Error Exploration for Automatic Abstract Meaning Representation Parsing

Maria Boritchev, Johannes Heinecke

Error Exploration for Automatic Abstract Meaning Representation Parsing

Maria Boritchev, Johannes Heinecke
Orange Innovation
2 avenue Pierre Marzin
22307 Lannion cedex, France
{maria.boritchev,johannes.heinecke}@orange.com

Abstract

Following the data-driven methods of evaluation and error analysis in meaning representation parsing presented in (Buljan et al., 2022), we performed an error exploration of an Abstract Meaning Representation (AMR) parser. Our aim is to perform a diagnosis of the types of errors found in the output of the tool in order to implement adaptation and correction strategies to accommodate these errors. This article presents the exploration, its results, the strategies we implemented, and the effect of these strategies on the performances of the tool. Though we did not observe a significative rise on average in the performances of the tool, we got much better results in some cases using our adaptation techniques.

1 Introduction

Semantic parsing of natural language is the task of extracting a formal meaning structure from a natural language sentence. Semantics of natural languages can be formalised in various ways, see for instance Bos (2011) and more recently Žabokrtský et al. (2020) for overviews; semantic parsing can be performed from any natural language into any of the semantic formalisms. One of these formalisms, Abstract Meaning Representations (AMR, Banarescu et al. (2013)) has been widely used in the context of deep semantic parsing of English and up to at least ten other languages, including French, German, Spanish, Italian, and Polish. There are two main types of approaches to multilingual AMRs: either the AMR graphs concepts are consistent with the target language (e.g. French concepts for French sentences), or the parsing results in an AMR graph with English concepts (Propbank-based). In this paper, we work in the scope of the latter approach. Machine semantic parsing of English has reached high-quality results, scoring over 83% Smatch score (Cai and Knight, 2013)[1] for the state of the art approaches (Yu and Gildea, 2022). In this context, we want to focus on the remaining 17%, and investigate both why the parser performs badly on these inputs and why the evaluation techniques would consider these parses as bad ones. We have limited our error explorations to languages we were familiar with; in particular, expert annotators familiar with Chinese should be involved in a follow-up study covering Chinese, for which a large amount of AMR annotation has been done. Machine AMR parsing works well, making the cases where it performs badly particularly interesting both linguistically and for deep learning studies. To make AMR parsing truly usable and reliable for real-life applications such as automatic summarization (Huang et al., 2022), question/answer generation (Deng et al., 2022), and neural machine translation (Li and Flanigan, 2022) we need to be able to trust it. We believe that this trust will come from a deep understanding of both our models and our data. The work presented in this article takes roots in explainability of artificial intelligence and computational linguistics. We conduct an error analysis and annotation exploration of the 50 worst examples from development corpora. We work in a multilingual context, on English (EN), French (FR), German (DE), Spanish (ES), Italian (IT), and Polish (PL). Our aim in this article is to share the error categories that we observed along with our attempts to remediate these errors, and the results of these attempts, in particular in terms of (non-significant) effects on the Smatch score. While our work constitutes a negative proof of concept, we still think it is an important contribution to share in the field to help to constitute a baseline for such adaptation techniques and encourage research and dialogues around them.

[1] A Python package is available at https://github.com/snowblink14/smatch/

Proceedings of the 15th International Conference on Computational Semantics pages 246–251

2 AMR Parsing

AMR parsing was explicitly developed for English only. Its goal was to represent sentences through relations between predicates and their semantic arguments. These representations are now machine-generated and have been extended to at least ten other languages.

Abstract Meaning Representations In AMR, each sentence is represented with a rooted, directed, acyclic graph with labelled edges, where nodes are instances, concepts or literals, and edges are relations (figure 1). A single AMR graph can represent several natural language sentences as AMRs do not map words in a sentence to parts of the graph, but rather represent the semantic links that appear in the sentence or sequence of sentences. The goal of AMR representations is to abstract from syntactic constraints: sentences that have the same meaning but different formulations are represented with the same AMR. The representations are based on frames from PropBank (Kingsbury and Palmer, 2002), and the concepts and relations are either extracted from PropBank or English lemmas.

```
(h / hear-01          # "is a" relation (instantiation)
   :ARG0 (w / woman)                    # relation
   :ARG1 (c / cat
      :quant 2))                        # attribute
```

Figure 1: AMR graph for "the woman heard two cats".

