Anchor and Broadcast: An Efficient Concept Alignment Approach for Evaluation of Semantic Graphs

Haibo Sun, Nianwen Xue
Brandeis University
415 South Street, Waltham, MA 02453
{hsun, xuen}@brandeis.edu

Abstract

In this paper, we present AnCast, an intuitive and efficient tool for evaluating graph-based meaning representations (MR). AnCast implements evaluation metrics that are well understood in the NLP community, and they include *concept F1*, *unlabeled relation F1*, *labeled relation F1*, and *weighted relation F1*. The efficiency of the tool comes from a novel *anchor broadcast* alignment algorithm that is not subject to the trappings of local maxima. We show through experimental results that the AnCast score is highly correlated with the widely used Smatch score, but its computation takes only about 40% the time.

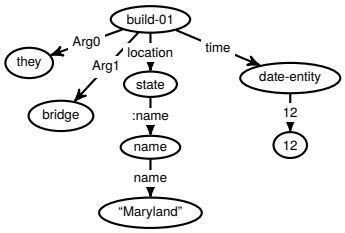
Keywords: Meaning representation evaluation, anchor broadcast, semantic parsing

1. Introduction

With the rapid advance of large language models as the background, there have been significant recent research activities on the building of graph-based meaning representation (MR) data sets (Knight et al., 2021; Van Gysel et al., 2021; Tu et al., 2024) and training meaning representation parsers to acquire such representations automatically (Bevilacqua et al., 2021; Bai et al., 2022; Chen et al., 2022). One particular meaning representation formalism that has received significant attention is Abstract Meaning Representation (AMR) (Banarescu et al., 2013), a sentence-level formalism designed for English that mathematically take the form of a rooted directed acyclic graph with nodes being *concepts* and edges representing *relations* between concepts. AMR has also been extended to cross-linguistic settings to create Uniform Meaning Representation (UMR) (Van Gysel et al., 2021), which additionally has a document-level representation that includes coreference, temporal, and modal relations that go beyond sentence-boundaries.

Linguistically, an AMR concept can be *concrete* or *abstract*. A concrete AMR concept can be either a lemma or a sense-disambiguated lemma that consists of the lemma and its sense ID. An abstract concept does not correspond to a specific word token in the sentence and is either inferred from the context or indicates a named entity type. An AMR relation either indicates a semantic role that an argument concept plays with respect to a predicate concept, or other types of semantic relations between parent and child concepts. An example with abstract and concrete concepts is provided in Figure 1.

To make advances in both building MR data sets and training MR parsers, it is important to have appropriate MR evaluation metrics. When building meaning representation data sets, it is critical to have an evaluation metric to measure the inter-annotator agreement (IAA) between annotators to ensure data consistency. When training MR parsers, an evaluation metric is needed to measure the performance of the parser by comparing its output against the human annotated gold standard.



(1) (b / build-01
:ARG0 (t / they)
:ARG1 (b2 / bridge)
:location (s / state
:name (n / name :op1 "Maryland"))
:time (d / date-entity :month 12))

Figure 1: An example AMR representation for the sentence "They built a bridge in Maryland in December."

A number of meaning representation evaluation metrics have been developed in the recent years, with the most widely used one being variants of Smatch (Cai and Knight, 2013; Opitz, 2023; Damonte et al., 2017). Smatch decomposes an AMR graph into a set of triples, either in the form of a $\langle var\ rel\ concept \rangle$ or $\langle var\ rel\ var \rangle$. The former are *concept triples* in which var is a variable, the relation rel is always "instance", and $concept$ is an AMR concept. In other words, this triple indicates that

LREC-COLING 2024, pages 1052–1062
20-25 May, 2024. © 2024 ELRA Language Resource Association: CC BY-NC 4.0

the variable represents an instance of a particular concept. The latter are *relation triples* that represent the relation between two variables, with each representing an instance of some concept. The Smatch score represents the fraction of triples that overlap between two AMR graphs.

As concept and relation triples are represented independently of one another, a relation triple match does not require that the pair of concepts between which the relation holds also match. This means that even if the concepts of two AMRs graphs are totally different, as long as a variable mapping exists between them, Smatch will assign non-zero scores to them, as illustrated in (2):

(2) He likes apples. She hates oranges.

```
(l / like                (h / hate
  :ARG0 (j / he)           :ARG0 (m / she)
  :ARG1 (a / apple))       :ARG1 (o / orange))
```

This runs counter to our intuition that a match in relation is premised on the requirement that the concepts should also match, as has been standard in the evaluation of dependency trees, which are special cases of graphs. In dependency-based parsing evaluation, it is standard to use labeled and unlabeled attachment as evaluation metrics (Nivre et al., 2020; Lo and Wu, 2011; Papineni et al., 2001). When calculating unlabeled attachment, both the parent and child are required to match. When calculating labeled attachment, the relation has to match as well, in addition to the match for parent and child. Translating the unlabeled and labeled attachment to graph-based meaning evaluation means that we will assign different scores to the three AMR graphs for the sentence in (3).

(3) He likes her.

```
(l / like                (l / like
  :ARG0 (h / he)           :ARG0 (s / she)
  :ARG1 (s / she))         :ARG1 (h / he))
(h / he
   :ARG0 (l / likes)
   :ARG1 (s / she))
```

AMR graphs differ from dependency trees in that there is no guarantee that the concepts from two graphs for the same sentence will be the same. This means that we need an additional metric for concept triples. By decomposing the meaning representation metric into three different scores, *concept F1*, *unlabeled relation F1*, and *labeled relation F1*, we will have a metric that is easily interpretable and familiar to users in the NLP community.

Unlike dependency trees, there is no inherent alignment between the nodes in an AMR graph and the word tokens for a sentence, as some word tokens may not align with any concepts in the AMR graph, while some AMR concepts (e.g., *abstract concepts*) in the graph may not map to any word tokens. So when evaluating the similarity of two AMR graphs for the sentence, the first step is to obtain an alignment between them. Smatch obtains this alignment by maximizing the F1 of the concept and relation triples between the two graphs by initializing a random alignment between them and then iteratively revising the alignment by maximizing the Smatch score through hill climbing. The hill-climbing approach to alignment can en at local optima and is also inefficient.

