SEMA: an Extended Semantic Evaluation Metric for AMR

Rafael T. Anchiêta[1], Marco A. S. Cabezudo[1], and Thiago A. S. Pardo[1]

Interinstitutional Center for Computational Linguistics (NILC)
Institute of Mathematical and Computer Sciences, University of São Paulo, Brazil
rta@usp.br, msobrevillac@usp.br, taspardo@icmc.usp.br

Abstract. Abstract Meaning Representation (AMR) is a recently designed semantic representation language intended to capture the meaning of a sentence, which may be represented as a single-rooted directed acyclic graph with labeled nodes and edges. The automatic evaluation of this structure plays an important role in the development of better systems, as well as for semantic annotation. Despite there is one available metric, *smatch*, it has some drawbacks. For instance, *smatch* creates a self-relation on the root of the graph, has weights for different error types, and does not take into account the dependence of the elements in the AMR structure. With these drawbacks, *smatch* masks several problems of the AMR parsers and distorts the evaluation of the AMRs. In view of this, in this paper, we introduce an extended metric to evaluate AMR parsers, which deals with the drawbacks of the *smatch* metric. Finally, we compare both metrics, using four well-known AMR parsers, and we argue that our metric is more refined, robust, fairer, and faster than *smatch*.

Keywords: Abstract Meaning Representation · Semantic Metric · Evaluation.

1 Introduction

Abstract Meaning Representation (AMR) is a semantic representation language designed to capture the meaning of a whole sentence [4]. AMR got the attention of the scientific community due to its relatively simpler structure, showing the relations among concepts and making them easy to read. The creation of AMR language was motivated by the need of providing to the research community corpora with annotations related to traditional tasks of Natural Language Processing (NLP), such as named entity recognition, semantic role labeling, word sense disambiguation, and coreference resolution [4]. Moreover, AMR structures are arguably easier to produce than traditional formal meaning representations [5].

In this way, several annotated corpora arose, for English[1], Chinese [12], Spanish [15], and Portuguese [3]. Consequently, a considerable number of semantic parsers emerged [9,7,16,2,13], and, with the available parsers, some applications

[1] https://amr.isi.edu/download.html

were developed and/or improved: automatic summarization [10], text generation [18], paraphrase detection [11], and others.

Given the growing interest in AMR language, the automatic evaluation of AMR structures plays a very important role for the AMR parsing task, as well as for semantic annotation tasks, which create linguistic resources for semantic parsing. Although there is one metric to automatically evaluate AMR structures, named *smatch* [6], it has some shortcomings:

1. *Smatch* does not take into account the dependence of the elements in the AMR structure, i.e., its analysis is very simple, masking several analysis problems. So, *smatch* often gives higher scores for AMRs that have different meanings in relation to the reference AMR.
2. *Smatch* creates a self-relation called *TOP* for the root of the AMR structure. That is, *smatch* gives more weight for the root of the graph than other elements, distorting the analysis.
3. *Smatch* has weights for different error types. As discussed by Damonte et al. [7], three named entity errors are considered more important than six wrong labels. Nevertheless, it is difficult to conclude which task should have a higher weight.

Smatch metric computes the degree of overlapping between two AMR structures. To evaluate an AMR generated by a parser against a reference manually produced AMR, *smatch* defines `M` the correct number of triples, `C` the produced number of triples by a parser, and `T` the total number of triples in reference AMR. So, precision and recall are calculated according to Eq. 1 and 2, respectively.

$$P = \frac{M}{C} \qquad (1) \qquad\qquad R = \frac{M}{T} \qquad (2)$$

For example, when evaluating the AMR graph in Fig. 2 against the AMR in Fig. 1, *smatch* returns `M` equal to four (`disaster`, `describe-01`, `man`, and `mission`), `C` equal to eight (`disaster`, `describe-01`, `man`, `mission`, `TOP`, `ARG0`, `ARG1`, and `ARG2`), and `T` equal to eight. So, precision and recall are equal to $4/8 = 0.5$.

