Evaluate AMR Graph Similarity via Self-supervised Learning

Ziyi Shou and **Fangzhen Lin**
Department of Computer Science and Engineering
HKUST-Xiaoi Joint Laboratory
The Hong Kong University of Science and Technology
{zshou, flin}@cse.ust.hk

Abstract

In work on AMR (Abstract Meaning Representation), similarity metrics are crucial as they are used to evaluate AMR systems such as AMR parsers. Current AMR metrics are all based on nodes or triples matching without considering the entire structures of AMR graphs. To address this problem, and inspired by learned similarity evaluation on plain text, we propose AMRSim, an automatic AMR graph similarity evaluation metric. To overcome the high cost of collecting human-annotated data, AMRSim automatically generates silver AMR graphs and utilizes self-supervised learning methods. We evaluated AMRSim on various datasets and found that AMRSim significantly improves the correlations with human semantic scores and remains robust under diverse challenges. We also discuss how AMRSim can be extended to multilingual cases.[1]

1 Introduction

An Abstract Meaning Representation (AMR; Banarescu et al., 2013) is a rooted, directed graph where nodes represent concepts and edges correspond to relations of concepts. The goal of an AMR metric is to evaluate the similarities of pairs of AMR graphs so that it can be used to evaluate the outputs of AMR generators such as parsers. Therefore, a good AMR metric is crucial for the design and evaluation of AMR parsers. However, the research on AMR metrics has so far lagged far behind the work on AMR parsing.

Current AMR metrics either transfer AMR graphs to triples and consider the one-to-one matching of variables (Cai and Knight, 2013) or linearize AMR graphs as sequences and calculate n-gram matching (Song and Gildea, 2019). These metrics fail to consider the entire AMR structure and lack flexibility, resulting in poor correlation with human annotations.

Inspired by plain-text automatic similarity assessment methods that encode sentences into latent semantic representations to measure a similarity of the two representations (Reimers and Gurevych, 2019), we propose to learn the automatic assessment of AMR graph similarity through a similar pipeline. Our proposed metric, called AMRSim, adopts the pre-trained language model BERT as the backbone and incorporates GNN adapters to capture the structural information of AMR graphs. To overcome the high cost of collecting training data, we utilize self-supervised learning methods. The training objective is to maximize the dot product between positive embeddings and to minimize the dot product between different encodings. In contrast to one-to-one matching metrics, our AMRSim is alignment-free by computing the cosine of contextualized token embeddings. The prediction process can be considerably accelerated by leveraging GPUs.

We experiment with AMRSim on the transformed STSB (Baudiš et al., 2016) and SICK (Marelli et al., 2014) datasets, which contain pairs of AMR graphs and corresponding similarity scores. Our experiments demonstrate that AMRSim achieves prominent improvements in correlation with human annotations. In further analysis, AMRSim retains the highest performance under various challenges compared to previous metrics. We also explore the potential of extending our metric to multilingual cases, taking into account the generic nature of the transformer structure and the fact that the pipeline of AMRSim is not constrained to any particular language.

The remaining paper is structured as follows. Section 2 gives an overview of existing AMR metrics; Section 3 introduces our proposed metric, AMRSim; Section 4 includes experimental settings and main results; We analyze the robustness and the efficiency of our metrics in Section 5 and conclude in Section 6.

[1]The code and datasets can be found at https://github.com/zzshou/AMRSim.

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 16112–16123

2 Existing Metrics

AMR similarity metrics play a vital role in evaluating the performance of AMR parsers. However, computing the degree of similarity between AMR graphs is not trivial. In this section, we summarise the existing AMR metrics.

SMATCH SMATCH (Cai and Knight, 2013) evaluates the overlap of structures as the similarity score. Each AMR graph is viewed as a conjunction of triples. SMATCH tries to find a one-to-one matching of variables that maximizes the numbers of exact matches of triples through the hill climbing method in a greedy style. The alignment process, which has been proved as NP-hard, limits the efficiency of SMATCH, especially as the sizes of AMR graphs increase. Another weakness is that the structural matching is insufficient for meaning similarity assessment and fragile in concept synonym replacement and structure deformation (Blloshmi et al., 2020; Opitz et al., 2021).

S^{2}MATCH To yield a better variable alignment, Opitz et al. (2020) propose S^{2}MATCH by allowing soft semantic match instead of matching only identical triples in SMATCH. Accounting for cosine similarity of concepts helps to assess the similarity of graphs, however, computational limitations and structure deformation confusion in SMATCH have not been addressed.

SEMA SMATCH adds a TOP relation to the structure, but SEMA (Anchiêta et al., 2019) argues that this addition can potentially distort the evaluation. Therefore, they ignored triples identifying the graph top. Furthermore, instead of computing the maximum score like SMATCH, SEMA works as a breadth-first search and produces a deterministic result, making it faster than one-to-one matching of variable employed by SMATCH.

