Creating a High Quality Abstract Meaning Representation Dataset Automatically

Johannes Heinecke, Munshi Asadullah, Frédéric Herledan, Géraldine Damnati
Orange Innovation
2 avenue Pierre Marzin
22300 Lannion, France
firstname.name@orange.com

Abstract

As only a few gold training datasets are available today, Abstract Meaning Representation (AMR) parsers are mainly trained on AMR 3.0, the largest dataset (Knight et al., 2020) which contains 55k sentences for training. Even if great progress has been made, leading to parsers that can reach Smatch scores higher than 83% on the AMR 3.0 test dataset, this is not accurate enough to be used in real world application pipelines. More data could help improve performance, but manually annotating sentences is costly. So, we have investigated an approach to automatically create synthetic data using different existing tools and models trained on AMR 3.0. This leads to better parsing performance with Smatch scores increased by 1 to 2 points (depending on the 3 *gold* test datasets used) with models trained on the augmented data.

Keywords: Abstract Meaning Representation, synthetic data

1. Introduction

Abstract Meaning Representation (AMR, Banarescu et al., 2013) is becoming increasingly used in order to represent the core meaning of sentences by graph structures. An AMR graph models actions, events, states and their participants. It relies on concepts defined in PropBank (Kingsbury and Palmer, 2002; Palmer et al., 2005). AMR graphs are usually noted in a serialised format (PENMAN, Kasper, 1989) where variables are instances of a concept (following the '/'). Instances are linked using labelled directed relations (cf. Figure 3 as an example).

In order to train models capable of parsing sentences from natural languages into AMR graphs, annotated corpora are needed. The largest available corpus is AMR 3.0 (Knight et al., 2020), provided by the Linguistic Data Consortium (LDC), LDC2020T02. This corpus is composed of nearly 60 000 sentences and corresponding AMR graphs (55 635 sentences in the training set, 1 722 in the validation set and 1 898 in the test set). The data is from various sources of discussion forums, news (partly translated from Chinese into English) etc[1]. Other, smaller, corpora exist, notably the AMR annotations of questions taken from the QALD-9 corpus (Usbeck et al., 2018), (Lee, 2022) or DRS2AMR (Pavlova et al., 2023)[2] which contains 102 sentences with AMR graphs, which were transformed from Discourse Representation Structures (DRS) as defined in Discourse Representation Theory (Kamp and Reyle, 1993).[3]

To evaluate predicted AMR graphs the Smatch metric is used (Cai and Knight, 2013)[4]. The Smatch algorithm attempts to match nodes of two graphs in an ideal way to have the smallest possible number of node and edge labels which differ. New ways to evaluate AMR graphs, which take into account semantic similarity, have recently been proposed and implemented (Opitz and Frank, 2021; Opitz, 2023). However, we decided to continue to use Smatch since it is still the most widely used evaluation tool for AMR parsing.

Even though text-to-AMR parsers such as AMRlib[5], X-AMR (Cai et al., 2021), or the AMR transition parser (Drozdov et al., 2022; Lee et al., 2022)[6] perform very well (Smatch F1 82.5% and above, depending on the size of the underlying model which is finetuned during training), improving scores is difficult because the available data (for English) to train AMR tools is small (Regan et al., 2024), (Mansouri, 2025, section 5.2). For instance, the AMR 3.0 training corpus covers only 60% of all concepts defined in PropBank. A model can by chance output a PropBank-concept never seen during training (notably those with the very frequent "`-01`" suffix), but we believe that more training data would improve the test score, as supported by Zhang et al. (2024) and Lin et al. (2024).

The contribution of our paper is an approach

[1] https://catalog.ldc.upenn.edu/LDC2020T02

[2] https://gitlab.inria.fr/semagramme-public-projects/drs2amr

[3] Even more smaller AMR datasets are available at https://github.com/flipz357/AMR-World/tree/main/data/reference_amrs

[4] https://github.com/snowblink14/smatch

[5] https://github.com/bjascob/amrlib

[6] https://github.com/IBM/transition-amr-parser

Proceedings of the Fifteenth Language Resources and Evaluation Conference (LREC 2026), pages 11907–11915
11–16 May 2026.

to create high-quality silver data without additional human annotation or validation. In order to do so, we automatically annotate a corpus of sentences and propose an automatic selection strategy, inspired by consensus approaches, which makes use of four different parsers. The main difficulty for graph selection from different parsers is to have a fair comparison between graphs. We introduce an original selection strategy which relies on two-by-two comparisons of candidate graphs based on the Smatch metric. We also validate our approach by evaluating parsers trained on our WikiAMR corpus[7].