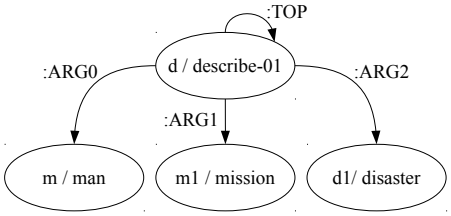

Fig. 1: Reference AMR

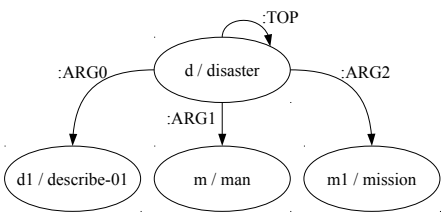

Fig. 2: Test AMR

As we may see, *smatch* adds a *TOP* relation in the structure. This self-relation is not provided by AMR language and it distorts the analysis because if a parser correctly identifies the root of the graph, *smatch* will compute the root concept and the *TOP* relation as correct, otherwise, it will compute only the root concept as correct. In addition, *smatch* is not considering the dependence of the elements. The other issues will be detailed later.

Thereby, we believe that assessing the dependence in which the elements are arranged in the AMR structure may help to better understand the semantic analyzers potentialities and limitations and to produce better applications.

Given these shortcomings and inspired by Damonte et al. [7] to better understand the limitations of AMR parsers and to find their strong points, we propose a new metric for evaluating AMR parsers, named SEMA (Semantic Evaluation Metric for AMR). Our metric deals with these issues of the *smatch* metric, presenting a new way to evaluate concepts and relations in AMR structures, computing precision, recall, and f-score values between two AMRs. Moreover, we compare *smatch* and SEMA, using four well-known AMR parsers in order to analyze the differences between the metrics and, finally, we discuss the obtained results.

In what follows, Sect. 2 presents the essential related work. In Sect. 3, we introduce the AMR fundamentals. Sect. 4 details our developed metric. In Sect. 5, we compare *smatch* and SEMA and, finally, Sect. 6 concludes the paper.

2 Related Work

Compared to traditional meaning representations, AMR is a relatively new representation, as well as AMR parsing is a new task. Thus, there are few works involving semantic representation measurements.

Allen et al. [1] adopted a logical form representation for evaluating its semantic representation. The authors proposed a metric that computes the maximum score by any alignment among logical form graphs. This representation needs an alignment between the input sentences and the semantic analysis. However, the authors did not address how to determine the alignments.

Dridan and Oepen [8] directly evaluated a semantic parser output by comparing semantic sub-structures. The authors also adopted a logical form representation for evaluating its semantic representation. For that, the authors required an alignment between sentence spans and semantic sub-structures. One limitation of that metric is the need for an alignment between the input sentences and their semantic analyses.

Cai and Knight [6] developed a metric named *smatch* that calculates the degree of overlap between two AMR structures. The metric computes the maximum f-score obtainable via one-to-one matching of variables between two AMRs.

As the *smatch* metric, our metric is also focused on AMR structures. However, our metric is more robust, because it deals with the several drawbacks that *smatch* has, as the dependence of elements (nodes, edges), the self-relation

created on the root of the graph, and the weights generated for different error types.

3 AMR Essentials

Abstract Meaning Representation (AMR) is a semantic representation language designed to capture the meaning of a sentence, abstracting away from elements of the surface syntactic structure, such as morphosyntactic information and word ordering [4]. Hence, words that do not significantly contribute to the meaning of a sentence are left out of the annotation.

AMR focuses on the predicate-argument structure of a sentence, as defined by the PropBank resource [17]. It may be represented as a single-rooted directed acyclic graph with labeled nodes (concepts) and edges (relations) among them. Nodes represent the main events and entities mentioned in a sentence, and edges represent semantic relationships among nodes. AMR concepts are either words in their lexicalized forms (e.g., `boy`, `girl`), PropBank framesets (`want-01`, `adjust-01`), or special keywords such as `date-entity`, `distance-entity`, `government-organization`, and others. PropBank framesets are essentially verbs linked to lists of possible arguments and their semantic roles. In Fig. 3, we show a PropBank frameset example. The frameset `edge.01`, which represents the "move slightly" sense, has six arguments (Arg 0 to 5).

Frameset **edge.01** "move slightly"	
Arg0: causer of motion	Arg3: start point
Arg1: thing in motion	Arg4: end point
Arg2: distance moved	Arg5: direction
Ex: [$_{\text{Arg0}}$Revenue] *edge* [$_{\text{Arg5}}$ up] [$_{\text{Arg2-EXT}}$ 3.4%] [$_{\text{Arg4}}$ to $904 million] [$_{\text{Arg3}}$ from $874 million] [$_{\text{ArgM-TMP}}$ in last year's third quarter]. (wsj_1210)	

Fig. 3: A PropBank frameset [17]

For semantic relationships, in addition to PropBank semantic roles, AMR adopts approximately 100 additional relations. We list some of them below. For more details, we suggest consulting the original paper [4].

