Smatch: an Evaluation Metric for Semantic Feature Structures

Shu Cai
USC Information Sciences Institute
4676 Admiralty Way, Suite 1001
Marina del Rey, CA 90292
shucai@isi.edu

Kevin Knight
USC Information Sciences Institute
4676 Admiralty Way, Suite 1001
Marina del Rey, CA 90292
knight@isi.edu

Abstract

The evaluation of whole-sentence semantic structures plays an important role in semantic parsing and large-scale semantic structure annotation. However, there is no widely-used metric to evaluate whole-sentence semantic structures. In this paper, we present *smatch*, a metric that calculates the degree of overlap between two semantic feature structures. We give an efficient algorithm to compute the metric and show the results of an inter-annotator agreement study.

1 Introduction

The goal of semantic parsing is to generate all semantic relationships in a text. Its output is often represented by whole-sentence semantic structures. Evaluating such structures is necessary for semantic parsing tasks, as well as semantic annotation tasks which create linguistic resources for semantic parsing.

However, there is no widely-used evaluation method for whole-sentence semantic structures. Current whole-sentence semantic parsing is mainly evaluated in two ways: 1. task correctness (Tang and Mooney, 2001), which evaluates on an NLP task that uses the parsing results; 2. whole-sentence accuracy (Zettlemoyer and Collins, 2005), which counts the number of sentences parsed completely correctly.

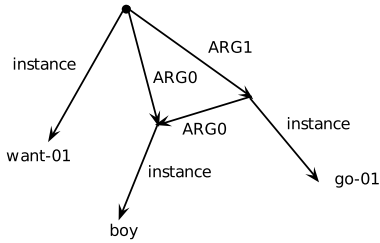
Nevertheless, it is worthwhile to explore evaluation methods that use scores which range from 0 to 1 ("partial credit") to measure whole-sentence semantic structures. By using such methods, we are able to differentiate between two similar whole-sentence semantic structures regardless of specific

tasks or domains. In this work, we provide an evaluation metric that uses the degree of overlap between two whole-sentence semantic structures as the partial credit.

In this paper, we observe that the difficulty of computing the degree of overlap between two whole-sentence semantic feature structures comes from determining an optimal variable alignment between them, and further prove that finding such alignment is NP-complete. We investigate how to compute this metric and provide several practical and replicable computing methods by using Integer Linear Programming (ILP) and hill-climbing method. We show that our metric can be used for measuring the annotator agreement in large-scale linguistic annotation, and evaluating semantic parsing.

2 Semantic Overlap

We work on a semantic feature structure representation in a standard neo-Davidsonian (Davidson, 1969; Parsons, 1990) framework. For example, semantics of the sentence "the boy wants to go" is represented by the following directed graph:



In this graph, there are three concepts: want-01, boy, and go-01. Both want-01 and go-01 are frames from PropBank framesets (Kingsbury and Palmer, 2002). The frame want-01 has two arguments connected with ARG0 and ARG1, and go-01 has an argument (which is also the same boy instance) connected with ARG0.

Following (Langkilde and Knight, 1998) and (Langkilde-Geary, 2002), we refer to this semantic representation as AMR (Abstract Meaning Representation).

Semantic relationships encoded in the AMR graph can also be viewed as a conjunction of logical propositions, or triples:

```
instance(a, want-01)  ∧
instance(b, boy)  ∧
instance(c, go-01)  ∧
ARG0(a, b)  ∧
ARG1(a, c)  ∧
ARG0(c, b)
```

Each AMR triple takes one of these forms: *relation*(*variable*, *concept*) (the first three triples above), or *relation*(*variable1*, *variable2*) (the last three triples above).

Suppose we take a second AMR (for "the boy wants the football") and its associated propositional triples:

```
instance(x, want-01)  ∧
instance(y, boy)  ∧
instance(z, football)  ∧
ARG0(x, y)  ∧
ARG1(x, z)
```

Our evaluation metric measures precision, recall, and f-score of the triples in the second AMR against the triples in the first AMR, i.e., the amount of propositional overlap.