In the remainder of this paper, after an overview on the background in section 2, we discuss our method to automatically expand our training dataset to a current total of 203 000 sentences (and an additional 6 700 sentences for the validation dataset) in section 3. We present results in section 4. The models trained with this dataset outperform those trained solely on the AMR 3.0 training data across three independent test sets. All training and experiments have been done on English data only.

In all cases we use an adapted version of AMRlib and the same hyperparameters in all cases to finetune Flan-T5-base (Chung et al., 2022) all trainings.

2. Background

In order to choose the best solution from set of solutions, machine learning provides the idea of hard or soft voting. Whereas the former simply consists of taking the most frequent solution provided by all participating systems (say, classifiers), soft voting takes into account any confidence score provided by the systems together with the solutions. This in contrast to our approach, where we use hard voting, i.e. no confidence available at vote time.

A well known voting system is ROVER (Fiscus, 1997) which aims to reduce word error rates (WER) for automatic speech recognition (ASR). In ROVER several ASR tools propose transcriptions of an audio input, which are then aligned. ROVER then proposes three scoring methods to choose the best sequence of word forms for a given input. Their results show that in total the WER of the combined output is lower than the WER of each of the ASR systems on their own. The major difference between ROVER and our approach is that the former votes for the best word at each transition, whereas the latter takes into account the entire graph. More recently model combination (and thus voting) is used for language identification (Goutte and Léger, 2017) where, similarly as ROVER, a single class (here language name) has to be chosen for a given input.

Our approach is a voting-based method on "inter-system annotation agreement" to achieve the best possible outcome. While the method bears a mild resemblance to Co-Training in that we use multiple parsers that are trained independently (Blum and Mitchell, 1998), it differs fundamentally in the following respects: (1) All the parsers were fully trained once on AMR 3.0 data for use as independent AMR parsers without any iterative addition of samples since our goal is to label data rather than to improve training performance via boosting; (2) There is no expectation that the parsers can produce separate yet complementary information (McClosky et al., 2006); (3) There is no notion of confidence or uncertainty in the outcomes of the parsers, nor any suggestion that they arrived at the outcomes based on independent feature sets (Charniak and Johnson, 2005); (4) Therefore, the concept of ranking, i.e. determining the most confident outcome, is absent. An extension to Co-training is Tri-training (Zhou and Li, 2005) which uses 3 classifiers. Here a training on classifier A is re-trained with the help of unlabeled examples that are labeled by the latest version of the other classifiers if they agree. The role of learner and teacher is rotated. Tri-training has been used to solve several NLP problems, and also in a problem similar to AMR parsing, namely dependency parsing (Søgaard and Rishøj, 2010; Wagner and Foster, 2021). Our method however, does not use an iterative training algorithm using consensus on prediction, rather a voting system to select the best possible annotation generated by independent systems. Even that is significantly different, for example, from the method described in Boulanger et al. (2022) as it kept the annotations with the best combined score compared to out selection of the graph that is most "central" among the graphs produced by the participating systems.

An approach which comes very close to our approach on synthetic AMR data generation is the one described by van Noord and Bos (2017, section 3.4). Here multiple parsers (2 in their case) parse incoming data. If the two generated graphs are too different in terms of Smatch (i.e. lower than 55%) the sentence is not added to the new corpus. The main difference between their and our approach is the way a graph from one of the parsers is chosen (or discarded). van Noord and Bos (2017) attempt taking all graphs from one or the other parser or a mix, but do not apply criteria to chose the first parser or the second for a particular sentence.

Lee et al. (2020) attempt another approach to create synthetic AMR data by generating AMR graphs from input text, regenerating a sentence

[7]The corpus will be available at https://github.com/Orange-OpenSource

from the AMR graph and comparing the original sentence with the sentence generated from the intermediate AMR graph using BLEU. We abandoned such an approach since even when the generated sentence was very close or identical to the original sentence, the AMR graph frequently was incorrect. Our experiments on the AMR 3.0 validation dataset showed that there was no correlation between the BLEU score of the two sentences and the quality of the AMR graph (cf. Fig. 1). On the one hand there are sentences where the sentence generated from a correct AMR graph is different from the original sentence (indicated by 'x', i.e. BLEU = 100%, Smatch < 100%). On the other hand there are also cases where the generated sentence is identical to the original, but the AMR graph is incorrect (marked by '+', i.e. Smatch = 100%, BLEU < 100%).

