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, adaptions 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 (Ühlig 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 adaptions 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 semantie 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 Anchiêta 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 \( <x, :rel, c> \), 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 \( :rel \).

Using variables such as \( x \) we can (co-)refer to different events and entities and capture complex events. Binary triples have the structure \( <x, :rel, y> \), where both the source \( x \) and the target \( y \) are variables.2

**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 \( \text{map}; \text{vars}(a) \leftrightarrow \text{vars}(b) \) 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 \( \text{map} \) to a graph \( a \) be denoted as \( a^\text{map} := \{t^\text{map} ; t \in a\} \), where \( t^\text{map} \) of a triple \( t = <x, :rel, y> \) is set to \( t^\text{map} = <\text{map}(x), :rel, \text{map}(y)> \) for binary triples, and \( t^\text{map} = <\text{map}(x), :rel, c> \).

---

1E.g., input standardization (§5) and sub-graph extraction for fine-grained aspectual matching (§7.3).
2E.g., \( <x, :\text{instance, cat}> \) would indicate that \( x \) is a cat, while \( <y, :\text{polarity, ->}> \) means \( y \) is negated.
3E.g., \( <x, :\text{location, y}> \), which means that \( x \) is located at \( y \) or \( <x, :\text{arg0, y}> \) 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 \( b \):

\[
f(a, b, \text{map}) = |a^{\text{map}} \cap b|.
\]

Ultimately we are interested in

\[
F = \max_{\text{map}} f(a, b, \text{map}),
\]

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

\[
P = |a|^{-1}F, \quad R = |b|^{-1}F,
\]

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 \( \text{match}(t, t') := I[t = t'] \) that returns 1 if two triples \( t \) and \( t' \) are the same, and zero else (we omit the \( \text{map} \) 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 \( <x, \text{:instance, c}> \) triples for studying cross-lingual MRs or MRs of different sentences. 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:

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

For matching concepts with embeddings, we can use an embedding similarity on the descriptive concept labels with \( \text{sim}(c, c') \) and the importance weight \( w^t_t = 1 \forall t, t' \). 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.

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 \( \text{sim}(c, c') := I[c = c'] \).

### 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 \( \text{map}^* \). iii) **Scoring** involves calculating final scores and statistics that are returned to the 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-

---

**Footnotes:**

4 Consider \( <x, \text{instance, cat}> \) extracted from one sentence and \( <y, \text{instance, kitten}> \) extracted from another sentence. A graded match is required to properly assess the similarity of the concepts.

5 That 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.

6 For a properly normalized final score if \( \exists (t, t'), w^t_t \neq 1 \), we may have to change denominators in Eq. 3.
tics are eliminated. Original SMATCH performs pre-processing as follows: i) lower-case strings, ii) de-invert edges (e.g., <x, :relation-of, y> → <y, :relation, x>). 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, ..., :op1, :op2,...) 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 <x, :location, y> relation. In this case, the rule is:

- location (de)reification:
  
  \[ \langle x, :\text{location}, y \rangle \iff \langle z, :\text{instance}, \text{beLocatedAt} \rangle \land \langle z, :\text{arg1}, x \rangle \land \langle z, :\text{arg2}, y \rangle. \]

where :\text{arg1} indicates the thing that is found at a location :\text{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.

<table>
<thead>
<tr>
<th>Data setup</th>
<th>X</th>
<th>Y</th>
<th>SMATCH</th>
</tr>
</thead>
<tbody>
<tr>
<td>Orig</td>
<td>gold dereify</td>
<td>gold reify</td>
<td>73.8</td>
</tr>
<tr>
<td>rfyStd</td>
<td>gold dereify</td>
<td>gold reify</td>
<td>100.0</td>
</tr>
<tr>
<td></td>
<td>gold standard</td>
<td>gold reify</td>
<td>73.9</td>
</tr>
<tr>
<td></td>
<td>gold standard</td>
<td>gold dereify</td>
<td>100.0</td>
</tr>
<tr>
<td>i)</td>
<td>parser dereify</td>
<td>gold reify</td>
<td>60.9</td>
</tr>
<tr>
<td>v)</td>
<td>parser reify</td>
<td>gold reify</td>
<td>82.8</td>
</tr>
<tr>
<td>vi)</td>
<td>parser dereify</td>
<td>gold dereify</td>
<td>81.4</td>
</tr>
<tr>
<td>vii)</td>
<td>parser standard</td>
<td>gold standard</td>
<td>81.4</td>
</tr>
<tr>
<td>viii)</td>
<td>parser standard</td>
<td>gold dereify</td>
<td>60.9</td>
</tr>
<tr>
<td>ix)</td>
<td>parser standard</td>
<td>gold dereify</td>
<td>81.4</td>
</tr>
</tbody>
</table>

