Garage Sale Semantic Parsing

Kira Droganova[1], **Andrey Kutuzov**[2], **Nikita Mediankin**[1], **and Daniel Zeman**[1]

[1]Charles University, Faculty of Mathematics and Physics, ÚFAL
[2]University of Oslo, Faculty of Mathematics and Natural Sciences, IFI
{droganova|mediankin|zeman}@ufal.mff.cuni.cz
andreku@ifi.uio.no

Abstract

This paper describes the "Prague 2" system submission to the shared task on Cross-Framework Meaning Representation Parsing (MRP, Oepen et al. 2019). The submission is based on several third-party parsers. Within the official shared task results, the submission ranked 12th out of 14 participating systems.

1 Introduction

The CoNLL 2019 shared task is on Meaning Representation Parsing, i.e., finding graphs of semantic dependencies for plain-text English sentences. There are numerous frameworks that define various kinds of semantic graphs; five of them have been selected as target representations in this shared task. The five frameworks are: Prague Semantic Dependencies **(PSD);** Delph-In bilexical dependencies **(DM);** Elementary Dependency Structures **(EDS);** Universal Conceptual Cognitive Annotation **(UCCA);** and Abstract Meaning Representation **(AMR).** See the shared task overview paper (Oepen et al., 2019) for a description of the individual frameworks.

Previous parsing experiments have been described for all these frameworks, and some of the parsers are freely available and re-trainable. Being novices in the area of non-tree parsing, we did not aim at implementing our own parser from scratch; instead, we decided to experiment with third-party software and see how far we can get. Our participation can thus be viewed, to some extent, as an exercise in reproducibility. The challenge was in the number and in the diversity of the target frameworks. No single parser can produce all five target representation types (or at least that was the case when the present shared task started).

Within the shared task, data of all five frameworks are represented in a common JSON-based interchange format (the **MRP** format). This format allows to represent an arbitrary graph structure whose nodes may or may not be anchored to spans of the input text. Using a pre-existing parser thus means that data have to be converted between the MRP interchange format and the format used by the parser; such conversion is not always trivial.

The shared task organizers have provided additional *companion data* where both the training and the test data were preprocessed by UDPipe (Straka and Straková, 2017), providing automatic tokenization, lemmatization, part-of-speech tags and syntactic trees in the Universal Dependencies annotation scheme (Nivre et al., 2016). We work solely with the companion data in our experiments; we do not process raw text directly.

2 Related Work

For the purposes of this work we considered previous work matching the following criteria:

- reporting reasonably good results;
- accompanied by open-source code available to use;
- with instructions sufficient to run the code;
- using only the resources from the shared task whitelist.

Peng et al. (2017) presented a neural parser that was designed to work with three semantic dependency graph frameworks, namely, DM, PAS and PSD. The authors proposed a single-task and two multitask learning approaches and extended their work with a new approach (Peng et al., 2018) to learning semantic parsers from multiple datasets.

The first specialized parser for UCCA was presented by Hershcovich et al. (2017). It utilized

novel transition set and features based on bidirectional LSTMs and was developed to deal with specific features of UCCA graphs, such as DAG structure of the graph, discontinuous structures, and non-terminal nodes corresponding to complex semantic units. The work saw further development in (Hershcovich et al., 2018), where authors presented a generalized solution for transition-based parsing of DAGs and explored multitask learning across several representations, showing that using other formalisms in joint learning significantly improved UCCA parsing.

Buys and Blunsom (2017) proposed a neural encoder-decoder transition-based parser for full MRS-based semantic graphs. The decoder is extended with stack-based embedding features which allows the graphs to be predicted jointly with unlexicalized predicates and their token alignments. The parser was evaluated on DMRS, EDS and AMR graphs. Lexicon extraction partially relies on Propbank (Palmer et al., 2005), which is not in the shared task whitelist. Unfortunately, we were not able to replace it with an analogous white-listed resource.

Flanigan et al. (2014) presented the first approach to AMR parsing, which is based around the idea of identifying concepts and relations in source sentences utilizing a novel training algorithm and additional linguistic knowledge. The parser was further improved for the SemEval 2016 Shared Task 8 (Flanigan et al., 2016). JAMR parser utilizes a rule-based aligner to match word spans in a sentence to concepts they evoke, which is applied in a pipeline before training the parser.

Damonte et al. (2017) proposed a transition-based parser for AMR not dissimilar to the ARCEAGER transition system for dependency tree parsing, which parses sentences left-to-right in real time.

