SMARAGD: Learning SMatch for Accurate and Rapid Approximate Graph Distance

Juri Opitz **Philipp Meier** **Anette Frank**
Dept. of Computational Linguistics
Heidelberg University
69120 Heidelberg
{opitz,meier,frank}@cl.uni-heidelberg.de

Abstract

The similarity of graph structures, such as Meaning Representations (MRs), is often assessed via structural matching algorithms, such as SMATCH (Cai and Knight, 2013). However, SMATCH involves a combinatorial problem that suffers from NP-completeness, making large-scale applications, e.g., graph clustering or search, infeasible. To alleviate this issue, we learn SMARAGD: Semantic Match for Accurate and Rapid Approximate Graph Distance. We show the potential of neural networks to approximate SMATCH scores, i) in linear time using a machine translation framework to predict alignments, or ii) in constant time using a Siamese CNN to directly predict SMATCH scores. We show that the approximation error can be substantially reduced through data augmentation and graph anonymization.

1 Introduction

Semantic graphs such as Meaning Representation (AMR) are directed, rooted and acyclic, and labeled. For instance, in AMR (Banarescu et al., 2013) labels indicate the events and entities of a sentence, and structures capture semantic roles and other key semantics such as coreference.

Often, pairs of MRs need to be studied, using MR metrics. Classically, MRs are compared to assess Inter Annotator Agreement in SemBanking or for the purpose of parser evaluation, typically using the *structural* SMATCH metric (Cai and Knight, 2013; Opitz, 2023). Going beyond these applications, researchers have leveraged SMATCH-based MR metrics for NLG evaluation (Opitz and Frank, 2021; Manning and Schneider, 2021), for re-inforcing AMR parsers (Naseem et al., 2019), as a basis for a COVID-19 semantics-based search engine (Bonial et al., 2020), comparison of cross-lingual AMR (Uhrig et al., 2021; Wein et al., 2022), and fine-grained argument similarity assessment (Opitz et al., 2021b). Many of these extended scenarios greatly profit from a *quick similarity computation*. Also, additional future applications can be anticipated that require fast metric inference: e.g., corpus linguists who want to find instantiations of abstract semantic patterns in a large corpus.

But graph metrics typically suffer from a high time complexity: Computation of SMATCH is NP-hard (Nagarajan and Sviridenko, 2009), and it can take more than a minute to compare some 1,000 AMR pairs (Song and Gildea, 2019). To understand that this can become problematic in many setups, consider a hypothetical user who desires exploring a (small) AMR-parsed corpus with only $n = 1,000$ instances via clustering. The (symmetric) SMATCH needs to be executed over $(n^2 - n)/2 = 499,500$ pairs, resulting in a total time of more than 6 hours.

This high time complexity is a well-known bottleneck and negatively impacts AMR evaluation time (Song and Gildea, 2019), as well as parsing efficency of approaches involving re-inforcement learning (Naseem et al., 2019) or graph ensembling (Hoang et al., 2021), where the SMATCH metric is executed with high frequency. Furthermore, given recent interest into larger meaning representations that cover multiple sentences, such as multi-sentence AMR (O'Gorman et al., 2018), dialogue AMR (Bonial et al., 2021) or discourse representation structures (Kamp, 1981; van Noord et al., 2018), we anticipate that this problem will become more pressing in the future.

Testing ways to mitigate these issues, we propose a method that learns to match semantic graphs from a teacher SMATCH, and show that this can reduce AMR clustering time from hours to seconds, with only little expected loss in accuracy.

Our contributions are:

1. We explore three different neural approaches to synthesize the combinatorial graph metric

SMATCH from scratch.

2. We show that we can approximate SMATCH up to a small error, by leveraging novel data augmentation tricks.

Our code is available at: `https://github.com/PhMeier/Smaragd/`.

2 Related work

Other metrics for MR similarity Recently, researchers have proposed AMR metrics beyond SMATCH. We can distinguish two lines of work: i) metrics aiming at extreme efficiency by skipping the alignment and extracting graph parts via breadth-first traversal (Song and Gildea, 2019; Anchiêta et al., 2019). ii) Weisfeiler-Leman graph metrics that aim to reflect human similarity ratings (Opitz et al., 2021a). Opitz et al. (2020) make an argument for the importance of graph alignment.

