Argument Sharing in Meaning Representation Parsing

Maja Buljan, Lilja Øvrelid, Stephan Oepen
Language Technology Group, University of Oslo
{majabu, liljao, oe}@ifi.uio.no

Abstract

We present a contrastive study of argument sharing across three graph-based meaning representation frameworks, where semantically shared arguments manifest as reentrant graph nodes. For a state-of-the-art graph parser, we observe how parser performance – in terms of output quality – covaries with overall graph complexity, on the one hand, and presence of different types of reentrancies, on the other hand. We identify common linguistic phenomena that give rise to shared arguments, and therefore node reentrancies, through a small-case and partially automated annotation study and parallel error anaylsis of actual parser outputs. Our results provide new insights into the distribution of different types of reentrancies in meaning representation graphs for three distinct frameworks, as well as on the effects that these structures have on parser performance, thus suggesting both novel cross-framework generalisations as well as avenues for focussed parser development.

1. Introduction

Over the past decade, there has been increasing interest in parsing into graph-based meaning representations, with a growing field of research across different linguistic traditions and frameworks for meaning representation in terms of labelled graphs. A range of parsing systems and approaches have been developed, as well as various frames of in-depth analyses into particular features and challenges of the task and individual frameworks. Unlike widely used representations of syntactic structure in the form of rooted trees, common meaning representation frameworks employ *general graphs*, which makes parsing into these representations more complex, due to, among other features, fewer structural constraints on elements of the graph and on correspondences to the underlying input string ("anchoring"), as well as, of course, the presence of graph nodes with an in-degree greater than one (henceforth "reentrancies").

We follow in this line of research by expanding the methodologies proposed in Buljan et al. (2022), and based on English data and systems featured in the 2020 Shared Task on Cross-Framework Meaning Representation Parsing (Oepen et al., 2020). We focus on PERIN (Samuel and Straka, 2020), the top-performing parsing system in the shared task, and conduct a contrastive error analysis over three frameworks (elaborated in Section 2) to identify common parsing errors, with a view to devising potential parser improvements.

Our research shows an unexpected outlier to the widely accepted wisdom that parsing accuracy deteriorates with growing structural complexity. In an effort to identify potential explanations of this behaviour, we look into the phenomenon of argument sharing in meaning representation graphs, giving rise to the aforementioned reentrant structures. Following the methodology of Szubert et al. (2020) and extending it to the two other frameworks, we attempt to set a foundation for expanding our understanding of the effects of framework design decisions, which will eventually allow for informing future annotation, as well as more targeted parser development.

Apart from these empirical findings, the technical contributions of this paper are: a substantial augmentation of `mtool`, the open-source graph analysis and scoring tool first introduced in the 2019 MRP (Meaning Representation Parsing) shared task (Oepen et al., 2019), which enables quantitative and qualitative analysis of reentrancies and their various subtypes; and a small-scale manual reentrancy annotation effort over gold standard data used for parser development in the shared task. Both contributions will be released openly upon publication.

The paper is organised as follows: Section 2 gives a broad summary of the methodological and technological context of our work; Section 3 describes our approach to parser performance analysis, presents the results, and motivates further investigation. In Section 4, we look into the underlying framework properties and how they inform our error analysis. Section 5 discusses different linguistic causes of reentrancy structures in meaning representation graphs, describes the setup of our pilot annotation effort, and presents its findings. Finally, Section 6 concludes the paper, and discusses pertinent next steps.

2. Background

The MRP 2019 and 2020 shared tasks on cross-framework meaning representation parsing were organised with the goal of advancing the state-of-the-art in parsing into graph-based representa-

5th Workshop on Designing Meaning Representations (DMR 2024) @LREC-COLING-2024, pages 77–87

tions of sentence meaning (Oepen et al., 2019, 2020). The task focussed on five semantic graph frameworks, and required participants to develop systems that predict sentence-level meaning representations for all five frameworks in parallel.

Of the five frameworks present in the shared task, we narrow our focus (and use development data pertaining) to three frameworks embodying distinct approaches to meaning representation, differing in their level of abstraction from the underlying surface string, as well as in formal construction and linguistic assumptions. The three frameworks are exemplified in Figures 1, 2, and 3 with the sentence *"Pierre Vinken, 61 years old, will join the board as a nonexecutive director Nov. 29."* (Oepen et al., 2020).

Elementary Dependency Structures (EDS; Oepen and Lønning, 2006, Figure 1) encode sentence meaning in an unordered semantic graph that is derived from the underspecified logical forms of the English Resource Grammar (Flickinger et al., 2017; Copestake et al., 2005). EDS nodes are explicitly anchored onto substrings of the underlying sentence, but these do not correspond one-to-one to surface lexical units, while edge labels denote argument positions into semantic predications.

Prague Tectogrammatical Graphs (PTG; Zeman and Hajic, 2020, Figure 2) present a conversion from the multi-layered (and somewhat richer) annotations in the tradition of Prague Functional Generative Description (FGD; Sgall et al., 1986), as adopted (among others) in the Prague Czech–English Dependency Treebank (PCEDT; Hajič et al., 2012). PTG nodes are mostly anchored to surface lexical units, but allowing for empty ("generated") nodes and discontinuous anchoring Edges in PTG denote fine-grained labelled relation types ("functors").

Abstract Meaning Representation (AMR; Banarescu et al., 2013, Figure 3), in contrast, makes no explicit connection between the surface sentence and elements of the graph, and is therefore considered unanchored (or free of specific assumptions about derivation and composition). Graph nodes are content words most frequently normalised to verbal senses, and edges are labelled with argument positions or more specific semantic relations, including e.g. fine-grained annotations of named entities and some lexical decomposition.