Lyu and Titov (2018) presented AMR parser that jointly learns to align and parse treating alignments as latent variables in a joint probabilistic model. The authors argue that simultaneous learning of alignment and parses benefits the parsing in the sense that alignment is directly informed by the parsing objective thus producing overall better alignments.

Zhang et al. (2019) recently reported results that outperform all previously reported SMATCH scores, on both AMR 2.0 and AMR 1.0. The proposed attention-based model is aligner-free and deals with AMR parsing as sequence-to-graph task. Additionally, the authors proposed an alternative view on reentrancy converting an AMR graph into a tree by duplicating nodes that have reentrant relations and then adding an extra layer of annotation by assigning an index to each node so that the duplicates of the same node would have the same id and could be merged to recover the original AMR graph. This paper looks very promising, but unfortunately we were not able to test the parser due to the paper being published after the end of the shared task.

3 System Description

3.1 DM and PSD

To deal with the DM and PSD frameworks we chose a parser that was described in (Peng et al., 2017). This work explores a single-task and two multitask learning approaches using the data from the 2015 SemEval shared task on Broad-Coverage Semantic Dependency Parsing (SDP, Oepen et al. 2015) and reports significant improvements on the state-of-the-art results for semantic dependency parsing. The parser architecture utilizes arc-factored interface and a bidirectional-LSTM composed with a multi-layer perceptron. Our first intention was to adapt the models that utilize the multitask learning approach. Unfortunately, the project seems to be stalled. The authors did not finish the adaptation of the multitask models for the new version of DyNet, so we were not able to run these models. We proceeded with the single-task model (NeurboParser), in which models for each formalism are trained completely separately. To reproduce the experiment from the paper we needed to perform the following steps:

- Convert the training data from the MRP format to the input format required by the parser. The input format is the same as the one used in the 2015 SemEval Shared Task[1] (see Figure 1 for an example).

- Download pre-trained word embeddings (GloVe, Pennington et al. 2014). We use the same version that is described in the paper – 100-dimensional vectors trained on Wikipedia and Gigaword.

[1] See detailed format description at http://alt.qcri.org/semeval2015/task18/index.php?id=data-and-tools

- Create an additional file with the following information: part-of-speech tag, token ID of the head of the current word, dependency relation. The parser considers syntactic dependencies before it predicts the semantic ones; note that we can obtain this information from the companion data and give it to the parser.

- Run the training script to train the model. The most challenging part was to install and compile the parser. The authors provided the training script with default hyperparameters; however, using some of the documented options resulted in errors on our system. Models are trained up to 20 epochs with Adadelta (Zeiler, 2012).

The single-task model does not predict the frame labels. This is a simple classification problem, similar to lemmatization, so as a quick workaround, we used UDPipe (Straka and Straková, 2017), namely its predictor of morphological features, to simulate such a classifier. First, we converted the training data to the CoNLL-U format[2] replacing morphological features in the sixth column with the frame labels. Next, we trained the model using the instructions from Reproducible Training section of the UDPipe manual.[3]

To produce the final output for the testing data, we first parsed it with the trained models. The input files were produced using companion data. To be more specific, for the UDPipe model input we used tokenization and word forms from companion data. NeurboParser takes the following information as input: token ID, word form, lemma, and part-of-speech tag. Then we merged the frame information predicted by UDPipe with the NeurboParser output and converted it back to the MRP interchange format.

3.2 EDS

We do not have any parser specifically for EDS. However, EDS is closely related to DM (DM is a lossy conversion of EDS, where nodes that do not represent surface words have been removed (Ivanova et al., 2012)). We thus work with the hypothesis that a DM graph is a subset of the corresponding EDS graph, and we submit our DM graph to be also evaluated as EDS.

3.3 AMR

For AMR, we chose the JAMR parser (Flanigan et al., 2014, 2016). The parser is based on a two-part algorithm that identifies concepts using a semi-Markov model and then identifies the relations by searching for the maximum spanning connected subgraph (MSCG) from an edge-labeled, directed graph representing all possible relations between the identified concepts. Lagrangian relaxation (Geoffrion, 1974) is used to ensure semantic well-formedness. For our experiments we used the version that was presented at the 2016 SemEval shared task on Meaning Representation Parsing (May, 2016), in which the authors implemented a novel training loss function for structured prediction, added new sources for concepts and improved features, and improved the rule-based aligner.