Algorithm synthesis Neural networks have been studied for solving other problems efficiently. Examples range from sorting numbers (Graves et al., 2014; Neelakantan et al., 2016) to solving elaborated tasks such as symbolic integration (Lample and Charton, 2019), the famous traveling salesman problem (Gambardella and Dorigo, 1995; Budinich, 1996; Bello et al., 2016; Zhang et al., 2021), and computer programs (Balog et al., 2016; Nye et al., 2020; Chen et al., 2021). The 'long-range arena' benchmark (Tay et al., 2021) includes algorithm synthesizing tasks, such as 'listOps' (learning to calculate), or Xpath (tracing a squiggly line), which prove challenging even for SOTA architectures. Since structural graph matching with SMATCH constitutes a very hard combinatorial problem, investigating efficient neural approximations seems an interesting challenge in general – beyond the use-case of rapid graph distance calculation.

3 Learning NP-hard graph alignment

The SMATCH metric measures the structural overlap of two graphs. We i) compute an alignment between variable nodes of graphs and ii) assess triple matches based on the provided alignment. Formally, we start with two graphs a and b with variable nodes $X = (x_1, ...x_n)$ and $Y = (y_1...y_m)$. The goal is then to find an optimal *alignment*

$$map^\star : X \rightarrow Y, \tag{1}$$

searching for a map that maximizes the number of *triple matches* for the two graphs. For instance, assume two AMR triples `(x, ARG0, y)` $\in \mathcal{G}$ and `(u, ARG0, v)` $\in \mathcal{G}'$. If $x = u$ and $y = v$, we count *one* triple match. Finally:

$$SMATCH = \max_{map} score(a, b, map) \tag{2}$$

Researchers typically use a harmonic mean based overlap $score = F1 = 2PR/(P + R)$, where $P = |triples(a) \cap triples(b)|/|triples(a)$ and $R = |triples(a) \cap triples(b)|/|triples(b|$.

3.1 Setup

Experimental data creation We create the data for our experiments as follows: 1. We parse 59,255 sentences of the LDC2020T02 AMR dataset with a parser (Lyu and Titov, 2018) to obtain graphs that can be aligned to reference graphs; 2. For every parallel graph pair (a, b), we use SMATCH (ORACLE) to compute an F1 score s and the alignment $map^\star$, yielding an extended data tuple $(a, b, s, map^\star)$ We shuffle the data and split it into training, development and test set (56255-1500-1500).

Objective and approach The task is to reproduce the teacher ORACLE as precisely as possible. We design and test three different approaches. The first is indirect, in that it predicts the alignment, from which we compute the score. The second directly predicts the scores. The third approach enhances the second, to make it even more efficient.

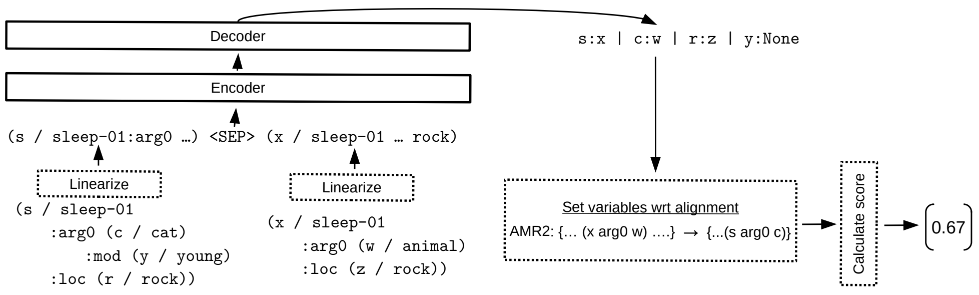
3.2 Synthesis option I: Alignment learning

Here, we aim to learn the alignment itself (Eq. 1) with an NMT model, as illustrated in Figure 1. For the input, we linearize the two AMRs and concatenate the linearized token sequences with a special `<SEP>` token. The output consists of a sequence $x_j{:}y_k$ ... $x_i{:}y_m$ ... where in every pair $u{:}v$, u is a variable node from the first AMR mapped to a node v from the second AMR. The SMATCH score is then calculated based on the predicted alignment.

