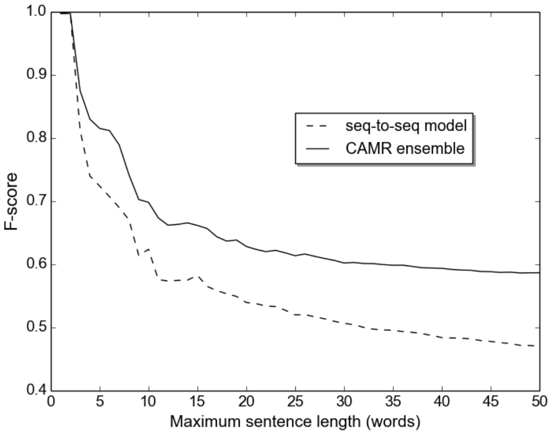

Figure 3: Comparison of CAMR and our seq-to-seq model for different sentence lengths.

3.3 Comparison with CAMR

Although CAMR outperformed our neural semantic parser by a large margin, the seq-to-seq model did produce a better AMR for 108 out of the 500 evaluation AMRs, based on Smatch score. If the CAMR + seq2seq ensemble was somehow able to always choose the best AMR, it obtains an F-score of 0.601, an increase of 2.2% instead of the current 0.2%. This suggests that the current method of combining neural semantic parsers with existing parsers is far from optimal, but that the neural methods do provide complementary information. A different way to incorporate this information would be to pick the most suitable parser based on the input sentence. A classifier that exploits the characteristics of the sentence could be trained to assign a parser to each individual (to be parsed) sentence.

Figure 3 shows the performance of the neural semantic parser and the CAMR ensemble per maximum sentence length. We see that seq-to-seq can keep up with CAMR for very short sentences, but is clearly outperformed on longer sentences. As the sentences get longer, the difference in performance gets bigger, but not much.

4 Conclusion and Future Work

We were able to reproduce the results of the character-level models for neural semantic parsing as proposed by Barzdins and Gosko (2016). Moreover, we showed improvement on their basic setting by using data-augmentation, part-of-speech as additional input, and using super characters. The latter setting showed that a combination of character and word level input might be optimal for neural semantic parsers. Despite these enhancements, the resulting AMR parser is still outperformed by more traditional, off-the-shelf AMR parsers. Adding our neural semantic parser to an ensemble including CAMR (Wang et al., 2015), a dependency-based parser, yielded no noteworthy improvements on the overall performance.

Do these results indicate that neural semantic parsers will never be competitive with more traditional statistical parsers? We don't think so. We have the feeling that we have just scratched the surface of possibilities that neural semantic parsing can offer us, and how they possibly can complement parsers using different strategies. In future work we will explore these.

Acknowledgements

We thank the two anonymous reviewers for their comments. We would also like to thank the Center for Information Technology of the University of Groningen for their support and for providing access to the Peregrine high performance computing cluster. We also used a Tesla K40 GPU, which was kindly donated by the NVIDIA Corporation. This work was funded by the NWO-VICI grant "Lost in Translation – Found in Meaning" (288-89-003).

References

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

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. http://www.aclweb.org/anthology/W13-2322.

Guntis Barzdins and Didzis Gosko. 2016. Riga at semeval-2016 task 8: Impact of smatch extensions and character-level neural translation on amr parsing accuracy. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. Association for Computational Linguistics, San Diego, California, pages 1143–1147. http://www.aclweb.org/anthology/S16-1176.

Johannes Bjerva, Johan Bos, and Hessel Haagsma. 2016. The meaning factory at semeval-2016 task 8: Producing amrs with boxer. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. Association for Computational Linguistics, San Diego, California, pages 1179–1184. http://www.aclweb.org/anthology/S16-1182.

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)*. Association for Computational Linguistics, Sofia, Bulgaria, pages 748–752. http://www.aclweb.org/anthology/P13-2131.

Stephen Clark, James R Curran, and Miles Osborne. 2003. Bootstrapping pos taggers using unlabelled data. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*. Association for Computational Linguistics, pages 49–55. http://www.aclweb.org/anthology/W03-0407.

Joachim Daiber, Max Jakob, Chris Hokamp, and Pablo N. Mendes. 2013. Improving efficiency and accuracy in multilingual entity extraction. In *Proceedings of the 9th International Conference on Semantic Systems (I-Semantics)*.

Marco Damonte, Shay B. Cohen, and Giorgio Satta. 2017. An incremental parser for abstract meaning representation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Association for Computational Linguistics, Valencia, Spain, pages 536–546. http://www.aclweb.org/anthology/E17-1051.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 33–43. http://www.aclweb.org/anthology/P16-1004.

Xiaochang Peng, Chuan Wang, Daniel Gildea, and Nianwen Xue. 2017. Addressing the data sparsity issue in neural amr parsing. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Association for Computational Linguistics, Valencia, Spain, pages 366–375. http://www.aclweb.org/anthology/E17-1035.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015. 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*. Association for Computational Linguistics, Denver, Colorado, pages 366–375. http://www.aclweb.org/anthology/N15-1040.