The instructions and training scripts were provided by the authors. To run the training, we needed to create a label-set file, which is a list of unique labels collected from the training data and then convert the training data to the parser input format.

The JAMR parser works with the traditional AMR format, which represents an AMR graph in bracketed form (Banarescu et al., 2013), therefore necessitating a two-way conversion between the MRP and AMR formats. The example sentence "There is no asbestos in our products now." would look the following way in AMR format (see also Figure 3 for a visualization of the graph):

```
(a / asbestos :polarity -
    :time (n / now)
    :location (t / thing
        :ARG1-of (p / produce-01
            :ARG0 (w / we))))
```

To facilitate the conversion, we created a Python3 script for each conversion direction.

The main features of conversion from the MRP format to the AMR format are as follows:

- For each sentence, a representation of the graph in the form of source-to-target mapping is obtained from the JSON representation of the list of edges.

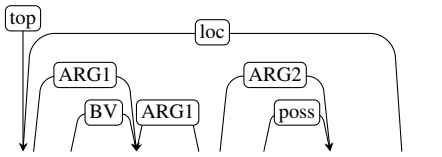
- The graph is traversed starting from the top using depth-first search algorithm outputting

[2] https://universaldependencies.org/format.html

[3] http://ufal.mff.cuni.cz/udpipe/models#universal_dependencies_24_reprodusible_training

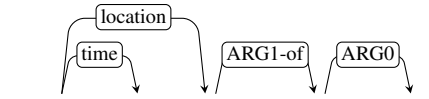
1	There	there	EX	-	-	_	_	_	_	_	_
2	is	be	VBZ	+	+	v_there:e-i	_	_	_	_	loc
3	no	no	DT	-	+	q:i-h-h	_	_	_	_	_
4	asbestos	asbestos	NN	-	-	n:x	ARG1	BV	ARG1	_	_
5	in	in	IN	-	+	p:e-u-i	_	_	_	_	_
6	our	we	PRP$	-	+	q:i-h-h	_	_	_	_	_
7	products	product	NNS	-	-	n:x	_	_	ARG2	poss	_
8	now	now	RB	-	+	time_n:x	_	_	_	_	_
9	.	.	.	-	-	_	_	_	_	_	_
10	"	”	''	-	-	_	_	_	_	_	_

Figure 1: An example of a sentence in the format required by NeurboParser. See Figure 2 for visualization of this DM graph.



“There is no asbestos in our products now.”

Figure 2: DM representation of the example sentence.



“There is no asbestos in our products now.”

Figure 3: AMR representation of the example sentence.

one node on a line in order the nodes are traversed.

- Nodes that were already visited are marked and are not traversed again in order to break possible infinite loops resulting from the cycles in the graph.
- Numeric node ids are substituted with alphanumeric values standard for AMR format: the first letter of the MRP node label is followed by an ordinal number if it is necessary to distinguish multiple nodes starting with the same letter.
- Properties of the node are output on the same line as the node.
- Property values that contain characters that are special for AMR representation, namely a colon (:), are enclosed in straight double quotes, as recommended by the parser documentation, e.g., 20:00 becomes "20:00".

The back conversion has the following features:

- For each sentence, its AMR representation is recursively split into a nested list structure reflecting the nestedness of bracket notation.
- The path starting from the top node is recursively retrieved from the nested list structure.
- The lists of nodes and edges are collected along the path and converted to the MRP format.
- Finally, the alphanumeric node ids are converted to numeric format: the root is assigned 0, then the incremental ids are assigned to the rest of the nodes in order they are visited by depth-first traverse, with the child nodes of the same parent node sorted by rough priority of their connecting edge label:
  - frame arguments are sorted in order of their numbers, e.g., :ARG0 precedes :ARG1;
  - frame arguments are precede semantic relations, e.g., :ARG0 precedes :date;
  - inverse relations are placed after straight ones of the same name, e.g., :ARG0 precedes :ARG0-of.