In this paper, we report AnCast[1], a Meaning Representation (MR) graph evaluation metric that implements an efficient alignment algorithm based on the idea that we can align two graphs by identifying *initial alignment anchors* in a pair of graphs. Initial anchors are pairs of concepts, one from each graph, that can be aligned with a high level of confidence either because the two concepts match and they are also unique within each graph, or if they are aligned to the same word token(s) in a sentence. We can determine the alignment for the rest of the concepts in the pair of graphs by observing that concepts with aligned anchors as neighbors also have a higher probability of being aligned. We can thus align two graphs by first identifying initial anchors and then iteratively propagating the alignment to their neighbors through a process called *anchor broadcast*.

The rest of the papers is organized as follows. In Section 2, we provide a detailed description of our alignment method and the evaluation metrics. We present experimental results in Section 3 to show that when evaluating AMR parsers, AnCast produces results that are highly correlated with Smatch scores but in a more efficient manner and does not get trapped in local maxima. We discuss related work in Section 4 and draw our conclusions in Section 5.

2. Method

The basic goal of an evaluation metric for MR graphs is to quantify the similarity between two graphs. We believe that a good evaluation metric for MR graphs needs to be interpretable, efficient, and light-weight. As we cannot assume that the concepts between the two graphs are aligned, the first step when designing an MR metric is to perform concept alignment between pairs of graphs.

AnCast performs alignment through an iterative process by first computing an *intrinsic similarity matrix* between two graphs based on the intrinsic properties of the nodes and an *initial anchor matrix* based on concept pairs that can be aligned with high confidence. This initial anchor matrix will

[1] https://github.com/sxndqc/ancast

be iterative updated through *anchor broadcast* until no more concepts can be aligned. Finally, *suboptimal alignment* is performed on the remaining unmatched concepts.

The output of the alignment process is a list of *aligned concepts* between the two graphs. Using the list of aligned of concepts, we compute *concept F1*, *unlabeled relation F1*, *labeled relation F1*, and *weighted relation F1*. We also compute a mock Smatch score to evaluate the correlation between the AnCast score and the Smatch score.

2.1. Intrinsic Similarity and Initial Anchor Matrices

The intrinsic similarity matrix and initial anchor matrix are computed solely based on the intrinsic properties of a node. The intrinsic properties of a node include the lemma, the sense of the lemma, and the attributes of a concept. Comparing the intrinsic properties of the nodes between two MR graphs results in an *intrinsic similarity matrix* S, a $|V^{(T)}| \times |V^{(G)}|$ matrix, where $V^{(T)}$ denotes the set of all nodes (vertices) of the test graph, and $V^{(G)}$ denotes nodes of the gold graph.

The similarity S_{ij} between a specific pair of nodes $(V_i^{(T)}, V_j^{(G)})$ is computed as

$$S_{ij} = \frac{S_{ij}^{(l)}(1 + \gamma(S_{ij}^{(s)} - 1)) + S_{ij}^{(a)}}{1 + \mathbb{1}(|R^{(i,j)}|)} \quad (1)$$

where $S_{ij}^{(l)}$ is the string similarity of the concept lemmas of $V_i^{(T)}$ and $V_j^{(G)}$, $S_{ij}^{(s)}$ represents whether the senses of two nodes match and $S_{ij}^{(a)}$ represents the average attribute similarity between two nodes. γ is the sense coefficient used to determine the level of importance of sense matching to the intrinsic similarity, set empirically at 0.1.

$S_{ij}^{(l)}$ measures the similarity between the lemmas of the two nodes and is equal to $sim(\text{LEMMA}(V_i^{(T)}), \text{LEMMA}(V_j^{(G)}))$, where the similarity function sim is defined in Equation 2:

$$sim(u_s, u_l) = \begin{cases} |u_s|/|u_l|, & u_s \subseteq u_l \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

If the two strings have an inclusion relationship, the similarity is the ratio of the shorter string u_s to the longer string u_l; if they are equal, it is 1. In any other case, the similarity is 0.

$S_{ij}^{(s)}$ represents whether the sense of two nodes matches, and is equal to $\mathbb{1}(\text{SENSE}(V_i^{(T)}) = \text{SENSE}(V_j^{(G)}))$, which evaluates 1 when they are same, and 0 otherwise.

$S_{ij}^{(a)}$ measures the average attribute similarity between two nodes, and is calculated using Equation 3:

$$S_{ij}^{(a)} = \sum_r \frac{\mathbb{1}(D_T^{(i,j)}(r) = D_G^{(i,j)}(r))}{|R^{(i,j)}|} \quad (3)$$

where $D_T^{(i,j)}$ and $D_G^{(i,j)}$ are attribute value dictionaries for $V_i^{(T)}$ and $V_j^{(G)}$, and $R^{(i,j)}$ represents the subset of attributes in both dictionaries, as specified in Equation 4. We stipulate that when $|R^{(i,j)}| = 0$, $S_{ij}^{(a)} = 0$.

$$R^{(i,j)} = \{r | r \in D_T^{(i,j)}(keys) \cap D_G^{(i,j)}(keys)\} \quad (4)$$

(4) illustrates the result of the intrinsic similarity calculation for a toy example. $S_{ij}^{(l)}$ is 0.375, $S_{ij}^{(s)}$ is 0, $S_{ij}^{(a)}$ is 0.5 because they have two shared attributes, but only one attribute has identical values. The final intrinsic similarity S_{ij} for this pair of AMR nodes is 0.41875.

(4)
(f / fry-03	(s / stir-fry-01
:quant 5	:quant 7
:polarity -	:polarity -)
:mode imperative)	

Initial Anchor Matrix Initial anchor pairs refer to pairs of concepts from two graphs that are aligned by the algorithm based solely on the lemma of the concepts. There are two methods to find such anchor pairs. If manual alignment between an MR graph and the word tokens in a sentence exist, the alignment between a pair of concepts can be inferred from this alignment by noting that a pair of concepts aligned to the same word tokens can be considered to be an initial anchor pair.

