Linearization Order Matters for AMR-to-Text Generation Input

Justin DeBenedetto
Villanova University
justin.debenedetto@villanova.edu

Abstract

Abstract Meaning Representation (AMR) is a semantic graph formalism designed to capture sentence meaning using a directed graph. Many systems treat AMR-to-text generation as a sequence-to-sequence problem, drawing upon existing models. The largest AMR dataset (AMR 3.0) provides a sequence format which is considered equivalent to the graph format. However, due to the position-sensitive nature of sequence-to-sequence models, graph traversal order affects system performance. In this work we explore the effect that different, valid orderings have on the performance of sequence-to-sequence AMR-to-text systems and find that changing the traversal order can result in a BLEU score drop of up to 17.5 on a state-of-the-art system.

1 Introduction

Abstract Meaning Representation (AMR) is a semantic graph formalism designed to capture sentence meaning using a directed graph. Nodes in the graph represent semantic concepts and edges between nodes represent relations between concepts. The original AMR paper (Banarescu et al., 2013) describes three "equivalent formats" for a given AMR (see Figure 1). The first is referred to as the "LOGIC" format and is composed of a conjunction of triples. The second is referred to as "AMR" format, which is based on PENMAN notation. The third is referred to as "graph" format and is the most common way to think about AMR graphs. The three formats can be seen applied to the same sentence in Figure 1.

The LOGIC format (logical triples) is used primarily when scoring AMR parses. The AMR parsing task seeks to generate AMR graphs from text data. These AMR parses are primarily scored using the SMATCH metric (Cai and Knight, 2013). SMATCH compares logical triples of the generated parse to the reference to score the semantic similarity. During SMATCH scoring, since conjunction is a commutative operation, there is no inherent ordering on the logical triples. Since AMR parsing is often used as part of a pipeline for AMR-to-text generation, it is important to keep the logical triples in mind when considering how these systems would be used in practice.

The AMR format is the way that most AMR datasets are distributed, including the AMR 3.0 release (Knight et al., 2021). This format is much more human readable than logical triples. This format is based on PENMAN notation (Matthiessen and Bateman, 1991) and is sometimes referred to in this way. AMR-to-text generation systems which use sequence-to-sequence networks typically operate on this format (see for example Mager et al. (2020) or Ribeiro et al. (2021)). Since these methods are sequence-to-sequence, the input is treated as a sequence with some ordering information. This is explored further throughout this current work.

As a semantic graph formalism, the graph format is the main format for AMR graphs. Graph-to-sequence AMR-to-text generation systems attempt to take advantage of the information captured in this graph structure. They typically feature a graph encoder with a sequence decoder. As the amount of available AMR data has grown, graph-to-sequence systems have grown in popularity. Despite these advances, both sequence-to-sequence and graph-to-sequence systems continue to push performance higher.

Increases in the performance of AMR systems has led to increased success in downstream tasks. The number of downstream tasks for which AMR has shown useful is continuously growing and includes summarization (Dohare et al., 2017), translation (Song et al., 2019), biomedical event extraction (Rao et al., 2017), understanding disrupted sentences in speech recognition systems (Addlesee and Damonte, 2023), human-robot communication (Bonial et al., 2023), and more. As these systems expand their usage, it is important to highlight the impact that certain representation decisions can have on performance. Systems which treat AMR graphs as sequence input through linearization have positional representations that rely on the ordering of that linearization.

In this work we investigate the effects of different choices for converting from graph format to AMR format on sequence-to-sequence AMR-to-text systems. This topic was considered in a recent system (Bevilacqua et al., 2021), comparing a breadth-first search traversal to a depth-first traversal for their data. Since they found that breadth-first search yielded inferior results, we do not explore that here. In that work they followed the edge order given by the dataset. Here, we focus on different choices for picking the order to visit child nodes when following a depth-first search traversal of a graph. We find that consistent ordering is essential and that the AMR 3.0 dataset has a specific ordering in its AMR format which provides useful information. Inconsistent ordering from training to dev sets can yield a drop of 5 points on the BLEU score. Randomizing the order in which child nodes are visited also yields a drop of nearly 4 BLEU points, showing that the order given in the AMR 3.0 dataset is important for successful sequence-to-sequence systems. The results are even more drastic on a state-of-the-art system which showed BLEU score drops of 9.4 and 17.5 when moving from the ordering provided by the LDC data to a random or reversed traversal ordering respectively.