3.4 UCCA

We decided to adapt JAMR parser that we had already set up to parse AMR data in order to train on UCCA data as well. Since both AMR and UCCA are unanchored, we had theorized that a parser

suitable for AMR might be able to be trained to predict non-surface nodes in UCCA graphs. For this, we needed to convert UCCA graphs from the uniform graph interchange format to AMR-like bracketed representation and vice versa, so the parser would be able to work with sentences in familiar format. The example sentence "There is no asbestos in our products now."" would look the following way in the AMR-like representation (see also Figure 4 for a visualization of the graph):

```
(_1 / _root
    :H (_2 / _h
        :S (t / There)
        :F (i / is)
        :D (n / no)
        :A (a / asbestos)
        :A (_3 / _a
            :R (i1 / in)
            :E (_4 / _e
                :S (o / our))
            :C (p / products))
        :T (n1 / now)
        :U (. / .)
        :U (_ / _quot)))
```

As demonstrated by this example, we introduced the following modifications to the AMR format in order to adapt it for UCCA:

- Since in UCCA nodes that do not directly correspond to surface tokens lack any labels at all, we assign them placeholder labels during conversion, which start with the underscore to differentiate them from labels of surface nodes. Top node is given the _root label, while the rest are given labels that are the same as the label on the edge connecting it with its parent node.

- In UCCA punctuation get its own nodes. In most cases we use the punctuation symbol as the node label, with one exception: we replace the double-quote character (") with _quot because the parser treats the double quote as a special character.

The conversion process is mostly the same as for AMR, with the following notable modifications:

- Conversion from MRP to the AMR-like format:
  - labels for the nodes corresponding to surface tokens are obtained from sentence text and anchors;
  - the list of possible special characters that necessitate the label to be enclosed in double quotes is extended to slash (/) and parentheses;
  - nodes with empty labels are assigned labels as described above;
  - the double-quote label is replaced with _quot as described above.

- Conversion from the AMR-like format to MRP:
  - anchors are recalculated from node labels and sentence text where needed;
  - alphanumeric ids are reassigned to numeric not based on the order the nodes emerge when depth-first traversing the tree, but first assigned to the surface nodes in order of their emergence in the sentence, then to the rest of the nodes, which seems to be the preferred way for UCCA graphs.

4 Results

The results are shown in Table 1. Unfortunately, our results for AMR and UCCA testing sentences were corrupted, thus the official results comprise only scores for DM, PSD and EDS frameworks. However, we do provide the scores for the post-evaluation run for AMR and UCCA frameworks. The results for the complete evaluation set and for the LPPS subset, a 100-sentence sample from The Little Prince annotated in all frameworks, are reported for both the official and unofficial runs.

For reference we provide previously reported original results measured by formalism-specific metrics for both the parsers that we use. Our results for DM and PSD are quite close to the original results reported in (Peng et al., 2017). Original SMATCH scores are reported in (May, 2016). The score reported on the LPPS subset is close to the original score, whereas the score measured on the whole test set is much lower. This difference can be explained by the back conversion procedure, which drops the graphs that contain cycles. There are 36% of such sentences in the whole evaluation set. However, the LPPS subset does not contain any graphs with cycles.

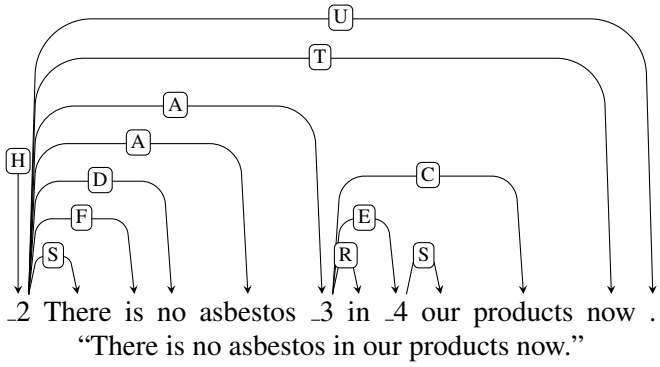


Figure 4: UCCA representation of the example sentence.

	MRP	MRP:DM	MRP:PSD	MRP:EDS	MRP:UCCA	MRP:AMR	SDP:DM	SDP:PSD	SMATCH:AMR
Official run	0.338	0.7902	0.5958	0.3064	0.0000	0.0000	0.8803	0.7689	0.0000
	0.334	0.7767	0.5663	0.3260	0.0000	0.0000	0.8879	0.7950	0.0000
Post-evaluation	0.434	0.7902	0.5958	0.3064	0.1118	0.3645	0.8803	0.7689	0.3513
	0.473	0.7767	0.5663	0.3260	0.1748	0.5194	0.8879	0.7950	0.5081
Previously reported							0.894	0.776	0.56

Table 1: Official run: official results; Post-evaluation: results that were achieved after the submission deadline; Previously reported: original results for utilized parsers. For every metric we show F1 score, except for SDP:DM and SDP:PSD, where we show labeled F1 score; for both runs we provide results for the complete evaluation set (upper line) and the LPPS subset (lower line).