General semantic relations: :mod, :manner, :location, :name, :polarity
Relations for quantity: :quant, :unit, :scale
Relations for date-entity: :day, :month, :year, :weekday, :dayperiod
Relations for list: :op1, :op2, :op3, and so on

In addition to the graph structure, AMR may be represented in two different notations: traditionally, in first-order logic; or in the PENMAN notation [14], for easier human reading and writing. For instance, Table 1 presents sentences

with similar senses, which are represented in the canonical form in PENMAN format and in the corresponding graph notation, in Figs. 4 and 5, respectively.

Table 1: Sentences with similar meaning

Sentences
The girl made adjustment to the machine.
The girl adjusted the machine.
The machine was adjusted by the girl.

```
(a / adjust-01
      :ARG0 (g / girl)
      :ARG1 (m / machine))
```

Fig. 4: PENMAN notation

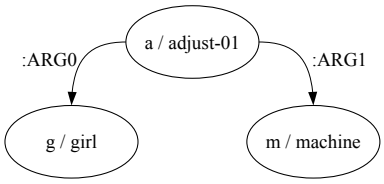

Fig. 5: Graph notation

As it is possible to see, AMR assigns the same representation to sentences with the same basic meaning. In the example, the concepts are `adjust-01`, `girl`, and `machine` and the relations are `:ARG0` and `:ARG1`, represented by labeled directed edges in the graph. In Figs. 4 and 5, the symbols "a", "g", and "m" are variables and may be re-used in the annotation, corresponding to reentrancies (multiple incoming edges) in the graph.

4 SEMA Metric

Following Cai and Knight [6], semantic relationships encoded in the AMR graph may also be viewed as a conjunction of logical propositions, or triples. For example, suppose that the sentence "Tolerance is certainly not fear, and sincerity does not have to be cowardice." produces triples according to Fig. 6 and its graph notation in Fig. 7.

Each AMR triple takes one of these forms: *relation (variable, concept)*, *relation (variable1, variable2)* or *relation (variable, constant)*. The first form encompasses the first seven triples, the second the six triples then, and the third the last two triples in Fig. 6.

Assuming a second AMR annotation for the same sentence, according to Fig. 8 and graphically in Fig. 9, we may compare the two structures considering, for instance, that one is produced by a parser and must be compared to the other one, which would be a reference AMR.

Our metric computes precision, recall, and f-score, evaluating the test triples against the reference triples, analyzing the root of the graphs and, then, relations and concepts, similar to a Breadth-First Search (BFS), taking into account its dependence.

instance (a, and) ^
instance (b, fear) ^
instance (c, certain) ^
instance (d, tolerance) ^
instance (e, obligate-01) ^
instance (f, cowardice) ^
instance (g, sincerity) ^
op1 (a, b)
op2 (a, e)
manner (b, c)
domain (b, d)
ARG2 (e, f)
domain (f, g)
polarity (b, '-')
polarity (e, '-')

Fig. 6: Reference triples

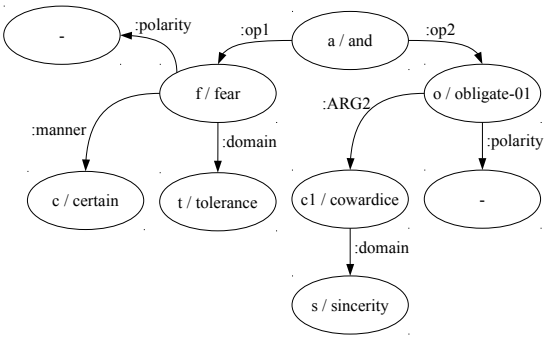

Fig. 7: Graph notation for reference triples

instance (a, and) ^
instance (b, fear-01) ^
instance (c, tolerate-01) ^
instance (d, certain) ^
instance (e, obligate-01) ^
instance (f, cowardice) ^
instance (g, sincerity) ^
op1 (a, b)
op2 (a, e)
ARG0 (b, c)
mod (b, d)
ARG2 (e, f)
ARG1 (e, g)
polarity (b, '-')
polarity (e, '-')