If an MR graph are not manually aligned to word tokens in a sentence, which is the case for the AMR annotation, the alternative is to extract a subset of highly similar pairs from a pair of graphs for the same sentence. For instance, if a pair of nodes (i) represent a concrete concept, (ii) have the same lemma, and (iii) are unique in their respective graphs, we can safely assume that they form an initial anchor pair. Note that the sense does not have to be the same as it does not affect the uniqueness of a concept. In other words, if the sense IDs of a concept pair are not the same, they can still be initial anchors. The pairs of initial anchors that have such properties will be assigned 1 in the initial anchor matrix $A^{(0,0)}$, and the rest of the concept pairs are assigned 0.

The initial anchor matrix will be updated in the iterative anchor broadcast step, while the intrinsic similarity matrix will only be computed once and will *not* be updated once it is computed.

2.2. Iterative Anchor Broadcast

Anchor broadcast is an $O(n^3)$ algorithm that iteratively propagates contextual information on a

graph. This algorithm consists of an outer loop and an inner loop, followed by a *finalization* phase. The inner loop broadcasts an anchor matrix by combining an *adjacency matrix* that encodes the shared contextual information of a pair of nodes and the intrinsic similarity matrix and returns a list of aligned concept pairs and an updated anchor matrix. The outer loop iteratively expands the list of aligned concepts through multiple iterations of the inner loop, and assigns a matching quality level to the concept alignment based on the order in which the aligned concepts are identified. The finalization phase conducts suboptimal matching for the remaining nodes and identifies unalignable nodes.

2.2.1. The Inner loop

Computing structural similarity of a node pair The inner loop is a revised SimRank algorithm (Jeh and Widom, 2002) for bipartite semantic graphs. Here, we use this algorithm to capture the intuition that a pair of nodes has a higher similarity if they have more anchor pairs in their neighborhood. We consider 1-hop and 2-hop neighbors of the i^{th} node of the test graph $V^{(T)}$ and the j^{th} node of the gold graph $V^{(G)}$, namely parent and grandparent nodes (ancestors) and child and grandchild nodes (descendants), denoted as N_P and N_C respectively. Edge labels between a parent and a child are not taken into consideration when identifying neighbors. We include indirect neighbors at distances of more than one because there are cases where the nodes are still similar even though their immediate neighbors are not due to errors in annotation or parsing. Based on the neighborhood similarity, we compute an *adjacency matrix* in which each cell represents the similarity between a pairs of nodes based on the similarity of their neighborhood.

When computing the neighborhood similarity for a pair of nodes, we consider ancestors separately from descendants. We compute two matrices, P for ancestors and C for descendants, where the similarity between a pair of nodes is calculated as the sum of the anchor values of all ancestor (descendant) node pairs as in Equations 5-7, where $A^{(q,t)}$ means the anchor matrix in the process of being propagated at the t^{th} iteration of the inner loop and the q^{th} iterationn of the outer loop. ΔN_{ij} represents the ratio of the number of neighbors (parent side or child side) between nodes $V_i^{(T)}$ and $V_j^{(G)}$ (greater than or equal to 1) as a coefficient for correcting false high similarities caused by mismatched numbers of neighbors. For example, if one node has 1 neighbor and the other has 5 neighbors, they should at most have 1 matching neighbor. However, if both nodes have 3 neighbors, then this correction coefficient is not needed. The parent, child, and adjacency matricies are computed as follows:

$$P_{ij}^{(t+1)} = \frac{1}{\Delta N_{Pij}} \sum_{k_t}^{N_P(V_i^{(T)})} \sum_{k_g}^{N_P(V_j^{(G)})} A_{k_t k_g}^{(q,t)} \quad (5)$$

$$C_{ij}^{(t+1)} = \frac{1}{\Delta N_{Cij}} \sum_{k_t}^{N_C(V_i^{(T)})} \sum_{k_g}^{N_C(V_j^{(G)})} A_{k_t k_g}^{(q,t)} \quad (6)$$

$$A_{ij}^{(q,t+1)} = \sqrt{(P_{ij}^{(t+1)} + 1)(C_{ij}^{(t+1)} + 1) - 1} \quad (7)$$

In Equation 5, k_g and k_t are the indices of nodes extracted from neighborhoods. The parent and child sides of the neighbor information are aggregated as the square root of the product, because the number of neighboring anchors makes a significant difference between 0 (no anchor in neighborhood) and 1 (one anchor pair in the neighborhood), but its significance decays with additional neighbors. The significance of having anchors on both sides is also very different from having anchors on only one side, so we use square root function to flatten the curve when the number of anchors is relatively large, but steepen the slope when it is within the range of 0-1. The broadcasted anchor is then regularized as in Equation 8. $||A^{(q,t+1)}||_\infty$ is the maximum of the matrix, which is also the infinity norm. Finally, all the anchor points in $A^{(q,t+1)}$ are reset to 1 to reinforce the initial anchor information as in Equation 9, where $A^{(q,0)}$ is an anchor matrix with values in $0, 1$ and the initial anchors for the q^{th} round in the outer loop.

$$\bar{A}^{(q,t+1)} = \frac{A^{(q,t+1)}}{||A^{(q,t+1)}||_\infty} \quad (8)$$

$$\bar{A}_{ij}^{(q,t+1)} = 1 \quad \text{if} \quad A_{ij}^{(q,0)} = 1 \quad (9)$$

The equations 5-7 can be computed in the form of matrix operations, where J is all-ones matrix, ΔN_P and ΔN_C are matrix neighbor number ratios, and Y, Z are respectively the combination of first and second order adjacency matrices on the parent side and the child side.

$$P^{(t+1)} = \frac{1}{\Delta N_P} Y_{(test)}^T A^{(q,t)} Y_{(gold)} \quad (10)$$

$$C^{(t+1)} = \frac{1}{\Delta N_C} Z_{(test)}^T A^{(q,t)} Z_{(gold)} \quad (11)$$

$$A^{(q,t+1)} = \sqrt{(P^{(t+1)} + 1)(C^{(t+1)} + 1) - J} \quad (12)$$

Aggregating structural information and intrinsic information At the end of broadcast will there will be a converged adjacency matrix with values between 0 and 1 denoted as $A^{(q,t_c)}$, as proved in (Jeh and Widom, 2002). The convergence is defined as every element of $A^{(q,t_c)}$ being within a range of ϵ to $A^{(q,t_c-1)}$, where ϵ is set to $1e-4$. This broadcasted anchor is multiplied with the intrinsic similarity to obtain an *adjusted similarity matrix* $F^{(q)}$ as in Equation 13.

$$F_{ij}^{(q)} = (S_{ij} + \alpha)(\bar{A}_{ij}^{(q,t_c)} + \beta) \quad (13)$$

The adjusted similarity matrix can then be used to update the anchor matrix. Two biases are introduced in Equation 13. The bias α is used to allow the matching between some pairs of nodes that have different abstract concepts but are in a similar structural configuration, indicating that the two abstract concepts might be semantically related. β is an adjustment factor for nodes that are far from anchors. If a node is far from an anchored node, it may not receive much propagated structural information and its structural similarity may still be near 0. As this is a very rare situation, β is very small compared to α. We define α to be 0.2 and β to be 0.01 in our metric. To avoid round-off errors of floats, the values in adjusted similarity matrix are rounded to the same precision of the convergence precision t, which is set to $1e{-}4$ in our setting.