SEMBLEU BLEU (Papineni et al., 2002), which assesses text quality by comparing n-grams, is frequently adopted in machine translation evaluation. To extend BLEU for matching AMR graphs, SEMBLEU (Song and Gildea, 2019) linearizes AMR graphs through breadth-first traversal and extracts n-gram for comparison. The metric is alignment-free and thus computationally efficient. Experimental results show that SEMBLEU achieved slightly higher consistency with human judgment than SMATCH.

WWLK Besides treating AMR graphs as triples and linearized grams, Weisfeiler-Leman AMR similarity metrics (Opitz et al., 2021) consider AMR graphs as high-dimensional objects. They first propagate node embeddings iteratively by incorporating contextualization and then employ Wasserstein Weisfeiler Leman kernel (WWLK) to calculate the minimum cost of transforming one graph to another. WWLK only considers node embeddings (GloVe embeddings) in AMR graphs, while edge labels have no corresponding embeddings. Therefore, WWLK$_\theta$ is extended to learn AMR edge labels, which requires additional training data.

Considering embeddings of AMR edges improves the performance of metrics. However, supervised learning methods require additional efforts to collect human-labeled data. In contrast, our proposed learned AMR similarity metric adopts self-supervised learning and achieves higher correlation performance.

3 The Proposed Approach

In this section, we introduce our proposed approach for learning AMR graphs similarity evaluation.

3.1 Problem Formulations

AMR graph similarity metrics compute the graded similarities of AMR graphs. Given a reference AMR graph $G = \{V, E, R\}$ where V, E and R denote the node set, edge set, and relation set of AMR graph G respectively, and a candidate AMR graph $\hat{G} = \{\hat{V}, \hat{E}, \hat{R}\}$, the primary motivation is to automatically learn the similarity $sim(G, \hat{G})$ between the two AMR representations. To do that, we propose to use network structures to derive the contextual embeddings of the two AMR graphs and then compute their similarity score as the cosine similarity of the embeddings. More precisely, we use $sim(G, \hat{G}) = sim(e, \hat{e}) = \frac{e^\top \hat{e}}{\|e\|\|\hat{e}\|}$, where e and $\hat{e}$ are the contextual embeddings of G and $\hat{G}$, respectively. Hence the key of our approach is learning contextual embeddings.

3.2 AMRSim

The use of self-supervised methods is a well-established approach in semantic text similarity tasks (STS, Carlsson et al., 2021; Gao et al., 2021). To assess AMR graph similarity efficiently and well-correlated with human judgments, we propose migrating the self-supervised training process in STS to AMR similarity evaluation.

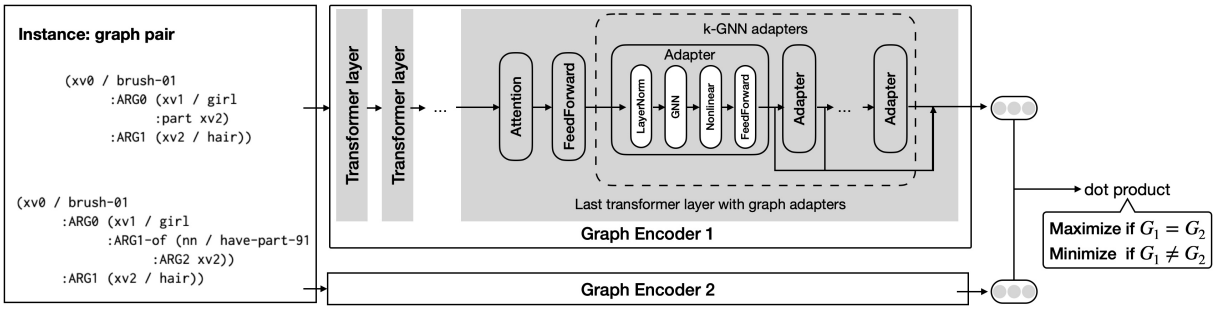

Figure 1: Illustration of the pipeline of AMRSim. Graph Encoder 1 and 2 have the same network structure. The training objective is to maximize dot products between contextual embeddings from positive instances and minimize dot products between contextual embeddings from negative instances.

3.2.1 Self-supervised Training

In plain text applications, e.g., STS tasks and text generation tasks, many learned metrics are trained to optimize correlation with human annotations (Lowe et al., 2017; Reimers and Gurevych, 2019). However, AMR graph similarity data collection is more time-consuming because AMR evaluation has a learning cost of understanding the semantics of graphs, which are not as straightforward as plain text. Thus, self-supervised learning methods are an alternative solution. We adopt an efficient self-supervised approach Contrastive Tension (CT; Carlsson et al., 2021) in AMRSim metrics. The basic assumption is that AMR graphs with adjacent distributions have similar meanings.