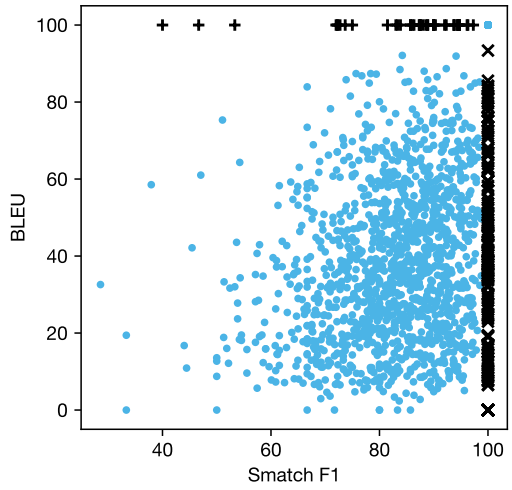

Figure 1: Smatch F1 score and BLEU score for parsed AMR graphs and regenerated sentences of AMR 3.0 validation set.

3. Multi-parsing

3.1. Description of our approach

We use four AMR parsers, 3 of which have different internal architectures, as well as different underlying models:

- AMRlib[8] (**A**),
- X-AMR[9] (**X**),
- and two versions of the AMR-transition-parser[10]:
  - AMR3-joint-ontowiki-seed44 model (**T1**),
  - and AMR3-structbart-L model (**T2**).

Our hypothesis is that if all AMR parsers (with quite different internal architectures) produce similar or identical AMR graphs for a given sentence, then the likelihood that one of these graphs is correct is high. The data for our new WikiAMR is taken mainly from the English version of Wikipedia (44%) and to a lesser extent from Wikinews (26%) and Wikiquote (30%). The main steps are listed below and illustrated in Fig. 2.

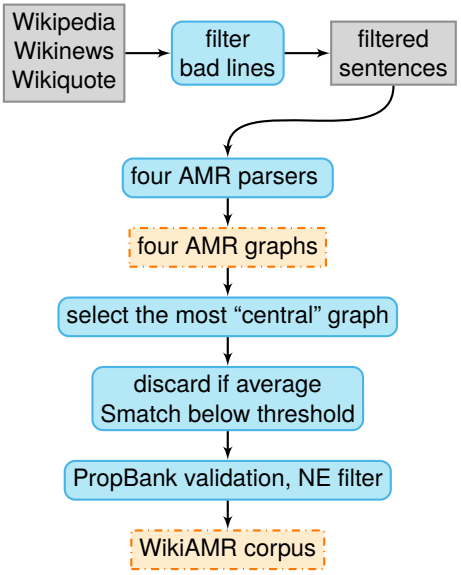

Figure 2: Flow diagram of multi-parsing

Filtering of input sentences: After parsing Wikipedia (etc.) pages, we only keep sentences which do not contain non-Latin characters, nor brackets or braces, and which have a final punctuation mark. We also ignore short sentences with fewer than 10 tokens (defined by white spaces), or sentences which contain long sequences of digits (like ISBN numbers, etc). This is necessary since these kinds of sentences are unlikely to be parsed correctly by any of our parsers. Finally, we delete duplicates. Filtering removes between 60 and 70% of all original sentences.

Multi-parsing: Each candidate sentence is parsed by all four parsers trained on AMR 3.0 resulting in four AMR graphs. For AMRlib, we have trained the model ourselves by finetuning a Flan-T5-Base model, for the other parsers we downloaded models provided by the authors of X-AMR and the AMR-transition parser.

Select most "central" graph: We then calculate the Smatch scores between all graphs of a

[8] https://github.com/bjascob/amrlib
[9] https://github.com/jcyk/XAMR
[10] https://github.com/IBM/transition-amr-parser

given sentence to find the graph which is most similar or "central" to all the others graphs. For example the comparisons of the graphs of a sentence result in a confusion matrix (Table 1). In our example the graph generated by parser A is the most central, i.e. it leads to the highest average Smatch score. If more than one graph has the best score, we take the first graph of those with the best score in the (arbitrary) order A, X, T1, T2.

	A	*X*	*T1*	*T2*	*average*
A	–	90.1	92.3	94.5	**92.3**
X	90.1	–	80.2	82.4	84.2
T1	92.3	80.2	–	84.0	85.5
T2	94.5	82.4	84.0	–	87.0

Table 1: Confusion matrix of Smatch between parser result pairs, parser A has best average (in bold)

Decide whether to keep the graph: The most central graph can nevertheless be rather incorrect. In order to decide whether to add the graph to WikiAMR or to discard it, we check whether its average Smatch score is above a certain threshold (which we set to 90% after experimenting with other values). Only 25% of all graphs pass this threshold.