2.2.2. The Outer loop

The updated anchor matrix $A^{(q+1,0)}$ is generated by setting $A_{ij}^{(q+1,0)}$ to 1 where $F_{ij}^{(q)}$ is the maximal value of both its rows and columns, which means the two nodes are mutually their best match under the combination of intrinsic and neighborhood information. If a node in graph A has multiple nodes in graph B bearing the same maximum similarity value, then these nodes in B with the same similarity will be, as the final step, compared by adding the similarity of edge labels from the nodes that are connected to them, denoted by $\tilde{F}_{ij}^{(q)}$. If there is a single maximum among these rival nodes, the one with the maximum value will be chosen as the best match; if there are more than one maximum, then that node and those nodes in the opposing graph corresponding to these maximum values will be compared based on the label sets of the relations around them. If the sets of relation labels are also the same (which is a rare scenario but may occur), then these nodes will not be matched in the current round and will be compared in the next round till more of their neighbors have been matched. Such approach ensures that nodes will not be missed or misaligned, as any node can only be defined by four properties: name, attribute set, neighbors, and the set of labels on edges connected to it. Two nodes are necessarily identical if they have share all these four properties.

The outer loop terminates if no such mutual best match can be found or the new anchor matrix would be the same as the previous anchor matrix as all such pairs are already anchors. Note that the intrinsic similarity matrix S is not updated and we do not use $F^{(q-1)}$ from the last outer loop in place of S, because $F^{(q-1)}$ already contains the results of the last anchor broadcast. On one hand, it is not appropriate to multiply two pieces of repetitive structural information. On the other hand, given the limited propagation distance of known anchors in a single round, some pairs of nodes that are far from known anchors might not have their neighborhood similarities accurately determined yet. Such nodes might possess high intrinsic similarity and should be matched, but if outdated anchor information is used, it might interfere with the calculation of their true similarity. Therefore, only the most recent anchor broadcast information should be used in each iteration.

$$A_{ij}^{(q+1,0)} = \begin{cases} 1 & \tilde{F}_{ij}^{(q)} > \max(\tilde{F}_{kj}^{(q)}, \tilde{F}_{il}^{(q)}), \forall k, l \neq i, j \\ 0 & \text{otherwise} \end{cases} \quad (14)$$

The outer loop typically consists of 2-3 inner loops and exits when no new anchor pair is added in the last inner loop, as illustrated in Algorithm 9. Each pair $(V_{i_k}^{(T)}, V_{j_k}^{(G)})$ in the matching set M is also marked with the round number q. q increases by 1 in each round as an indicator of quality level. A higher value of q usually implies a lower degree of confidence that the two nodes are similar, either because they have different lemmas or they have slightly different structural contexts, usually caused by differences in reentrancy.

Algorithm 1: Anchor and Broadcast

Data: Initial anchor matrix $A^{(0,0)}$
Result: List of matched pairs M

repeat
  repeat
    Broadcast anchor $A^{(q,t)}$ using semantic bipartite SimRank;
    Regularize $A^{(q,t)}$ and reset all previous anchor position to 1;
  until *The $A^{q,t}$ has converged*;
  Combine $A^{(q,t_c)}$ with intrinsic similarity S and add newly matched pairs into M and return a new anchor $A^{(q+1,0)}$;
until *The new $A^{(q+1,0)}$ is the same as the previous anchor matrix $A^{(q,0)}$*;
Compute suboptimal matches for remaining nodes;
Link all the redundant nodes to null;

2.2.3. Finalization Phase

The finalization phase deals with nodes that cannot find a mutual best match. Generally, such nodes are situated in areas with significant annotation inconsistencies or parsing errors. Nonetheless, we can still find signals in concept lemmas or their structural context that allow us to match them by lowering the matching criteria. As illustrated in Algorithm 9, based on the adjusted similarity matrix from the final round $F^{(q_c)}$, we first remove from the matrix all matched nodes with positive q mark because nodes are only allowed in 1-1

matches. We can thus acquire a submatrix $\tilde{F}^{(q_c)}$, from which we repeatedly retrieve the global maximal value $\tilde{F}^{(q_c)}_{xy}$, and remove the newly matched nodes $(V^{(T)}_x, V^{(G)}_y)$, until all the rows or columns are removed. If there are still multiple maximum values in the residual matrix, the nodes at these maximum value positions will be compared with respect to the similarities of the sets of relation labels connected to them, and the pair with the most overlapping relation labels will be selected as the best match. If there is still a tie, then the matching pair will be selected among these maximums of overlapped relation labels by the numbering order in the matrix.

The final step of the finalization phase sets unmatched nodes to *null*. Only one graph will have such nodes as all the nodes of the smaller graph will be matched in previous step.

Algorithm 2: Suboptimal Greedy Match

```
Let M be the set of already matched pairs in the outer loop;
for (i, j) ∈ M do
    Remove row i and column j from F^(q_c) ;
F̃^(q_c) is the reduced matrix from F^(q_c) ;
while rows and columns remain in F̃^(q_c) do
    (x, y) ← argmax(F̃^(q_c)) ;
    Remove row x and column y from F̃^(q_c) ;
for remaining row / column index z of F̃^(q_c) do
    z ← null
```

An example of broadcast process of (5) is illustrated as in Figure 2

(5) She is reading my book in the house.

```
(r1 / read-01                (r1 / read-03
 :location (h2 / house)       :location (h2 / home)
 :ARG0 (h / he)               :ARG0 (s / she)
 :ARG1 (b / book)             :ARG1 (p / paper)
        :poss (i / i))               :part (i / i))
```

2.3. Metrics

With a list of aligned concepts produced by the anchor broadcast algorithm, we can now calculate a number of MR evaluation metrics, measuring how two MR graphs are similar from different perspectives. We first provide a mock Smatch score to simulate the calculation of Smatch, a metric that measures the similarity of concepts and relations independently, as we have discussed in Section 1. We argue that the Smatch score does not provide the most intuitive results, but we want to provide this metric to evaluate how well the anchor broadcast algorithm correlates with the hill climbing algorithm implemented in Smatch.

