

SMATCH++: Standardized and Extended Evaluation of Semantic Graphs

Juri Opitz

Heidelberg University

opitz.sci@gmail.com

Abstract

The SMATCH metric is a popular method for evaluating graph distances, as is necessary, for instance, to assess the performance of semantic graph parsing systems. However, we observe some issues in the metric that jeopardize meaningful evaluation. E.g., opaque pre-processing choices can affect results, and current graph-alignment solvers do not provide us with upper-bounds. Without upper-bounds, however, fair evaluation is not guaranteed. Furthermore, adaptations of SMATCH for extended tasks (e.g., fine-grained semantic similarity) are spread out, and lack a unifying framework.

For better inspection, we divide the metric into three modules: pre-processing, alignment, and scoring. Examining each module, we specify its goals and diagnose potential issues, for which we discuss and test mitigation strategies. For pre-processing, we show how to fully conform to annotation guidelines that allow structurally deviating but valid graphs. For safer and enhanced alignment, we show the feasibility of optimal alignment in a standard evaluation setup, and develop a lossless graph compression method that shrinks the search space and significantly increases efficiency. For improved scoring, we propose standardized and extended metric calculation of fine-grained sub-graph meaning aspects. Our code is available at <https://github.com/flipz357/smatchpp>

1 Introduction

Semantic graphs such as meaning representations (MRs) aim at capturing the meaning of a text. Typically, these graphs are rooted, directed, acyclic, and labeled. Vertices denote semantic entities, and edges represent semantic relations (e.g., *instrument*, *cause*, etc.). A prominent MR framework is *Abstract Meaning Representation (AMR)*, proposed by Banarescu et al. (2013), which anchors in a propositional knowledge base (Palmer et al., 2005).

Using a metric such as SMATCH (Cai and Knight, 2013), we can measure a distance (or similarity) between graphs, by aligning nodes, and counting matching graph triples. In fact, SMATCH measurement has various applications. It is used for selecting parsing systems that project AMR structures (Flanigan et al., 2014; May and Priyadarshi, 2017; Xu et al., 2020; Hoang et al., 2021a; Bevilacqua et al., 2021) and various other semantic graphs (van Noord et al., 2018; Zhang et al., 2018; Oepen et al., 2020; Stengel-Eskin et al., 2020; Martínez Lorenzo et al., 2022; Lin et al., 2022), for MR-based evaluation and diagnostics of text generation systems (Opitz and Frank, 2021; Manning and Schneider, 2021; Ribeiro et al., 2021; Hoyle et al., 2021), as backbone in an ensemble parsing algorithm (Hoang et al., 2021b), and for studying cross-lingual phenomena (Uhrig et al., 2021; Wein et al., 2022). Through SMATCH measured on sub-graphs, we can assess similarity of linguistic phenomena such as semantic roles, negation, or coreference (Damonte et al., 2017), a property that can be leveraged in neural text embeddings (Opitz and Frank, 2022b).

However, SMATCH measurement is non-trivial and lacks specification. For instance, SMATCH involves an NP-hard *optimization problem* of structural *graph alignment*, which distinguishes it from most metrics used in other evaluation tasks. In practice, a solution of this problem is found by employing a hill-climber. However, a hill-climber terminates at local optima, and it cannot inform us about a score upper-bound. In the end, this means that we lack information about the quality of the returned solution, potentially lowering our trust in the final evaluation. To mitigate this issue, we would like to study the possibility of optimal solution, or solution with a tight upper-bound. There are also other issues, on which we lack understanding. E.g., we do not know to what extent different pre-processing choices may affect the evaluation results, and we miss specification of SMATCH’s

popular fine grained sub-graph metrics (Damonte et al., 2017), where it is unclear how sub-graphs should be best extracted and compared.

Paper structure and contributions First, we describe and generalize the SMATCH metric (§3), and summarize recent SMATCH variants in one framework. Then we break the metric down into three modules (§4), which lets us better distribute our attention over its key components. For each module, we discuss specification of goals and mitigation of issues. In the pre-processing module (§5), we motivate graph standardization to allow safer matching of equivalent MR graphs with different structural choices. In the optimization module (§6), we test strategies for solving the alignment problem with optimality guarantees. In the scoring module (§7), we discuss standardized and extended scoring of fine-grained semantic aspects, such as causality, tense, and location.

2 Related work

Metric standardization An inspiration for us is the work of Post (2018), who propose the popular SACREBLEU framework for fairer comparison of machine translation systems with a standardized BLEU metric (Papineni et al., 2002). Specifically, SACREBLEU ships BLEU *together with* a specified tokenizer – prior to this, BLEU differences between systems could depend on different tokenization protocols. Facing the challenging problem of graph evaluation, a main contribution of our work is that we i) analyze weak spots in the current evaluation setup and ii) discuss ways of mitigating these issues, aiming at best evaluation practices.

MR metrics Cai and Lam (2019) introduce a variant of SMATCH (Cai and Knight, 2013) that penalizes dissimilar structures if they are situated in proximity of the graph root, motivated by their assumption that ‘core-semantics’ are located near the root of MR graphs. Furthermore, Opitz et al. (2020) introduce a SMATCH variant that performs a graded match of semantic concepts (e.g., *cat* vs. *kitten*), aiming at extended use-cases beyond parsing evaluation, where MRs of different sentences need to be compared. Similarly, Wein and Schneider (2022) adapt an embedding-based variant of SMATCH for cross-lingual MR comparison. We show that the different SMATCH adaptations can be viewed through the same lens with a generalized notion of triple match. Furthermore, Damonte et al.

(2017) propose fine-grained SMATCH that measure MR agreement in different aspects, such as *semantic roles*, *coreference* or *polarity*. We diagnose and mitigate issues in the aspectual assessment, and show how to extend the measured aspects.

Conceptually different MR metrics have been proposed by Anchieta et al. (2019) and Song and Gildea (2019) who aim at increased efficiency using structure extraction via breadth-first traversals, or Opitz et al. (2021) who compare MRs of different sentences with Wasserstein Weisfeiler-Leman kernels (Weisfeiler and Leman, 1968; Togninalli et al., 2019). Since significant parts of this paper are independent from SMATCH-specific scoring¹, other MR metrics can profit from our work.

3 SMATCH: Overview and generalization

We introduce SMATCH and define a generalized SMATCH, so that we can summarize recent SMATCH variants in one framework.