PropBank validation and Named Entity filter : If the graph passes this test, we further check whether all verbal and adjectival concepts (i.e. those with a numerical suffixes like "`-01`") are in fact defined in PropBank, and that all their outgoing relations ARG*n* in the graph are also defined for the given concept in PropBank.

A final check concerns named entities (NE) which are sometimes distorted in the predicted AMR graph. For instance, the Wikipedia sentence "The church also contains the tombs of Francesco Geminiani and Luigi *Boccherini*" results in a graph with an incorrect representation of "Boccherini" as "Baccolini" (cf. Fig. 3). We detect these errors by trying to find the names mentioned in the AMR graph in the corresponding sentence. Apart from some place names (adjective in the sentence ("...British...") but noun in the graph `(n / name :op1 "Britain")` which are taken from a list) this is an easy task.

We suspect that these errors are due to the underlying language models of the 4 parsers (like Flan-T5 in our version of AMRlib), since the training data (AMR 3.0) for the 4 parsers does not contain these kind of errors. We detect these errors by verifying whether the concatenated strings of the `:opN` relations occur in the sentence. We use a list of named entities which occur in an adjectival form in the sentence, but as proper nouns in the

```
(c / contain-01
   :ARG0 (c2 / church)
   :ARG1 (t / tomb
      :poss (a / and
         :op1 (p / person
            :name (n / name
               :op1 "Francesco"
               :op2 "Geminiani"))
         :op2 (p2 / person
            :name (n2 / name
               :op1 "Luigi"
               :op2 "Baccolini"))))
   :mod (a2 / also))
```

Figure 3: Incorrect representation of "Luigi Boccherini" as "Luigi Baccolini" in a AMR graph generated from the sentence "The church also contains the tombs of Francesco Geminiani and Luigi *Boccherini*".

graph to be able to verify these too (e.g. *French* : `:op1 "France"`). If they do not occur, we delete sentences with an incorrect name from WikiAMR (about 5.6% of all sentences).

3.2. Statistical description of the resulting corpus

Table 2 shows how much each parser contributed to the final dataset. For 12% of sentences, the average Smatch for all parsers was identical, in 9.4% of sentences, all parsers yielded identical graphs.

best parser	*%*	*best parser*	*%*
A	7.9	**T1**	8.5
A, T1	2.1	**T1**, T2	9.9
A, T1, T2	5.9	**T1**, T2, X	4.7
A, T1, T2, X	12.0	**T1**, X	1.5
A, T1, X	1.4	**T2**	22.3
A, T2	6.2	**T2**, X	4.6
A, T2, X	4.2	**X**	6.2
A, X	2.6		

Table 2: Contribution of each parser to WikiAMR. The parser marked in bold has the chosen graph, more than one parser name in a line indicates that all had identical average Smatch scores. The total contribution for each parser was: A: 42.3%, X: 6.2%, T1: 24.6%, and T2: 26.9%.

One could argue that the approach would only select short and simple sentences. Actually, even though the average length of discarded sentences is higher than the length of selected ones, the overall average length of sentences in WikiAMR is higher than the average length of sentences from AMR 3.0 (cf. Table 3). For more details see Figures 4, 5 and 6.

We stop the generation of WikiAMR after having created a training data set of about 203 000

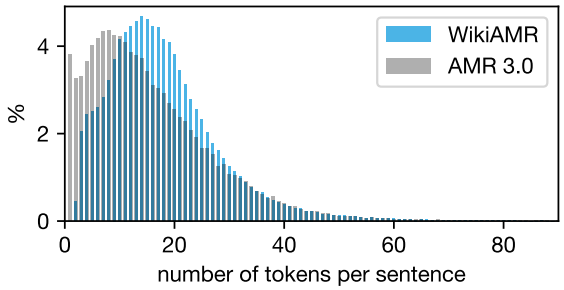

Figure 4: Distribution of tokens per sentence for WikiAMR in % of all sentences. Mean length is 17.86, median 16 and standard deviation 10.17. For AMR 3.0 (gray) mean is 15.68, medium 13 and standard deviation 11.75.