The MRP 2020 English validation data for the cross-framework track, from which we draw data for our work, comprises gold annotated meaning representation graphs of sentences, counting 3302 datapoints for EDS, 1664 for PTG, and 3560 for AMR, respectively (Oepen et al., 2020).

When analysing parser performance, we focus on the top-scoring, state-of-the-art parser from the MRP 2020 shared task: PERIN (Samuel and Straka, 2020). The PERIN parser is a general neural network architecture for learning to predict the mapping from surface strings to various types of linguistic structure in the form of general graphs. Using an XLM-R and transformer-based encoder-decoder architecure, the parser is language- and framework-agnostic, and therefore applicable across different meaning representation frameworks and languages with the adjustment of pre- and post-processing steps. It also uses a novel permutation-invariant approach to parallel graph node prediction, which is well suited to the task of predicting orderless semantic graphs. Furthermore, as PERIN is not a seq2seq model, but based on a specialized node, edge, and label prediction architecture, there is room for follow-up engineering in light of findings such as those presented in this study.

In the broader sphere of parsing data analysis, we build on methodologies inspired by contrastive approaches introduced in, among other works, McDonald and Nivre (2011) and Kulmizev et al. (2019) for dependency treebanks and parsers. We follow and expand upon the quantitative and qualitative approach to error analysis in MRP outlined in Buljan et al. (2020, 2022). We also look to Szubert et al. (2020) for a discussion of reentrancies in AMR and underlying linguistic phenomena.

We report performance using `mtool`[1], the cross-framework graph analyser used in the MRP shared tasks. Other notable framework-specific graph similarity metrics are discussed by Cai and Knight (2013) and Opitz (2023).

3. Analysing Parser Performance

Following the methodology outlined by Buljan et al. (2022), we examine the performance of the PERIN parser on the MRP 2020 shared task, retrained on the official training data, and using the validation data for our study. To make our results robust to fluctuation that could arise from random initialization, we set out to compare five separate training and testing runs. By and large, our observations are stable across all runs.

Graph complexity We begin by dividing the data into ten decile bins, according to sentence-level graph complexity in terms of the number of nodes. Following the official metric of the MRP shared tasks (Oepen et al., 2019, 2020), we focus on the micro-average F_1 score over tuple types that encode various graph properties, where Buljan et al. (2022) observe that it can be beneficial

[1] https://github.com/cfmrp/mtool

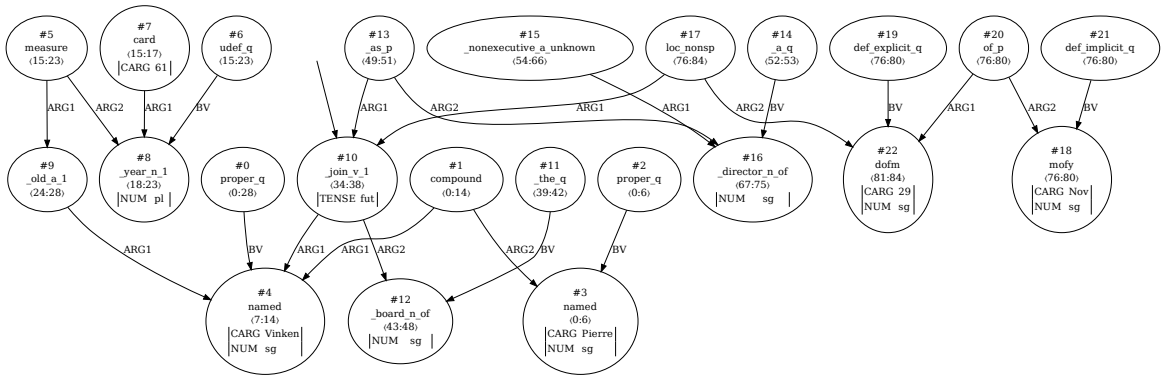

Figure 1: EDS semantic graph for the running example.

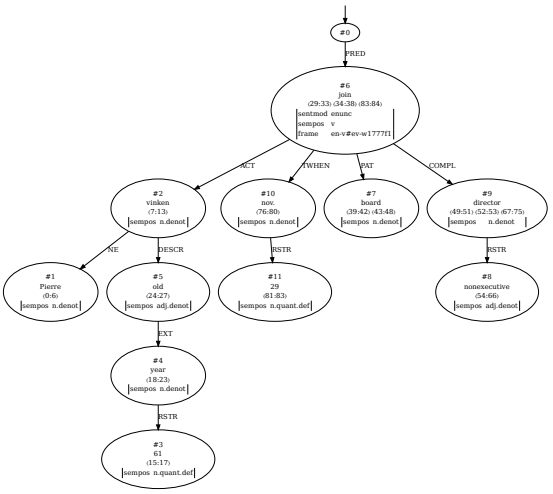

Figure 2: PTG semantic graph for the running example.