Preliminary I: MR graph If not mentioned otherwise, we view an MR graph a as a set of triples, where a triple has one of two types. Unary triples have the structure $\langle x, :rel, c \rangle$, where the source x is a variable and the target c is a descriptive label that shows the type or an attribute of x , depending on the edge label $:rel$.² Using variables such as x we can (co-)refer to different events and entities and capture complex events. Binary triples have the structure $\langle x, :rel, y \rangle$, where both the source x and the target y are variables.³

Preliminary II: SMATCH The idea of SMATCH is to measure structural similarity of graphs via the amount of triples that are shared by a and b . To obtain a meaningful score, we must know an alignment $map: vars(a) \leftrightarrow vars(b)$ that tells us how to map a variable in the first MR to a variable in the second MR. In this alignment, every variable from a can have at maximum one partner in b (and vice versa). Let an application of a map to a graph a be denoted as $a^{map} := \{t^{map} ; t \in a\}$, where t^{map} of a triple $t = \langle x, :rel, y \rangle$ is set to $t^{map} = \langle map(x), :rel, map(y) \rangle$ for binary triples, and $t^{map} = \langle map(x), :rel, c \rangle$

¹E.g., input standardization (§5) and sub-graph extraction for fine-grained aspectual matching (§7.3).

²E.g., $\langle x, :instance, cat \rangle$ would indicate that ‘ x is a cat’, while $\langle y, :polarity, - \rangle$ means y is negated.

³E.g., $\langle x, :location, y \rangle$, which means that x is located at y , or $\langle x, :arg0, y \rangle$ which usually indicates that y participates as the agent in the event referred to by x .

for unary triples.

Under any alignment map , we can calculate an overlap score f . In original SMATCH, f is the size of the triple overlap of a and b :

$$f(a, b, map) = |a^{map} \cap b|. \quad (1)$$

Ultimately we are interested in

$$F = \max_{map} f(a, b, map), \quad (2)$$

Finding a maximizer map^* lies at the heart of SMATCH, and we will dedicate ourselves to it later in §6. For now, we assume that we have map^* at our disposal. Therefore, we can calculate *precision* (P) and *recall* (R):

$$P = |a|^{-1}F, \quad R = |b|^{-1}F, \quad (3)$$

to obtain a final F1 evaluation score: $2PR/(P + R)$. With such a score, we can assess the similarity of MRs, and compare and select parsing systems.

Generalizing SMATCH In SMATCH, two triples are said to match if they are identical under a mapping. I.e., we match with $match(t, t') := I[t = t']$ that returns 1 if two triples t and t' are the same, and zero else (we omit the map for simplicity). Recently, SMATCH has been adapted and tailored to different use-cases. E.g., SMATCH has been extended to incorporate word embeddings (Opitz et al., 2020; Wein and Schneider, 2022) to match $\langle x, :instance, c \rangle$ triples for studying cross-lingual MRs or MRs of different sentences.⁴ On the other hand, Cai and Lam (2019) propose a root-distance bias, based on the assumption that ‘core-semantics’ lie in the proximity of an MR’s root.

We find that we can summarize such variants in one framework. We achieve this by introducing a *scaled triple matching* function:

$$match(t, t') = w_t^{t'} \cdot \begin{cases} I[t = t'], & \text{if } t_2, t'_2 \neq :inst. \\ I[t_1 = t'_1] \cdot sim(t_3, t'_3) & \text{else} \end{cases}$$

For matching concepts with embeddings, we can use an embedding similarity on the descriptive concept labels with $sim(c, c')$ and the importance

⁴Consider $\langle x, :instance, cat \rangle$ extracted from one sentence vs. $\langle y, :instance, kitten \rangle$ extracted from another sentence. A graded match is required to properly assess the similarity of the concepts.

“(d / dog :location (h / house))”

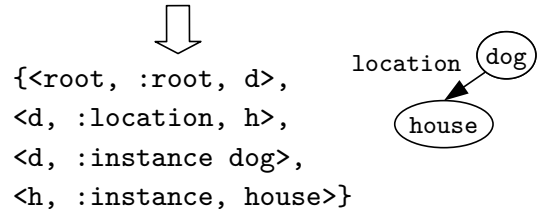


Figure 1: A serialized MR string is read into a graph.

weight $w_t^{t'} = 1 \forall t, t'$.⁵ For Root-distance biased SMATCH as proposed by Cai and Lam (2019) we set $w_t^{t'}$ such that we discount triple matches that are distant to the root.⁶

Our generalization does not change or constrain the original SMATCH. Instead, our goal was to define a more general framework of SMATCH-type metrics that unifies recently proposed SMATCH variants and show possibilities for further extension. For the following studies, we set SMATCH++ to basic SMATCH, which is recovered by setting $\forall t, t' : w_t^{t'} = 1$ and $sim(c, c') := I[c = c']$.

4 A modular view on SMATCH

To set the stage for inspection, we break SMATCH down into three modules. i) *Preprocessing*, ii) *Alignment*, and iii) *Scoring*. In particular, i) *Preprocessing* discusses any graph reading and processing in advance of the alignment. ii) *Alignment* revolves around the search mechanism used for finding an optimal mapping map^* . iii) *Scoring* involves calculating final scores and statistics that are returned to a user. For each module, we will specify its goals, assess potential weak spots and discuss mitigation.

5 Module I: Pre-processing

5.1 Module goal and current implementation

MRs are typically stored and distributed in a ‘Penman’ string format, which can serialize any rooted and directed graph into a string. The goal of this module is to project two serialized textual MRs onto two sets of triples, as outlined in Figure 1.

The target domain of this projection should be a *standardized* MR graph space, where format divergences that do not impact graph seman-

⁵That is, if the triples are not instance triples, we check whether the triples are equivalent (as in standard SMATCH), but if both triples are instance relation triples and the variables t_1, t'_1 are set to equal each other, we calculate the similarity between their descriptive concept labels.

⁶For a properly normalized final score if $\exists (t, t'), w_t^{t'} \neq 1$, we may have to change denominators in Eq. 3

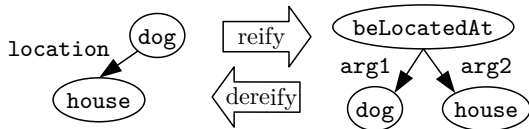


Figure 2: Outline of *location*-reification.

tics are eliminated. Original SMATCH performs pre-processing as follows: i) lower-case strings, ii) de-invert edges (e.g., $\langle x, :relation-of, y \rangle \rightarrow \langle y, :relation, x \rangle$). However, while these steps seem sensible, more steps can be undertaken to enhance evaluation.

5.2 Two structures, one meaning: reification

Some MR guidelines, including the AMR guideline, allow meaning-preserving structural graph translations (Banarescu et al., 2019; Goodman, 2019) with so called *reifications* (or *de-reification* as an inverse mechanism). A subset of relations is selected to constitute a semantic relation core set (e.g., $:arg0, :arg1, \dots, :op1, :op2, \dots$) and for all other remaining relations (e.g., $:location, :time$), we use rules to map the relation to a sub-graph, where the rule-triggering relation label is projected onto a node, and the former source and target of the relation are attached with outgoing core relations. E.g., consider Figure 2, where a reification is applied to a $\langle x, :location, y \rangle$ relation. In this case, the rule is:

- location (de)reification:

$$\begin{aligned} &\langle x, :location, y \rangle \\ &\iff \\ &\langle z, :instance, beLocatedAt \rangle \\ &\wedge \langle z, :arg1, x \rangle \\ &\wedge \langle z, :arg2, y \rangle, \end{aligned}$$

where $:arg1$ indicates the thing that is found at a location $:arg2$.