In addition, we propose a number of metrics that should be fairly intuitive to the NLP community. The first one is *Concept F1*, which is a score that measures how well the two MR graphs overlap in terms of their nodes. The second one is *Labeled Relation F1*, which measures the degree to which relation triples from the two graphs match. The Labeled Relation F1 assumes that the concepts from the triple as well as their parent-child dependency would have to match as well, in addition to the relation (the edge label) itself, and it is thus a more rigorous metric than Smatch. We also calculate *Unlabeled Relation F1*, which is calculated similarly to Labeled Relation F1 but does not require that the edge label itself match. Finally, we provide a Weighted Relation F1, which attempts to capture the intuition that the matches for some relations are more important than others. Below we explain how each of these metrics are applied in detail.

2.3.1. Mock SMatch Score

The essence of the SMatch metric is that it evaluates concepts and relations independently of each other. It consists of four parts: (i) root match, whether the roots of the two graphs are a match, denoted as $\langle TOP\ \ TOP\ \ var \rangle$; (ii) instance match, whether the matched variables correspond to the exact same lemma and sense, and there is no match, for example, between $\langle r\ instance\ read-03 \rangle$ and $\langle r\ instance\ read-01 \rangle$; (iii) relation match, where $\langle var\ rel\ var \rangle$ are compared as a whole, requiring the relation labels to be completely the same and the two pairs of variables, one for the parents and one for the children, to be both matched as well; (iv) attribute match, which requires an exact match for attributes such as $\langle a7, quant, 2.5 \rangle$. Since we also have list of aligned nodes, we can simulate the calculation of the SMatch score. We will show in our experiments that our mock Smatch score has an extremely high correlation with the actual Smatch score, with a Pearson correlation of over 0.97.

2.3.2. Our metrics

Concept F1 Concept F1 measures the degree to which the two graphs have similar concepts. We first approach this from the perspective of one graph, examining the intrinsic similarity of every node in the graph to its matched counterpart. If there is no match, the similarity $S(v, M(v))$ is 0 if $v \notin V$, with M being the list of aligned concepts. This is calculated for both the test and the gold graphs, and the F score is derived from the two Γ's, to deal with the situation where the test and gold graphs are different in size.

$$\Gamma' = \frac{1}{|V|} \sum_{v}^{V} S(v, M(v)) \tag{15}$$

$$\Gamma = \text{F-score}(\Gamma'_T, \Gamma'_G) \tag{16}$$

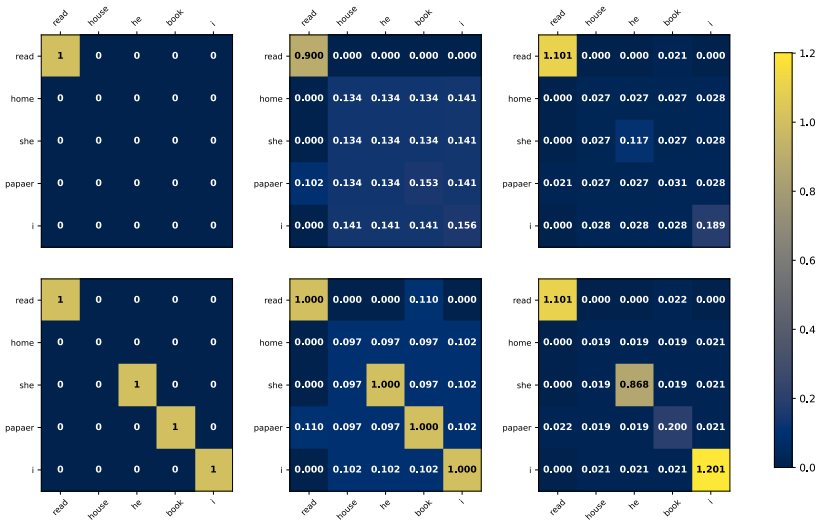


Figure 2: The anchor, broadcasted anchor and adjusted similarity matrix for (5). The initial anchor only contains $read - read$, as i is not the same form as my in the text. In the first round of outer loop, the newly established anchor are $paper - book$, $i - i$ and $he - she$, but the $home - house$ relation is not established as their best choices are not each other. Neither can the second round link $home - house$ because they are still not each other's best match, so they are suboptimally matched after removing all current anchor pairs.

Labeled Relation F1 This is an overall score that measures the similarity for both concepts and relations, defined as *the F score of average relation overlap*. We do not add an extra node "TOP" because the root node does not affect the semantics of an AMR graph.

Consider a parent-child tuple v_1 and v_2 on an MR graph. If both v_1 and v_2 have aligned concepts in the graph it is measured against, we will have a mirror parent-child tuple (w_1, w_2) from the other graph. First we calculate how similar these two tuples are by calculating s_c in Equation 17, which is the average of intrinsic similarity of the two tuples. If there is no such mirror tuple, s_c is 0. The tuple represents a parent-child dependency and is thus directional, so relation tuples like $\langle read-03\ ARG0\ book\rangle$ and $\langle book\ \ ARG0\ \ read-03\rangle$ are considered to be totally different.

$$s_c = (S_{v_1 w_1} + S_{v_2 w_2})/2 \tag{17}$$

In an MR representation, there may be one or more relations between this parent-child pair p, whose relation labels compose a set denoted as L_p. Let s_{ol} denote how many labels in L_p that are also in (w_1, w_2), their counterpart's label set L'. Intuitively we should assign a higher weight to parent-child pairs with multiple relations. We thus multiply s_c by the size of the set of overlapping relations s_{ol} to calculate the relation F1 s_p for this parent-child pair (v_1, v_2).