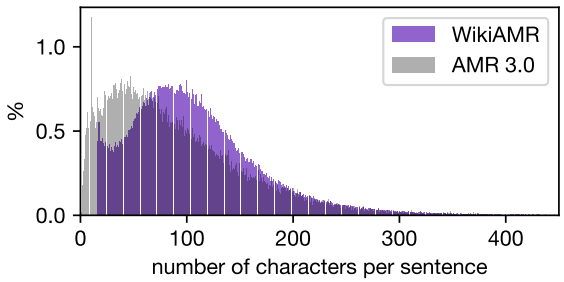

Figure 5: Distribution of characters per sentence for WikiAMR in % of all sentences. Mean length is 108.57, median 99, standard deviation 62.63. For AMR 3.0 (gray) mean is 91.03, median: 75 and standard deviation 69.27

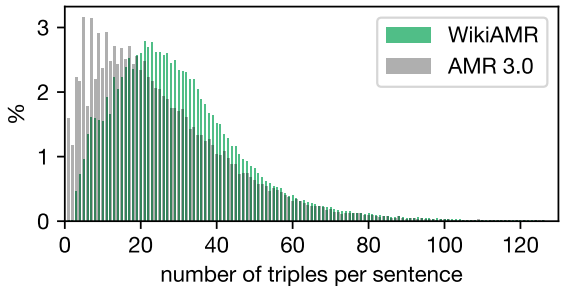

Figure 6: Distribution of triples per graph for WikiAMR in % of all sentences. Mean length is 29.96, median 27, standard deviation 17.40. For AMR 3.0 (gray) mean is 25.23, median: 21 and standard deviation 18.85.

sentences and a validation dataset of about 6 700. At this moment only 5% of concepts and 8% of relations present in the AMR 3.0 test dataset, are missing in WikiAMR. As a comparison, all relations in the AMR 3.0 test dataset are also found in the

corpus	*tokens*	*characters*	*triples*
AMR 3.0	15.58	91.03	25.23
WikiAMR	17.86	108.57	29.96
discarded	34.19	208.57	52.03

Table 3: Average length of the 3 test datasets

AMR 3.0 train dataset, and only 4.5% of concepts found in the AMR 3.0 test dataset are not found in the AMR 3.0 train dataset.

For our experiments, we finetune Flan-T5-Base (256M parameters) using the entire dataset of WikiAMR for training and validation. We use a single RTX-3090 GPU, with a batchsize of 7, a maximum of 30 epochs and early stopping. the learning rate is initialised at 10^{-5}, the LR-scheduler type is set to "linear". The training stopped after 12 epochs (early stopping with a patience of 3) and takes nearly 35 hours.

4. Results

We ran the three gold test corpora mentioned above (AMR 3.0 test corpus with 1898 sentences, QALD-9 test with 150 sentences and DRS2AMR with 102 sentences), on the model trained on WikiAMR and all tests had better results than using a model trained on the AMR 3.0 training corpus (see Table 4). In particular the results for the QALD-9 test corpus improved (+1.9 points). This improvement can be explained by the bigger size (WikiAMR train contains 203 000 sentences compared to the 55 635 sentences in the train set of AMR 3.0). But there are other explanations: AMR 3.0 contains nearly no questions, whereas QALD-9 contains predominantly short, quiz-like questions. Our WikiAMR contains nearly 700 questions (short and long, with and without *to do*-periphrasis), so we can attribute this improvement to the WikiAMR dataset.

All trainings have been made using AMRlib and Flan-T5 as the base model. We tried to finetune The AMR transition parsers, but did not succeed (old version of python, cuda, pytorch and fairseq which we could not install on our hardware).

In Table 5 the baselines are shown. All except the text on DRS2AMR are lower than our results on a model trained on WikiAMR.

4.1. Error Analysis

In order to know why a model finetuned with WikiAMR is more performant we first checked which model generates which errors. In a first step we had a closer look to concepts in the test data sets which are either already seen or unseen in the training dataset (Table 6). With both train sets the number of unseen and seen concepts is more

train↓ test→	*AMR 3.0*	*QALD-9*	*DRS2AMR*
AMR 3.0	82.5	87.3	83.2
WikiAMR	82.8 (+0.3)	89.2 (+1.9)	84.2 (+1.0)
both	84.4 (+1.9)	88.8 (+1.5)	84.3 (+1.1)

Table 4: Smatch F1 Test results on a model trained on AMR 3.0 (baseline), WikiAMR, and the combined AMR 3.0 and WikiAMR training corpora (improvement w.r.t. baseline in parentheses)

parser↓ test→	*AMR 3.0*	*QALD-9*	*DRS2AMR*
A	82.5	87.3	83.2
X	81.7	86.2	82.9
T1	84.0	88.4	84.5
T2	82.7	86.9	83.6

Table 5: Smatch F1 Test results of the 4 parsers trained on AMR 3.0. Models for X-AMR (X) and AMR-transition parser (T1, T2) provided by authors.

or less similar even though they are note identical. We then counted how often the seen and unseen concepts are correctly (true positives, TP) or incorrectly (False positives, FP) predicted or ignored (false negatives, FN), cf. Table 7. These numbers suggest that the model trained on WikiAMR is better on seen concepts. On unseen concepts a AMR 3.0 models looks better at first sight, however the very low number of unseen concepts does not allow a conclusive interpretation of the outcome.