To predict the node alignments/mapping of variables, we use a transformer based encoder-decoder NMT model. Details about the network structure and hyperparameters are stated in Appendix A.1.

3.3 Synthesis option II: SMATCH prediction

In this setup, we aim to predict SMATCH F1 scores for pairs of AMRs directly, in a single step. This means that we directly learn Eq. 2 with a neural network and our target is the ORACLE F1 score.



Decoder
Encoder
s:x | c:w | r:z | y:None
(s / sleep-01:arg0 …) <SEP> (x / sleep-01 … rock)
Linearize
Linearize
(s / sleep-01 :arg0 (c / cat) :mod (y / young) :loc (r / rock))
(x / sleep-01 :arg0 (w / animal) :loc (z / rock))
Set variables wrt alignment
AMR2: {... (x arg0 w) ....} → {...(s arg0 c)}
Calculate score
0.67

Figure 1: Seq2seq SMATCH alignment-learner.

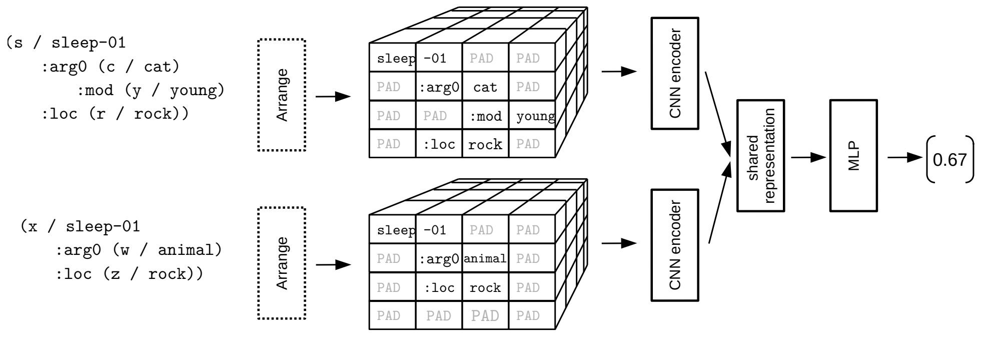


Figure 2: Implicit CNN-based SMATCH graph metric predictor.

To learn this mapping, we adapt the convolutional neural network (CNN) of Opitz (2020), as shown in Figure 2. The model was originally intended to assess AMR accuracy (Opitz and Frank, 2019), i.e., measuring AMR parse quality without a reference. Taking inspiration from human annotators, who exploit a spatial 'Penman' arrangement of AMR graphs for better understanding, it models directed-acyclic and rooted graphs as 2d structures, employing a CNN for processing, which is highly efficient. To feed a pair of AMRs, we remove the dependency graph encoder of the model and replace it with the AMR graph encoder. Moreover, we increase the depth of the network by adding one more MLP layer after convolutional encoding. A basic mean squared error is employed as loss function. More details about hyperparameters are stated in Appendix A.2.

3.4 Synthesis option III: AMR Vector learning

Inspired by Reimers and Gurevych (2019), we aim to make the CNN even more efficient, by alleviating the need for pair-wise model inferences. Instead of computing a shared representation of two CNN-encoded graphs, we process each representation with an MLP (w/ shared parameters), to obtain two vectors $NN(a)$ and $NN(b)$. These vectors are then tuned with signal from ORACLE(s):

$$\mathcal{L} = \sum_{(a,b,s)} \Big([1 - |NN(a) - NN(b)|] - s \Big)^2, \quad (3)$$

where || is returns a vector distance $\in [0, 1]$. This approach enables extremely fast search and clustering: the required (clustering-)model inferences are $O(n)$ instead of $O(n^2)$, since the similarity is achieved with simple linear vector algebra.

3.5 Data compression and extension tricks

Vocabulary reduction trick The SMATCH metric measures the structural overlap of two graphs. This means that we can greatly reduce our vocabulary, by assigning each graph pair a *local vocabulary* (see Figure 3, 'anonymize').