In CT, two independent encoders are initialized identically. The training objective is to maximize the dot product between positive embeddings and to minimize the dot product between different encodings. CT constructs positive and negative pairs in each batch. For each randomly selected AMR graph G, copy G into an identical graph pair to construct the positive instance, and sample other K graphs to construct negative instances by pairing G. The $K + 1$ instances are included in the same batch. The training contrastive loss $\mathcal{L}$ is binary cross-entropy between the generated similarity scores and labels.

$$\mathcal{L}(G, \hat{G}) = \begin{cases} -\log \sigma(e \cdot \hat{e}), & \text{if } G = \hat{G} \\ -\log \sigma(1 - e \cdot \hat{e}), & \text{if } G \neq \hat{G} \end{cases}$$

where σ refers to the Logistic function. Figure 1 demonstrates the pipeline of AMRSim. An instance containing two AMR graphs is input to graph encoders to generate contextual embeddings, then the dot product of the embeddings is used to compute the loss.

3.2.2 Network Structures

Compared to plain text, AMR graphs contain more structural information. We propose to incorporate graph neural networks into transformers to adapt to AMR graph structures and derive more descriptive contextual embeddings.

Transformers Transformer based neural networks have demonstrated exemplary performance in the fields of natural language processing but they only accept inputs as sequence data. Therefore, we first convert AMR graphs to sequences. In AMR graph G, the labeled edge $(u, r, v) \in E$, where $u, v \in V$ and $r \in R$ is a relation type, means that there is an edge labeled as r from node u to node v. Similar to Ribeiro et al. (2022), we convert each AMR graph G into an unlabeled graph by replacing each labeled edge (u, r, v) with two unlabeled edges $\{(u, r), (r, v)\}$. Unlabeled $G' = \{V', E'\}$ where V' include original nodes as well as additional nodes converted from relations. This preprocessing method facilitates BERT to learn word embeddings for edges.

The unlabeled AMR graph is then linearized. Position embeddings are crucial for modeling sequence order in transformers. The widely used position embedding takes the advantage of absolute positions of the input sequence. However, we argue that the linearization order should not affect AMR graph encoding, because of the inherent nature of AMR graphs that the relationships between nodes are mainly defined by the underlying semantic connections rather than their linear positioning. Therefore, instead of absolute position encoding, the shortest path lengths between all nodes and the root node are encoded as relative position embeddings.

In pre-trained language models like BERT, tokens may be split into smaller subwords due to

the fixed vocabulary. We refuse to split AMR relations and instead add them to the vocabulary as new words. However, concept nodes are tokenized as usual. The main reason is that relation words are artificial, while nodes are always concepts, closer to linguistic tokens in the vocabulary. For unlabeled edge (u, r), if node u is split into subword $u_1, \ldots, u_k$ in tokenization, we consider each subword as a new node and connect each subword to r, thus (u, r) is replaced by $\{(u_1, r), \ldots, (u_k, r)\}$.

Graph Neural Networks Adapter To generalize transformers to capture the structural information of AMR graphs, we incorporate graph neural networks as adapter into transformer layers. In particular, we refer to the idea of the k-dimensional Graph Neural Networks (k-GNN; Morris et al., 2019), which are based on the k-dimensional Weisfeiler and Leman algorithm (k-WL). For a given k, a k-set in $[G]^k$, $s = \{V_s, E_s\}$ is a k-element subgraph over G, then the neighborhood of s is $N(s) = \{t \in [G]^k || s \cap t| = k - 1\}$. The GNN at layer t computes new features: $h_k^t(s) = \sigma(h_k^{t-1}(s) \cdot W_1^t + \sum_{u \in N(s)} h_k^{t-1}(u) \cdot W_2^t)$, where σ is the activation function, W_1^t and W_2^t are layer parameters. Different from the graph adapter in Ribeiro et al. (2022), we employ an adapter module after the feed-forward sub-layer of the last layer of BERT. The experimental results comparison can be found in section 5.6. The normalization layer before the GNN layer and the projection after the GNN layer is kept, see figure 1. So given the hidden states h_v for node v, the GNN adapter layer computes: $z_v = W \cdot \text{GNN\_layer}(\mathsf{LN}(h_v)) + h_v$, where W is the adapter parameters, $\mathsf{LN}(\cdot)$ denotes layer normalization and GNN_layer represents the calculation process of GNN layers.

4 Experiments

4.1 Data Construction

A major advantage of the self-supervised method is that no human-annotated data is required for training. Following data preparation in AMR-DA (Shou et al., 2022), AMRSim utilized SPRING (Bevilacqua et al., 2021) to parse one-million sentences randomly sampled from English Wikipedia[2] to AMR graphs. Generated silver AMR graphs were linearized by a depth-first traversal algorithm (the choice of linearization method does not have impact on embeddings, refer to section 5.1), meanwhile, all edges were recorded as pairs (u, r). Computing relative positions of tokens was implemented by Dijkstra's Method [3].