In a second step we calculated the difference between the Smatch score of the graph of each test sentence predicted by our WikiAMR-model and the Smatch score of the graph predicted by the AMR 3.0 trained model. The differences ranged from 55 points in favour of WikiAMR to 47 points in favour of the AMR 3.0 trained model.

In 655 out of the 1898 sentences of the AMR 3.0 test set, the WikiAMR model scored better, in 593 cases our model scored less well. The differences, however, cannot be easily categorized. In general the WikiAMR-model was better in identifying long named entities (organisations, official country names like "the Democratic Republic of …" or more correct PropBank rolesets, e.g. `up-02` vs. `increase-02` in the sentence wb.eng_0003.9 of the AMR 3.0 test set.

train set	(test dataset) unseen concept		
	AMR 3.0	QALD-9	DRS2AMR
AMR 3.0	181	17	24
WikiAMR	224	11	24

train set	(test dataset) seen concepts		
	AMR 3.0	QALD-9	DRS2AMR
AMR 3.0	21919	898	383
WikiAMR	21876	904	383

Table 6: Absolute number of unseen (top) and seen concepts in test files with respect to the training dataset.

		test set (unseen concepts)		
train set		AMR 3.0	QALD-9	DRS2AMR
AMR	FN	0.55%	0.00%	0.00%
	FP	45.05%	64.71%	66.67%
	TP	**54.4%**	**35.29%**	**33.33%**
WikiAMR	FN	3.03%	0.00%	0.00%
	FP	76.62%	81.82%	75.00%
	TP	**20.35%**	**18.18%**	**25.00%**

		test set (seen concepts)		
train set		AMR 3.0	QALD-9	DRS2AMR
AMR	FN	6.74%	2.92%	2.54%
	FP	10.63%	7.14%	8.14%
	TP	**82.62%**	**89.95%**	**89.31%**
WikiAMR	FN	6.74%	2.27%	2.79%
	FP	9.89%	6.92%	8.38%
	TP	**83.37%**	**90.81%**	**88.83%**

Table 7: False Negatives/Postives and True Positives (bold) with Unseen (top) and seen concepts

5. Conclusion and Outlook

In this paper we present a novel approach to creating a high quality training corpus for AMR. Three independent gold test datasets prove that this approach is valid, since results obtained on these test datasets were better with the model trained on this new data than with a model trained on the standard AMR 3.0 training corpus. Robust filtering on the input side and relatively high thresholds in the selection phase ensures that graphs which do not represent the corresponding sentence correctly, are not integrated into the final corpus.

Since we are not assigning simple classes to words or sentences but trying to find the best structure for a given input, this approach should also work to create a corpus of sentences annotated with syntax such as for instance the treebanks provided by the Universal Dependencies project (Nivre et al., 2020)[11]. We plan to use at least three (architecturally) different UD parsers which already produce syntax trees with reasonable labelled attachment score (LAS) to test this hypothesis.

6. Limitations

Even though we experimented with other (major) languages, currently WikiAMR only contains English sentences and their corresponding AMR graph. One reason is that we did not have access

[11] https://universaldependencies.org

to several AMR parsers (or rather models) that perform well on additional languages which would have allowed us to do the voting. Existing work on other languages in the literature always produced much lower Smatch scores than English. It is not yet clear whether our system could have worked as well in this case.

We also did not provide an ablation study. For instance how would a model perform which is trained on 210k graphs provided just by one parser?

Another limitation is linked to the quality ot the existing data to train the parsers. If, for instance, we need to create a corpus on noisy data, but the existing parsers have all been trained on AMR 3.0, then there is a high probability that the graphs selected by our multiparsing algorithm will still be similar but imperfect, and consequently the generated corpus could not be used to create a model which performs better than a model trained on manually curated data.

7. Bibliographical 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, Sofia, Bulgaria. Association for Computational Linguistics.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, COLT' 98, page 92–100, New York, NY, USA. Association for Computing Machinery.

Hugo Boulanger, Thomas Lavergne, and Sophie Rosset. 2022. Generating unlabelled data for a tri-training approach in a low resourced NER task. In *Proceedings of the Third Workshop on Deep Learning for Low-Resource Natural Language Processing*, pages 30–37, Hybrid. Association for Computational Linguistics.