The question whether an annotator should use either means of representation, is answered in the guidelines as follows: *whenever they feel like it* (Banarescu et al., 2019). Therefore, a parser should not be penalized or rewarded for projecting reified (or non-reified) structures.

Empirical assessment of effect To understand the effect that reification can have on the final SMATCH score, it is interesting to study an edge-case: evaluating graphs that are fully reified against graphs that are fully de-reified. As a data set we take LDC2017T10, a standard AMR benchmark.

	Data setup		SMATCH	
	X	Y	Orig	rlyStd
i)	gold dereify	gold reify	73.8	100.0
ii)	gold standard	gold reify	73.9	100.0
iii)	gold standard	gold dereify	100.0	100.0
iv)	parser dereify	gold reify	60.9	82.8
v)	parser reify	gold reify	82.8	82.8
vi)	parser dereify	gold dereify	81.4	82.8
vii)	parser standard	gold standard	81.4	82.8
viii)	parser standard	gold reify	60.9	82.8
ix)	parser standard	gold dereify	81.4	82.8

Table 1: Results of meaning-preserving translations. rlyStd: score when we project X and Y into standardized reified space.

Additionally, we gather automatic parses by applying an AMR parser (Xu et al., 2020).

The results of this experiment are shown in Table 1. In the first three lines (i-iii) we compare *equivalent* translated versions of the test partition (gold vs. gold). We find that two equivalent gold standards can be judged to be very different (73.9 points, -26.1 points). A similar phenomenon can be observed when looking at the parses. The best parser score is achieved when comparing parses and references in the domain of reified graphs (82.8 points). On the other hand, if only the reference is reified, the parser score drops by 20 points (viii).

However, we also see that the results of a basic evaluation (vii) is practically the same as the result when evaluating with de-reified graphs (vi), indicating that both parser and gold annotation abstain from reification, where possible.

Discussion Having established that rule-based graph translations can enhance evaluation fairness, we pose the question: *should we prefer reification or de-reification for space standardization?*

The answer should be *reification*, since it can be seen as a form of generalization. More precisely, we note that reification of non-core relations is *always* possible. In fact, an interesting effect of reified structures is that they equip us with the means to attach further structure, or features, to semantic relations. On the other hand, however, de-reification is not always possible. It is only well-defined if there is no incoming edge into the node that corresponds to the non-core relation⁷, and if

⁷It is not clear to which node the incoming edge (that now does not have a target) should be re-attached: the $arg0$ or $arg1$ of the outgoing edges of the former node? Either choice would likely come with a change in meaning.

there are not more than two outgoing edges⁸.

However, there are also (practical) arguments against reification. Consider that de/non-reified MRs are smaller and have more edge label differentiation. This i) may facilitate more intuitive display for humans and ii) shrinks the alignment search space. Indeed, a large solution space may have ramifications for evaluation optimality and efficiency (in §6, we empirically study this issue). Therefore, when taking into account that the empirical effect size appears neglectable in the average case, these trade-offs may not always be justified, and we may instead use de-reification, where possible.

5.3 Triple removals

Duplicate triples are triples that occur more than once. We find that they are sometimes produced by some parsers. Additionally, some parsers introduce a node more than once, which results in two triples $\langle x, :instance, a \rangle$ and $\langle x, :instance, b \rangle$. Currently, SMATCH removes all such introductions of a second concept, but does not remove duplicate triples. By contrast, we propose to remove all duplicate triples, since they have no clear semantics, and stay agnostic to second introductions of a concept (in some MRs, it may be acceptable that an entity is the instance of two concepts), keeping all such triples (if they are not identical).⁹

6 Module II: Alignment

The goal of this module is solving Eq. 2, finding a map^* for optimal matching score.

SMATCH uses a hill-climber for solving Eq. 2. An issue with this is that such a heuristic terminates at local optima and cannot provide us with any *upper-bounds*. Upper-bounds, however, can inform users about the *quality* of the outputted solution and thus increase the trustworthiness of the final score (and any parser comparison that is based thereupon). Therefore, we can conclude that using a hill-climber seems **practical but may not be optimal**, especially when considering cases where fair comparison needs to be *guaranteed*. Instead, we would like to use an Integer Linear Program (ILP) to obtain the (optimal) solution. Alternatively, at least, we would like to know a tight upper-bound to inform ourselves about the trustworthiness of

⁸I.e., since reification can potentially be used to model n -ary relations, only in the case where $n = 2$ we can model the structure with a single (labelled) edge

⁹Due to rare occurrence of such phenomena in our parsed data, we find the effects of either choice to be negligible.

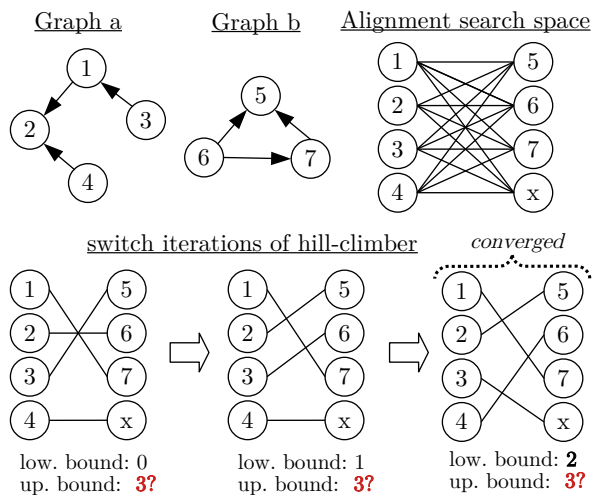


Figure 3: Sketch of search space (top) and hill-climber run (bottom). Every hill-climber step constitutes an improved lower bound, but we cannot obtain a tight upper-bound (an accessible trivial upper-bound is the amount of triples in the smaller of two graphs: 3).

the final score. But ILP is NP hard, and therefore it seems **optimal but possibly not practical**, a conception that might favor the usage of a hill-climber.

Triggered by these considerations, we review the hill-climber and the ILP and assess their effects on MR evaluation, with two desiderata in mind: evaluation quality and efficiency. Additionally, we propose a strategy for loss-less MR compression that can improve efficiency of any solver.

6.1 Practical but not optimal: hill-climber

SMATCH hill-climbing uses two operations, which we denote as *switch*, and *assign*. The *assign*-operation assigns a variable from $vars(a)$ to an unaligned variable from $vars(b)$: $(i, \emptyset) \rightarrow (i, j = map'(i))$, where map' is a candidate map. The *switch* operation does an alignment cross-over with respect to two alignment pairs, i.e.: $(i, j = map(i)) \wedge (k, l = map(k)) \rightarrow (i, l = map'(i)) \wedge (k, j = map'(k))$, where map is the current alignment and map' the candidate alignment. In each iteration, we examine all possible *switch*- and *assign* options, and greedily choose the best one.¹⁰ An example alignment procedure is shown in Figure 3.