4.2 Experimental Setups

We implemented AMRSim with sentence transformers (Reimers and Gurevych, 2019). During training, we set the positive ratio to be 4/16. In a batch of 16, there were 4 positive graph pairs and 12 negative pairs. This indicates that we sampled 4 graphs and created one positive pair and three negative pairs for each graph. The transformer parameters were initialized from uncased BERT base model (Devlin et al., 2019), and parameters for graph adapters were initialized randomly. We carried out a search of sequence length $\in \{64, 128, 256\}$ and determined the length of linearized AMR graphs as 128. Other hyperparameters were set as follows: learning rate as 1e-5, dropout rate as 0.1, and graph adapter size as 128. GNN embeddings from k layers were concatenated and input to a projection function to generate final embeddings. Our experiments were done using GeForce RTX 2080 Ti GPU. We trained our models for one epoch, which took approximately two and a half hours and reported in the table the average performance of our models over five repeated experiments with different seeds.

4.3 Main Results

We compared AMRSim with other AMR similarity metrics on evaluation datasets modified from semantic textual similarity datasets, STSB (Baudiš et al., 2016) and SICK (Marelli et al., 2014). Original datasets contain a set of pairs of sentences with a human-labeled similarity score. Opitz et al. (2021) utilized a strong parser to construct AMR graph pairs from sentence pairs and normalized similarity scores to the range [0, 1] to facilitate standardized evaluation. There are 1379 and 4927 test instances in the two datasets, respectively. Over 95% randomly selected data from generated AMR graphs were assessed as gold or with minor errors.

The best performance of baseline metrics on the test dataset was included in the comparison. For example, $k = 2$ achieved superior performance in STS and SICK than the default value $k = 3$

[2]The one-million sentences sampled from Wikipedia is taken from datasets for SimCSE (Gao et al., 2021), which can be downloaded from https://huggingface.co/datasets/princeton-nlp/datasets-for-simcse/tree/main.

[3]The implementation code comes from networkx: https://networkx.org.

in SEMBLEU, so we included SEMBLEU$_{k=2}$ as SEMBLEU.

	Metrics	STSB	SICK
Baselines	SMATCH	58.45	59.72
	S^{2}MATCH	58.82	60.42
	SEMA	55.90	55.32
	SEMBLEU	60.62	59.86
	WWLK	63.15	65.58
	WWLK$_\theta$	66.94	67.64
Ours	AMRSim$_1$	69.61±0.31	72.17±0.49
	AMRSim$_2$	70.10±0.31	71.95±0.79
	AMRSim$_3$	70.59±0.64	72.82±0.46
	AMRSim$_4$	70.88±0.61	**73.10±0.42**
	AMRSim$_5$	**70.94±0.74**	72.64±0.44

Table 1: Comparison of different AMR metrics. Results are Pearson correlation (x100) on STSB and SICK test set. We repeated our experiments five times and reported the average score with the standard variance. AMRSim$_k$ indicates that the graph encoder adopts k-GNN as the adapter.

Table 1 shows the comparison of the evaluation results of various metrics on two test datasets. Baseline results are from Opitz et al.'s (2021) work. We conducted five experiments for each different k-GNN setups and reported the mean and standard deviation of Pearson correlation scores. Our proposed AMRSim significantly outperformed previous AMR metrics and achieved the highest score on correlation with human annotation. Specifically, AMRSim$_5$ with 5-GNN adapter improved the previous best Pearson score from 66.94% to 70.94% on STSB, and AMRSim$_4$ with 4-GNN adapter improved the score from 67.64% to 73.10% on SICK dataset. When k increased from one to four, the average performance of AMRSim in both datasets increased, however, when increasing from four to five, the average performance of the model decreased. More parameters to optimize made the model harder to train without more improvement. Considering the average Pearson score and standard deviation, we took AMRSim$_4$ as our final metric. It is worth mentioning that WWLK$_\theta$ learned edge encodings through supervised learning methods. By contrast, our proposed AMRSim adopted self-supervised learning, and no human-labeled data was required.

5 Analysis

AMRSim shows a high correlation with human annotations. In this section, we further analyze the robustness and efficiency of AMRSim.

5.1 Robustness Analysis

At first, we explore in depth how AMRSim performs under various challenges.

Reification Challenges Reification challenges import AMR rephrasing, which means changing the structure of graphs, but not its meaning (Banarescu et al., 2013; Goodman, 2019; Opitz et al., 2021). The reification rule is that edge (u, r, v) in the original graph induced $(u, r_1, ins_r) \wedge (ins_r, r_2, v)$, where ins_r is r's reification and r_ns are new generated edges. For example, in the following AMR graph:

```
(xv0 / brush-01
      :ARG0 (xv1 / girl
            :part xv2)
      :ARG1 (xv2 / hair))
```

Edge label :part has its reification :have-part-91, so for the above AMR graph, $(girl, \text{:part}, hair)$ can be replaced by $(girl, \text{:ARG-of}, \textit{have-part-91}) \wedge (\textit{have-part-91}, \text{:ARG2}, hair)$. The modified AMR graph keeps the meaning displayed as:

```
(xv0 / brush-01
      :ARG0 (xv1 / girl
            :ARG1-of (nn / have-part-91
                  :ARG2 xv2))
      :ARG1 (xv2 / hair))
```

Metrics	STSB	Δ	SICK	Δ
SMATCH	57.98	-0.47	61.81	2.09
S2MATCH	58.08	-0.74	62.25	1.83
SEMA	55.51	-0.39	56.16	0.84
SEMBLEU	54.84	-5.78	57.70	-2.16
WWLK	59.78	-3.37	65.53	-0.05
WWLK$^\theta$	64.34	-2.60	65.49	-2.15
AMRSim	**70.54**	-0.34	**73.42**	0.32

Table 2: Comparison of AMR metrics for reification challenges. Results are Pearson correlation (x100) on reified STSB and SICK test dataset. Δ means the difference in score before and after reification.