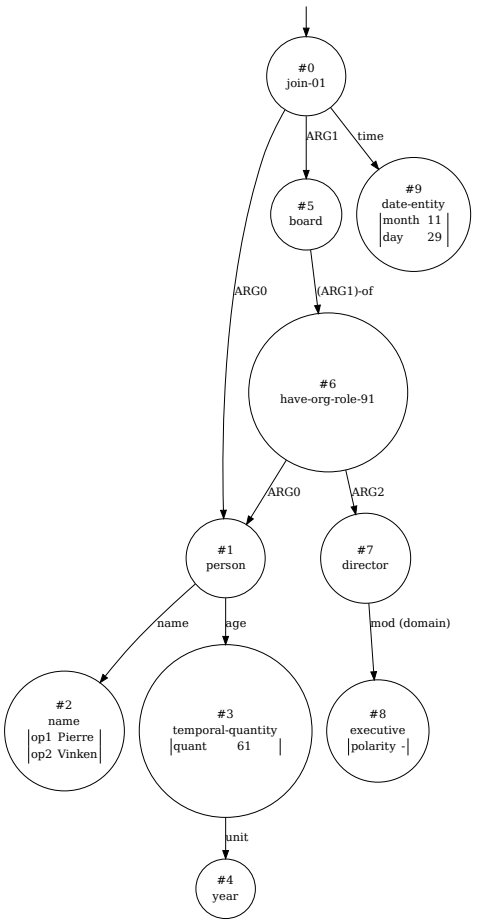

Figure 3: AMR semantic graph for the running example.

to tease apart two distinct subtasks in graph prediction: (a) predicting graph nodes and their decorations, e.g. labels and other node-local properties vs. (b) core graph structure in terms of (labelled) edges and identification of the top node. Figure 4 reports PERIN performance across frameworks and decile bins. The top plot of each framework-specific group charts the overall MRP F_1 score; the middle figure charts F_1 considering only node decoration[2]; and the bottom figure charts F_1 over structural properties only (root nodes (tops) and edges).

We observe a drop in parser performance in the overall F_1 score charts, for PTG and AMR particularly, correlated with rising graph complexity. As discussed in Section 1 above, this is expected behaviour given an assumed correlation between output structure size, sentence length, and related complexity of the parsing problem. However, EDS subverts these expectations, showing instead only a drop of a couple percentage points in the first

[2]Prediction of node anchoring is disregarded, for the sake of result comparability, as it is not applicable to AMR nodes.

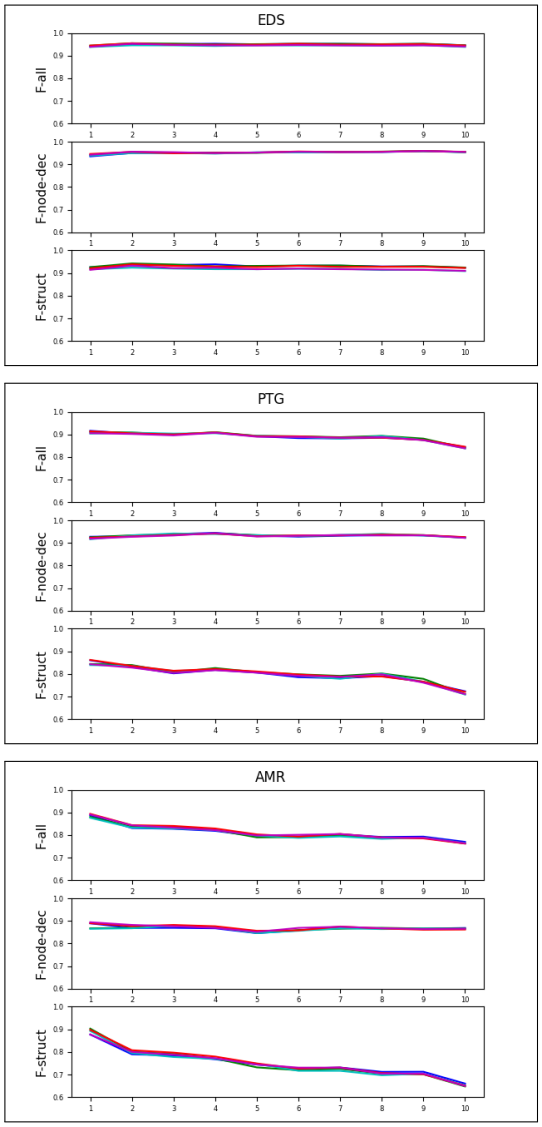

Figure 4: Average parser performance per complexity bin, in terms of overall F_1 (top in each graph), F_1 for node-local properties only (middle), and F_1 for structural properties only (bottom), across the three frameworks.

and last bins – which represent somewhat uneven groups of very short and very long sentences.

Node-local properties The performance over all three frameworks is fairly consistent across the complexity bins when evaluating only on node decoration (node labels and properties). This indicates that node-local information may be easier (for PERIN) to predict with consistent quality.

Structural properties When we examine the parser performance on structural properties of the graphs (edges and root nodes), a clearer picture emerges of the performance drop with greater graph complexity for PTG and AMR. Compared to a drop of 15 and 9 percentage points, respectively, between the highest and lowest performing bin in PTG and AMR overall, the drop in performance is 26 and 16 percentage points for the structural properties. Again, though, EDS remains the outlier, with relatively consistent scores across the bins (within 3-5 percentage points), and moderate divergence between the different training and scoring runs.

	EDS	**PTG**	**AMR**$^{-1}$
Average Nodes / Graph	23.4	17.9	10.4
Edge Labels	10	67	84
$\%_g$ Rooted Trees	0.3	23.9	27.0
$\%_g$ Treewidth One	66.9	23.9	53.7
Average Treewidth	1.33	2.07	1.52
Maximal Treewidth	3	6	5
Average Edge Density	1.02	1.18	1.09
$\%_n$ Reentrant	33.4	15.7	19.6
$\%_g$ Cyclic	0.0	29.9	0.3

Table 1: Some graph statistics (validation data).