$$s_{ol} = \sum_{r \in L_p} \mathbb{1}(r \in L'_p) \tag{18}$$

$$s_p = s_c s_{ol} \tag{19}$$

The total labeled relation F1 Ψ is computed as in Equation 20-22, where P is the set of all parent-child concept pairs in one graph, p denotes one particular parent-child pair and T is the set of all triples, as the weight of a parent-child pair with multiple triples is already encoded in its s_p. and an F-score of the two Ψ' is calculated as well.

$$|T| = \sum_{p}^{P} |L_p| \tag{20}$$

$$\Psi' = \frac{1}{|T|} \sum_{p}^{P} s_p \tag{21}$$

$$\Psi = \text{F-score}(\Psi'_{test}, \Psi'_{gold}) \tag{22}$$

Unlabeled Relation F1 Sometimes we only need to check the structural similarity between two graphs without being concerned with specific relation labels. We thus set all s_{ol} to their maximal possible value $\min(|L_p|, |L'_p|$, and the unlabeled relation F1 for one parent-child pair is calculated as in Equation 23, with the rest of the metric calculated the same way as its labeled counterpart by substituting s_p with s^u_p.

$$s^u_p = s_c \cdot \min(|L_p|, |L'_p|) \tag{23}$$

Weighted Relation F1 It is debated whether there is "core semantics" in our perception of MR graphs (**?**Cai and Lam, 2019a), which claims that nodes closer to the root or having more children are semantically more important than those that are further away from the root or have fewer children. We provide this metric for further study. The weight of each parent-child pair w_p is determined by the two numbers of parent and child's descendants, namely their arguments, grand-arguments and so on, respectively d_1 and d_2, with attribute nodes having no descendants, and we have:

$$w_p = \sqrt{d_1 d_2} + 1 \tag{24}$$

This gives higher weight to relations that are higher in the concept hierarchy in the MR graph, and guarantees that every relation at least has a weight of 1. Subsequently, we have:

$$s_p^w = w_p s_p \tag{25}$$

$$\Psi'_W = \frac{\sum_p^P s_p^w}{\sum_p^P w_p |L_p|} \tag{26}$$

3. Experiment and Discussion

We selected five well-maintained parsers with available checkpoints trained on the AMR 3.0 dataset from the top 13 parsers on LDC2020T02 benchmark. *IBM transition-based* parser(Drozdov et al., 2022) parser is evaluated with AMR3-structbart-L checkpoint; *BiBL* (Cheng et al., 2022) is evaluated with the AMR3.0 model on its github page; *ATP-AMR* (Chen et al., 2022)'s model is ATP_SRL_AMR3.0_Ensemble; for *AMRBART* (Bai et al., 2022) we use the output for AMRBART-large (AMR3.0), and for *SPRING* (Bevilacqua et al., 2021) we use its AMR3.parsing-1.0 model.

Table 1 shows the results. The Smatch score is inflated compared to Labeled Relation F1 as it also includes the concept F1 score and it does not consider concept match in comparing relation triples. Unlabeled Relation F1 can be about 5% higher than LR F1, which indicates that it's not rare that concepts are correctly attached but labels are wrong. Weighted Relation F1's are generally lower than LR Macro F1, meaning that the errors may happen equally to all levels of nodes. In terms of speed, our running time is only about 40% the that of Smatch but yields the same result. SemBleu (Song and Gildea, 2019) is significantly faster, but it sacrifices interpretability and fails to parse the result from ATP-AMR and SPRING because it cannot accept late specification where the variable is declared first and specified afterwards, where it is marked with "X" in Table 1.

We are interested in answering two main questions with the experiments. First, can our metrics replace Smatch by having a high correlation with the Smatch score? Second, can our new metric overcome the shortcomings of Smatch by uncovering more meaningful discrepancies between MR graphs?

Similarities with Smatch score Regarding the first question, for each of the five parsers, our mock Smatch has a very high Pearson correlation of over 0.97 with the Smatch score among the 1898 AMRs in the AMR3.0 test set, indicating that our evaluation method yields a score that is equivalent to Smatch but with higher efficiency.

Difference with Smatch score The second question leads us to examine the change in ranking between *BiBL* and *SPRING* based on our own metrics. An interesting observation in parsing the 37^{th} sentence from AMR3.0-Lorelei test set is that the Smatch score for *BiBL*'s parsing result is higher than our Mock Smatch score (0.71 v. 0.64), but

BiBL

```
(z0 / contrast-01
 :ARG1 (z1 / take-01
  :polarity -
  :ARG0 (z2 / we)
  :ARG1 (z3 / man))
 :ARG2 (z4 / prepare-02
  :polarity -
  :ARG1 z2
  :ARG2 (z5 / issue-02
   :ARG0 (z6 / or
    :op1 (z7 / boyfriend)
    :op2 (z8 / thing
     :ARG1-of (z9 / resemble-01
      :ARG2 z7)))
   :ARG1 (z10 / security))))
 :concession-of (z11 / want-01
  :ARG0 z2
  :ARG1 (z12 / assure-01
   :ARG0 z2
   :ARG1 (z13 / privacy
    :beneficiary (z14 / woman
     :mod (z15 / other))))
  :mod (z16 / main)))
```

Spring

```
(z0 / contrast-01
  :ARG1 (z1 / take-01
    :polarity -
    :ARG0 (z2 / we)
    :ARG1 (z3 / man))
  :ARG2 (z4 / want-01
    :ARG0 z2
    :ARG1 (z5 / assure-01
  :ARG0 z2
      :ARG1 (z6 / privacy
        :poss (z7 / woman
          :mod (z8 / other))))
    :mod (z9 / main))
  :ARG1-of (z10 / cause-01
    :ARG0 (z11 / prepare-02
      :polarity -
      :ARG1 z2
      :ARG2 (z12 / or
        :op1 (z13 / issue-02
          :ARG0 (z14 / security)
          :ARG1 (z15 / person
            :ARG0-of (z16 / have-rel-role-91
              :ARG1 z2
              :ARG2 (z17 / boyfriend))))
        :op2 (z18 / et-cetera)))))
```

Gold

```
(c / contrast-01
  :ARG1 (a / and
    :op1 (t / take-in-06 :polarity -
      :ARG0 (w / we)
      :ARG1 (m / man))
    :op2 (p / prepare-02
      :ARG1 w
      :ARG2 (i2 / issue-02
        :ARG0 (a2 / and
          :op1 (p2 / person
            :ARG0-of (h / have-rel-role-91
              :ARG2 (b / boyfriend)))
          :op2 (p3 / person
            :ARG1-of (r / resemble-01
              :ARG2 p2)))
        :topic (s / security))))
  :ARG2 (w2 / want-01
    :ARG0 w
    :ARG1 (a3 / assure-01
      :ARG0 w
      :ARG2 (p4 / privacy
        :beneficiary (w3 / woman
          :mod (o / other))))
    :mod (m2 / main)))
```

Figure 3: Parsing results of BiBL and SPRING, and the standard gold graph.

ours is higher than Smatch in *spring*'s parsing result (0.65 v. 0.63). The two parsing results and the gold graph are shown in Figure 3.