First, we gather all nodes from two graphs a and b, computing a joint vocabulary over the concept nodes. We then relabel the concepts with integers starting from 1. E.g., consider AMR a: *(r / run-01 :ARG0 (d / duck))*, and AMR b: *(x / run-01 :ARG0 (y / duck) :mod (z / fast))*. The gold alignment is $map^\star = \{(r, x), (d, y), (\emptyset, z)\}$. Now, we set the shared concepts and relations to the same index *run=run=1* and *duck=duck=2* and *:ARG0=:ARG0=3* and distribute the rest of the indices *r=4, d=5, x=6, y=7, z=8, fast=9, :mod=10*. This yields equivalent AMRs a' = *(4 / 1 :3 (5 / 2))*

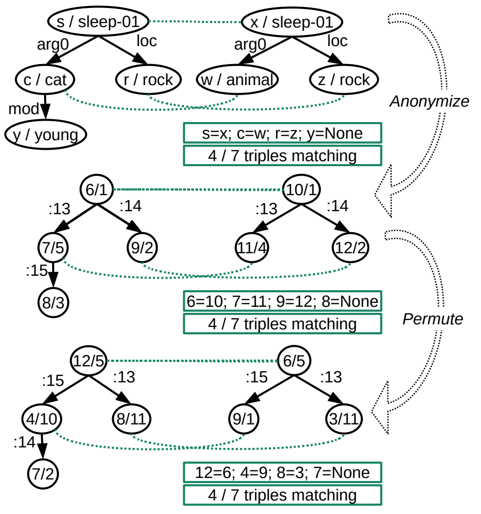


Figure 3: AMR graph anonymization and permutation.

	data trick	Eq. 2	Pea's ρ	time$^{(secs)}$
ORACLE	na	77.5	100	28680
rand. baseline	na	13.5	22.2	0.4
align. synthesis		39.0	52.8	1089
align. synthesis	voc	64.5	80.0	1089
align. synthesis	voc+aug	76.4	**98.4**	1089
score synthesis		na	<u>87.5</u>	140
score synthesis	voc	na	*82.0*	140
score synthesis	voc+aug	na	96.8	140
vector synthesis		na	84.7	0.7
vector synthesis	voc	na	75.6	0.7
vector synthesis	voc+aug	na	94.2	0.7

Table 1: Results of experiments. time: Approximate time for computing a pair-wise distance matrix on 1k AMRs on a TI 1080 GPU.

and $b' =$ *(6 / 1 :3 (7 / 2) :10 (8 / 9))*. The target alignment then equals $map^{\star} = \{(4,6),(5,7),(\emptyset,8)\}$. This strategy greatly reduces the vocabulary size, in our case from 40k tokens to less than 700.

Auxiliary data creation trick We also find that we can cheaply create auxiliary gold data. We re-assign different indices to AMR tokens, and correspondingly modify the ORACLE alignment (Figure 3, 'permute'). In our experiments, we permute the existing token-index vocabularies 10 times, resulting in a ten-fold increase of the training data. We expect that, with this strategy, the model will better learn properties of permutation invariance, which in turn will help it synthesize the algorithm.

3.6 Evaluation

Output post-processing For the score synthesis (Option II) and vector synthesis (Option III), no further post-processing is required, since we directly obtain the estimated SMATCH scores as output. In the explicitly synthesized alignment algorithm, however, we get map, which is the predicted alignment from the sequence-to-sequence model. In this case, we simply feed map as an argument into Eq. 2, to obtain the scores.

Evaluation We compare the predicted scores $\hat{y}$ against the gold scores y with Pearson's ρ. However, for the model that predicts the explicit alignment (Option I), we can compute another interesting and meaningful metric. For this, we first calculate the average SMATCH score over AMR pairs given the gold alignment $map^{\star}$, and then we calculate the average SMATCH score over AMR pairs given the predicted alignment $\widehat{map}$ using Eq. 2. Note, that the SMATCH score based on the gold alignment constitutes an upper bound (max). Therefore, the SMATCH score based on the predicted alignment shows us how close we are to this upper bound. Our baseline consists of scores that are computed from a random alignment (*random*).