Shu Cai and Kevin Knight. 2013. Smatch: an Evaluation Metric for Semantic Feature Structures. In *51st Annual Meeting of the Association for Computational Linguistics*, page 748–752, Sofia, Bulgaria. Association for Computational Linguistics.

Yitao Cai, Zhe Lin, and Xiaojun Wan. 2021. Making Better Use of Bilingual Information for Cross-Lingual AMR Parsing. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, page 1537–1547, Online. Association for Computational Linguistics.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 173–180, Ann Arbor, Michigan. Association for Computational Linguistics.

Hyung Won Chung, Le Hou, Shayne Longpre, Zoph Barret, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehgani, Siddhartha Brahma, Albert Webson, Shane Shixiang Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincen Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling Instruction-Finetuned Language Models. https://arxiv.org/abs/2210.11416.

Andrew Drozdov, Jiawei Zhou, Radu Florian, Andrew McCallum, Tahira Naseem, Yoon Kim, and Ramón Astudillo. 2022. Inducing and using alignments for transition-based AMR parsing. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1086–1098, Seattle, United States. Association for Computational Linguistics.

Jonathan G. Fiscus. 1997. A post-processing system to yield reduced word error rates: Recognizer Output Voting Error Reduction (ROVER). In *IEEE Workshop on Automatic Speech Recognition and Understanding Proceedings*, pages 347–354. 1997.

Cyril Goutte and Serge Léger. 2017. Exploring optimal voting in native language identification. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 367–373, Copenhagen, Denmark. Association for Computational Linguistics.

Hans Kamp and Uwe Reyle. 1993. *From Discourse to Logic. Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory*. Studies in Linguistics and Philosophy 42. Kluwer, Dordrecht.

Robert T. Kasper. 1989. A flexible interface for linking applications to Penman's sentence gen-

erator. In *Speech and Natural Language: Proceedings of a Workshop Held at Philadelphia, Pennsylvania, February 21-23, 1989*.

Paul Kingsbury and Martha Palmer. 2002. From TreeBank to PropBank. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC'02)*, Las Palmas, Canary Islands - Spain. European Language Resources Association (ELRA).

Young-Suk Lee, Ramón Astudillo, Hoang Thanh Lam, Tahira Naseem, Radu Florian, and Salim Roukos. 2022. Maximum Bayes Smatch ensemble distillation for AMR parsing. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5379–5392, Seattle, United States. Association for Computational Linguistics.

Young-Suk Lee, Ramón Fernandez Astudillo, Tahira Naseem, Revanth Gangi Reddy, Radu Florian, and Salim Roukos. 2020. Pushing the limits of AMR parsing with self-learning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3208–3214, Online. Association for Computational Linguistics.

Haowei Lin, Baizhou Huang, Haotian Ye, Qinyu Chen, Zihao Wang, Sujian Li, Jianzhu Ma, Xiaojun Wan, James Zou, and Yitao Liang. 2024. Selecting large language model to fine-tune via rectified scaling law.

Behrooz Mansouri. 2025. Survey of abstract meaning representation: Then, now, future.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 152–159, New York City, USA. Association for Computational Linguistics.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Jan Hajič, Christopher Manning, Sampo Pyysalo, Sebastian Schuster, Tyers Francis, and Daniel Zeman. 2020. Universal Dependencies v2: An Evergrowing Multilingual Treebank Collection. In *LREC 2020*, Marseille. ELRA.

Juri Opitz. 2023. SMATCH++: Standardized and extended evaluation of semantic graphs. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 1595–1607, Dubrovnik, Croatia. Association for Computational Linguistics.

Juri Opitz and Anette Frank. 2021. Towards a decomposable metric for explainable evaluation of text generation from AMR. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1504–1518, Online. 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.

Michael Regan, Shira Wein, George Baker, and Emilio Monti. 2024. MASSIVE multilingual Abstract Meaning Representation: A dataset and baselines for hallucination detection. In *Proceedings of the 13th Joint Conference on Lexical and Computational Semantics (*SEM 2024)*, pages 1–17, Mexico City, Mexico. Association for Computational Linguistics.

Anders Søgaard and Christian Rishøj. 2010. Semi-supervised dependency parsing using generalized tri-training. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 1065–1073, Beijing, China. Coling 2010 Organizing Committee.

Ricardo Usbeck, Ria Hari Gusmita, Muhamad Saleem, and Axel-Cyrille Ngonga Ngomo. 2018. 9th challenge on question answering over linked data (QALD-9). In *Joint proceedings of the 4th Workshop on Semantic Deep Learning (SemDeep-4) and NLIWoD4*.