Machine AMR Parsing The AMR parser we explore is based on AMRlib[2] for which the underlying language model T5 was changed for the multilingual MT5. We only used the T5/MT5 models since at the time we began our study they gave the best results (on English). AMRlib is based on a seq2seq model and outputs a "raw" AMR graph (without instance variables). The variables are inserted in a postprocessing step. If the raw graph contains too many errors (e.g. missing or additional quotes or parentheses), the postprocessing step loops through the raw graph until it has found a clean beginning. In this case, the final AMR graph lacks some instances and relations.

Data The training data for our parser is based on the English corpus of AMR 3.0 (LDC2020T02[3]). These corpora are mainly based on news reels. To obtain multilingual parsing, we trained our modified AMRlib using MT5 (instead of the monolingual T5) on data obtained by machine translation of English data to French, German, Spanish, Italian, and Polish. To reinforce the training, the parser was trained for each language on both corpora in English and in the target language. The AMR test corpus has been translated manually into German, Italian, Spanish, and Chinese (LDC2020T07). We evaluated on the first three of these and added the machine-translated versions for French and Polish since there is no manual translation of the sentences of the test corpora for these languages available. The results of our evaluation are listed in table 1.

IT	ES	DE	FR	PL
73.9	74.4	71.0	74.0	72.2

Table 1: Results for multilingual parsing evaluation.

3 Error Exploration

We performed an error analysis of the parser's outputs. We identified and implemented two strategies based on this analysis. Our results show improvements in the parsing results that are qualitatively interesting but quantitatively not significant enough.

Our analysis was done on the development corpus of AMR 3.0 (LDC2020T02), as we wanted to avoid introducing any bias in our study by using the test corpus. The sentences were machine translated into the given language, parsed using a model trained on the machine-translated training corpus, and then evaluated using the Smatch Python package against the gold development corpus. Then, the sentences were ranked by worst Smatch score, and the 50 first ones were annotated. Table 2 shows the Smatch scores for the first and 50th worst sentences, per language.

Smatch	EN	FR	DE	ES	IT	PL
worst	25.0	22.2	13.3	13.3	20.7	11.8
50th	60.9	49.6	46.8	48.9	49.5	47.4

Table 2: Worst and 50th worst Smatch per language.

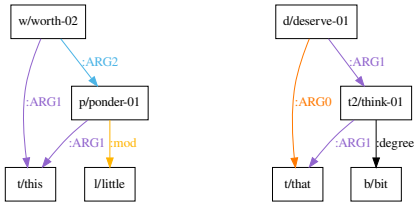
3.1 Error Categories

We identified seven categories of errors in the development data: (1) translation, (2) coordination, (3) input-based errors, (4) incomplete output, (5) reification, (6) errors in gold annotation, (7) other. These categories are listed in the order used for the exclusive annotation: if an error is annotated as a translation one, we did not try to annotate it further as belonging to another category as well.

[2] https://github.com/bjascob/amrlib
[3] https://catalog.ldc.upenn.edu/LDC2020T02

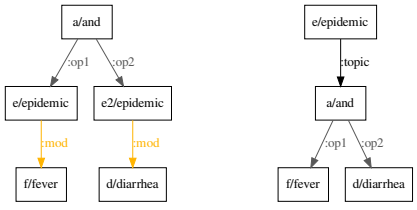
(1) Translation Translation-based errors have different origins: some of these come from wrong translations from English to the target language, which yields a bad parsing; others come from sentences for which the wordings in English and in the target language are structurally or lexically very different. Note that for DE, ES, and IT we used the official translations provided by LDC, so for these languages translation errors are of the second category. Some of the sentences contain technical terms which are badly or inexactly translated. The translation can also contain hallucinations because of the underlying seq2seq model, which adds parts that were not in the original sentence to the translated one. Lastly, the translation introduces synonymy in the AMR concepts that are used to build the graph. For instance, the English verb *break* was translated correctly into French *casser*, however, the AMR parser uses the concept *smash-01* instead of *break-01* found in the gold AMR graph.
Example: "This is worth pondering a little!" / "Cela mérite réflexion un peu !"

gold predicted (on French)