Fig. 8: Test triples

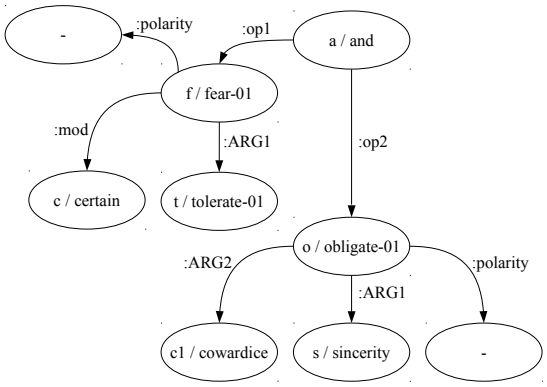

Fig. 9: Graph notation for the test triples

First, our metric analyzes if the root of the test graph (`and`) belongs to the reference graph, that is `and`. We may verify that the two concepts are equal. Thus, the metric computes the concept (`and`) as correct (`M`), one produced element `and` (`C`), and one reference element `and` (`T`). Table 2 presents the root analysis by SEMA.

Table 2: Root analysis

Reference graph	Test graph	M	C	T
and	and	and	and	and

Continuing the evaluation, considering the neighbor relations of the root, our metric analyzes if the relations `:op1` and `:op2` of the test graph and their parent, which is the root of the graph, belong to the reference graph.

Although the two relations are present in reference graph, our metric correctly identifies only the `:op2` relation, as the relation `:op1`, in test graph, is connected to the concept `fear-01` that is different from the reference graph that is `fear`. In Table 3, we show the relations analysis.

Table 3: Relations analysis neighbor to the root

Reference graph	Test graph	M	C	T
`:op1, :op2`	`:op1, :op2`	`:op2`	`:op1, :op2`	`:op1, :op2`

After analyzing the relations, our metric analyzes the neighbor concepts of the root, that is, it verifies if the concepts `fear-01`, and `obligate-02` of the test graph and their parent, which is the root of the graph, belong to the reference graph.

Table 4: Concepts analysis

Reference graph	Test graph	M	C	T
`fear, obligate-01`	`fear-01, obligate-01`	`obligate-01`	`fear-01, obligate-01`	`fear, obligate-01`

As one may see, the concept `obligate-01` is correct and the concept `fear-01` is wrong, since the correct concept is `fear`. So, the metric computes correctly one element `fear`, shown in Table 4.

In the same manner, our metric will calculate the remaining relations and concepts. At the end of the evaluation, our metric returns six correct triples {*instance (a, and), instance (e, obligate-01), op2 (a, e), instance (f, cowardice), ARG2 (e, f), polarity (e, '-')*} and both test and reference AMRs produced fifteen triples. So, precision, recall, and f-score are equal to $6/15 = 0.40$, respectively.

Analyzing the previous example, the *smatch* metric returns as precision, recall, and f-score values equal to 0.69 for each measure. *Smatch* considers as correct the triples {*instance (a, and), instance (e, obligate-01), instance (d, certain), instance (g, sincerity), instance (f, cowardice), op1 (a, b), op2 (a, e) ARG2 (e, f), polarity (b, '-'), polarity (e, '-'), TOP (a, 'and')*} . The metric tries to maximize the f-score, so, it does not evaluate the dependence of the elements

in the AMR structure. Besides that, the *smatch* scores the root `and` and its self-relation `:TOP`, distorting the analysis[2] (see Fig. 10).

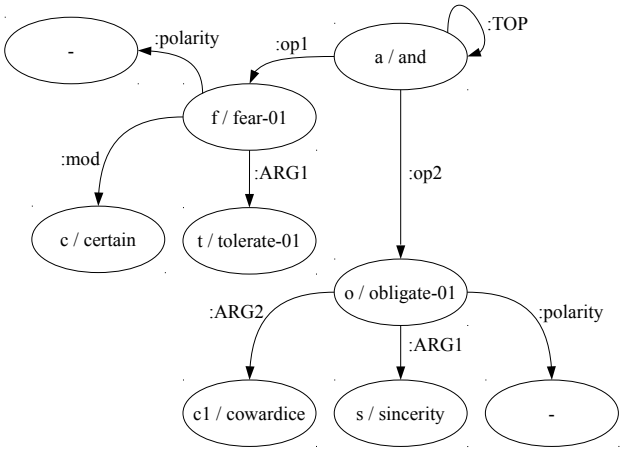

Fig. 10: AMR considered by *smatch* metric

In contrast to the *smatch* metric, our metric considers the dependence of the elements arranged on a graph, i.e., the metric evaluates the relations/concepts and their parents. Furthermore, our metric does not create a `:TOP` relation at the root of the graph, not distorting the evaluation and making the analysis fairer than *smatch* metric. More than that, our metric produces a deterministic result since it works as a Breadth-first search where in the worst-case the performance is $O(|V| + |E|)$, which is faster than to compute the maximum score via one-to-one matching of variable, as the *smatch* metric.