LOGIC format:

```
instance(c, contrast-01) ∧
instance(d, disappoint-01) ∧
instance(q, quite) ∧
instance(a2, also) ∧
instance(a, acquire-01) ∧
instance(f, fall-through-06) ∧
ARG2(c, d) ∧
ARG0(d, f) ∧
degree(d, q) ∧
mod(q, a2) ∧
time(d, f) ∧
ARG1(f, a)
```

AMR format:

```
(c / contrast-01
    :ARG2 (d / disappoint-01
        :ARG0 f
        :degree (q / quite
            :mod (a2 / also))
        :time (f / fall-through-06
            :ARG1 (a / acquire-01))))
```

Graph format:

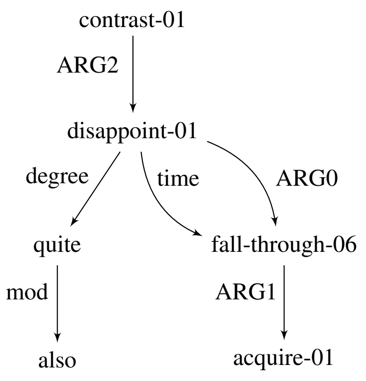

Figure 1: Three formats for the AMR graph for the sentence "However, when the acquisition did fall through, it was also quite disappointing."

2 Related Works

The work by Hoyle et al. (2020) investigates a similar phenomenon of different graph traversal orderings. While the work is motivated by a similar question, their approach to traversal orderings is quite different from ours. In particular, they consider three traversals. The first is a "canonical" ordering which is identical to our "LDC" ordering. The second is a "reconfigured" ordering in which the top (root) node remains the same, but all other nodes may be reordered and edges themselves may be reversed (for example "ARG0(a,b)" may be reversed to "ARG0-of(b,a)"). In our work we do not allow reversal of edges to be part of the graph traversal choice. The third of their traversals is "randomized", which is similar to "reconfigured", but may not even respect the top (root) node placement. Again, this type of scrambling of the AMR graph does not feel like a natural part of the sequence-to-sequence pipeline, so we do not consider these types of traversals in our work.

In the work by Konstas et al. (2017), there are three linearization orderings tried. The first is "Human" which is identical to our "LDC" ordering. The second is "Global-random" which imposes a strict ordering on labels and always applies that ordering when traversing a graph. The third is "Random" which, by its description, appears to match our "Random" traversal process. In their work, they find that each of the three traversal orderings provided very similar BLEU scores. This is in contrast to our findings. They do not present any training mismatch scenario as we do (trained with one traversal, tested on another). Furthermore, the contrast between the relatively small BLEU score differences in their work and the large BLEU score differences we show on a current state-of-the-art system demonstrates that this problem may be getting worse as our systems have become more tailored to the ordering given by the data.

After the work presented here was completed, work by Gao et al. (2023) was published which focuses on leveraging traversal order to improve AMR parsing systems. In that work, child order is determined fully by the order given in the LDC release of the data. They define left-to-right (L2R) as the order given ("LDC" ordering in our work) and right-to-left (R2L) as the reverse order of children given in the dataset ("reversed" in our work). By combining both representations as part of their training process, they are able to achieve a 0.5 SMATCH improvement on the AMR 3.0 dataset. This shows a recent application of graph traversal ordering improving a system. Since they are not working on AMR-to-text generation, the concern about mismatched training data compared to testing data is not considered in their work.

3 Converting Between Formats

The motivation behind the three different formats are to offer three equivalent ways to express the semantic information contained in the AMR graph. When you begin with an AMR in LOGIC format, you can convert to either of the other two formats. Upon converting back into the LOGIC format, you will have the same set of triples that you started with and, in this regard, the formats are viewed as equivalent. However, when you convert from graph to the sequence based AMR format, there may be multiple ways to linearize the graph which result in different sequences.

3.1 Graph Traversal Order

When converting between the graph format and the linearized AMR format, changes to the order in which graph nodes are visited can produce different resulting linearizations. In Figure 1 the linearization begins with "contrast-01" followed by "disappoint-01", but then there is a choice that has to be made. There are three edges coming out from "disappoint-01" which can be followed in any order. The choice made in the AMR 3.0 dataset was to follow the "ARG0" edge, then the "degree" edge, then the "time" edge. Additionally, the edge going to "acquire-01" was not followed until all edges to its parent were processed.