Results (Table 1) Our best model is the NMT approach using both data augmentation tricks. Obtaining 98.4 ρ, it very closely approximates the ORACLE, while being about 30 times faster than ORACLE and 76.2 points better then the random baseline. Perhaps the best tradeoff between speed and approximation performance is gained by the simple CNN score synthesis (96.8 ρ, 200x faster than ORACLE), also using both data tricks. The vector synthesis falls a bit shorter in performance (94.2 ρ), but it is extremely fast and achieves a 40,000x speed-up compared to ORACLE and about 1500x compared to the NMT approach.[1]

Consistently, the data extension (*aug*) is very useful. However, the vocabulary reduction (*voc*) is only useful for the NMT model (+27.2 points), whereas the scores are lowered for the CNN-based models (-5.5 for score synthesis, -9.1, *vector synthesis*). We conjecture that the CNNs learn SMATCH more indirectly by exploiting token similarities in the global vocabulary, and therefore struggle more to build a generalizable algorithm, in contrast to the bigger NMT transformer that learns to assess tokens fully from their given graph context.

[1] Note also that all models in Table 1 are significantly better (p<0.001) than the random baseline (one-sided test w/ z-transform).

4 Conclusion

We tested methods for learning to solve the hard structural graph matching problem that is key to many applications where we compare meaning representations. To this aim, we explored different neural architectures, and data augmentation strategies that help models to generalize. Our best models increase metric calculation speed by a large factor while incurring only small losses in accuracy that can be tolerated in many use cases. Our work paves the way to emergent use-cases of meaning representation that involve pair-wise analysis: e.g., semantic clustering or semantic pattern-based search for corpus linguistic studies.

Limitations

An issue of the tested methods concerns the alignment of larger graphs with many variables. On one hand, when the alignment candidate space increases, the runtime of SMATCH increases exponentially, while our considered approaches remain fast. However, in such a scenario, the neural models are bound to trade in some accuracy. Table 4 (Appendix A.3) assesses the effect size for differently sized alignment candidate spaces: while the model overall copes with different search space sizes, the accuracy loss is more considerable for large problems. We conclude that the fast and accurate alignment of *larger* AMR graphs remains a challenging and unsolved problem. However, note that such a bottleneck even exists for the algorithmic metrics, which either use a hill-climber that suffers from worsening sub-optimality or require a costly ILP procedure that may be infeasible for larger graphs (see Opitz (2023) for discussion and analysis). In this regard, we believe that our proposed data extension trick in combination with long-sequence transformers (Beltagy et al., 2020; Rae et al., 2020; Choromanski et al., 2021) may provide valuable means to address this limitation, or provide useful tradeoffs.

Other limitations are: i) the models that were trained without our proposed anonymization protocol were tested on graphs that contain English concepts, and therefore depend on an English vocabulary. ii) For loading the models, our tested methods require more RAM memory than SMATCH, which can be calculated on a low-budget computer.

References

Rafael Torres Anchiêta, Marco Antonio Sobrevilla Cabezudo, and Thiago Alexandre Salgueiro Pardo. 2019. Sema: an extended semantic evaluation for amr. In *(To appear) Proceedings of the 20th Computational Linguistics and Intelligent Text Processing*. Springer International Publishg.

Matej Balog, Alexander L. Gaunt, Marc Brockschmidt, Sebastian Nowozin, and Daniel Tarlow. 2016. Deepcoder: Learning to write programs. *CoRR*, abs/1611.01989.

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.

Irwan Bello, Hieu Pham, Quoc V Le, Mohammad Norouzi, and Samy Bengio. 2016. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*.

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Claire Bonial, Mitchell Abrams, David Traum, and Clare Voss. 2021. Builder, we have done it: Evaluating & extending dialogue-amr nlu pipeline for two collaborative domains. In *Proceedings of the 14th International Conference on Computational Semantics (IWCS)*, pages 173–183, Groningen, The Netherlands (online). Association for Computational Linguistics.