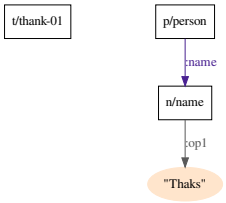


(2) Coordination This category corresponds to several subtypes of errors, including sentence coordination/multiple sentences, that can yield `multi-sentence` annotations, trigger `and`-concepts or not be annotated at all. First, we labelled here the errors that have to do with a bad parsing of conjunctions such as "and" or "but". Then, the ones that have to do more largely with sentence segmentation: sentences which were split incorrectly in two graphs linked with the `multi-sentence`-concept, or, on the contrary, two sentences which were merged using the `and`-concept.
Example: "There is an epidemic of fever and diarrhea."

gold predicted

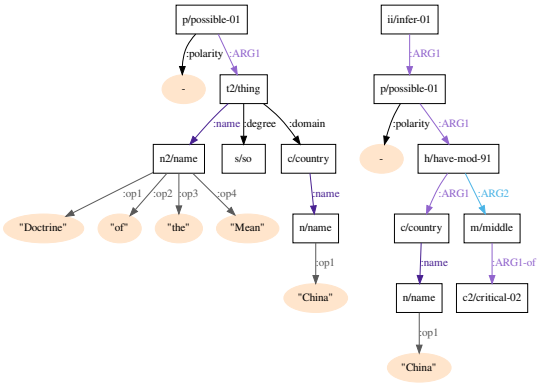


(3) Input There are errors in the input corpus, which can in turn yield errors in the output. In particular, some of the input sentences are too long for the model; when confronted with too long sentences, the model cuts off the sentence after the maximal input length has been reached, yielding incomplete AMR graphs. For Romance languages (FR, ES, IT), translated sentences are generally longer than the EN original ones. We also identified several cases in which the input sentence contains a misspelt word, or a word that is not in the model's vocabulary, or data in a format that is not identified by the model (ex: date), or named entities not recognized by the model as such.
Example: "Thaks."

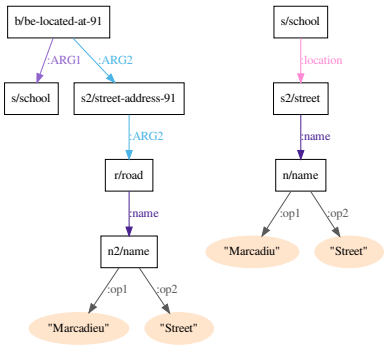


(4) Incomplete output Sometimes, we cannot identify any of the first three categories of errors, and the output AMR graph is still incomplete.
Example: "China can not be so 'Doctrine of the Mean' " / "Chiny nie mogą być więc 'Doktryną środka' "

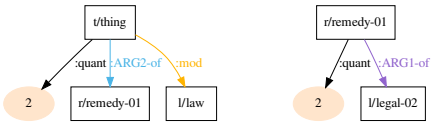
gold (left) & predicted on Polish



(5) Reification In the AMR 3.0 documentation, some relations (e.g. `:location`) can be reified into concepts (e.g. `be-located-at-91`), in order to be able to add a third argument. A reification without additional relations is considered semantically equivalent to the non-reified relation, thus in the gold annotations, both types of annotations are used. However, the standard evaluation script Smatch does not detect these equivalencies and produces a bad score.
Example: "the school is on marcadieu street"



(6) Gold In very few cases, the drop in the Smatch score comes from a mistake in the gold annotation and not from one in the parser's output.
Example: "Legally, there are two remedies."



(7) Other After establishing the 6 previous categories and conducting the annotation, we found other mistakes, which did not constitute a category on their own and could not be assigned to any of the previous categories. These errors have been annotated in this last category.
Example: "That was one hell of an over-reaction."/ "To była cholernie przesadna reakcja."

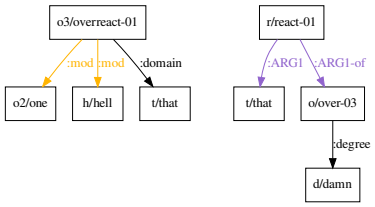


Table 3 shows the distribution of errors across these categories according to the annotation we performed on the 50 examples with the worst Smatch scores, for each language. Coordination is the most important error category for English and French; for the other languages, it is the second most important one after Translation. Then come the categories Input and Reification. Our annotation shows that the 2 other categories (not counting Other) are not significant enough with respect to the worst Smatch score examples.