Complexity in parsing In syntactic parsing, and to some degree also in semantic parsing, it has long been established that longer sentences are more difficult to parse (McDonald and Nivre, 2011; Van Noord et al., 2018). This is commonly attributed to increasing probability of linguistically more complex structures, as well as to error propagation. However, in the semantic parsing and meaning representation sphere, there is (to the best of our knowledge) little research into what the possible causes of this behaviour are, and whether it is a universal phenomenon across frameworks. Therefore, what we observe with EDS in Figure 4 is more surprising than, intuitively, the PTG and AMR scores, and raises questions about what causes these differences. Hypothetically, either the PERIN parser could be particularly tuned to parse into EDS with great accuracy (which is, for all we know, not the case), or there is a specific property of the EDS framework that differentiates it from the other two frameworks in parser performance related to structural graph complexity.

4. Graph Statistics

In the previous section, we analysed parser performance across three frameworks. Our findings raised questions about the difference between expected and observed behaviour with regards to graph complexity. We now provide a more detailed analysis of the underlying properties of the three frameworks.

We begin by presenting an analysis of struc-

Framework	All	01	02	03	04	05	06	07	08	09	10
EDS	*33.4*	31.15	33.17	33.64	34.01	33.78	33.73	33.55	33.23	33.46	33.39
PTG	*15.7*	3.65	8.25	11.61	13.41	15.39	15.09	16.42	17.67	17.60	21.52
AMR^{-1}	*19.6*	0.19	9.65	16.93	16.55	19.29	20.33	20.59	20.95	20.78	21.35

Table 2: Per-framework percentages of reentrant nodes broken down by graph size decile bins.

tural graph statistics, according to Kuhlmann and Oepen (2016). Table 1 shows a subset of properties computed by the `mtool` graph analyser. EDS graphs, on average, have the largest number of nodes, with AMR graphs being substantially smaller; in terms of the number of distinct edge labels, the order is reversed. The next four rows in Table 1 seek to quantify degrees of "treeness". Unlike in EDS, about one quarter of PTG and AMR graphs actually are rooted trees. Conversely, the EDS graphs have lower average and maximal treewidth, with PSG appearing least "tree-like" in this perspective. The average number of edges per node and percentage of reentrant nodes indicate that EDS has comparatively low edge density but that reentrancies nevertheless occur in one of three nodes, compared to around 16% and 20% for PTG and AMR, respectively. Finally, EDS and AMR exclude cyclic graphs by design, whereas cycles are both allowed and common in PTG.

From the graph structure properties in Table 1, a noteworthy difference between EDS and the other two frameworks is the frequency of reentrant nodes. As discussed previously, reentrancies are central properties that distinguish graph-based structures from tree-based structures and make them more challenging to parse into (Szubert et al., 2020). From previous research, one might assume that this makes reentrant nodes harder to predict, so this datapoint is, again, somewhat surprising.

In Table 2 we look further into the "%$_n$" row in Table 1 and break the statistics down by complexity bins. We find that the percentage of reentrant nodes is consistently high across bins in EDS, while the percentage of reentrant nodes grows with graph complexity for PTG and AMR (in correlation with a drop in performance). EDS still remains the outlier, while based on observations on PTG and AMR, it could be hypothesised that reentrancies are harder to predict.

To explore this hypothesis, we further refine the methodology used in Section 3, and compare structural parser performance considering the reentrant status of edges in the graphs. The results are charted in Figure 5. The leftmost column shows parser performance considering only edges that are not part of a reentrancy, i.e. do not point to a reentrant node. The rightmost column shows parser performance considering only reentrant edges, i.e. edges that point to a reentrant node. To facilitate comparison, the middle column repeats the data shown in Figure 4, showing performance on all edges.

We observe that EDS performance drops by nearly ten percentage points when going from scoring all edges to only scoring non-reentrant edges, and furthermore observe a slight improvement when considering reentrant edges only.

In the case of PTG and AMR, while there is little difference in performance overall across the three scoring methods, we do see some improvement in the lower decile bins specifically when scoring reentrant edges only.

Considering that PTG and AMR show no deterioration in performance on average for reentrant edges compared to non-reentrant ones, and that EDS performs much better on reentrant edges overall, it would appear that our initial hypothesis is not confirmed – and arguably even disproven.

This motivates a more detailed analysis of reentrant nodes – their causes and kinds of manifestations in the frameworks, which we present in the following section.

5. Reentrancy

The description papers and annotation guidelines for each of the three frameworks in focus mention reentrant nodes to varying degrees, but unlike discussions of reentrancies in AMR in work such as Szubert et al. (2020); Van Noord and Bos (2017), there is (to the best of our knowledge) little in-depth discussion of linguistic phenomena that give rise to reentrancies, or the structures in which they manifest, for EDS and PTG.

In this section we investigate the most frequent and overlapping causes for reentrancies in all three frameworks, with the hopes of learning more about the difficulties – or advantages – of parsing into reentrancies.