Another decision which must be made when traversing graphs for linearization is whether to take an edge-centric or node-centric view. In a node-centric view, when you process a node (such as "fall-through-06" in Figure 1), you process all edges coming into the node (here both the "time" edge and the "ARG0" edge). In an edge-centric view, you process one out-edge at a time without regard to whether multiple edges lead to the same node or not. For example (referring back to Figure 1 again), the ordering in which the edges are processed could be "ARG2", "time", "ARG1", "degree", "mod", and finally "ARG0".

3.2 AMR 3.0

The largest high quality AMR dataset currently available is the AMR 3.0 dataset (Knight et al., 2021) which contains nearly $60,000$ sentences paired with their corresponding AMR graph. This dataset, and its prior two releases, have been widely used to advance the study of AMR systems. While typically AMR 3.0 follows a depth-first search approach by introducing node concepts at the first use of their variable, this does not always happen. In particular, out of $55,635$ training AMR graphs in AMR 3.0 release, $4,775$ of them have at least one variable written before its corresponding concept. An example of this can be seen in Figure 1 since the variable f comes before the corresponding concept of "fall-through-06".

The order in which out-edges should be processed is not entirely determined by depth-first search nor breadth-first search. For example, as previously mentioned, parallel edges are of concern for consistency in processing. Within the training AMR graphs, there are 159 which contain occurrences of parallel edges which are separated in the given linearization. Figure 1 shows this with the two parallel edges from "disappoint-01" to "fall-through-06" being separated in the AMR format by the edge to "quite". Overall, out of $55,635$ training AMR graphs, $4,886$ have at least one of these two phenomena and 48 have both (including the example from Figure 1).

There are also many instances of inconsistent ordering within very similar subtrees across the training AMR graphs. For example, subtrees rooted at a "date-entity" node feature all of the following edge orders within the training set:

- day then month
- month then day

- month then year
- year then month
- year then month then day
- day then month then year
- month then day then year
- month then year then day

This means that conversion from the given AMR formatted data into either graph or logical triples will result in certain ordering information loss. While the edge traversal order given within the AMR 3.0 dataset does not always match the sentence token ordering (see for example Figure 2), there may be some useful ordering information encoded in the ordering.

4 Experiments

To measure the effects of different linearization decisions on an AMR-to-text system, we trained a simple sequence-to-sequence model on the data using various linearization orders. In all experiments, the same preprocessing steps were taken and the model parameters were the same. Our preprocessing steps followed steps common to other work (Pourdamghani et al., 2016; Konstas et al., 2017). These steps were to remove wiki entries, remove variable names, remove quotation marks, remove sense information, and finally to tokenize using the Moses tokenizer (Koehn et al., 2007). Our scripts are available on GitHub[1].

Our sequence-to-sequence model for the experiments was a Fairseq (Ott et al., 2019) implementation of a Transformer model (Vaswani et al., 2017). We used a 4 layer encoder with embedding size 256, feedforward size 512, and 2 attention heads. The decoder had 6 layers with embedding size 512, feedforward size 1024, and 4 attention heads. The learning rate was set to 5e-4 and early stopping was used based on best dev BLEU score. We did not use any external data such as silver training data nor pretrained language models.

It is worth noting that the focus here was to train a simple model to demonstrate the effects of linearization order. While many other optimizations can be made to improve the performance, these models were kept as generic and simple as possible to be more widely applicable to other systems.

[1]https://github.com/jdebened/AMR-sequence-ordering.git

To demonstrate the potential impact on a current state-of-the-art system, we also tested a pretrained system (described below in section 4.1) on the same data.

4.1 AMRBART

Additional experiments were run on a pretrained AMRBART system (Bai et al., 2022). As the name suggests, AMRBART it built upon the BART (Lewis et al., 2020) model. AMRBART focuses on an AMR-to-AMR pretraining process at its core. This then allows them to use this as the basis for either an AMR parser or for an AMR-to-text generation system. In our work, the AMR-to-text system is used for comparison. This is one of the best performing systems currently for AMR-to-text generation and uses the sequence-to-sequence framework (BART). This pretrained model was used without any further modifications nor any additional finetuning done to the model. The data from each of the four traversal scenarios was fed into the model for evaluation. Since AMR parsing models are largely evaluated by SMATCH score (which is identical across our traversals), this set of experiments demonstrates what could happen when pipelining an AMR parsing system with an AMR-to-text generation system.