3.2 Adaptation Strategies and Results

Errors in the input are difficult to correct, as we would risk overfitting and even worsening the situation when our parser would be confronted with new input mistakes outside the kind it would have been prepared to adapt to. Thus, we focused on coordination and reification phenomena for the development of our adaptation and correction strategies.

Reification To check whether the comparison between reified and non-reified relations impacts the evaluation, we wrote a script that reified every occurrence of reifiable relations in both the gold and system output of the development corpus and checked whether the Smatch score increases. However, the impact is minimal, instead of a Smatch score of 85.4, after reification we got 86.0 for EN.

Syntax-based Sentence Splitting Since we observed that long sentences are cut off when the number of tokens is bigger than the MT5 model can handle, we decided to test two sentence-splitting methods. We parsed all sentences of the development corpus with a dependency parser trained on Universal Dependency data[4]. In the first test, we focused on coordination by splitting sentences at the `parataxis` dependency relation (see black on white and white on black parts in figure 2). We then processed each partial sentence and merged the AMR graphs using the `multi-sentence` concept, e.g.:
original sentence: "The first stage splashed down in the Sea of Japan, the second stage crossed the main island of Japan."
partial sentences: (1) "The first stage splashed down in the Sea of Japan" (2) "the second stage crossed the main island of Japan."

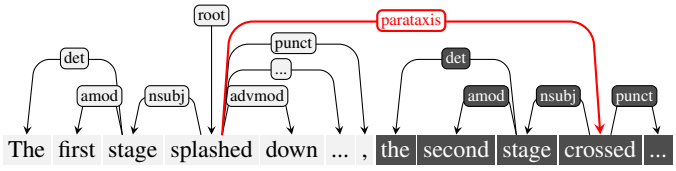


Figure 2: Dependency syntax tree for adjuncts (truncated).

For the second test, we extracted relative clauses (white on black in cf. figure 3) from the sentence (black on white in 3), replaced the relative pronoun ("who") by the head of the relative clause (to have a complete sentence). We then ran the AMR parser on each partial sentence and recombined the AMR graphs by merging the variables of the head of the relative clause (here "man") in both AMR graphs.
original sentence: "The man who saw the dog was afraid"
partial sentences: (1) "the man was afraid" (2) "the man saw the dog"

[4] https://universaldependencies.org

Language	Translation	Coord.	Input	Incomplete output	Reification	Gold error	Other
EN	n.a.	**21** (42%)	6 (12%)	0 (0%)	7 (14%)	3 (6%)	13 (26%)
FR	13 (26%)	**14** (28%)	1 (2%)	0 (0%)	10 (20%)	2 (4%)	10 (20%)
DE	**32** (64%)	7 (14%)	7 (14%)	3 (6%)	0 (0%)	0 (0%)	1 (2%)
ES	**25** (50%)	7 (14%)	7 (14%)	3 (6%)	0 (0%)	1 (2%)	7 (14%)
IT	**26** (52%)	12 (24%)	3 (6%)	5 (10%)	0 (0%)	0 (0%)	4 (8%)
PL	**22** (44%)	14 (28%)	1 (2%)	1 (2%)	5 (10%)	2 (4%)	5 (10%)

Table 3: Results of error annotations of the 50 first parses with the worst Smatch scores, per language.

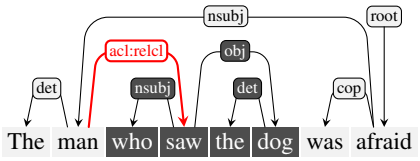


Figure 3: Dependency syntax tree for relative clause.

individual AMR graphs:

```
(v1 / fear-01                (v3 / see-01
  :ARG0 ( v2 / man))           :ARG0 ( v4 / man)
                               :ARG1 (v5 / dog))
```

joined graph (instances `v2` and `v4` merged into `m`):

```
(f / fear-01
    :ARG0 ( m / man
        :ARG0-of (s / see-01
            :ARG1 (d / dog))))
```

Even though for some complex sentences we got much better results with this splitting technique, for others this resulted in additional errors. On average the results in terms of Smatch score did not change.