Claire Bonial, Stephanie M. Lukin, David Doughty, Steven Hill, and Clare Voss. 2020. InfoForager: Leveraging semantic search with AMR for COVID-19 research. In *Proceedings of the Second International Workshop on Designing Meaning Representations*, pages 67–77, Barcelona Spain (online). Association for Computational Linguistics.

Marco Budinich. 1996. A self-organizing neural network for the traveling salesman problem that is competitive with simulated annealing. *Neural Computation*, 8(2):416–424.

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)*, pages 748–752, Sofia, Bulgaria. Association for Computational Linguistics.

Xinyun Chen, Dawn Song, and Yuandong Tian. 2021. Latent execution for neural program synthesis beyond domain-specific languages. In *Advances in Neural Information Processing Systems*, volume 34, pages 22196–22208. Curran Associates, Inc.

Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J Colwell, and Adrian Weller. 2021. Rethinking attention with performers. In *International Conference on Learning Representations*.

Luca M Gambardella and Marco Dorigo. 1995. Ant-q: A reinforcement learning approach to the traveling salesman problem. In *Machine learning proceedings 1995*, pages 252–260. Elsevier.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401*.

Thanh Lam Hoang, Gabriele Picco, Yufang Hou, Young-Suk Lee, Lam Nguyen, Dzung Phan, Vanessa Lopez, and Ramon Fernandez Astudillo. 2021. Ensembling graph predictions for amr parsing. In *Advances in Neural Information Processing Systems*, volume 34, pages 8495–8505. Curran Associates, Inc.

Hans Kamp. 1981. A theory of truth and semantic representation. *Formal semantics-the essential readings*, pages 189–222.

Guillaume Lample and François Charton. 2019. Deep learning for symbolic mathematics. *arXiv preprint arXiv:1912.01412*.

Chunchuan Lyu and Ivan Titov. 2018. AMR parsing as graph prediction with latent alignment. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 397–407, Melbourne, Australia. Association for Computational Linguistics.

Emma Manning and Nathan Schneider. 2021. Referenceless parsing-based evaluation of AMR-to-English generation. In *Proceedings of the 2nd Workshop on Evaluation and Comparison of NLP Systems*, pages 114–122, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Viswanath Nagarajan and Maxim Sviridenko. 2009. On the maximum quadratic assignment problem. *Mathematics of Operations Research*, 34(4):859–868.

Tahira Naseem, Abhishek Shah, Hui Wan, Radu Florian, Salim Roukos, and Miguel Ballesteros. 2019. Rewarding Smatch: Transition-based AMR parsing with reinforcement learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4586–4592, Florence, Italy. Association for Computational Linguistics.

Arvind Neelakantan, Quoc V. Le, and Ilya Sutskever. 2016. Neural programmer: Inducing latent programs with gradient descent. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Rik van Noord, Lasha Abzianidze, Hessel Haagsma, and Johan Bos. 2018. Evaluating scoped meaning representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*, Miyazaki, Japan. European Languages Resources Association (ELRA).

Maxwell Nye, Armando Solar-Lezama, Josh Tenenbaum, and Brenden M Lake. 2020. Learning compositional rules via neural program synthesis. In *Advances in Neural Information Processing Systems*, volume 33, pages 10832–10842. Curran Associates, Inc.

Tim O'Gorman, Michael Regan, Kira Griffitt, Ulf Hermjakob, Kevin Knight, and Martha Palmer. 2018. AMR beyond the sentence: the multi-sentence AMR corpus. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3693–3702, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Juri Opitz. 2020. AMR quality rating with a lightweight CNN. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 235–247, Suzhou, China. Association for Computational Linguistics.

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, Angel Daza, and Anette Frank. 2021a. Weisfeiler-Leman in the Bamboo: Novel AMR Graph Metrics and a Benchmark for AMR Graph Similarity. *Transactions of the Association for Computational Linguistics*, 9:1425–1441.

Juri Opitz and Anette Frank. 2019. Automatic accuracy prediction for AMR parsing. In *Proceedings of the Eighth Joint Conference on Lexical and Computational Semantics (*SEM 2019)*, pages 212–223, Minneapolis, Minnesota. 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.