In practice, we can resort to multiple random restarts, to find better optima. However, this hardly

¹⁰*Assign* is just a special instance of the more general *switch* so we can ablate the *assign* step. Then, *assign* becomes $(i, \emptyset = map(i)) \wedge (k, j = map(k)) \rightarrow (i, j = map'(i)) \wedge (k, \emptyset = map'(k))$, which is a *switch*.

addresses the underlying issue: we lack any information on upper-bounds, which may decrease trustworthiness of results, especially when facing larger graphs with lots of local optima.

6.2 ILP: Optimal, but less practical?

We would like to use Integer Linear Programming (ILP) for optimal solution of the graph alignment.

Problem statement Assume two graphs g, g' with node sets V, V' . Let $u(i, j)$ denote the amount of unary triple matches, given we align i from V to j from V' , counting matches of triples that involve one MR variable. On the other hand, $b(i, j, k, l)$ will denote the amount of structural binary triple matches, given we align i from V to j from V' and k from V to l from V' . Here, we count matching binary triples that involve two MR variables. Usually, these data are pre-computed. Let x indicate our current *map*, i.e., if $x_{ij} = 1$ then we align i from V to j from V' . We find our solution at

$$\begin{aligned} \max \quad & \sum_{\substack{(i,j) \in \\ V \times V}} u(i, j)x_{ij} + \sum_{\substack{(i,j,k,l) \in \\ (V \times V)^2}} b(i, j, k, l)x_{ij}x_{kl} \\ \text{st} \quad & \sum_j x_{ij} \leq 1; \quad \sum_i x_{ij} \leq 1 \\ & x_{ij} \in \{0, 1\} \quad \forall (i, j) \in V \times V' \end{aligned}$$

The constraint ensures that every node from one graph is aligned, at maximum, to one node from the other graph. By linearization, and introducing structural variables y , we obtain the equivalent ILP:

$$\begin{aligned} \max \quad & \sum_{\substack{(i,j) \in \\ V \times V'}} u(i, j)x_{ij} + \sum_{\substack{(i,j,k,l) \in \\ (V \times V')^2}} b(i, j, k, l)y_{ijkl} \\ \text{st} \quad & \sum_j x_{ij} \leq 1; \quad \sum_i x_{ij} \leq 1 \\ & y_{ijkl} \leq x_{ij}, \quad \forall (i, j, k, l) \in (V \times V')^2 \\ & y_{ijkl} \leq x_{kl}, \quad \forall (i, j, k, l) \in (V \times V')^2 \\ & x_{ij} \in \{0, 1\} \quad \forall (i, j) \in V \times V' \\ & y_{ijkl} \in \{0, 1\} \quad \forall (i, j, k, l) \in (V \times V')^2, \end{aligned}$$

where the structural variables, if active, show us countable binary triple matches. This is an NP complete problem, imposing limits on its capability to provide us with optimal solutions for larger graphs (note, however, that we can retrieve intermediate solutions and upper-bounds).

6.3 Reduced search space with lossless graph compression

We observe that in an MR a , every variable $x \in vars(a)$ is related to a concept c , e.g., $\langle x, :instance, cat \rangle$. This means that a concept c does *identify* a variable $x \in vars(a)$ iff $\forall y \in vars(a) : \langle y, :instance, c \rangle \Rightarrow y = x$. Therefore, if x denotes a *cat*, and there is no other entity in the MR that also denotes a *cat*, then x may be referred to simply by *cat*. This carries over to pairs of MRs: which are the focus of the paper – instead of considering $vars(a)$, we simply consider $vars(a) \cup vars(b)$. Therefore, we can replace all n variables from $vars(a) \cup vars(b)$ that are *identified* by concepts, with the corresponding concepts (see Appendix A.1 for a full example). This shrinks the search space by reducing the amount of variables that the optimizer has to consider. Note that such a compression is *lossless*, in the sense that the possibility of full reconstruction of the original MR is ensured. This implies that if two compressed MRs are assessed as (non-)isomorphic, then the uncompressed MRs are also (non-)isomorphic.

6.4 Solver experiments

Two questions are of main interest: 1. *RQ1, solution quality*: (How) do the final SMATCH results depend on the solver? 2. *RQ2, solution efficiency*: How does the evaluation time depend on the solver? In addition, we would like to assess how our answers to RQ1 and RQ2 might be affected by reification (resulting in a bigger search space) and MR compression (resulting in a smaller search space).

Setup We simulate a standard AMR parsing evaluation setting. We parse the LDC2017T10 testing data with six parsers: \mathcal{P}_1 (Xu et al., 2020), \mathcal{P}_2 (Cai and Lam, 2020), \mathcal{P}_3 (Lindemann et al., 2020), \mathcal{P}_4 (Zhang et al., 2019), \mathcal{P}_5 (Lyu and Titov, 2018), \mathcal{P}_6 (Cai and Lam, 2019). We evaluate the parsers using ILP or hill-climber (denoted by \triangle). As is standard, we show F1 micro corpus scores. For reference, we also run evaluation with the standard SMATCH hill-climbing script (denoted as *previous*). We observe that we successfully reproduce the scores from the standard SMATCH script with our \triangle implementation (first two lines of Table 2).¹¹

		parser scores (ranked)							time	# vars	quality
		\mathcal{P}_1	\mathcal{P}_2	\mathcal{P}_3	\mathcal{P}_4	\mathcal{P}_5	\mathcal{P}_6	secs	(tot., avg., max.)	(yield, bound)	
all vars	basic	prev.	81.4 ₍₁₎	80.3 ₍₂₎	77.0 ₍₃₎	76.3 ₍₄₎	74.5 ₍₅₎	73.1 ₍₆₎	50.4	(20346, 15, 129)	(217702, +?)
	basic	\triangle_4	81.4 ₍₁₎	80.3 ₍₂₎	77.0 ₍₃₎	76.2 ₍₄₎	75.1 ₍₅₎	73.1 ₍₆₎	49.9	see above	(217716, +?)
	basic	ILP	81.5 ₍₁₎	80.4 ₍₂₎	77.1 ₍₃₎	76.5 ₍₄₎	75.2 ₍₅₎	73.3 ₍₆₎	98.0	see above	(218072, +0)
	reify	\triangle_4	82.8 ₍₁₎	81.3 ₍₂₎	78.3 ₍₃₎	77.7 ₍₄₎	76.9 ₍₅₎	74.7 ₍₆₎	134.5	(27812, 20, 174)	(288597, +?)
	reify	ILP	83.5 ₍₁₎	82.1 ₍₂₎	79.3 ₍₃₎	78.7 ₍₄₎	77.7 ₍₅₎	75.8 ₍₆₎	300.5	see above	(291370, +13)
	<hr/>										
compress	basic	\triangle_1	72.9 ₍₁₎	70.9 ₍₂₎	67.1 ₍₃₎	66.2 ₍₄₎	64.6 ₍₅₎	61.5 ₍₆₎	7.3	(5568, 4, 62)	(74163, +?)
	basic	ILP	73.3 ₍₁₎	71.3 ₍₂₎	67.5 ₍₃₎	66.3 ₍₄₎	65.0 ₍₅₎	62.1 ₍₆₎	11.7	see above	(75036, +0)
	reify	\triangle_1	74.9 ₍₁₎	72.8 ₍₂₎	69.5 ₍₃₎	68.7 ₍₄₎	67.7 ₍₅₎	64.6 ₍₆₎	31.4	(10704, 8, 106)	(124323, +?)
	reify	ILP	76.7 ₍₁₎	74.1 ₍₂₎	71.3 ₍₃₎	70.6 ₍₄₎	69.5 ₍₅₎	66.4 ₍₆₎	27.3	see above	(129019, +0)