The difficulty is that variable names are not shared between the two AMRs, so there are multiple ways to compute the propositional overlap based on different variable mappings. We therefore define the *smatch* score (for semantic match) as the *maximum f-score obtainable via a one-to-one matching of variables between the two AMRs*.

In the example above, there are six ways to match up variables between the two AMRs:

	M	P	R	F
x=a, y=b, z=c:	4	4/5	4/6	0.73
x=a, y=c, z=b:	1	1/5	1/6	0.18
x=b, y=a, z=c:	0	0/5	0/6	0.00
x=b, y=c, z=a:	0	0/5	0/6	0.00
x=c, y=a, z=b:	0	0/5	0/6	0.00
x=c, y=b, z=a:	2	2/5	2/6	0.36
smatch score:				0.73

Here, `M` is the number of propositional triples that agree given a variable mapping, `P` is the precision

of the second AMR against the first, R is its recall, and F is its f-score. The smatch score is the maximum of the f-scores.

However, for AMRs that contain large number of variables, it is not efficient to get the f-score by simply using the method above. Exhaustively enumerating all variable mappings requires computing the f-score for $n!/(n-m)!$ variable mappings (assuming one AMR has n variables and the other has m variables, and $m \leq n$). This algorithm is too slow for all but the shortest AMR pairs.

3 Computing the Metric

This section describes how to compute the smatch score. As input, we are given AMR1 (with m variables) and AMR2 (with n variables). Without loss of generality, $m \leq n$.

Baseline. Our baseline first matches variables that share concepts. For example, it would match *a* in the first AMR example with *x* in the second AMR example of Section 2, because both are instances of *want-01*. If there are two or more variables to choose from, we pick the first available one. The rest of the variables are mapped randomly.

ILP method. We can get an optimal solution using integer linear programming (ILP). We create two types of variables:

- (Variable mapping) v_{ij} = 1 iff the ith variable in AMR1 is mapped to the jth variable in AMR2 (otherwise v_{ij} = 0)
- (Triple match) t_{kl} = 1 iff AMR1 triple k matches AMR2 triple l, otherwise t_{kl} = 0. A triple $relation1(xy)$ matches $relation2(wz)$ iff relation1 = relation2, v_{xw} = 1, and v_{yz} = 1 or y and z are the same concept.

Our constraints ensure a one-to-one mapping of variables, and they ensure that the chosen t values are consistent with the chosen v values:

$$\text{For all } i, \quad \sum_j v_{ij} \leq 1$$

$$\text{For all } j, \quad \sum_i v_{ij} \leq 1$$

For all triple pairs $r(xy)r(wz)$ (r for relation),

$$t_{r(xy)r(wz)} \leq v_{xw}$$

$$t_{r(xy)r(wz)} \leq v_{yz}$$

when y and z are variables.

Finally, we ask the ILP solver to maximize:

$$\sum_{kl} t_{kl}$$

which denotes the maximum number of matching triples which lead to the smatch score.

Hill-climbing method. Finally, we develop a portable heuristic algorithm that does not require an ILP solver[1]. This method works in a greedy style. We begin with m random one-to-one mappings between the m variables of AMR1 and the n variables of AMR2. Each variable mapping is a pair $(i, map(i))$ with $1 \leq i \leq m$ and $1 \leq map(i) \leq n$. We refer to the m mappings as a variable mapping state.

We first generate a random initial variable mapping state, compute its triple match number, then hill-climb via two types of small changes:

1. Move one of the m mappings to a currently-unmapped variable from the n.

2. Swap two of the m mappings.

Any variable mapping state has $m(n-m) + m(m-1) = m(n-1)$ neighbors during the hill-climbing search. We greedily choose the best neighbor, repeating until no neighbor improves the number of triple matches.

We experiment with two modifications to the greedy search: (1) executing multiple random restarts to avoid local optima, and (2) using our Baseline concept matching ("smart initialization") instead of random initialization.