Reification challenges require that metrics have the capacity to handle structure variants. Table 2 shows the comparison of AMR metrics under reification challenges. AMRSim ranked first on both reified datasets. On reified STSB test set, AMRSim achieved the highest score of 70.54% and the lowest loss of 0.43% compared with performance

on STSB dataset. Similar results were presented on SICK dataset. AMRSim achieved the highest score of 73.42% and the second lowest loss of 0.31%. Encoding the entire AMR graph and comparing contextual embeddings is a contributing factor that improves robustness under reification challenges.

Synonym Challenges Another challenge is synonym challenge, which is conceptual synonym substitution while preserving meaning. AMR nodes are iterated over and replaced by their (near-) synonyms in PropBank, or their synset in WordNet (Opitz et al., 2021). The synonym transformation is not trivial for the reason that the single node may be replaced by a graph substructure. For example, $instance(x, tofu)$ can be extended as $(x, :\text{mod}, y) \wedge instance(x, curd) \wedge instance(y, bean)$, so the AMR graph

```
(xv0 / cut-01
 :ARG0 (xv2 / woman)
 :ARG1 (xv1 / tofu))
```

is transformed as:

```
(xv0 / cut-01
      :ARG0 (xv2 / womanhood)
      :ARG1 (xv1 / curd
            :mod (nn52 / bean)))
```

Metrics	STSB	Δ	SICK	Δ
SMATCH	56.14	-2.31	57.39	-2.33
S2MATCH	56.70	-2.12	57.92	-2.50
SEMA	50.16	-5.74	48.87	-6.45
SEMBLEU	52.82	-7.80	53.47	-6.39
WWLK	59.40	-3.75	59.98	-5.60
WWLK$^\theta$	60.11	-4.23	62.29	-5.35
AMRSim	**66.50**	-4.38	**67.73**	-5.37

Table 3: Comparison of AMR metrics for synonym challenges. Results are Pearson correlation (x100) on transformed STSB and SICK test dataset. Δ means the score difference before and after concept node transformation.

Table 3 shows the performance of various AMR metrics on synonym challenges. From the results, synonym challenges are more complex than reification challenges. Nevertheless, AMRSim still ranked first on both test datasets, with 66.50% and 67.73% respectively. SEMBLEU that matches n-gram of linearized AMR graphs is the most vulnerable under this challenge. Besides, WWLK$_\theta$ and AMRSim suffer more loss than reification challenges. One possible reason is that both WWLK$_\theta$ and AMRSim are learned metrics, and they have a common generalization problem to some extent. However, our proposed AMRSim outperformed WWLK$_\theta$ by a large margin, but with a close score loss.

Linearization Challenges The basic assumption of AMRSim is that the similarity score is concerned with the entire AMR graph and will not be affected regardless of the linearization of the AMR graph. As the transformer model treats each token as independent, position embedding is added to retain the order information of tokens. To test AMRSim's ability to handle the linearization challenge, we compared model outputs for different inputs: (i) A pair of AMR graphs that are isomorphic but have different linearization sequences; (ii) a pair of different AMR graphs with the same linearized sequence after adopting the original position embedding strategy in BERT; (iii) a pair of different AMR graphs with the same linearized sequence after adopting relative position embedding strategy in AMRSim. For example, four AMR graphs are listed below:

```
AMR A:
(xv0 / play-01
  :ARG0 (xv2 / group
        :consist-of (xv4 / boy))
  :ARG1 (xv1 / soccer)
  :location (xv3 / beach))
```

```
AMR B:
(xv0 / play-01
  :location (xv3 / beach)
  :ARG0 (xv2 / group
        :consist-of (xv4 / boy))
  :ARG1 (xv1 / soccer))
```

```
AMR C:
(xv0 / play-01
  :ARG0 (xv2 / group
        :consist-of (xv4 / boy
            :ARG1 (xv1 / soccer
              :location (xv3 / beach)))))
```

```
AMR D:
(xv0 / play-01
  :ARG0 (xv2 / group)
  :ARG1 (xv1 / soccer)
  :location (xv3 / beach
        :consist-of (xv4 / boy)))
```

AMR graphs A and B are isomorphic but the position of :location *beach* differs in the linearized sequences. Graphs A and C are different, however, token positions ranked by the ascending algorithm in BERT are the same. Graphs A and D are also inconsistent, but the relative positions of all tokens are consistent. For example, the distance between :consist-of and the root *play*-01 is 4 on both graphs. To demonstrate the effectiveness of GNN adapters in AMRSim, we compare AMRSim with the original BERT and BERT with relative position embeddings (AMRSim without GNN). Table 4 concluded similarity scores from BERT, BERT with relative position embedding, and AMRSim. BERT was confused by various linearizations of the same graph whereas BERT$_{relative}$ has a shortage of capturing structure information. With GNN adapters, AMRSim can handle the linearization challenge. In terms of the other metrics, only SemBleu requires linearization, and it exhibits a high level of sensitivity towards the chosen linearization.