Table 2: Parser evaluation. *time* refers to the approximate total time needed to evaluate a single parser (i.e., processing 1371 graph pairs). $\triangle-N$ indicates hill-climber optimizer with N restarts. *quality*: solution quality of solver – first number is the amount of matching triples summed over all six parser evaluations (yield); second number indicates the tightest found upper-bound (which is only known by ILP).

6.4.1 RQ1: solution quality

Insight: Better alignment → safer evaluation

Importantly, we see that the ILP yields score increments for all parsers, which signals the occurrence of alignment problems, where the \triangle (despite multiple restarts) did not find the optimal solution. The effect-size is larger for reified graphs. We find differences of up to 1 point F1 score (Table 2: reify \triangle_4 vs. reify ILP). This can be explained by the growth of the alignment search space – reification makes graphs larger and introduces more MR variables. This explanation is further supported by contrasting the amount of unique final objective values against the size of the alignment space with different random initializations of the hill-climber (Appendix A.2, Figure 6). We see that i) for many graph pairs there are multiple local optima, and ii) the likelihood of finding a non-global optimum with the \triangle increases for larger/reified graphs.

We further study upper-bounds and solution quality (right column of Table 2). The ILP found the optimal solution in all cases, yielding 218072 matching triples. The \triangle_4 finds 217,700 matching triples (99.83%), which misses the mark by 350 triples. When evaluating reified graphs, the ILP returns 291370 matches and thus misses its temporary tightest upper-bound by 13 triples, indicating that in a few cases, a sub-optimal solution might have been found.¹² The \triangle_4 , however, yields only

288,597 matches (99.04%) and misses the temporary ILP upper-bound by 2,786. The growing gap underlines the degrading quality of the hill-climber when facing larger graphs.

Finally, the (slight) *differences* in increments among parsers when we evaluate them on reified graphs indicate that different parsers do make different decisions on when to reify an edge. For instance the score difference Δ for reified graphs vs. non-reified graphs (using ILP) of $\mathcal{P}_5, \mathcal{P}_6$ is 2.5 points, for \mathcal{P}_1 2 points and for \mathcal{P}_2 1.7 points. This supports our theoretical insights from §5.2 – reification can make parser comparison fairer.

6.4.2 RQ2: Solution efficiency

Insight I: ILP isn’t that impractical It seems to be commonly presumed that original SMATCH uses a hill-climber to make evaluation more practical and fast. However, our results qualify this presumption. For evaluating a full corpus (1371 graph pairs), SMATCH with ILP needs only about 48 seconds longer than original SMATCH with hill-climber (50s vs 98s). When the search space grows (due to reification) the time gap widens to a difference of 165 seconds. However, the consistent improvement of scores due to ILP (signaling sub-optimal hill-climber solutions) can make the time increase acceptable for evaluations where fairness is critical.

Insight II: MR compression increases evaluation speed

Viewing the last four rows of Table 2, we see that the MR compression i) did not lead to switched system ranks and ii) increased the evaluation speed by a large factor. Using MR compression, the ILP runs a full system evaluation in 11.7

¹¹An improvement is obtained for \mathcal{P}_5 . We find that we can mostly attribute this to a bug in the original script that prevents proper graph reading of some parses of \mathcal{P}_5 .

¹²Indeed, we find one graph by \mathcal{P}_2 , and one graph by \mathcal{P}_6 , where the ILP terminates after a 240s timeout that we set, and returns a temporary solution.

avg.	parser scores					
	\mathcal{P}_1	\mathcal{P}_2	\mathcal{P}_3	\mathcal{P}_4	\mathcal{P}_5	\mathcal{P}_6
mic.	81.5 ^{81.1} _{80.7}	80.4 ^{81.2} _{79.6}	77.1 ^{77.8} _{76.2}	76.5 ^{77.2} _{75.6}	75.2 ^{75.8} _{74.5}	73.3 ^{74.1} _{72.4}
mac.	82.6 ^{81.8} _{83.3}	81.4 ^{82.1} _{80.7}	79.0 ^{79.8} _{78.2}	78.3 ^{79.1} _{77.5}	76.2 ^{77.0} _{75.4}	75.9 ^{75.0} _{76.6}

Table 3: Evaluation with additional macro statistics and confidence intervals. Solver: ILP.

seconds for standard graphs and 27.3 seconds for the reified graphs. Given that the MR compression is lossless (c.f. §6.2), it provides us with an option for more efficient evaluation that is also safe (i.e., optimal).

7 Module III: scoring

7.1 Main scores: Precision, Recall and F1

The goal of this module is to provide the user with a final result. As discussed in §3, the main scores (Precision, Recall, and F1) follow directly from the *map**. The final score is typically micro averaged, summing matching statistics across all graph pairs before they are normalized. SMATCH++ makes two additions, macro-scoring and confidence intervals. Macro-averaging scores over graph pairs can be a useful complementary signal, specifically when comparing high-performance parsers (Opitz and Frank, 2022a). Additionally, we adopt the bootstrap assumption (Efron, 1992) for calculating confidence intervals. To make calculation feasible, bootstrapping is performed after the alignment stage. Table 3 shows results of the additional statistics. Confidence intervals range between $\pm[0.5, 1]$ points for all parsers. Macro score shows an outlier, where \mathcal{P}_6 (+2.6 points) is more positively affected than other parsers ($\pm[1.0, 1.9]$ points).¹³

7.2 Measuring aspectual semantic similarity

We observe considerable interest in applying fine-grained aspectual MR metrics (Damonte et al., 2017) for inspecting linguistic aspects captured by MRs (e.g., semantic roles, negation, etc.). Applications range from parser diagnostics (Lyu and Titov, 2018; Xu et al., 2020; Bevilacqua et al., 2021; Martínez Lorenzo et al., 2022), to NLG system diagnostics and sentence similarity (Opitz and Frank, 2021, 2022b). Formally, given an aspect of interest *asp* and an MR *g*, we apply a sub-graph-extraction

¹³We find a potential explanation in a motivation of \mathcal{P}_6 's creators to focus on semantics in proximity of an MR's top node (the proportion of such semantics increases when the graph is smaller, and smaller graphs have more influence on macro average than on micro average).