To illustrate our approach, Figure 6 shows the EDS graph for the example sentence

(1) *The high interest rates and outlooks announced today surprised and shocked investors.*

This example exhibits some interesting linguistic

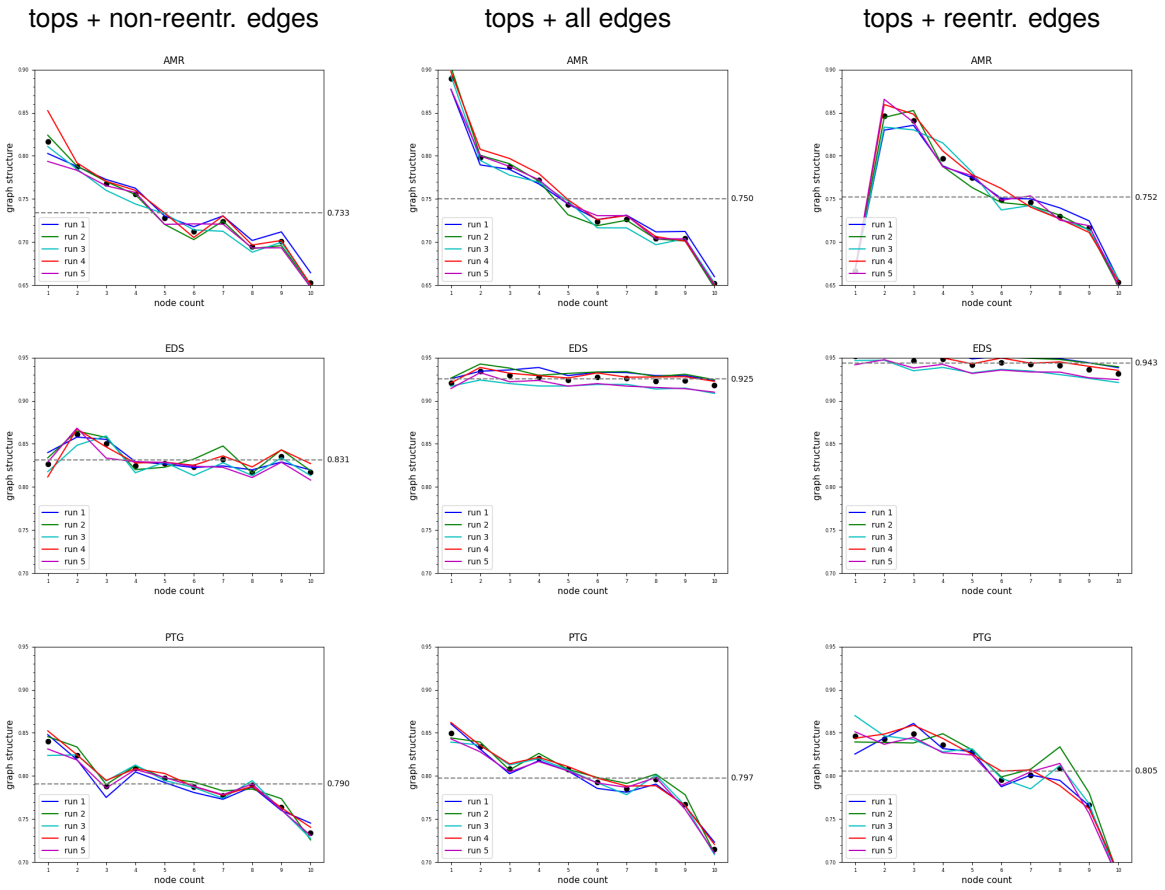

Figure 5: Graph structure F-score, scoring tops and (left) all edges except incoming reentrancies; (center) all edges; (right) only incoming reentrancies, over five train-and-test runs of the PERIN parser.

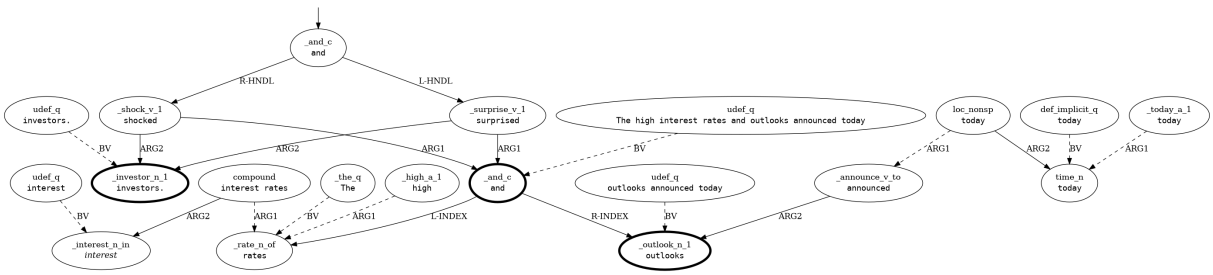

Figure 6: EDS graph for Example (1). Ten automatically annotated reentrant edges are shown as dashed arrows; three remaining reentrant nodes are highlighted with bold edges. In addition to EDS-specific node labels, each node indicates (in typewriter font) the corresponding sub-string of the example.

complexity, including a nominal compound, nominal and verbal coordination, where in the latter both the subject and an extracted object are shared arguments between the conjuncts, a reduced relative clause, and a semantically decomposed temporal modifier. Argument sharing in coordinate structures and relative clauses gives rise to reentrant nodes in the graph, both in EDS and in the other frameworks in our study. Additionally, restrictive modification typically causes reentrancies in EDS, e.g. the attributive adjective, analysis as the compound structure parallel to an unexpressed preposition, and the attachment and internal structure of the temporal modifier. If translated to a more conventional logical-form representation, this would correspond to something like $\_\text{high\_a\_1}(x) \wedge \_\text{rate\_n\_of}(x)$. Similar reentrancies related to modifier structures will arise in AMR, though not in PTG, where modifiers tend to be dependents of the nodes they modify. Finally, among our three frameworks, EDS has the unique property of encoding quantificational structure, using

Framework	Type	Freq.
EDS	quantification	.377
	compound	.062
	modification	.070
	numeric	.022
	preposition	.072
	other modification	.027
PTG	paratactic structures	.341
AMR	modification	.417

Table 3: Relative frequencies of reentrancy types labelled automatically.)

designated BV ("bound variable") edges. This applies to both determiners that introduce quantificational force (_the_q in our example) and to covert (unexpressed) quantificational predicates, e.g. on bare nominals and in the decomposition of *today* (udef_q and def_implicit_q). These edges, again, reflect the underlying logical structure and are a very frequent source of reentrancies in EDS which is not present in the other frameworks.