The discrepancy stems from different approaches to variable alignment. Both parsers yield poor results, and our metric prioritizes the matching concepts with the same lemma, while Smatch tries to get the highest possible f-score. In this case, the two parsing results are different in only one alignment.

In *BiBL*, Ancast aligns the concept "boyfriend" to the concept with the same lemma in another graph, while Smatch aligns "boyfriend" in the test graph to "person" in the gold graph, which has a more similar structural environment. This makes BiBL's Smatch score higher than Mock Smatch.

For *spring*, the only difference is the matching of a hallucinated "cause-01" in the test graph. Our metric aligns it with "and" in the gold graph probably because they share a common descendant in "prepare". Smatch aligns this hallucinated concept to "resemble" which is unmatched in our metric, leaving the concepts of "and" unmatched. This shows that Smatch is stuck in a local optima here and failed to acquire the maximum F-1 score with 4 random starts.

This case study proves that even though AnCast never prioritizes a maximum relational F1 score, it can sometimes do so as a side effect of our alignment algorithm.

Using Multiple F1's Label Relation F1 is about 10% lower than Smatch score, and it ranks the five parsers identically to Smatch as macro F1. In AMR 3.0 dataset, the Macro F1 is computed by averaging the scores of 1898 sentences, and Micro F1 is calculated by averaging all triple-pair scores across the whole dataset. Shorter sentences thus have higher weights in Macro F1. From the experiment results, we can see that BiBL and Transition-based parsers might perform worse in parsing shorter sentences than the other three. ATP-AMR has a slightly better concept score than AMRBART, but these concepts are worsely connected as it has lower unlabeled relation F1.

(%)	Transition	BiBL	AMRBART	ATP-AMR	SPRING
Smatch (4 random starts)	81.15	83.83	86.17	85.80	84.55
Mock Smatch	79.59	82.63	85.01	84.57	83.23
Pearson Correlation	97.34	98.36	98.21	97.93	98.07
Sembleu	64.66	71.00	72.43	X	X
Labeled Relation Micro F1	70.14	76.61	76.16	75.99	74.76
Labeled Relation Macro F1	70.00	75.28	79.02	78.50	76.39
Concept F1	90.29	90.13	92.67	92.83	91.54
Unlabeled Relation F1	74.83	78.51	82.27	81.73	79.57
Weighted Relation F1	70.10	73.85	77.59	76.91	74.69
Runtime of Smatch	36.56s	33.97s	33.41s	37.32s	31.95s
Runtime of Sembleu	0.08s	0.08s	0.08s	X	X
Runtime of A & B	14.33s	15.79s	15.63s	15.68s	15.63s

Table 1: Comparison of different methods on 1898 AMRs in AMR 3.0 dataset. All non-micro F1's are macro F1, including Smatch, Mock Smatch and Sembleu. Sembleu is incapable of parsing graphs with early re-entrancies.

Robustness Test Bamboo is a benchmark proposed in (Opitz et al., 2021) to evaluate the robustness of AMR parsing metrics. This benchmark can be used to assess the degree to which the scores given by an AMR evaluation metric between a pair of AMRs correlate with human judgment, and whether the metric is robust to meaning preserving and meaning altering transformations of AMRs.

As shown in Table 2, our metric achieved an extremely high correction with human judgment on the role label confusion task. This is because in this task the role labels on the AMR graphs have been altered and pairs of AMR graphs that are similar are no longer similar after the transformation. Such label alternations barely affect our concept matching approach to alignment, but other metrics are more affected by such changes as changes in role labels lead to changes in alignment.

		Main			Reify		
Metric	STS	SICK	PARA	STS	SICK	PARA	amean
Smatch	58.45	59.72	41.25	57.98	61.81	39.66	51.28
SemBleu(k=2)	60.62	59.86	36.88	57.68	59.64	36.24	48.87
WWLK$^{\Theta}$	66.94	67.64	37.91	64.34	65.49	39.23	54.90
Ancast	56.46	57.17	36.09	55.31	58.67	36.09	59.95
		SynoS			Args		
Metric	STS	SICK	PARA	STS	SICK	PARA	hmean
Smatch	56.14	57.39	39.58	48.05	70.53	24.75	47.50
SemBleu(k=2)	57.34	56.18	33.26	44.54	67.54	16.60	42.13
WWLK$^{\Theta}$	60.11	62.29	35.15	55.03	75.06	29.64	50.26
Ancast	53.65	54.70	33.52	**91.14**	**98.32**	**88.22**	53.34

Table 2: Robustness tests on the Bamboo benchmark.

4. Related Work

Previous works on semantic graph evaluation can be divided into alignment and alignment-free algorithms. The former group, which includes Smatch (Cai and Knight, 2013) and Smatch++ (Opitz, 2023), and provides interpretable metrics. The latter group, which includes SemBleu (Song and Gildea, 2019), WWLK(Opitz et al., 2021), are usually faster, but they lack in interpretability.

Smatch is a simple and light-weight evaluation metric that measures how many of the relation and concept triples overlap under a fast enough variable alignment algorithm. However, it does not separate the evaluation of concepts from that of relations, making it hard to perform detailed discrepancy analysis. In addition, its variable alignment method uses hill climbing to approximate the non-polynomial time graph matching, sometimes resulting in non-optimal alignment. Later work like (Damonte et al., 2017) proposed pre-processing techniques such as removing wikification or relation labels before using Smatch for a more comprehensive evaluation, but its speed is still slow. Smatch++ trieds to optimize Smatch by reducing the search space via identifying points that can be aligned with high confidence and applying an ILP solver but cannot guarantee its run-time efficiency. SEMA (Anchiêta et al., 2019) tries to use concept names but the comparison algorithm is too simple and may miss many potential matches.