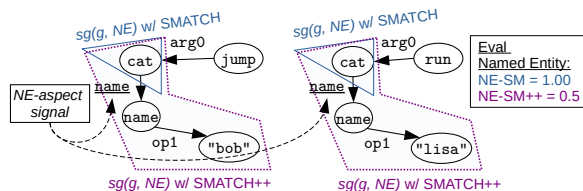


Figure 4: Named Entity (NE) sub-graph extraction with SMATCH vs. SMATCH++

function $sg(g, asp)$ to build an aspect-focused sub-graph, and compute a matching score (e.g., F1).

Review of previous implementation We study the description in Damonte et al. (2017) and the most frequently used implementation (Lyu, 2018). The treated aspects¹⁴ are divided in two broad groups: **i) alignment-based matching:** For some aspects, we extract aspect-related genuine sub-graphs, on which we calculate an optimal alignment. **ii) bag-of-label matching:** for other aspects, we detect aspect-related variables and gather associated node labels¹⁵ in a bag/list, to compute an overlap score based on simple set intersection.

E.g., *SRL*-aspect belongs to the first category (**i**): we extract $\langle x, :arg_n, y \rangle$ relations, and their corresponding *instance* triples (here: $\langle x, :instance, c \rangle$, and $\langle y, :instance, c' \rangle$). Then we calculate SMATCH on such SRL-subgraphs. The *Negation*, *Named Entity* (NEs) and *Frames* aspect is put into the second group (**ii**). We look for a relation/node-label that signals a particular aspect, e.g., $\langle x, :polarity, - \rangle$ (for negation) or $\langle x, :name, y \rangle$ (for NEs), we extract *x*, and replace *x* with the descriptive label *c* from $\langle x, :instance, c \rangle$. For *Frames*, we search for $\langle x, :instance, c \rangle$ where *c* is a PropBank predicate, and collect *c*. Finally, we can evaluate without an alignment, using set intersection.

Open questions We pose two questions:

1. Can the sub-graph extraction be improved?
2. Are there other aspects that we can measure?

7.3 Improving sub-graph extraction

Sensible range of extraction For some phenomena, the current extraction range is clearly too limited. For instance, let us consider named entities, which can be captured in more complex and nested

¹⁴See Appendix A.3 for a full overview.

¹⁵I.e., from $\langle x, :instance, label \rangle$ triples

MR structure. E.g., in AMR, one node typically indicates the type of the named entity (NE), and another multi-node structure represents its name and other attributes. Consider two AMRs a and b , from which we want to extract NE structures to measure the agreement of the graphs w.r.t. NE similarity. As shown in Figure 4, assume that one graph is about *a cat named Bob*¹⁶, while the other graph is about *a cat named Lisa*¹⁷. Obviously, the MRs have similarities in their NE structure (since there are named cats), but also differences (since the cats have different names). However, NE-focused SMATCH only extracts *cat* and *cat*, and returns maximum score.

Hence, for all finer-grained aspects that are captured by non-atomic MR structures (e.g., Named Entities), we propose to gather the full sub-graph starting at the aspect-indicating relation or node label. In the NE example, as shown in Figure 4, we would be provided a score of 0.5, better reflecting the similarity of the two NE structures.

Sub-graph compression, align and match We find a middle-ground in the advantages of the coarse matching (concreteness, efficiency) and graph alignment (safe matching) by using alignment with lossless MR compression. This is optimal and efficient, and alleviates the need to switch among fine and coarse extraction methods.

7.4 Extending fine-grained scores

Beyond negation and named entities – other semantic aspects We find that the fine-grained SMATCH metrics by Damonte et al. (2017) miss some interesting features captured by MRs. For instance, four interesting AMR aspects that are currently not captured are *cause*, *location*, *quantification*, and *tense*. SMATCH++ allows their integration in a straightforward way. An example for tense extraction is displayed in Figure 5, where our SMATCH++ sub-graph extraction extracts the complete temporal sub-graph, triggered by the edge label `:time` (if we would resort to the style of fine-grained SMATCH, we would miss larger parts of the temporal structure, only extracting the node label *end*).

Results of fine-grained parser diagnostics for *cause*, *location*, *quantification*, and *tense* are shown

¹⁶Triples: `<x, :instance, cat>`, `<y, :instance, name>`, `<x, :name, y>`, `<y, :opl, "bob">`.

¹⁷Triples: `<x, :instance, cat>`, `<y, :instance, name>`, `<x, :name, y>`, `<y, :opl, "lisa">`.

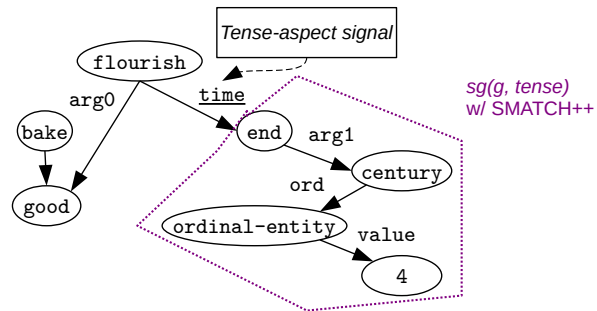


Figure 5: Temporal sub-graph extraction with SMATCH++ for an MR capturing “Baked goods flourished at the end of the fourth century”.

aspect	parser scores					
	\mathcal{P}_1	\mathcal{P}_2	\mathcal{P}_3	\mathcal{P}_4	\mathcal{P}_5	\mathcal{P}_6
cause	47.8	47.4	44.4	35.7	31.4	31.2
location	61.8	53.2↓	54.7↑	49.2↓	51.7↑	40.0
quant	69.4	67.4	58.4	56.8	56.5	55.8
tense	67.7	62.3	58.5	56.5	50.3	48.4

Table 4: Evaluation for *causal* and *temporal* structures. ↓↑ indicate switched ranks. Solver: ILP.

in Table 4.

Interestingly, we see that projecting *causality* seems hard: all parsers tend to struggle when assessing causal structures (31.2 up to 47.8 F1 points), showing much room for improvement. The temporal structures, on the other hand, can be assessed with somewhat higher accuracy (48.4 up to 67.7 points). We also see some switched ranks, indicating different parser strengths. Overall, parser score differences seem notably more pronounced than when calculating SMATCH (++) on the full graphs, showing the difficulty of capturing finer phenomena, and highlighting strengths of more recent parsers.