Table 1: Results of meaning-preserving translations. 

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 relation

\(^7\) It is not clear to which node the incoming edge (that now does not have a target) should be re-attached: the \text{arg0} or \text{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 edges\(^8\).

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 negligeble 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 \(<x, :\text{instance}, a>\) and \(<x, :\text{instance}, b>\). 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 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.: \((i, j = map(i)) \land (k, l = map(k)) \rightarrow (i, l = map'(i)) \land (k, j = map'(k))\), 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

---

\(^8\)I.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

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

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

\[
\begin{align*}
\max & \sum_{(i,j) \in V \times V} u(i,j)x_{ij} + \sum_{(i,j,k,l) \in (V \times V)^2} b(i,j,k,l)x_{ij}x_{kl} \\
\text{st} & \sum_{j} x_{ij} \leq 1; \quad \sum_{i} x_{ij} \leq 1 \\
& x_{ij} \in \{0, 1\} \quad \forall (i,j) \in V \times V'
\end{align*}
\]

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:

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

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., $<x, \text{:instance, cat}>$. This means that a concept $c$ does identify a variable $x \in vars(a)$ if $\forall y \in vars(a) : <y, \text{:instance, c}> \Rightarrow y = x$. 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: $P_1$ (Xu et al., 2020), $P_2$ (Cai and Lam, 2020), $P_3$ (Lindemann et al., 2020), $P_4$ (Zhang et al., 2019), $P_5$ (Lyu and Titov, 2018), $P_6$ (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 implementation (first two lines of Table 2).
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 \( \Delta \) (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 \( \Delta \) 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 \( \Delta \) 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 \( \Delta \) 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 the temporary upper-bound by 13 triples, indicating that in a few cases, a sub-optimal solution might have been found.\(^{12}\) The \( \Delta \), however, yields only 288,597 matches (99.04%) and misses the temporary 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 \( \Delta \) for reified graphs vs. non-reified graphs (using ILP) of \( P_5, P_6 \) is 2.5 points, for \( P_1 \) 2 points and for \( P_2 \) 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 seconds longer than without compression.
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 \( \text{map}^* \). 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 +[-0.5, 1] points for all parsers. Macro score shows an outlier, where \( P_6 (+2.6 \text{ points}) \) is more positively affected than other parsers (+[1.0, 1.9] points).\(^{13}\)

<table>
<thead>
<tr>
<th>Parsers</th>
<th>avg.</th>
<th>( P_1 )</th>
<th>( P_2 )</th>
<th>( P_3 )</th>
<th>( P_4 )</th>
<th>( P_5 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>mac.</td>
<td>81.5</td>
<td>84.5</td>
<td>87.77</td>
<td>76.57</td>
<td>75.32</td>
<td>73.32</td>
</tr>
<tr>
<td>mac.</td>
<td>82.63</td>
<td>84.27</td>
<td>79.05</td>
<td>78.31</td>
<td>76.24</td>
<td>75.87</td>
</tr>
</tbody>
</table>