NP-completeness. There is unlikely to be an exact polynomial-time algorithm for computing smatch. We can reduce the 0-1 Maximum Quadratic Assignment Problem (0-1-Max-QAP) (Nagarajan and Sviridenko, 2009) and the subgraph isomorphism problem directly to the full smatch problem on graphs.[2]

We note that other widely-used metrics, such as TER (Snover et al., 2006), are also NP-complete. Fortunately, the next section shows that the smatch methods above are efficient and effective.

[1]The tool can be downloaded at http://amr.isi.edu/evaluation.html.

[2]Thanks to David Chiang for observing the subgraph isomorphism reduction.

4 Using Smatch

We report an AMR inter-annotator agreement study using smatch.

1. Our study has 4 annotators (A, B, C, D), who then converge on a consensus annotation E. We thus have 10 pairs of annotations: A-B, A-C, . . . , D-E.

2. The study is carried out 5 times. Each time annotators build AMRs for 4 sentences from the Wall Street Journal corpus. Sentence lengths range from 12 to 54 words, and AMRs range from 6 to 29 variables.

3. We use 7 smatch calculation methods in our experiments:

   - Base: Baseline matching method
   - ILP: Integer Linear Programming
   - R: Hill-climbing with random initialization
   - 10R: Hill-climbing with random initialization plus 9 random restarts
   - S: Hill-climbing with smart initialization
   - S+4R: Hill-climbing with smart initialization plus 4 random restarts
   - S+9R: Hill-climbing with smart initialization plus 9 random restarts

Table 1 shows smatch scores provided by the methods. Columns labeled 1-5 indicate sentence groups. Each individual smatch score is a document-level score of 4 AMR pairs.[3] ILP scores are optimal, so lower scores (in bold) indicate search errors.

Table 2 summarizes search accuracy as a percentage of smatch scores that equal that of ILP. Results show that the restarts are essential for hill-climbing, and that 9 restarts are sufficient to obtain good quality. The table also shows total runtimes over 200 AMR pairs (10 annotator pairs, 5 sentence groups, 4 AMR pairs per group). Heuristic search with smart initialization and 4 restarts (S+4R) gives the best trade-off between accuracy and speed, so this is the setting we use in practice.

Figure 1 shows smatch scores of each annotator (A-D) against the consensus annotation (E). The

[3]For documents containing multiple AMRs, we use the sum of matched triples over all AMR pairs to compute precision, recall, and f-score, much like corpus-level Bleu (Papineni et al., 2002).