8 Conclusion

SMATCH++ is the first specification of a standardized, extended, and extensible SMATCH metric. We aim at i) standardized and transparent comparison of graph parsing systems, and ii) improved extensibility for custom applications.¹⁸ The applications can include finer parser diagnostics and measuring semantic sub-graph similarities such as *quantification*, *cause*, or *tense* with our fine-grained metrics.

Acknowledgments

We thank our reviewers for their helpful feedback.

¹⁸See Appendix A.4 for a summary of the default setup.

Limitations

We have to leave some questions open. First, we would have liked to shed more light on the solvers' behaviors when facing large graphs, in isolation. On one hand, our benchmark corpus indeed contains some large MRs with many variables, including reified MRs and MRs that represent multiple sentences (up to 174 variables, cf. Table 2). We have shown that ILP could cope with these harder problems, providing optimal solutions in reasonable time. When facing bigger graphs, however, we can expect that the solution quality of the hill-climber quickly degrades, while the ILP will struggle to find optimal solutions. While our graph compression strategy can help mitigate this issue by reducing the alignment search space, it would be interesting to study the quality of temporary solutions, or of solutions of LP relaxation. There are also relaxed ILP solvers (Klau, 2009) that iteratively tighten the lower and the upper-bound. They could prove useful for aligning larger MR graphs, or, at least, to find useful upper-bounds.

Second, in this paper we studied SMATCH (++) that measures *structural overlap* and assigns each triple the *same weight*. But structural differences of similar degree can have a different impact on overall meaning similarity as perceived by humans, which can have ramifications for measuring sentence similarity (Opitz et al., 2021) and meaningful evaluation of strong AMR parsers (Opitz and Frank, 2022a). Therefore, for a deeper assessment of MR similarity we may have to use conceptually different metrics, or explore SMATCH++-based strategies and (sensibly) weigh triples depending on label importance or compose an overall score by weighting measured sub-aspect similarities.

References

- Rafael Torres Anchieta, Marco Antonio Sobrevilla Cabezedo, 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.
- 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.
- Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2019. [Amr guidelines](https://github.com/amrisi/amr-guidelines/blob/master/amr.md). <https://github.com/amrisi/amr-guidelines/blob/master/amr.md>.
- 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*.
- Deng Cai and Wai Lam. 2019. [Core semantic first: A top-down approach for AMR parsing](#). 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 3799–3809, Hong Kong, China. Association for Computational Linguistics.
- Deng Cai and Wai Lam. 2020. [AMR parsing via graph-sequence iterative inference](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1290–1301, Online. Association for Computational Linguistics.
- 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.
- Marco Damonte, Shay B. Cohen, and Giorgio Satta. 2017. [An incremental parser for Abstract Meaning Representation](#). In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 536–546, Valencia, Spain. Association for Computational Linguistics.
- Bradley Efron. 1992. *Bootstrap methods: another look at the jackknife*. Springer.
- Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. 2014. [A discriminative graph-based parser for the Abstract Meaning Representation](#). In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1426–1436, Baltimore, Maryland. Association for Computational Linguistics.
- Michael Wayne Goodman. 2019. [AMR normalization for fairer evaluation](#). *CoRR*, abs/1909.01568.
- Thanh Lam Hoang, Gabriele Picco, Yufang Hou, Young-Suk Lee, Lam Nguyen, Dzung Phan, Vanessa López, and Ramon Fernandez Astudillo. 2021a. Ensembling graph predictions for amr parsing. *Advances in Neural Information Processing Systems*, 34.
- Thanh Lam Hoang, Gabriele Picco, Yufang Hou, Young-Suk Lee, Lam Nguyen, Dzung Phan, Vanessa Lopez, and Ramon Fernandez Astudillo. 2021b. [Ensembling](#)

- graph predictions for amr parsing. In *Advances in Neural Information Processing Systems*, volume 34, pages 8495–8505. Curran Associates, Inc.
- Alexander Miserlis Hoyle, Ana Marasović, and Noah A. Smith. 2021. Promoting graph awareness in linearized graph-to-text generation. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 944–956, Online. Association for Computational Linguistics.
- Gunnar W Klau. 2009. A new graph-based method for pairwise global network alignment. *BMC bioinformatics*, 10(1):1–9.
- Zi Lin, Jeremiah Liu, and Jingbo Shang. 2022. Neural-symbolic inference for robust autoregressive graph parsing via compositional uncertainty quantification. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 4759–4776, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Matthias Lindemann, Jonas Groschwitz, and Alexander Koller. 2020. Fast semantic parsing with well-typedness guarantees. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3929–3951, Online. Association for Computational Linguistics.
- Chunchuan Lyu. 2018. Fine-grained smatch implementation. <https://github.com/ChunchuanLv/amr-evaluation-tool-enhanced>.
- 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.
- Abelardo Carlos Martínez Lorenzo, Marco Maru, and Roberto Navigli. 2022. Fully-Semantic Parsing and Generation: the BabelNet Meaning Representation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1727–1741, Dublin, Ireland. Association for Computational Linguistics.
- Jonathan May and Jay Priyadarshi. 2017. Semeval-2017 task 9: Abstract meaning representation parsing and generation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 536–545.
- Stephan Open, Omri Abend, Lasha Abzianidze, Johan Bos, Jan Hajic, Daniel Hershcovich, Bin Li, Tim O’Gorman, Nianwen Xue, and Daniel Zeman. 2020. Mrp 2020: The second shared task on cross-framework and cross-lingual meaning representation parsing. In *Proceedings of the CoNLL 2020 Shared Task: Cross-Framework Meaning Representation Parsing*, pages 1–22.
- 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 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 and Anette Frank. 2022a. Better Smatch = better parser? AMR evaluation is not so simple anymore. In *Proceedings of the 3rd Workshop on Evaluation and Comparison of NLP Systems*, pages 32–43, Online. Association for Computational Linguistics.
- Juri Opitz and Anette Frank. 2022b. SBERT studies meaning representations: Decomposing sentence embeddings into explainable semantic features. In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 625–638, Online only. 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.
- Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.
- Kishore Papineni, Salim Roukos, Todd Ward, and Weijing 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, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.
- Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics.
- Leonardo F. R. Ribeiro, Yue Zhang, and Iryna Gurevych. 2021. Structural adapters in pretrained language models for AMR-to-Text generation. In *Proceedings of the 2021 Conference on Empirical Methods*