In addition to the above shortcomings, Damonte and Cohen [7] detected that *smatch* has weights for different error types. For example, considering two parses for the sentence "Silvio Berlusconi gave Lucio Stanca his current role of modernizing Italy's bureaucracy", in Fig. 11.

At the left, the output of a parser (*Parser 1*) is not able to deal with named entities. At the right, in the output of other parser (*Parser 2*), except for `:name`, `:op`, and `:wiki` the relation label `:ARG0` is always used. The *smatch* scores for two parses are 0.56 and 0.78 for f-score, respectively. Despite both parses make obvious mistakes, three named entity errors in *Parse 1* are considered more important than six wrong labels in *Parse 2*, according to Damonte et al. [7]. SEMA metric solves that issue by assigning equal weights to all relations, making the evaluation more robust than *smatch*.

[2] The result may be confirmed at `https://amr.isi.edu/eval/smatch/compare.html`. We also checked the available source code `https://github.com/snowblink14/smatch`

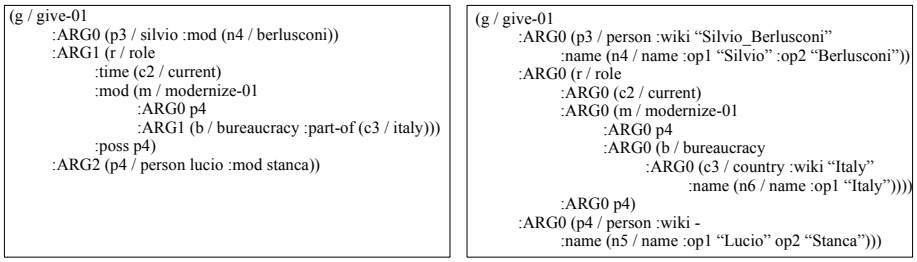

Fig. 11: Sentence "Silvio Berlusconi gave Lucio Stanca his current role of modernizing Italy's bureaucracy" parsed by two parsers [7]

By analyzing AMRs according to SEMA, we may measure precision, recall, and f-score for instance and relation identification tasks, and thus, understand better the AMR parsing task due to a more fine-grained analysis. A demo version and the source code of our metric is available at `https://github.com/rafaelanchieta/sema`. In what follows, we compared our metric with *smatch* using four well-known AMR parsers.

5 Evaluation

In order to compare our metric with *Smatch*, we chose four AMR parsers for English: JAMR parser [9], AMREager parser [7], Neural AMR Parser [16], and AMR Graph Prediction Parser [13]. These parsers were chosen because they handle the parsing task differently and they are publicly available.

We focused on two datasets: LDC2015E86 (R1), which consists of 16,833, 1,368, and 1,371 sentences in training, development, and testing sets, respectively, and LDC2016E25 (R2), which contains 36,521 training sentences, and the same sentences for development and testing as R1. Table 5 shows the comparison between the SEMA and *smatch* metrics on the test set.

Table 5: Comparison between SEMA and Smatch metrics on the test set

Parser	**Train. Data**	**SEMA**			**Smatch**		
		P	R	F	P	R	F
JAMR	R1	0.61	0.57	0.59	0.70	0.64	0.67
AMREAger	R1	0.59	0.54	0.56	0.67	0.62	0.64
Neural AMR	R2	0.67	0.59	0.63	0.76	0.67	0.71
AMR Graph P.	R2	0.67	0.64	0.66	0.75	0.72	0.74

As shown in Table 5, our metric is stricter than *smatch* metric. In order to understand these values and how the metrics deal with graphs of different sizes, we carried out a detailed evaluation.

We calculated the average number of relations in the test set and found that each sentence has 19.8 relations on average. Thus, we organized the test set into two sets: those sentences with number of relations below the average (799 sentences) and those with number of relations above the average (572 sentences) and compared the SEMA and *smatch* metrics. Tables 6 and 7 present the results.