Metrics	$sim(A,B)$	$sim(A,C)$	$sim(A,D)$
Expected	=1	$\neq$1	$\neq$1
BERT	0.98	1	0.99
BERT$_{relative}$	1	0.97	1
AMRSim	1	0.97	0.98

Table 4: Similarity scores for graph pairs. A, B, C, D are four AMR graphs listing earlier where $A = B \neq C \neq D$. Expected denotes the expected similarity scores for graph pairs. BERT means BERT with original position encoding, and BERT$_{relative}$ means BERT with relative position encodings. AMRSim adds GNN adapters to BERT$_{relative}$.

5.2 Scatter Plot Analysis

To visually analyze the distribution patterns and the relationship between human annotations and scores obtained from different metrics, we utilize scatter plots to show instances distributions and best fit lines to depict the trend. All scores are normalized to the range [0,1]. In figure 2, the y-axis of each subplots represents human annotated scores, while the x-axis represents the scores calculated using various metrics. By plotting individual examples on scatter plots, we observed that WWLK-related metrics and our AMRSim outperform other metrics significantly, with higher concentration around the fit and fewer outlier. However, they exhibit biases towards instances with distinct human scores. WWLK-related metrics tend to underrate examples with higher human scores while AMRSim tends to overrate examples with lower human scores.

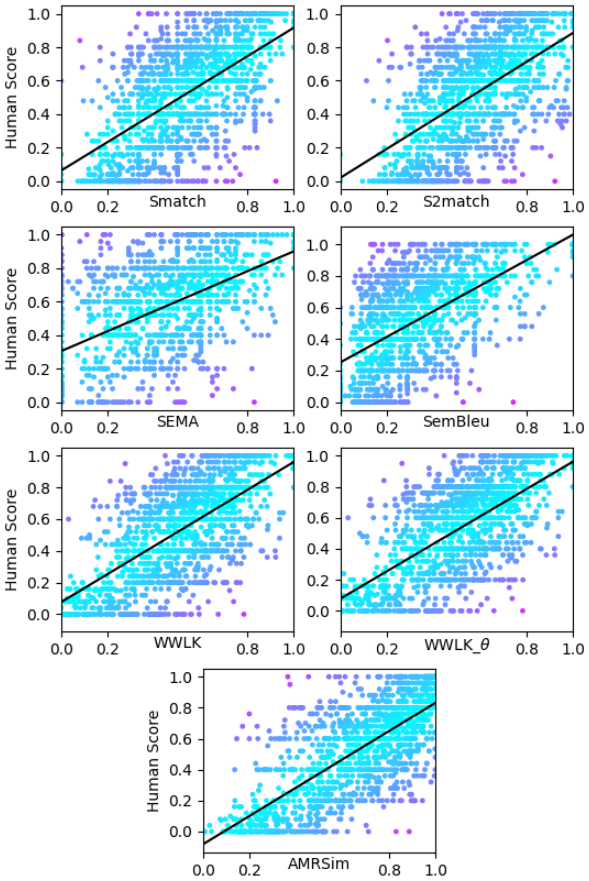

Figure 2: Scatter plots showing the relationship between human annotated scores (y-axis) and scores obtained using various metrics (x-axis). The lines represent the best fit lines, and the color scale indicates the distance away from the best fit lines.

5.3 Case Study

Disagreements in preference for AMR graphs can affect the ranking of AMR parsers. Thus we employ a case study to study metrics' preference for parsed results. Here is an example from Opitz et al.'s (2020) paper. CAMR and JAMR are two AMR parsers.

```
::snt: Legally, there are two remedies.
Gold AMR:
(t / thing :quant 2
    :ARG2-of (r / remedy-01)
    :mod (l / law)
```

```
CAMR Parsed Result:
(x6 / remedy-01
    :quant 2)
```

```
JAMR Parsed Result:
(l / legally
```

```
:manner-of (r / remedy-01
    :quant 2))
```

The scores obtained from different metrics are as follows: SMATCH (0.2, 0.167); S2MATCH (0.2, 0.252); SEMA (0, 0); SEMBLEU (0.215, 0.215); WWLK (0.335, 0.352); WWLK$^{\theta}$ (0.329, 0.345); AMRSim (0.81, 0.92). Among these metrics, SMATCH gives higher scores to CAMR parsed results. Conversely, SEMBLEU and SEMA assign the identical score to both parsed results. SEMA employs a more strict evaluation method as it assesses not only the individual node but also its dependencies. However, S2MATCH, WWLK, WWLK$^{\theta}$ and AMRSim grant higher scores to the second parsed results. In this particular example, $legally$ is indispensable information of semantics. Our preference leans towards the second parsed results.