5.1. Pilot Annotation

Based on the methodology of Szubert et al. (2020), we begin by empirically observing reentrancies in the frameworks, and assign labels based on the linguistic phenomena they embody. With the goal of quantifying our findings, we carry out a small-scale pilot annotation study on sample sets from each of the three frameworks. For each framework, we build a sample of 150 sentence graphs, randomly selected to include at least one reentrant node, and balanced proportionally across the decile bins.

5.1.1. Predictable reentrancies

From our initial observations of the samples, we find a number of frequent and predictable reentrancy patterns in each of the frameworks that take up a not inconsiderable portion of the data and human labour during annotation. We automate the annotation of these "predictable" reentrancies, and focus manual annotation efforts on the remaining reentrant nodes. Table 3 lists these predictable reentrancies and their relative frequencies in the sample sets.

In the case of EDS, the largest portion of these is taken up by determiners and other quantificational nodes, denoted with the BV edge label (in further discussion, we label these as "category 1" reentrancies). Apart from being a very frequent cause of reentrancies, this edge type is a framework idiosyncrasy and, thus, not a comparable linguistic feature across frameworks. Similarly, we automatically annotate compounds, adjectival modifiers, cardinal and ordinal number, prepositions, and other predictable instances of what EDS analyses as restrictive modification (appositions, possessives), all of which we consider "category 2" reentrancies.

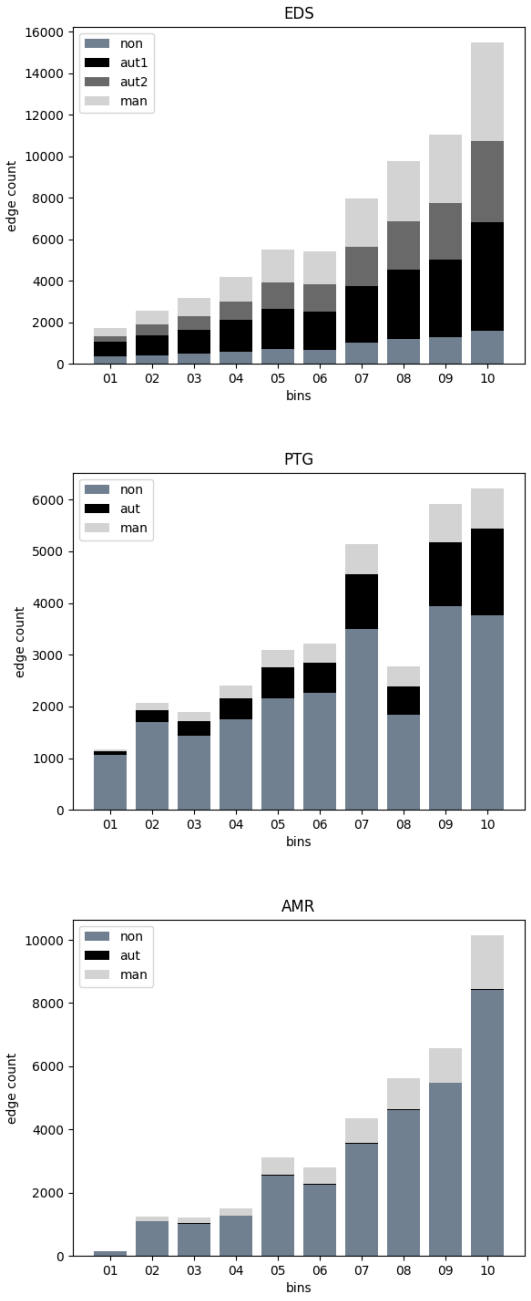

Figure 7: Proportions of edges not involving reentrancy vs. automatically and manually annotated reentrant ones

In the case of PTG, the majority of predictable reentrancies is caused by the framework formality of introducing member and effective edges for all paratactic clause-like structures, be they clauses or compounds, predominantly involving coordina-

tion.

Finally, in the case of AMR, a relatively frequent and predictable cause of reentrancy is restrictive modification, as uniquely denoted by the domain edge label.

Figure 7 charts proportions of edges in the full dataset, by framework and complexity, according reentrancy status: not reentrant, part of an automatically annotatable reentrancy (categories 1 and 2 for EDS), or part of a reentrancy requiring manual annotation. These figures give a sense of the portion of reentrancies caused by "predictable" framework formalities like quantification or paratactic structures in EDS and PTG, respectively.

5.1.2. Manual annotation

Following the approach of Szubert et al. (2020), we empirically observe the occurrence of reentrancies and note their causes, starting with the relation set discussed in the original paper, and expanding with more reentrancy type labels as needed. The results of the manual annotation are presented in Table 4, highlighting the most frequent reentrancy phenomena for each of the frameworks.