4 Related Work

There are to our knowledge not many publications presenting systematic explorations of machine AMR parsing mistakes for the purpose of improving the explored tool. This observation might come from a publication bias, as a scientific community, we tend to publish positive results over negative ones. In Buljan et al. (2022), the authors present a discussion of methodological choices for diagnostic evaluation and error analysis in the context of four semantic parsers, two of which output AMR graphs. This article also explores one of the alternatives to Smatch, developed for several semantic representations of language (not only AMR), as part of the *meaning representation parsing* task. Damonte et al. (2017) presents another way of measuring the quality of automatic parses by using Smatch to compute more fine-grained metrics. Stemming from this work, Szubert et al. (2020) focuses on reentrancy phenomena in AMR graphs, categorizes their types, and shows results of experiments performed via an oracle correcting these errors, augmenting the overall parsing performance by 5%. Smatch is also being questioned in a multilingual context. In Wein and Schneider (2022), the authors argue for the necessity of a multilingual AMR evaluation metric and present a multilingual adaptation of S2match called XS2match. The work presented in our article is inspired by the previous work of the same authors (Wein and Schneider, 2021); in this work, they annotate translation divergences between a corpus of English and a corpus of Spanish data, grounding their annotation schema in AMR and labelling type and cause of divergences.

5 Discussion and Conclusion

As shown in table 3, the errors for the AMR graphs on languages other than English mostly concern the machine translation. Either the (English) input had typos (like “thaks” for “thanks”) or contained some named entities spelt in lowercase without any quotes which were translated literally into the target languages and not identifiable as named entities thereafter. The most frequent translation-related error is when a concept slightly differs from the concept in the gold. Even though we can consider these errors as minor, Smatch cannot identify close synonyms and classifies these differing concepts as plain errors.

The next steps for our research are twofold. On one hand, we will continue the diagnostic of our approach, in particular for languages other than English, by evaluating our parser using scores such as XS2match and exploring the errors that get the lower scores; conjointly, as translation issues were majoritary in our analyses, we will investigate how manual correction of translations can improve the parsing’s quality. On the other hand, we will investigate other approaches for our parser. Several categories of errors we diagnosed come from the seq2seq method and from the machine translation tools we use to produce the non-English corpora.

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.

Johan Bos. 2011. A survey of computational semantics: Representation, inference and knowledge in wide-coverage text understanding. *Language and Linguistics Compass*, 5(6):336–366.

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, Sofia, Bulgaria. Association for Computational Linguistics.

Marco Damonte, Shay B. Cohen, and Giorgio Satta. 2017. An incremental parser for abstract meaning representation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 536–546, Valencia, Spain. Association for Computational Linguistics.

Zhenyun Deng, Yonghua Zhu, Yang Chen, Michael Witbrock, and Patricia Riddle. 2022. Interpretable amr-based question decomposition for multi-hop question answering. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence (IJCAI-22)*, pages 4093–4099, Vienna, Austria.

Kuan-Hao Huang, Varun Iyer, Anoop Kumar, Sriram Venkatapathy, Kai-Wei Chang, and Aram Galstyan. 2022. Unsupervised syntactically controlled paraphrase generation with Abstract Meaning Representations. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 1547–1554, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Paul Kingsbury and Martha Palmer. 2002. From TreeBank to PropBank. In *Proceedings of the Third International Conference on Language Resources and Evaluation*, Las Palmas, Canary Islands - Spain. European Language Resources Association.

Changmao Li and Jeffrey Flanigan. 2022. Improving neural machine translation with the Abstract Meaning Representation by combining graph and sequence transformers. In *Proceedings of the 2nd Workshop on Deep Learning on Graphs for Natural Language Processing (DLG4NLP 2022)*, pages 12–21, Seattle, Washington. Association for Computational Linguistics.

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.

Shira Wein and Nathan Schneider. 2021. Classifying divergences in cross-lingual AMR pairs. In *Proceedings of The Joint 15th Linguistic Annotation Workshop (LAW) and 3rd Designing Meaning Representations (DMR) Workshop*, pages 56–65, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Shira Wein and Nathan Schneider. 2022. Accounting for Language Effect in the Evaluation of Cross-lingual AMR Parsers. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 3824–3834, Gyeongju, Republic of Korea. Association for Computational Linguistics.

Chen Yu and Daniel Gildea. 2022. Sequence-to-sequence AMR Parsing with Ancestor Information. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 571–577, Dublin, Ireland. Association for Computational Linguistics.

Zdeněk Žabokrtský, Daniel Zeman, and Magda Ševčíková. 2020. Sentence meaning representations across languages: what can we learn from existing frameworks? *Computational Linguistics*, 46(3):605–665.