- in *Natural Language Processing*, pages 4269–4282, Online and Punta Cana, Dominican Republic. 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.
- Elias Stengel-Eskin, Aaron Steven White, Sheng Zhang, and Benjamin Van Durme. 2020. **Universal decompositional semantic parsing**. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8427–8439, Online. Association for Computational Linguistics.
- Matteo Togninalli, Elisabetta Ghisu, Felipe Llinares-López, Bastian Rieck, and Karsten Borgwardt. 2019. **Wasserstein weisfeiler-lehman graph kernels**. In *Advances in Neural Information Processing Systems*, volume 32, pages 6436–6446. Curran Associates, Inc.
- 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.
- 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).
- 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-XVI) within LREC2022*, pages 97–102, Marseille, France. European Language Resources Association.
- 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, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.
- Boris Weisfeiler and Andrei Leman. 1968. The reduction of a graph to canonical form and the algebra which appears therein. *NTI, Series*, 2(9):12–16.
- Dongqin Xu, Junhui Li, Muhua Zhu, Min Zhang, and Guodong Zhou. 2020. **Improving AMR parsing with sequence-to-sequence pre-training**. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2501–2511, Online. Association for Computational Linguistics.
- Sheng Zhang, Xutai Ma, Kevin Duh, and Benjamin Van Durme. 2019. **AMR parsing as sequence-to-graph transduction**. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 80–94, Florence, Italy. Association for Computational Linguistics.
- Sheng Zhang, Xutai Ma, Rachel Rudinger, Kevin Duh, and Benjamin Van Durme. 2018. **Cross-lingual decompositional semantic parsing**. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1664–1675, Brussels, Belgium. Association for Computational Linguistics.

A Appendix

A.1 Lossless graph pair reduction example

Consider *the cat scratches another cat*:

$$a = \{ \langle s, :instance, scratch \rangle, \langle c, :instance, cat \rangle, \langle d, :instance, cat \rangle, \langle s, :arg0, c \rangle, \langle s, :arg1, d \rangle \}$$

and *the gray cat scratches the small plant*:

$$b = \{ \langle x, :instance, scratch \rangle, \langle y, :instance, cat \rangle, \langle z, :instance, plant \rangle, \langle w, :instance, small \rangle, \langle v, :instance, gray \rangle, \langle x, :arg0, y \rangle, \langle x, :arg1, z \rangle, \langle y, :mod, v \rangle, \langle z, :mod, z \rangle \}.$$

The lossless compression is

$$a' = \{ \langle c, :instance, cat \rangle, \langle d, :instance, cat \rangle, \langle scratch, :arg0, c \rangle, \langle scratch, :arg1, d \rangle \} \quad \text{and}$$

$$b' = \{ \langle y, :instance, cat \rangle, \langle scratch, :arg0, y \rangle, \langle scratch, :arg1, plant \rangle, \langle y, :mod, gray \rangle, \langle plant, :mod, small \rangle \}.$$

The alignment search space is reduced from a size of more than 100 candidates to 2 candidate options ($y = c$, or $y = d$).

A.2 Assessing solution quality variability in dependence of variables

We use the parses of an example parser (\mathcal{P}_5)¹⁹. For every evaluation pair, we re-start the hillclimber 20 times, and collect the scores related to the found local optima. The Y-axis in Figure 6 shows the amount of unique scores found among the 20 tries (note that there could be more unique alignments

¹⁹We ran the experiment also with parses from other systems but always ended up with essentially the same results

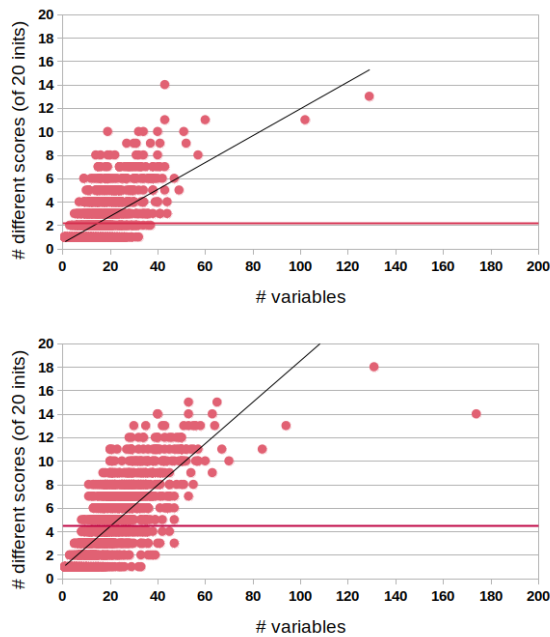


Figure 6: Assessing solution quality variability. Top: basic graphs, bottom: reified graphs. Diagonal line: linear trend. Horizontal line: arithmetic mean. See text in §A.2 for more description and §6.4.1 for discussion.

that would result in the same score – this is not captured in this Figure). The X-axis shows the amount of alignment variables. In different terms, a higher point in this Figure is equivalent to a larger pool of local optima of different quality, and thus we can conjecture a greater likelihood that the optimal solution is not returned by the hill-climber.

A.3 Aspect overview

Previously measured aspects For all aspects we retrieve F1, Precision, and Recall.

1. Measured under alignment

- (a) SRL: extract $\langle x, \text{arg}_n, y \rangle$ triples and corresponding instance triples.
- (b) Coreference/Re-entrancies: extract $\langle x, \text{rel}, y \rangle$ triples for which there is another triple $\langle z, \text{:rel}', y \rangle$ (meaning y is a re-entrant node) and also extract corresponding instance triples.

2. Measured via bag-of-structure extraction and set operations

- (a) Concepts: collect all node labels.
- (b) Frames: collect all node labels where the label is a PropBank predicate frame.
- (c) NonSenseFrames: see above, but with sense label removed

- (d) NE: Named entities, collect all node labels that have an outgoing `:name` relation.
- (e) Negation: collect all node labels that have an outgoing `:polarity` relation.
- (f) Wikification: collect all node labels that have an incoming `:wiki` relation.
- (g) IgnoreVars: replace all variables in triples with concepts, collect triples.

SRL, Named Entities, coreference (re-entrant nodes)

Additional aspects measured by us: *Cause*, *Tense*, *Location*, *Quantifier*.

We change: Add default option for extracting aspect sub-graphs, measure all aspects under alignment.

Aspects we added:

- *Cause*: Cause is modeled via `cause-01`. We extract label of `:arg1` (what is caused?) and subgraph of `:arg2`, the cause itself.
- *tense*: Tense is modeled via $\langle x, \text{:time}, y \rangle$ edge. We extract label of the thing that happens and subgraph of y , the temporal description where it happens.
- *location*: Similar to above but with `:location` edge.
- *quantifier*: Similar to above but with `:quant` edge.

A.4 Best practice

To provide a balance between efficiency, safety and meaningfulness of scores, default procedure of SMATCH++ is currently set to:

1. Pre-processing: lower-casing, duplicate-removal, de-reify where applicable.
2. Alignment: Solver: ILP. Triple-match: $w_t^t = 1 \forall t, t'; \text{sim}(c, c') := I[c = c']$
3. Scoring: Precision, Recall, F1, Bootstrap confidence intervals

An option to increase efficiency without incurring a loss in safety and meaningfulness is achieved by adding graph compression to the pre-processing. It is set as the default for fine semantic aspect scores. Also, to ensure utmost safety, we have to consider applying reification standardization (incurring a significantly longer evaluation time).