5.4 Computational Time

In this section, we emphasize the importance of computational time as a crucial factor to consider for practical use. While the time complexity of AMRSim is O(n), it is necessary to consider that the processing with transformers possibly impact the overall time required. We provide an analysis of various metrics' reference time based on SICK test dataset consisting of 4927 samples. The reference time is listed as follows: SMATCH: 5.24s, S2MATCH: 37.51s, SEMA: 1.132s, SEMBLEU: 0.67s, WWLK: 46.60s, WWLK$^{\theta}$: 48.13s. Our AMRSim differs slightly from other metrics as we employ neural networks to predict graph similarity, allowing us to utilize both CPU and GPU for the task. When using CPU, the reference time is approximately 67.66s. However, this is not the most optimal choice, as matrix computations can be accelerated by GPUs. By utilizing a single GPU (RTX2080ti in our experiments), for instance, with a batch size of 16, the reference time reduces to around 14.20s. Scaling up to a batch size of 256 further reduces the reference time to 8.04 seconds. Further increasing the batch size has limited benefits as the processing time for input and output becomes the dominant factor. In term of O(n) time complexity, the advantages of our AMRSim become more pronounced as the graph sizes increase.

5.5 Multilingual Metrics

The lack of multilingual AMR evaluation metrics becomes an obstacle to developing cross-lingual AMR parsers. It is common practice to translate other languages into English or use multilingual embeddings for matching, which does not take into account that the source language may affect the structure of the AMR (Wein and Schneider, 2022). AMRSim has the potential to be extended to multilingual cases given the generic nature of our pipeline: encoding AMR graphs and computing the cosine similarity of graph embeddings, not limited to specific languages. However, annotating the similarity of a multilingual dataset is more challenging and subject due to varying levels of language proficiency. So we performed a preliminary test by collecting a small-scale test dataset comprising 20 Chinese-English AMR pairs, then adopting a well trained multilingual model[4] to encode the corresponding text and calculating cosine similarity of text embeddings as the graph similarity score. To train our multilingual AMRSim, we employed the multilingual version of BERT base model and added a Chinese AMR dataset containing about 2,500 Chinese AMR graphs from Li et al. (2016) to English training data in Section 4.1. Results of metrics are concluded in table 5. SEMA lacks support for multilingual cases, resulting in it returning 0 for any multilingual inputs. Additionally, due to the absence of a multilingual dataset for WWLK$^{\theta}$ supervised training, we exclude them from our comparison. A specific example is illustrated in Figure 3. The predicted similarity score for these two graphs needs to be close to 1 because they correspond to the same sentence in the Chinese and English versions of The Little Prince. We used different metrics to evaluate the similarity of these two graphs and concluded the score in table 6. While AMRSim surpasses other metrics, its performance is poor compared to the English version of AMRSim. The possible explanation is the lim-

	SMATCH	S2MATCH	SEMBLEU	WWLK	AMRSim
Pearson	42.02	52.72	40.08	30.25	63.87

Table 5: Pearson correlation (x100) on 20 Chinese-English AMR Pairs.

	SMATCH	S2MATCH	SEMBLEU	WWLK	AMRSim
Score	0.174	0.252	0.101	-0.055	0.576

Table 6: Similarity scores for AMR graphs in Figure 3.

[4]The multilingual encoding model is downloaded from https://huggingface.co/sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2.

```
# ::snt 这 倒 并没有 使 我 感到 太 奇怪 。
(x10 / 感到-01
    :arg0  (x12 / 我)
    :arg1  (x13 / 奇怪-02)
    :polarity  (x17 / -)
    :cause  (x18 / 这))

# ::snt But that did not really surprise me much .
(c / contrast-01
    :ARG2 (s / surprise-01 :polarity -
        :ARG0 (t / that)
        :ARG1 (i / i)
        :degree (m / much)
        :ARG1-of (r / real-04)))
```

Figure 3: AMR graphs for the same sentence from The Little Princes in Chinese and English.

ited training data hinders effective self-supervised learning. A comprehensive study on multilingual applications is considered to be a future work.

5.6 Ablation Studies

We investigated the impact of GNN adapters' position. Comparison of models' performance is included in table 7. GNN represents that the model only contains GNN layers; GNN-Each means that one GNN adapter is included in each transformer layer, same as Ribeiro et al. (2022); GNN-Before denotes that GNN adapter is before transformer layers and AMRSim indicates that GNN adapters are incorporated after transformer layers. Models other than GNN-Each come with 4-GNN. GNN

	GNN	GNN-Each	GNN-Before	AMRSim
STSB	33.26	62.46	69.63	70.88

Table 7: Comparison of different neural networks. Results are Pearson correlation (x100) on STSB test set.

achieved the worst correlation score because it is notoriously hard for GNN to train (Li et al., 2019). However, setting transformers as the backbone and incorporating GNN as adapters are more acceptable in our experimental settings. GNN-Each underperformed in the experiments, possibly because too many GNN parameters appear in each layer to be optimized for self-supervised training. AMRSim achieved superior performance, which is intuitive because BERT encoded sequence information, and then GNN captured structural information.