4.2 Linearization Orders

We compared four different linearization orders. The first is the order given in the original dataset, unchanged. This is called “LDC” in our results. The second is obtained by removing the phenomena discussed above (Section 3.2). Specifically, a depth-first search traversal is respected in all cases and a node-centric view of this traversal is taken. The order in which child nodes were visited matches the LDC ordering. However, since we convert to a graph before converting back to AMR format, phenomena such as splitting parallel edges cannot occur. Similarly, there are no longer occurrences of a variable prior to its corresponding node concept. This is called “Graph and back” in our results. The third order is obtained by choosing a random child to visit first during the depth-first search. This seems quite natural when viewing the graph as a set of logical triples. This is called “Random” in our results. The final ordering tried simply reverses the order given in the original dataset. Thus when choosing which child to visit first, it makes the opposite decision from the AMR 3.0 choice. This is called “Reversed” in our results.

	LDC	Graph and Back	Random	Reversed
LDC	16.0	16.1	12.5	10.8
Graph and Back	15.7	15.7	12.3	10.4
Random	12.3	12.4	12.1	11.9
Reversed	10.9	10.9	11.9	14.4
All	14.3	14.3	13.6	14.0
AMRBART	51.2	51.0	41.8	33.7

Table 1: Best dev BLEU scores for train/dev pairings. The row gives the training traversal method and the column gives the dev traversal method. The "All" row concatenated the training data from all 4 other methods as a form of data augmentation. AMRBART is a pre-trained model shown here to demonstrate the effects on a state-of-the-art model.

LDC and Graph and Back:

```
(d / date-entity
    :year 2012
    :month 2
    :day 29)
```

Reversed

```
(d / date-entity
    :day 29
    :month 2
    :year 2012)
```

Random (2 of 6 possibilities shown)

```
(d / date-entity    OR  (d / date-entity
    :month 2                :day 29
    :day 29                 :year 2012
    :year 2012)             :month 2)
```

Figure 2: Different linearization choices for the AMR graph corresponding to the sentence "February 29, 2012" in the AMR 3.0 training data. Here the LDC and Graph and Back methods both match, but the Reversed and Random methods produce different orderings. In this case, Random produced the order closest to the sentence order, but due to its random nature this may not always be the case on this example.

These linearization options can be seen in Figure 2.

We wanted to know not only if one choice is better than others for linearization, but also how important consistency is between the linearization choice for training and testing. As shown in Table 1, we trained each model four times to maximize the BLEU score on the development set with the various traversal methods. The AMR 3.0 given orderings were the best to train with unless the dev data was reversed. Training on reversed data had the worst performance when the dev set was not reversed. Overall, it is clear that the traversal order should match your test set for best results. When your training data traversal does not match your dev/test data you can experience a performance drop of 5 BLEU points as seen when comparing LDC to reversed.

When considering which linearization order to use, another possible option is to concatenate the results of all of the different methods into one larger training set. This views linearization order as a data augmentation technique. The "All" row in Table 1 shows the results of training in this way. This method showed the most consistency across dev linearizations, but was only the best choice when the dev data followed the random linearization. This demonstrates that adding other linearization techniques is not an effective data augmentation technique on its own. It is worth noting that "All" does not appear as a column. This is because it is not simply imposing a specific ordering to the dev data, but rather changing the data by duplicating it once for each of the four orderings. Therefore it is suitable to try as a training method (thus the "All" row), but not as an evaluation method (thus no "All" column).

5 Conclusion

In this work we have explored different traversal orders for AMR graphs. While most of the AMR 3.0 dataset follows a depth-first search traversal, approximately 8.6% of the training graphs violate this order. This dataset is distributed with a certain edge ordering that leads to better BLEU scores than other possible edge traversal orders. We experimented with various traversal orders and showed that consistent ordering from training to dev sets is important and that the order given in the dataset contains important ordering information for training with sequence-to-sequence systems. Conversions from AMR format to graph or logical triple format and back lose certain information.

We looked at one example, date-entity subtrees, where an inconsistent edge ordering across training graphs leads to some of the information loss.

AMR parsing systems are often judged by their SMATCH scores. Since SMATCH is a metric comparing AMR graphs in their logical triple form, it does not provide information about the node or edge traversal order. In this work we showed that this decision is important for the performance of any system that uses these AMR graphs. In particular, all four traversal orderings tested receive identical SMATCH scores since the set of logical triples is identical across the four traversal orderings. As more systems incorporate AMR as an intermediate form for capturing semantic information, graph traversal order is a problem that requires attention to maximize the system performance. This is demonstrated through experiments on a pretrained AMRBART system which had a BLEU score drop of 9.4 when the random traversal ordering was applied and a BLEU score drop of 17.5 when the reversed traversal ordering was applied. Since SMATCH does not differentiate between these traversals, AMR parsing systems evaluated primarily on SMATCH scoring may benefit from additional ordering information before being combined with an AMR-to-text generation system.