- Characteristic of EDS, but not captured by the modification-labelling step of the automatic annotation, *comparative* comprises edges incoming from nodes denoting comparative and superlative modifications of adjectives and verbs, as in the sentence fragment *"the largest and most prized market"*.
- The *Control structures* label encompasses various types of argument sharing found in control structures, such as subject and object control, adjunct control, etc., including nominal control.
- Both *coordination* and *coreference* may give rise to argument sharing, and hence reentrancies, as in the example sentence *"the trust said it has rebuilt reserves and improved operations"*.
- The *modal* label covers all reentrancies occurring as a result of modal verb structures, such as *"they may rise to mountainous proportions"*, where the subject is the argument of both the modal verb and the main verb (in the case of PTG), or the main verb gets an additional incoming edge from the modal verb (in the case of EDS and AMR).
- In EDS and PTG, *modification* includes reentrancies occurring from adjectival participles in restrictive modification, as in *"We make waves under controlled conditions and learn where there are buried rock structures."*
- In PTG, *named entities* and similar compound structures of proper nouns also give rise to reentrancies, by linking each constituent to the predicate node, and each other via a *named entity* edge label, such as in the example sentence *"Goldman, Sachs & Co. will manage the offering."*
- In AMR, *partitive* encompasses a wide range of part-of relations that cause reentrancies, as the example of *finger* and *king* in *"I am more powerful than the finger of a king."*
- *Possessive* relations give rise to reentrancies in all three frameworks, by linking the possessor and the object of possession, as in the sentence fragment *"with regard to man's life in society"*.
- For all three frameworks, the *relative clause* is a common cause of reentrancy, with multiple incoming nodes for the shared argument, as in the fragment *"a tile bridge spanning a stream that flows into the building from outside"*.
- Most frequent in AMR, reentrancies labelled *verbalisation* arise from the annotation convention of maximising the use of predicates. This most often manifests as adjectival participles, or nouns as in the example of the (*govern-01, organization*) node pair representing the noun *government*.
- Finally, the *other* label comprises other causes for reentrancies that had less than five occurrences in the sample data for all three frameworks, such as object raising or various discourse elements.

Since data for the three frameworks are not drawn from the same source, the relative frequencies in Table 4 are not horizontally comparable. However, within the frameworks, certain highlights emerge. For example, both EDS and AMR reentrancies prominently feature verbalisation, particularly adjectival participles. Regardless of the different source material, coreference is a frequent cause of reentrancy in both PTG and AMR, while the highly reentrant EDS has an arguably more balanced occurrence of many of the discussed reentrancy types.

Alongside the relative frequencies of reentrancy types in the sample data, Table 4 shows the error rates of the PERIN parser for the respective reentrancy types, bringing us back to the original question of parser performance and where we might see particular areas of improvement.

For example, in the case of coreference in PTG and AMR, the PERIN parser fails to produce the correct graph structure 38% and 43% of the time,

	EDS		PTG		AMR	
Type	Freq.	Error	Freq.	Error	Freq.	Error
clause-like structures	.067	.058	-	-	.039	.428
comparative	.027	.000	-	-	-	-
control structures	.063	.250	**.136**	.291	**.106**	.342
coordination	**.127**	.125	**.165**	.258	**.134**	.166
coreference	.027	.142	**.401**	.382	**.243**	.436
modal	.011	.333	.014	.400	.008	.333
modification	**.167**	.071	**.176**	.451	-	-
named entity	-	-	.031	.818	-	-
partitive	-	-	-	-	.058	.285
possessive	.039	.500	.008	.666	.008	.666
relative clause	**.183**	.065	.022	.125	.008	.666
verbalisation	**.159**	.075	.002	.000	**.326**	.358
other	.122	.173	.039	.571	.067	.541

Table 4: Relative frequencies and error rates of manually annotated reentrancy types.

Framework	**EDS**	**PTG**	**AMR**
Missed/Total	.106	.345	.405

Table 5: Ratio of reentrant edges the parser failed to produce vs. total number of reentrant edges, per framework (sample set).

respectively, implying that, given the prominence of coreference in the frameworks, correct coreference resolution has an impact on parser performance, especially in longer sentences with more occurrences of this reentrancy structure.

Similarly, with verbalisation being a frequent cause of reentrancies in AMR, the parser demonstrates a 35% error rate on this reentrancy type. As with the previous example, better performance on this task would likely significantly increase parser performance overall.

In the case of possessives, it is interesting to note that, although this particular reentrancy cause makes up a relatively small proportion of reentrancies observed in the annotation set, the parser shows an error rate of 50% or greater for possessives in all three frameworks.

Table 5 summarises the error rates over the total number of reentrant edges per framework, in the sample annotation set. Note that this view does not include the "predictable" reentrancies from the automatic annotation step. Even disregarding these frequent reentrancy types that are a result of framework-specific regularities and, therefore, may be somewhat easier for the parser to correctly predict, the parser retains the lowest parsing error rate on EDS, with just 11% of reentrant edges not produced. Unlike for PTG and AMR, EDS annotations were guided by a large-scale computational grammar, i.e. automatically confirmed to obey formal principles of derivation and composition, which may both lead to higher degrees of predictability and overal greater consistency of the annotations.

6. Conclusion

Building on previous research into parser performance for different frameworks of meaning representation graphs, we carried out a contrastive study focussing on patterns of parser behaviour in sentences of increasing graph complexity, and the presence and frequency of reentrant nodes in the target graph. We performed a small-scale, semi-automated annotation effort over a sample of our datasets, and discussed observations on common linguistic phenomena that give rise to reentrant structures, and how successful a state-of-the-art parser is in producing them. This work sets the foundation for future focussed parser development, as well as further discussions of the particularities of framework design and annotation guidelines.