As shown in Tables 6 and 7, in both configurations *smatch* values were superior to SEMA values. This is due to two main factors:

1. The distorted analysis of the relation *TOP*;
2. A large number of concepts and relations not properly evaluated by *smatch*.

In the first factor, in 44.75% of the number of relations below the average and in 77.5% of the number of relations above the average, the parsers did not correctly produce the root of the graph, and, even so, *smatch* considered the roots as correct because the concepts were present in the graph.

Table 6: For number of relation below the average

Parser	Train. Data	SEMA			Smatch		
		P	R	F	P	R	F
JAMR	R1	0.61	0.55	0.58	0.71	0.65	0.68
AMREAger	R1	0.59	0.53	0.56	0.69	0.63	0.66
Neural AMR	R2	0.66	0.62	0.64	0.76	0.72	0.74
AMR Graph P.	R2	0.66	0.64	0.65	0.75	0.73	0.74

Table 7: For Number of relations above the average

Parser	Train. Data	SEMA			Smatch		
		P	R	F	P	R	F
JAMR	R1	0.62	0.58	0.60	0.69	0.64	0.66
AMREAger	R1	0.58	0.54	0.56	0.66	0.61	0.63
Neural AMR	R2	0.68	0.57	0.62	0.74	0.63	0.68
AMR Graph P.	R2	0.68	0.65	0.67	0.75	0.72	0.73

For the second factor, consider the sentence "How long are we going to tolerate Japan?", which was manually annotated as in Fig. 12. The AMR graph has six relations and seven concepts (11 triples). For the same sentence, an AMR parser generated the AMR graph in Fig. 13, which has ten relations and concepts (17 triples).

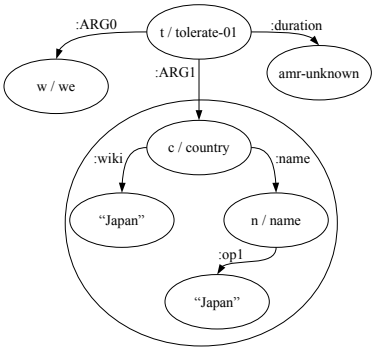

Fig. 12: Reference AMR graph

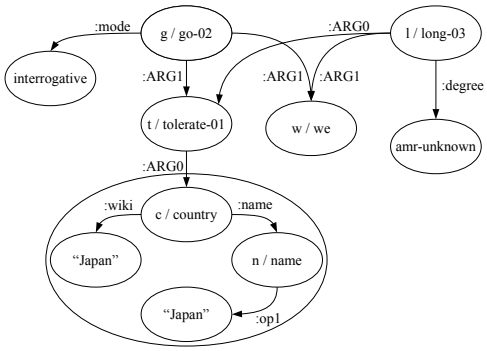

Fig. 13: AMR graph generated by a parser

We may see that the AMR parser produced a subgraph similar to a subgraph that was manually annotated. Despite there are other concepts in the AMR graph produced by the parser that are present in reference AMR graph, as: `tolerate-01`, `we` and `amr-unknown`, their dependents and/or relations are wrong. Hence, the SEMA metric considers these concepts as wrong. For instance, the concept `tolerate-01`, in the reference AMR graph, is the root of the graph, whereas, in the AMR produced by the parser, the root is the concept `go-02`. The root `go-02` is connected to the concept `tolerate-01` through the `:ARG1` relation. Finally, the concept `tolerate-01` in both graphs is connected to the concept `country` but by different relations: `:ARG0` and `:ARG1` for the AMR generated by the parser and reference AMR graph, respectively.

Due to these distinctions, our metric evaluates the connection with the subgraph as wrong since its relation is different from the reference AMR graph. On the other hand, the *smatch* metric evaluates as correct the concepts `we` and `amr-unknown`, although they are not connected to the concept `tolerate-01`. Thus, the *smatch* returns 0.44, 0.67, and 0.53, while the SEMA returns 0.29, 0.45, and 0.36, for precision, recall, and f-score, respectively.

Even though our metric is stricter than *smatch* metric, we believe that SEMA is fairer and more robust than *smatch*. As AMR parsing task is on the semantic level, the dependence of the elements in AMR structure should be analyzed. More than that, SEMA metrics neither creates a TOP self-relation on the root of the graph nor assigns weights for different error types, not distorting the analysis. In the way *smatch* is currently computed, several parsing problems are overlooked.