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

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 \( \text{asp} \) and an MR \( g \), we apply a subgraph-extraction function \( \text{sg}(g, \text{asp}) \) to build an aspect-focused subgraph, 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 aspects\(^{14}\) are divided in two broad groups: i) alignment-based matching: For some aspects, we extract aspect-related genuine subgraphs, on which we calculate an optimal alignment. ii) bag-of-label matching: For other aspects, we detect aspect-related variables and gather associated node labels\(^{15}\) 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 \(<x, :\text{arg}_n, y>\) relations, and their corresponding instance triples (here: \(<x, :\text{instance}, c>\) and \(<y, :\text{instance}, c'>\>). 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., \(<x, :\text{polarity}, ->\) (for negation) or \(<x, :\text{name}, y>\) (for NEs), we extract \( x \), and replace \( x \) with the descriptive label \( c \) from \(<x, :\text{instance}, c>\). For Frames, we search for \(<x, :\text{instance}, c>\) 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

---

\(^{13}\)See Appendix A.3 for a full overview.

\(^{14}\)I.e., from \(<x, :\text{instance}, \text{name}>\) 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 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.

\(^{16}\)Triples: \(<x, :instance, cat>, <y, :instance, name>, <x, :name, y>, <y, :op1, "bob">.\)

\(^{17}\)Triples: \(<x, :instance, cat>, <y, :instance, name>, <x, :name, y>, <y, :op1, "lisa">.\)

\(^{18}\)See 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


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.


Thanh Lam Hoang, Gabriele Picco, Yufang Hou, Young-Suk Lee, Lam Nguyen, Dzung Phan, Vanessa Lopez, and Ramon Fernandez Astudillo. 2021b. Ensembling


in Natural Language Processing, pages 4269–4282, Online and Punta Cana, Dominican Republic. Association for 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.


A Appendix

A.1 Lossless graph pair reduction example

Consider the cat scratches another cat:

\[
\begin{align*}
&a = \{<s, :instance, scratch>, \\
&\quad <c, :instance, cat>, \\
&\quad <d, :instance, cat>, <s, :arg0, c>, \\
&\quad <s, :arg1, d>\}
\end{align*}
\]

and the gray cat scratches the small plant:

\[
\begin{align*}
&b = \{<x, :instance, scratch>, \\
&\quad <y, :instance, cat>, \\
&\quad <z, :instance, plant>, \\
&\quad <w, :instance, small>, \\
&\quad <v, :instance, gray>, \\
&\quad <x, :arg0, y>, <x, :arg1, z>, \\
&\quad <y, :mod, v>, <z, :mod, z>\}.
\end{align*}
\]

The lossless compression is

\[
\begin{align*}
&d' = \{<c, :instance, cat>, \\
&\quad <d, :instance, cat>, \\
&\quad <\text{scratch}, :arg0, c>, \\
&\quad <\text{scratch}, :arg1, d>\}
\end{align*}
\]

and

\[
\begin{align*}
&b' = \{<y, :instance, cat>, \\
&\quad <\text{scratch}, :arg0, y>, \\
&\quad <\text{scratch}, :arg1, plant>, \\
&\quad <y, :mod, gray>, \\
&\quad <\text{plant}, :mod, small>\}.
\end{align*}
\]

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

A.2 Assessing solution quality variability in dependence of variables

We use the parses of an example parser \((P_5)\)\(^{19}\). 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

\(^{19}\text{We ran the experiment also with parses from other systems but always ended up with essentially the same results}\)
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 $<x, \text{arg}_n, y>$ triples and corresponding instance triples.
   (b) Coreference/Re-entrancies: extract $<x, \text{rel}, y>$ triples for which there is another triple $<z, \text{rel}', y>$ (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 $:\text{name}$ relation.
   (e) Negation: collect all node labels that have an outgoing $:\text{polarity}$ relation.
   (f) Wikification: collect all node labels that have an incoming $:\text{wiki}$ 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 $\text{cause-01}$. We extract label of $:\text{arg1}$ (what is caused?) and subgraph of $:\text{arg2}$, the cause itself.
- tense: Tense is modeled via $<x, :\text{time}, y>$ 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 $:\text{location}$ edge.
- quantifier: Similar to above but with $:\text{quant}$ 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: $w_t = 1 \forall t, t'$; $\text{sim}(c, c') := I[c = c']$
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).