Exploring a POS-based Two-stage Approach for Improving Low-Resource AMR-to-Text Generation

Marco Antonio Sobrevilla Cabezudo and **Thiago Alexandre Salgueiro Pardo**
Interinstitutional Center for Computational Linguistics (NILC)
Institute of Mathematical and Computer Sciences, University of São Paulo
São Carlos/SP, Brazil
msobrevillac@usp.br, taspardo@icmc.usp.br

Abstract

This work presents a two-stage approach for tackling low-resource AMR-to-text generation for Brazilian Portuguese. Our approach consists of (1) generating a masked surface realization in which some tokens are masked according to its Part-of-Speech class and (2) infilling the masked tokens according to the AMR graph and the previous masked surface realization. Results show a slight improvement over the baseline, mainly in BLEU (1.63) and METEOR (0.02) scores. Moreover, we evaluate the pipeline components separately, showing that the bottleneck of the pipeline is the masked surface realization. Finally, the human revision suggests that models still suffer from hallucinations, and some strategies to deal with the problems found are proposed.

1 Introduction

Abstract Meaning Representation (AMR) is a semantic formalism that encodes the meaning of a sentence into a labeled directed and rooted graph (Banarescu et al., 2013). This representation comprises semantic information related to semantic roles, named entities, and co-references, among others.

AMR is a widely-studied research topic in the semantic representation field and has been proven helpful in many Natural Language Processing tasks (Liao et al., 2018; Song et al., 2019). Its success is partially based on its broad use of mature linguistic resources, such as PropBank (Palmer et al., 2005), and its attempt to abstract away from syntax. Figure 1 shows an example of an AMR graph and its corresponding PENMAN notation for the sentence "*The boy must not go.*".

AMR-to-text generation aims to "translate" an Abstract Meaning Representation graph into its corresponding text. This task has been widely tackled by diverse approaches, starting from statistical, transducer-based and transition-based ones (Pourdamghani et al., 2016; Flanigan et al., 2016; Lampouras and Vlachos, 2017), until end-to-end neural ones (Mager et al., 2020; Ribeiro et al., 2021a), recently.

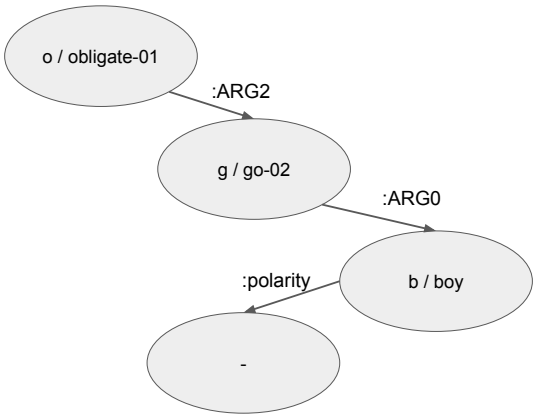

Figure 1: AMR example for the sentence "The boy must not go."

In particular, end-to-end neural models -mainly those based on pre-trained models- have largely outperformed the initial methods, achieving state-of-the-art results (Ribeiro et al., 2021b). These models can generate fluent text. However, they are prone to generate hallucinations, i.e., texts that are irrelevant to or contradicted with the input (Reiter, 2018).

Another drawback is that these models are usually data-hungry, i.e., they need to be trained on a large dataset to achieve a good performance. It can be a problem when we deal with low-resource domains, languages, or tasks (Sobrevilla Cabezudo and Pardo, 2022). Even when the results may be better than those obtained by statistical methods,

Proceedings of the 2nd Workshop on Natural Language Generation, Evaluation, and Metrics (GEM), pages 531 - 538

they are still far from good results. For example, Ribeiro et al. (2021b) show that fine-tuning T5 (Raffel et al., 2020) on a small portion of a big dataset (∼500 instances) produces a ∼10-15 BLEU score.