We intend to explore both of these tracks in future work, specifically to reach out to the PERIN developers and discuss properties of the parsing architecture that may explain our findings (and potential revisions to mitigate their negative impact on parser performance). In the framework analysis track, we will explore redefining graph complexity (and, subsequently, binning) by reentrancy count, in contrast to the currently used node-count approach. We also intend to refine and scale up the reentrancy annotation effort, to produce a larger dataset with phenomena categories aligned across frameworks, and to the highest degree possible over the same strings.

7. Acknowledgements

We would like to thank the reviewers for their insightful comments and suggestions regarding improvements to the paper and future work.

8. Bibliographical References

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.

Maja Buljan, Joakim Nivre, Stephan Oepen, and Lilja Ovrelid. 2020. A tale of three parsers: Towards diagnostic evaluation for meaning representation parsing. In *12th International Conference on Language Resources and Evaluation (LREC), MAY 11-16, 2020, Marseille, FRANCE*, pages 1902–1909. European Language Resources Association (ELRA).

Maja Buljan, Joakim Nivre, Stephan Oepen, and Lilja Øvrelid. 2022. A tale of four parsers: methodological reflections on diagnostic evaluation and in-depth error analysis for meaning representation parsing. *Language Resources and Evaluation*, 56(4):1075–1102.

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.

Ann Copestake, Dan Flickinger, Carl Pollard, and Ivan A. Sag. 2005. Minimal Recursion Semantics. An introduction. *Research on Language and Computation*, 3(4):281 – 332.

Dan Flickinger, Stephan Oepen, and Emily M. Bender. 2017. Sustainable development and refinement of complex linguistic annotations at scale. In Nacy Ide and James Pustejovsky, editors, *Handbook of Linguistic Annotation*, pages 353 – 377. Springer, Dordrecht, The Netherlands.

Jan Hajič, Eva Hajičová, Jarmila Panevová, Petr Sgall, Ondřej Bojar, Silvie Cinková, Eva Fučíková, Marie Mikulová, Petr Pajas, Jan Popelka, Jiří Semecký, Jana Šindlerová, Jan Štěpánek, Josef Toman, Zdeňka Urešová, and Zdeněk Žabokrtský. 2012. Announcing Prague Czech-English Dependency Treebank 2.0. In *Proceedings of the 8th International Conference on Language Resources and Evaluation*, pages 3153 – 3160, Istanbul, Turkey.

Marco Kuhlmann and Stephan Oepen. 2016. Towards a catalogue of linguistic graph banks. *Computational Linguistics*, 42(4):819–827.

Artur Kulmizev, Miryam de Lhoneux, Johannes Gontrum, Elena Fano, and Joakim Nivre. 2019. Deep contextualized word embeddings in transition-based and graph-based dependency parsing–a tale of two parsers revisited. *arXiv preprint arXiv:1908.07397*.

Ryan McDonald and Joakim Nivre. 2011. Analyzing and integrating dependency parsers. *Computational Linguistics*, 37(1):197–230.

Stephan Oepen, Omri Abend, Lasha Abzianidze, Johan Bos, Jan Hajic, Daniel Hershcovich, Bin Li, Tim O'Gorman, Nianwen Xue, and Daniel Zeman. 2020. MRP 2020: The second shared task on cross-framework and cross-lingual meaning representation parsing. In *Proceedings of the CoNLL 2020 Shared Task: Cross-Framework Meaning Representation Parsing*, pages 1–22, Online. Association for Computational Linguistics.

Stephan Oepen, Omri Abend, Jan Hajic, Daniel Hershcovich, Marco Kuhlmann, Tim O'Gorman, Nianwen Xue, Jayeol Chun, Milan Straka, and Zdenka Uresova. 2019. Mrp 2019: Cross-framework meaning representation parsing. In *Proceedings of the Shared Task on Cross-Framework Meaning Representation Parsing at the 2019 Conference on Natural Language Learning*, pages 1–27.

Stephan Oepen and Jan Tore Lønning. 2006. Discriminant-based mrs banking. In *LREC*, pages 1250–1255.

Juri Opitz. 2023. Smatch++: Standardized and extended evaluation of semantic graphs. *arXiv preprint arXiv:2305.06993*.

David Samuel and Milan Straka. 2020. UFAL at MRP 2020: Permutation-invariant semantic parsing in perin. *arXiv preprint arXiv:2011.00758*.

Petr Sgall, Eva Hajičová, and Jarmila Panevová. 1986. *The Meaning of the Sentence and Its Semantic and Pragmatic Aspects*. D. Reidel Publishing Company, Dordrecht, The Netherlands.

Ida Szubert, Marco Damonte, Shay B Cohen, and Mark Steedman. 2020. The role of reentrancies in abstract meaning representation parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2198–2207.

Rik Van Noord, Lasha Abzianidze, Antonio Toral, and Johan Bos. 2018. Exploring neural methods for parsing discourse representation structures. *Transactions of the Association for Computational Linguistics*, 6:619–633.

Rik Van Noord and Johannes Bos. 2017. Dealing with co-reference in neural semantic parsing. In *Proceedings of the 2nd Workshop on Semantic Deep Learning (SemDeep-2)*, pages 49–57. Association for Computational Linguistics (ACL).

Daniel Zeman and Jan Hajic. 2020. FGD at MRP 2020: prague tectogrammatical graphs. In *Proceedings of the CoNLL 2020 Shared Task: Cross-Framework Meaning Representation Parsing*, pages 33–39.