		B					C					D					E				
		1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
	Base	**0.68**	**0.74**	0.84	**0.71**	**0.83**	**0.69**	**0.70**	0.80	**0.69**	**0.78**	**0.77**	**0.72**	0.75	**0.68**	**0.63**	**0.79**	**0.86**	0.92	**0.85**	**0.89**
	ILP	0.74	0.80	0.84	0.76	0.88	0.75	0.78	0.80	0.77	0.88	0.83	0.77	0.75	0.72	0.76	0.85	0.92	0.92	0.89	0.92
	R	0.74	**0.79**	0.84	**0.75**	**0.86**	**0.74**	**0.75**	0.80	0.77	0.88	0.83	**0.76**	0.75	0.72	**0.75**	0.85	0.92	0.92	0.89	**0.89**
A	10R	0.74	0.80	0.84	0.76	0.88	0.75	0.78	0.80	0.77	0.88	0.83	0.77	0.75	0.72	0.76	0.85	0.92	0.92	0.89	0.92
	S	0.74	0.80	0.84	**0.75**	0.88	0.75	0.78	0.80	**0.76**	0.88	0.83	0.77	0.75	0.72	0.76	0.85	0.92	0.92	0.89	0.92
	S+4R	0.74	0.80	0.84	0.76	0.88	0.75	0.78	0.80	0.77	0.88	0.83	0.77	0.75	0.72	0.76	0.85	0.92	0.92	0.89	0.92
	S+9R	0.74	0.80	0.84	0.76	0.88	0.75	0.78	0.80	0.77	0.88	0.83	0.77	0.75	0.72	0.76	0.85	0.92	0.92	0.89	0.92
	Base	-	-	-	-	-	**0.72**	**0.68**	0.74	**0.69**	**0.79**	**0.71**	**0.72**	0.76	**0.65**	**0.57**	**0.68**	**0.71**	0.83	**0.79**	**0.86**
	ILP	-	-	-	-	-	0.74	0.83	0.74	0.75	0.85	0.78	0.83	0.76	0.68	0.73	0.76	0.81	0.83	0.83	0.89
	R	-	-	-	-	-	0.74	0.83	**0.72**	**0.72**	**0.83**	0.78	0.83	0.76	0.68	**0.68**	**0.74**	0.81	0.83	0.83	0.89
B	10R	-	-	-	-	-	0.74	0.83	0.74	0.75	0.85	0.78	0.83	0.76	0.68	0.73	0.76	0.81	0.83	0.83	0.89
	S	-	-	-	-	-	**0.73**	0.83	0.74	0.75	0.85	0.78	0.83	0.76	0.68	0.73	0.76	0.81	0.83	0.83	0.89
	S+4R	-	-	-	-	-	0.74	0.83	0.74	0.75	0.85	0.78	0.83	0.76	0.68	0.73	0.76	0.81	0.83	0.83	0.89
	S+9R	-	-	-	-	-	0.74	0.83	0.74	0.75	0.85	0.78	0.83	0.76	0.68	0.73	0.76	0.81	0.83	0.83	0.89
	Base	-	-	-	-	-	-	-	-	-	-	**0.68**	**0.68**	0.74	**0.69**	**0.65**	**0.64**	**0.64**	0.87	**0.79**	**0.83**
	ILP	-	-	-	-	-	-	-	-	-	-	0.74	0.79	0.74	0.78	0.81	0.74	0.76	0.87	0.85	0.89
	R	-	-	-	-	-	-	-	-	-	-	0.74	0.79	0.74	**0.75**	**0.78**	**0.71**	0.76	0.87	0.85	0.89
C	10R	-	-	-	-	-	-	-	-	-	-	0.74	0.79	0.74	0.78	0.81	0.74	0.76	0.87	0.85	0.89
	S	-	-	-	-	-	-	-	-	-	-	0.74	0.79	0.74	**0.77**	0.81	0.74	0.76	0.87	0.85	0.89
	S+4R	-	-	-	-	-	-	-	-	-	-	0.74	0.79	0.74	0.78	0.81	0.74	0.76	0.87	0.85	0.89
	S+9R	-	-	-	-	-	-	-	-	-	-	0.74	0.79	0.74	0.78	0.81	0.74	0.76	0.87	0.85	0.89
	Base	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	**0.68**	**0.69**	0.81	**0.74**	**0.64**
	ILP	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0.77	0.78	0.81	0.78	0.79
	R	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0.77	**0.73**	0.81	0.78	0.79
D	10R	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0.77	0.78	0.81	0.78	0.79
	S	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0.77	0.78	0.81	0.78	0.79
	S+4R	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0.77	0.78	0.81	0.78	0.79
	S+9R	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0.77	0.78	0.81	0.78	0.79

Table 1: Inter-annotator smatch agreement for 5 groups of sentences, as computed with seven different methods (Base, ILP, R, 10R, S, S+4R, S+9R). The number 1-5 indicate the sentence group number. Bold scores are search errors.

	Base	ILP	R	10R	S	S+4R	S+9R
Accuracy	20%	100%	66.5%	100%	92%	100%	100%
Time (sec)	0.86	49.67	5.85	64.78	2.31	28.36	59.69

Table 2: Accuracy and running time (seconds) of various computing methods of smatch over 200 AMR pairs.

plot demonstrates that, as time goes by, annotators reach better agreement with the consensus.

We also note that smatch is used to measure the accuracy of machine-generated AMRs. (Jones et al., 2012) use it to evaluate automatic semantic parsing in a narrow domain, while Ulf Hermjakob[4] has developed a heuristic algorithm that exploits and supplements Ontonotes annotations (Pradhan et al., 2007) in order to automatically create AMRs for Ontonotes sentences, with a smatch score of 0.74 against human consensus AMRs.