5 Conclusion

We have described the “Prague 2” submission to the CoNLL 2019 shared task on cross-framework meaning representation parsing. This submission stands on three parsers that were previously proposed, implemented and made available by other researchers: NeurboParser, JAMR, and UDPipe. We added several conversion scripts to make the parsers work with the shared task data. We were not able to implement other improvements within the time span of the shared task; we also do not list other publicly available parsers that we thought of testing but failed to make them work.

The main purpose of the present paper is to provide some context to our numbers in the shared task results; the results themselves are far from optimal. Using the official MRP shared task metric (and looking at the unofficial post-evaluation run, which includes AMR and UCCA results), we were relatively successful only in parsing DM. Parsing PSD is obviously harder (these figures are comparable, as we applied the same processing to PSD and DM), and, perhaps unsurprisingly, AMR is the most difficult target of the three. We achieved non-zero score on EDS by simply pretending that the DM graph is EDS. Finally, training an AMR parser on the UCCA representation did not turn out to be a good idea, and our UCCA score is the worst among all the target representations.

Acknowledgements

We thank the authors of the parsers that we used as parts of our system. Their well-documented work allowed us to reproduce their results and conduct our own experiments.

This work is partially supported by the GA UK grant 794417, the SVV project number 260 453, and the grant no. LM2015071 of the Ministry of Education, Youth and Sports of the Czech Republic.

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*, pages 178–186.

Jan Buys and Phil Blunsom. 2017. Robust incremental neural semantic graph parsing. In *ACL*.

Marco Damonte, Shay B. Cohen, and Giorgio Satta. 2017. An incremental parser for abstract meaning representation. In *Proceedings of EACL*.

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, San Diego, California. Association for Computational Linguistics.

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)*, pages 1426–1436.

Arthur M Geoffrion. 1974. Lagrangean relaxation for integer programming. In *Approaches to integer programming*, pages 82–114. Springer.

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2017. A transition-based directed acyclic graph parser for UCCA. In *Proc. of ACL*, pages 1127–1138.

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2018. Multitask parsing across semantic representations. *arXiv preprint arXiv:1805.00287*.

Angelina Ivanova, Stephan Oepen, Lilja Øvrelid, and Dan Flickinger. 2012. Who did what to whom?: A contrastive study of syntacto-semantic dependencies. In *Proceedings of the 6th Linguistic Annotation Workshop*, pages 2–11. Association for Computational Linguistics.

Chunchuan Lyu and Ivan Titov. 2018. Amr parsing as graph prediction with latent alignment. In *ACL*.

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, San Diego, California. Association for Computational Linguistics.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Haji, Christopher Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal Dependencies v1: A Multilingual Treebank Collection. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*, pages 1659–1666, Paris, France. European Language Resources Association.

Stephan Oepen, Omri Abend, Jan Hajič, Daniel Hershcovich, Marco Kuhlmann, Tim O'Gorman, and Nianwen Xue. 2019. MRP 2019. Cross-framework Meaning Representation Parsing. In *Proceedings of the Shared Task on Cross-Framework Meaning Representation Parsing at the 2019 Conference on Natural Language Learning*, pages 1 – 20, Hong Kong, China.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinková, Dan Flickinger, Jan Hajič, and Zdeňka Urešová. 2015. SemEval 2015 task 18: Broad-coverage semantic dependency parsing. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 915–926, Denver, Colorado. Association for Computational Linguistics.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational linguistics*, 31(1):71–106.

Hao Peng, Sam Thomson, and Noah A Smith. 2017. Deep multitask learning for semantic dependency parsing. *arXiv preprint arXiv:1704.06855*.

Hao Peng, Sam Thomson, Swabha Swayamdipta, and Noah A Smith. 2018. Learning joint semantic parsers from disjoint data. *arXiv preprint arXiv:1804.05990*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Milan Straka and Jana Straková. 2017. Tokenizing, pos tagging, lemmatizing and parsing ud 2.0 with udpipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99, Vancouver, Canada. Association for Computational Linguistics.

Matthew D. Zeiler. 2012. Adadelta: An adaptive learning rate method. *ArXiv*, abs/1212.5701.

Sheng Zhang, Xutai Ma, Kevin Duh, and Benjamin Van Durme. 2019. Amr parsing as sequence-to-graph transduction. In *ACL*.