6 Conclusion

We have proposed a learning based AMR graph similarity metric called AMRSim. Its underlying network structure incorporates GNNs into a transformer network and employs self-supervised learning. Our proposed similarity metric addresses two common pitfalls of existing metrics: firstly, a low correlation with human annotations due to lack of considering the entire structures of AMR graphs. AMRSim effectively encodes AMR graphs, exhibiting the highest correlations with human evaluations and maintaining robust against various challenges. The second is computational inefficiency associated with matching or alignment of AMR graphs. AMRSim is alignment-free, allowing for a significant reduction in inference time when utilizing GPUs. We also show the potential of extending AMRSim to multilingual AMR similarity metrics.

Limitations

AMRSim has high prediction efficiency but the training process is time-consuming. In our experiments, one epoch training on one GeForce RTX2080Ti took about two and a half hours. Self-supervised learning requires a large amount of training data. Parsing wiki sentences into graphs requires time, but the advantage is that it can be processed offline. Another issue is the length limit. Transformers can only handle limited sequence lengths due to the computational and memory complexity of attention calculation. Therefore, encoding large AMR graphs is challenging. Possible solutions include applying sliding window algorithm to split a large AMR graph into several subgraphs and merge the scores.

References

Rafael T Anchiêta, Marco AS Cabezudo, and Thiago AS Pardo. 2019. Sema: an extended semantic evaluation metric for amr. *arXiv preprint arXiv:1905.12069*.

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.

Petr Baudiš, Jan Pichl, Tomáš Vyskočil, and Jan Šedivỳ. 2016. Sentence pair scoring: Towards unified framework for text comprehension. *arXiv preprint arXiv:1603.06127*.

Michele Bevilacqua, Rexhina Blloshmi, and Roberto Navigli. 2021. One SPRING to rule them both: Symmetric AMR semantic parsing and generation without a complex pipeline. In *Proceedings of AAAI*.

Rexhina Blloshmi, Rocco Tripodi, and Roberto Navigli. 2020. Xl-amr: Enabling cross-lingual amr parsing with transfer learning techniques. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2487–2500.

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.

Fredrik Carlsson, Amaru Cuba Gyllensten, Evangelia Gogoulou, Erik Ylipää Hellqvist, and Magnus Sahlgren. 2021. Semantic re-tuning with contrastive tension. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910.

Michael Wayne Goodman. 2019. Amr normalization for fairer evaluation. *arXiv preprint arXiv:1909.01568*.

Bin Li, Yuan Wen, Weiguang Qu, Lijun Bu, and Nianwen Xue. 2016. Annotating the little prince with chinese amrs. In *Proceedings of the 10th Linguistic Annotation Workshop held in Conjunction with ACL 2016 (LAW-X 2016)*, pages 7–15.

Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. 2019. Deepgcns: Can gcns go as deep as cnns? In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9267–9276.

Ryan Lowe, Michael Noseworthy, Iulian Vlad Serban, Nicolas Angelard-Gontier, Yoshua Bengio, and Joelle Pineau. 2017. Towards an automatic turing test: Learning to evaluate dialogue responses. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1116–1126.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A sick cure for the evaluation of compositional distributional semantic models. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 216–223.

Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. 2019. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, pages 4602–4609.

Juri Opitz, Angel Daza, and Anette Frank. 2021. 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, Letitia Parcalabescu, and Anette Frank. 2020. Amr similarity metrics from principles. *Transactions of the Association for Computational Linguistics*, 8:522–538.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

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.

Leonardo F. R. Ribeiro, Mengwen Liu, Iryna Gurevych, Markus Dreyer, and Mohit Bansal. 2022. FactGraph: Evaluating factuality in summarization with semantic graph representations. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3238–3253, Seattle, United States. Association for Computational Linguistics.

Ziyi Shou, Yuxin Jiang, and Fangzhen Lin. 2022. AMR-DA: Data augmentation by Abstract Meaning Representation. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3082–3098, Dublin, Ireland. 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.

Shira Wein and Nathan Schneider. 2022. Accounting for language effect in the evaluation of cross-lingual amr parsers. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 3824–3834.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Limitations

☑ A2. Did you discuss any potential risks of your work?
Limitations

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Abstract and introduction

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

Experiments

☑ B1. Did you cite the creators of artifacts you used?
Experiments

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Experiments

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Experiments

☑ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Experiments

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Experiments

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Experiments

C ☑ Did you run computational experiments?

Experiments

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Experiments

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Experiments

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Experiments

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Experiments

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
Not applicable. Left blank.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
Not applicable. Left blank.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
Not applicable. Left blank.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
Not applicable. Left blank.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
Not applicable. Left blank.