While we highlighted certain inconsistencies in the AMR 3.0 datasets representation, there is useful ordering information encoded in the training set. Further study could investigate if it is possible to reliably and automatically generate similar orderings for AMR graphs produced through AMR parsing systems. Additionally, graph-to-sequence AMR-to-text systems could use the ordering information encoded in the linearization order given by AMR 3.0 as additional information not directly provided by the graph format. While graph encoders were applied to AMR graphs as a means of reducing the information loss encountered in linearization and other preprocessing, this ordering information already encoded in the dataset acts as a form of information loss in graph-based systems.

If each of the three representations for AMRs are to be considered equivalent, then the “useful” ordering information in the training set should be viewed as information leakage. With this perspective, AMR-to-text systems should be invariant to the linearization ordering applied. While this can be achieved in a variety of ways (imposing an order on edge or node labels, using an order invariant neural architecture, etc), it remains an important consideration for any system. For any system which is not traversal order invariant, it is important for your data to match the expected ordering for best performance. Otherwise, as shown in our experiments above (See Table 1), the system may experience a BLEU score drop of more than 5 on a simple system or 17.5 on a state of the art system.

Acknowledgements

This work used the Augie High-Performance Computing cluster, funded by award NSF 2018933, at Villanova University.

References

Angus Addlesee and Marco Damonte. 2023. Understanding disrupted sentences using underspecified abstract meaning representation.

Xuefeng Bai, Yulong Chen, and Yue Zhang. 2022. Graph pre-training for AMR parsing and generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6001–6015, Dublin, Ireland. Association for Computational Linguistics.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186.

Michele Bevilacqua, Rexhina Blloshmi, and Roberto Navigli. 2021. One spring to rule them both: Symmetric amr semantic parsing and generation without a complex pipeline. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 12564–12573.

Claire Bonial, Julie Foresta, Nicholas C Fung, Cory Hayes, Philip Osteen, Jacob Arkin, Benned Hedegaard, and Thomas Howard. 2023. Abstract meaning representation for grounded human-robot communication. In *Proceedings of the Fourth International Workshop on Designing Meaning Representations*, pages 34–44.

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.

Shibhansh Dohare, Harish Karnick, and Vivek Gupta. 2017. Text summarization using abstract meaning representation. *arXiv preprint arXiv:1706.01678*.

Bofei Gao, Liang Chen, Peiyi Wang, Zhifang Sui, and Baobao Chang. 2023. Guiding amr parsing with reverse graph linearization. *arXiv preprint arXiv:2310.08860*.

Alexander Hoyle, Ana Marasović, and Noah Smith. 2020. Promoting graph awareness in linearized graph-to-text generation. *arXiv preprint arXiv:2012.15793*.

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

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the association for computational linguistics companion volume proceedings of the demo and poster sessions*, pages 177–180.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural amr: Sequence-to-sequence models for parsing and generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 146–157.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Manuel Mager, Ramón Fernandez Astudillo, Tahira Naseem, Md Arafat Sultan, Young-Suk Lee, Radu Florian, and Salim Roukos. 2020. GPT-too: A language-model-first approach for AMR-to-text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1846–1852, Online. Association for Computational Linguistics.

Christian MIM Matthiessen and John A Bateman. 1991. Text generation and systemic-functional linguistics: experiences from english and japanese. *(No Title)*.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

Nima Pourdamghani, Kevin Knight, and Ulf Hermjakob. 2016. Generating english from abstract meaning representations. In *Proceedings of the 9th International Natural Language Generation conference*, pages 21–25, Edinburgh, UK. Association for Computational Linguistics.

Sudha Rao, Daniel Marcu, Kevin Knight, and Hal Daumé III. 2017. Biomedical event extraction using abstract meaning representation. *BioNLP 2017*, pages 126–135.

Leonardo F. R. Ribeiro, Martin Schmitt, Hinrich Schütze, and Iryna Gurevych. 2021. Investigating pretrained language models for graph-to-text generation. In *Proceedings of the 3rd Workshop on Natural Language Processing for Conversational AI*, pages 211–227, Online. Association for Computational Linguistics.

Linfeng Song, Daniel Gildea, Yue Zhang, Zhiguo Wang, and Jinsong Su. 2019. Semantic neural machine translation using amr. *Transactions of the Association for Computational Linguistics*, 7:19–31.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.