Juri Opitz, Philipp Heinisch, Philipp Wiesenbach, Philipp Cimiano, and Anette Frank. 2021b. Explainable unsupervised argument similarity rating with Abstract Meaning Representation and conclusion generation. In *Proceedings of the 8th Workshop on Argument Mining*, pages 24–35, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Juri Opitz, Letitia Parcalabescu, and Anette Frank. 2020. Amr similarity metrics from principles. *Transactions of the Association for Computational Linguistics*, 8:522–538.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. 2020. Compressive transformers for long-range sequence modelling. In *International Conference on Learning Representations*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Linfeng Song and Daniel Gildea. 2019. SemBleu: A robust metric for AMR parsing evaluation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4547–4552, Florence, Italy. Association for Computational Linguistics.

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. 2021. Long range arena : A benchmark for efficient transformers. In *International Conference on Learning Representations*.

Sarah Uhrig, Yoalli Garcia, Juri Opitz, and Anette Frank. 2021. Translate, then parse! a strong baseline for cross-lingual AMR parsing. In *Proceedings of the 17th International Conference on Parsing Technologies and the IWPT 2021 Shared Task on Parsing into Enhanced Universal Dependencies (IWPT 2021)*, pages 58–64, Online. Association for Computational Linguistics.

Shira Wein, Wai Ching Leung, Yifu Mu, and Nathan Schneider. 2022. Effect of source language on AMR structure. In *Proceedings of The 16th Linguistic Annotation Workshop (LAW)*, Marseille, France. European Language Resources Association (ELRA).

Zizhen Zhang, Hong Liu, MengChu Zhou, and Jiahai Wang. 2021. Solving dynamic traveling salesman problems with deep reinforcement learning. *IEEE Transactions on Neural Networks and Learning Systems*.

parameter	value
embedding size	512
encoder	4 transformer layers w/ 4 heads
decoder	4 transformer layers w/ 4 heads
feed forw. dim	2048
loss	cross-entropy
weight init	xavier
optimizer	adam
learning rate	0.0002
batch size	8192 (tokens)

Table 2: Overview of NMT hyper-parameters.

parameter	value
emb. dimension	100
'pixels'	60x15
CNN encoder	concatenate(
	256 3x3 convs, 3x3 max pool
	128 5x5 convs, 5x5 max pool)
MLP	relu layer followed by lin. regressor
weight init	xavier
optimizer	adam
learning rate	0.001
batch size	64

Table 3: Overview of CNN hyper-parameters.

A Appendix

A.1 Sequence-to-sequence network parameters

Hyper-parameters for the NMT approach are displayed in Table 2. The best model is determined on the development data by calculating BLEU against the reference alignments.

A.2 CNN network parameters

Hyper-parameters for the CNN approach are displayed in Table 2. The best model is determined on the development data by calculating Pearson's ρ correlation of predicted scores and gold scores.

A.3 Analysis of performance on different problem sizes

See Table 4.

		Δ vs. ORACLE		
data type	data size	Eq. 2	Pea's ρ	better
full	1500	-1.1	-1.6	-
$<$ 5 vars	505	-0.6	-1.2	yes
$<$ 10 vars	1041	-0.7	-1.2	yes
$<$ 15 vars	1206	-0.9	-1.2	yes
$<$ 20 vars	1353	-0.9	-1.2	yes
$<$ 25 vars	1449	-0.9	-1.3	yes
$>$ 5 vars	940	-1.5	-2.2	no
$>$ 10 vars	476	-2.1	-3.6	no
$>$ 15 vars	318	-2.3	-5.4	no
$>$ 20 vars	183	-3.0	-10.1	no
$>$ 25 vars	83	-4.7	-19.3	no
$>$ 30 vars	37	-8.0	-25.5	no
$>$ 35 vars	20	-12.5	-41.1	no
single snt AMRs	1421	-1.0	-1.5	yes
multi snt AMRs	79	-2.7	-9.6	no

Table 4: Experiments on different test subsets that represent different problem complexities predicted with our best model (*align. synthesis+voc+aug*). $<>$ x vars means that one of two graphs contains $<>$ x variables. *better*: is the drop in accuracy of the model vs. ORACLE smaller compared with the model tested on all data?