6 Final Remarks

In this paper, we presented a new metric for evaluating AMR structures. This metric analyzes the dependence in which the elements are arranged in the AMR structure and deals with other shortcomings of the *smatch* metric, as a self-relation produced on the root of the graph, which distorts the analysis, and weights for different error types. We compared our metric with the *smatch* metric, using four AMR parser and showed that, in general, our metric is stricter than *smatch* metric. However, we believe that our metric is fairer and robust than *smatch* since several parsing problems are being overlooked by *smatch*. In addition, we also showed that for both small and large graphs, the parsers have difficulty in learning the dependence of the elements, and even so, *smatch* considers as correct several elements.

As future work, we intend to investigate how to adapt our metric to other semantic representations.

Acknowledgments

The authors are grateful to FAPESP and IFPI for supporting this work.

References

1. Allen, J.F., Swift, M., De Beaumont, W.: Deep semantic analysis of text. In: Proceedings of the 2008 Conference on Semantics in Text Processing. pp. 343–354 (2008)
2. Anchiêta, R.T., Pardo, T.A.S.: A rule-based amr parser for portuguese. In: Simari, G.R., Fermé, E., Gutiérrez Segura, F., Rodríguez Melquiades, J.A. (eds.) Advances in Artificial Intelligence - IBERAMIA 2018. pp. 341–353 (2018)
3. Anchiêta, R.T., Pardo, T.A.S.: Towards amr-br: A sembank for brazilian portuguese. In: Proceedings of the 11th International Conference on Language Resources and Evaluation. pp. 974–979 (2018)
4. Banarescu, L., Bonial, C., Cai, S., Georgescu, M., Griffitt, K., Hermjakob, U., Knight, K., Palmer, M., Schneider, N.: Abstract meaning representation for sembanking. In: Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse. pp. 178–186 (2013)
5. Bos, J.: Expressive power of abstract meaning representations. Computational Linguistics **42**, 527–535 (2016)
6. Cai, S., Knight, K.: 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). pp. 748–752 (2013)
7. Damonte, M., Cohen, S.B., Satta, G.: An incremental parser for abstract meaning representation. In: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers. pp. 536–546 (2017)
8. Dridan, R., Oepen, S.: Parser evaluation using elementary dependency matching. In: Proceedings of the 12th International Conference on Parsing Technologies. pp. 225–230 (2011)
9. Flanigan, J., Thomson, S., Carbonell, J.G., Dyer, C., Smith, N.A.: A discriminative graph-based parser for the abstract meaning representation. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics. pp. 1426–1436 (2014)
10. Hardy, H., Vlachos, A.: Guided neural language generation for abstractive summarization using abstract meaning representation. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. pp. 768–773 (2018)
11. Issa, F., Damonte, M., Cohen, S.B., Yan, X., Chang, Y.: Abstract meaning representation for paraphrase detection. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers). vol. 1, pp. 442–452 (2018)
12. Li, B., Wen, Y., Weiguang, Q., Bu, L., Xue, N.: Annotating the little prince with chinese amrs. In: Proceedings of the 10th Linguistic Annotation Workshop held in conjunction with ACL 2016. pp. 7–15 (2016)
13. Lyu, C., Titov, I.: Amr parsing as graph prediction with latent alignment. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 397–407 (2018)
14. Matthiessen, C., Bateman, J.A.: Text generation and systemic-functional linguistics: experiences from English and Japanese. Pinter Publishers (1991)
15. Migueles-Abraira, N., Agerri, R., de Ilarraza, A.D.: Annotating Abstract Meaning Representations for Spanish. In: Proceedings of the 11th International Conference on Language Resources and Evaluation. pp. 3074–3078 (2018)

16. van Noord, R., Bos, J.: Neural semantic parsing by character-based translation: Experiments with abstract meaning representations. Computational Linguistics in the Netherlands Journal **7**, 93–108 (2017)
17. Palmer, M., Gildea, D., Kingsbury, P.: The proposition bank: An annotated corpus of semantic roles. Computational linguistics **31**(1), 71–106 (2005)
18. Song, L., Zhang, Y., Wang, Z., Gildea, D.: A graph-to-sequence model for amr-to-text generation. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 1616–1626 (2018)