5 Related Work

Related work on directly measuring the semantic representation includes the method in (Dridan and Oepen, 2011), which evaluates semantic parser output directly by comparing semantic substructures, though they require an alignment between sentence spans and semantic sub-structures. In contrast, our metric does not require the align-

[4]personal communication

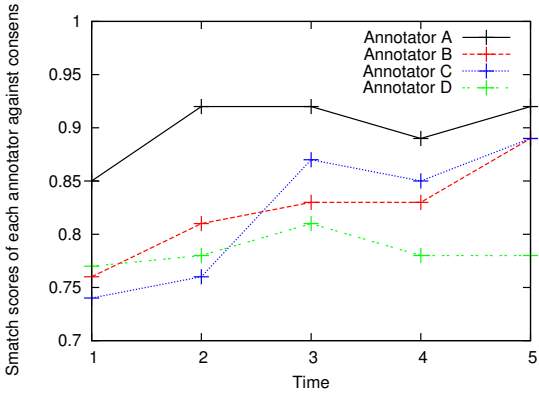


Figure 1: Smatch scores of annotators (A-D) against the consensus annotation (E) over time.

ment between an input sentence and its semantic analysis. (Allen et al., 2008) propose a metric which computes the maximum score by any alignment between LF graphs, but they do not address how to determine the alignments.

6 Conclusion and Future Work

We present an evaluation metric for whole-sentence semantic analysis, and show that it can be computed efficiently. We use the metric to measure semantic annotation agreement rates and parsing accuracy. In the future, we plan to investigate how to adapt smatch to other semantic representations.

7 Acknowledgements

We would like to thank David Chiang, Hui Zhang, other ISI colleagues and our anonymous reviewers for their thoughtful comments. This work was supported by NSF grant IIS-0908532.

References

J.F. Allen, M. Swift, and W. Beaumont. 2008. Deep Semantic Analysis of Text. In *Proceedings of the 2008 Conference on Semantics in Text Processing*.

D. Davidson. 1969. The Individuation of Events. In *Nicholas Rescher (ed.) Essays in Honor of Carl G. HempeL Dordrecht: D. Reidel*.

R. Dridan and S. Oepen. 2011. Parser Evaluation using Elementary Dependency Matching. In *Proceedings of the 12th International Conference on Parsing Technologies*.

B. Jones, J. Andreas, D. Bauer, K. M. Hermann, and K. Knight. 2012. Semantics-Based Machine Translation with Hyperedge Replacement Grammars. In *Proceedings of COLING*.

P. Kingsbury and M. Palmer. 2002. From Treebank to Propbank. In *Proceedings of LREC*.

I. Langkilde and K. Knight. 1998. Generation that Exploits Corpus-based Statistical Knowledge. In *Proceedings of COLING-ACL*.

I. Langkilde-Geary. 2002. An Empirical Verification of Coverage and Correctness for a General-Purpose Sentence Generator. In *Proceedings of International Natural Language Generation Conference (INLG'02)*.

V. Nagarajan and M. Sviridenko. 2009. On the Maximum Quadratic Assignment Problem. *Mathematics of Operations Research*, 34.

K. Papineni, S. Roukos, T. Ward, and W. Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*.

T. Parsons. 1990. *Events in the Semantics of English*. The MIT Press.

S. S. Pradhan, E. Hovy, M. Marcus, M. Palmer, L. Ramshaw, and R. Weischedel. 2007. Ontonotes: A Unified Relational Semantic Representation. In *Proceedings of the International Conference on Semantic Computing (ICSC '07)*.

M. Snover, B. Dorr, R. Schwartz, L. Micciulla, and J. Makhoul. 2006. A Study of Translation Edit Rate with Targeted Human Annotation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas (AMTA-2006)*.

L. R. Tang and R. J. Mooney. 2001. Using Multiple Clause Constructors in Inductive Logic Programming for Semantic Parsing. In *Proceedings of the 12th European Conference on Machine Learning*.

L. S. Zettlemoyer and M. Collins. 2005. Learning to Map Sentences to Logical Form: Structured Classification with Probabilistic Categorial Grammars. In *Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence*.