SemBleu (Song and Gildea, 2019) foregoes variable alignment and instead approximates MR graph evaluation by transforming and concatenating relations into bigrams and trigrams, to computes the Bleu Score on the resulting N-graphs. While this method is indeed fast, it lacks granularity and interpretability. WWLK(WWLK-Θ) (Opitz et al., 2021) also uses neighbor information by adding word embeddings of directly neighboring nodes to that of the node under evaluation. It relies on word vectors that are unavailable when evaluating low-resource language annotations, and the parameter learning for relation labels is potentially biased by the selection of human annotators.

5. Conclusion

We present AnCast, a meaning representation evaluation tool that implements intuitive metrics that include concept F1, unlabeled and labeled relation F1, as well as weighted relation F1. AnCast is also efficient in that it implements an anchor broadcast algorithm that has a polynomial runtime. This represents a significant improvement over Smatch, the most widely used MR metric. The alignment algorithm also has the nice property of not being subject to the trappings of local maxima in its search for optimal alignment.

Acknowledgements

This work is supported by grants from the CNS Division of National Science Foundation (Awards no: NSF_2213804) entitled “Building a Broad Infrastructure for Uniform Meaning Representations”. Any opinions, findings, conclusions or recommendations expressed in this material do not necessarily reflect the views of NSF. We also wish to extend our appreciation to Cloudbank, which provided an indispensable computational resource for our experiments.

Limitations

The proposed anchor and broadcast algorithm is applicable to semantic graphs that have at least some nodes that can serve as anchors and these are nodes that can be matched with high confidence. The algorithm will not work on semantic graphs that do not have reliable anchors as a starting point. The proposed evaluation metrics are for sentence-level semantic graphs such as Abstract Meaning Representation or the sentence-level Uniform Meaning Representation, but they have to be extended in order to be used to evaluate document-level meaning representation graphs.

6. Bibliographical References

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

Xuefeng Bai, Yulong Chen, and Yue Zhang. 2022. Graph pre-training for AMR parsing and generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6001–6015, Dublin, Ireland. 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. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th linguistic annotation workshop and interoperability with discourse*, pages 178–186.

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. 2019a. 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.

Deng Cai and Wai Lam. 2019b. 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.

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.

Liang Chen, Peiyi Wang, Runxin Xu, Tianyu Liu, Zhifang Sui, and Baobao Chang. 2022. Atp: Amrize then parse! enhancing amr parsing with pseudoamrs. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 2482–2496.

Ziming Cheng, Zuchao Li, and Hai Zhao. 2022. Bibl: Amr parsing and generation with bidirectional bayesian learning. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 5461–5475.

Ann Copestake, Dan Flickinger, Carl Pollard, and Ivan A Sag. 2005a. Minimal recursion semantics: An introduction. *Research on language and computation*, 3:281–332.

Ann Copestake, Dan Flickinger, Carl Pollard, and Ivan A Sag. 2005b. Minimal recursion semantics: An introduction. *Research on language and computation*, 3:281–332.

Marco Damonte, Shay B Cohen, and Giorgio Satta. 2017. An incremental parser for abstract meaning representation. In *15th EACL 2017 Software Demonstrations*, pages 536–546. Association for Computational Linguistics (ACL).

Donald Davidson. 1969. The individuation of events. In *Essays in honor of Carl G. Hempel: A tribute on the occasion of his sixty-fifth birthday*, pages 216–234. Springer.

Andrew Drozdov, Jiawei Zhou, Radu Florian, Andrew McCallum, Tahira Naseem, Yoon Kim, and Ramon Fernandez Astudillo. 2022. Inducing and using alignments for transition-based amr parsing. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Glen Jeh and Jennifer Widom. 2002. Simrank: a measure of structural-context similarity. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 538–543.

Kevin Knight, Bianca Badarau, Laura Baranescu, Claire Bonial, Madalina Bardocz, Kira Griffitt, Ulf Hermjakob, Daniel Marcu, Martha Palmer, Tim O'Gorman, et al. 2021. Abstract meaning representation (amr) annotation release 3.0.

Jiangming Liu, Shay B Cohen, and Mirella Lapata. 2020. Dscorer: A fast evaluation metric for discourse representation structure parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4547–4554.

Chi-kiu Lo and Dekai Wu. 2011. Meant: An inexpensive, high-accuracy, semi-automatic metric for evaluating translation utility based on semantic roles. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 220–229.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Jan Hajic, Christopher D Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, and Daniel Zeman. 2020. Universal dependencies v2: An evergrowing multilingual treebank collection. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4034–4043.

Juri Opitz. 2023. Smatch++: Standardized and extended evaluation of semantic graphs. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 1550–1562.

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. 2022. 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.

Kishore Papineni, Salim Roukos, Todd Ward, and W"BLEU Zhu. 2001. A method for automatic evaluation of machine translation". *the Proceedings of ACL-2002, ACL, Philadelphia, PA, July 2002*.

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.

Jingxuan Tu, Timothy Obiso, Bingyang Ye, Kyeongmin Rim, Keer Xu, Liulu Yue, Windisch Susan Brown, Martha Palmer, and James Pustejovsky. 2024. Glamr: Augmenting amr with gl-verbnet event structure. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation*.

Jens EL Van Gysel, Meagan Vigus, Jayeol Chun, Kenneth Lai, Sarah Moeller, Jiarui Yao, Tim O' Gorman, Andrew Cowell, William Croft, Chu-Ren Huang, et al. 2021. Designing a uniform meaning representation for natural language processing. *KI-Künstliche Intelligenz*, 35(3-4):343–360.

Chuan Wang and Nianwen Xue. 2017. Getting the most out of amr parsing. In *Proceedings of the 2017 conference on empirical methods in natural language processing*, pages 1257–1268.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

Pengcheng Yin and Graham Neubig. 2018. Tranx: A transition-based neural abstract syntax parser for semantic parsing and code generation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (Demo Track)*.