Rik van Noord and Johannes Bos. 2017. Neural semantic parsing by character-based translation: Experiments with abstract meaning representations. *Computational Linguistics in the Netherlands Journal*, 7:93–108.

Joachim Wagner and Jennifer Foster. 2021. Revisiting tri-training of dependency parsers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9457–9473, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Biao Zhang, Zhongtao Liu, Colin Cherry, and Orhan Firat. 2024. When scaling meets llm finetuning: The effect of data, model and finetuning method.

Zhi-Hua Zhou and Ming Li. 2005. Tri-training: exploiting unlabeled data using three classifiers. *IEEE Transactions on Knowledge and Data Engineering*, 17(11):1529–1541.

8. Language Resource References

Kevin Knight and Bianca Badarau and Laura Baranescu and Claire Bonial and Madalina Bardocz and Kira Griffitt and Ulf Hermjakob and Daniel Marcu and Martha Palmer and Tim O'Gorman and Nathan Schneider. 2020. *Abstract Meaning Representation (AMR) Annotation Release 3.0*. Linguistic Data Consortium. distributed via LDC: LDC2020T02, 3.0, ISLRN 676-697-177-821-8.

Lee, Young-Suk. 2022. *Human annotations of AMR graphs for QALD-9-AMR treebank*. IBM.

Pavlova, Siyana and Amblard, Maxime and Guillaume, Bruno. 2023. *Bridging Semantic Frameworks: mapping DRS onto AMR*. Association for Computational Linguistics.

A. Appendix

A.1. Data size and distribution

The diagrams in Figures 4 and 5 (above) show the number of sentences in WikiAMR (coloured bars) for a given length in terms of tokens and characters; Figure 6 shows the number of graphs with a given number of triples. The black overlay are the corresponding values for the AMR 3.0 training data in comparison. The longest sentence in WikiAMR has 111 tokens (732 characters), the longest graph has 173 triples. For the AMR 3.0 training data the longest sentence has 216 tokens or 1269 characters, the longest graph has 313 triples.

As mentioned in section 3, the sentences which were discarded by our decision algorithm are in average longer than the sentences of WikiAMR. The longest sentence has 284 tokens (1987 characters), the longest graph has 205 triples (AMRlib parser); cf. Fig. 7, 8 and 9.

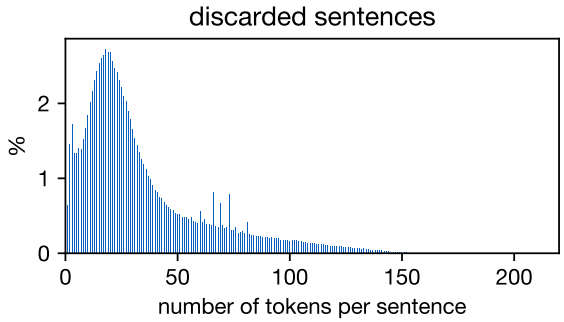

Figure 7: Distribution of number of tokens per sentence for sentences which were *discarded* in % of all sentences. Mean length is 34.19, median 25 and standard deviation 28.11

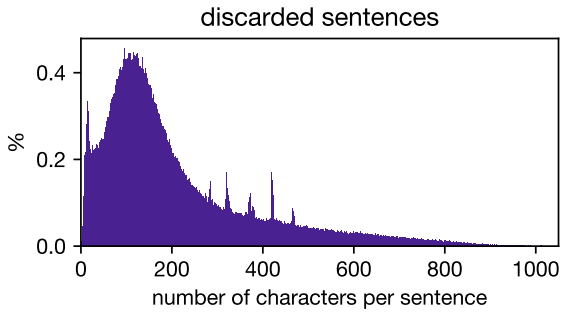

Figure 8: Distribution of number of characters per sentence which were *discarded* in % of all sentences. Mean length is 208.57, median 152, standard deviation 172.45

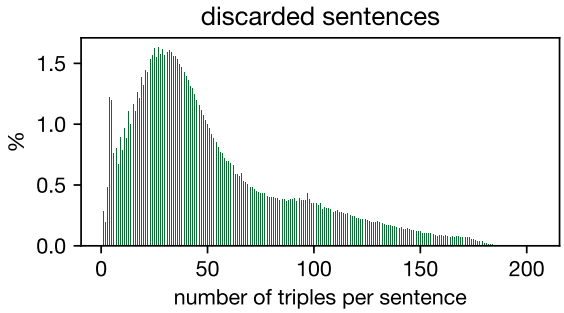

Figure 9: Distribution of number of triples per graph for sentences which were *discarded* in % of all sentences. Mean length is 52.03, median 41, standard deviation 37.87