In general, an approach to have more control over the decoding process (and avoid hallucinations) is to use a pipeline-based method in which the model of each pipeline's module is implemented with neural models (Castro Ferreira et al., 2019; Ma et al., 2019; Puduppully and Lapata, 2021). Another alternative is to use templates, infill concepts in these templates, and then define a strategy to transform them into sentences/paragraphs (Kasner and Dušek, 2020; Mota et al., 2020). Both approaches have proven to be helpful in text generation tasks. However, the main issue for the latter one is that it only can be applied in restricted domains as it is necessary to define a set of templates.

In this work, we approach the AMR-to-text generation task in two stages. Firstly, generating a masked surface realization in which some tokens are masked according to its Part-of-Speech (POS) classes. Then, finally, infilling the masked tokens according to the AMR graph and the previous masked surface realization[1].

The intuition for masking some tokens this way is that some POS classes are more difficult to be predicted during text decoding and can harm the performance. On the other hand, filling-in-the-blank is commonly used on current SotA architectures, such as T5 (Raffel et al., 2020) during the pre-training phase. This way, we can leverage the learned knowledge to infill the masked tokens in the previous stage adequately.

Experiments are conducted on low-resource an AMR-to-text generation task for Brazilian Portuguese (Inácio et al., 2022) to show how this method behaves even when a large dataset is unavailable.

In general, our main contributions are:

- we propose a simple two-stage method that consists of generating masked surface realization and infilling the masked tokens with a transformer-based architecture;
- we conduct a manual revision on the outputs of the best approaches and the end-to-end approach.

[1]The code is available at https://github.com/msobrevillac/DICO-AMR2Text.

2 Related Work

AMR-to-Text generation Modular approaches have been mainly focused on converting AMR graphs into syntax trees via transition-based methods (Lampouras and Vlachos, 2017), end-to-end methods (Cao and Clark, 2019) or rule-based graph-transducers (Mille et al., 2017) and use an off-the-shelf method (neural or statistical) to generate the text. These methods usually have got a low performance on test sets (May and Priyadarshi, 2017).

AMR is more open-ended than other datasets such as WebNLG (Gardent et al., 2017). This way, extracting templates can be a complex task. Some attempts to get templates in the form of rules are presented by Flanigan et al. (2016) and Song et al. (2017). However, these approaches need some manually created rules and have been surpassed by current models.

On the other hand, current neural models have achieved SotA results. However, they need a large dataset to get high performance. On the contrary, a small portion of an extensive dataset produces lower scores (Ribeiro et al., 2021a,b)[2].

Data-to-Text generation Currently, most data-to-text methods are based on end-to-end neural approaches. In particular, methods that fine-tunes a pre-trained model, such as BART (Lewis et al., 2020) or T5 (Raffel et al., 2020), on its specific generation task have achieved SotA results.

Other works have tried to approach this kind of tasks using pipeline approaches (Castro Ferreira et al., 2019; Ma et al., 2019; Puduppully and Lapata, 2021) and template-based approaches (Kasner and Dušek, 2020; Mota et al., 2020). In particular, pipeline approaches have advantages in low-resource settings and unseen domains. On the other hand, template-based approaches tend to infill the templates with concepts and then use them to generate the complete sentence.

3 Experimental Setup

3.1 Dataset

We conduct all experiments on the journalistic section of the AMR-PT corpus (Inácio et al., 2022) (named AMRNews)[3]. The AMRNews corpus comprises 870 sentences with up to 23 tokens each from

[2]Ribeiro et al. (2021b) show an impressive improvement using structural adapters. However, this is not part of this study.

[3]AMRNews is freely available at https://github.com/nilc-nlp/AMR-BP/tree/master/AMRNews.

Brazilian news texts manually annotated according to adapted AMR guidelines (Sobrevilla Cabezudo and Pardo, 2019). Besides, it is divided into 402, 224, and 244 instances for training, development, and testing, respectively.

3.2 Architecture

Aiming to leverage the "fill-in-the-blank" potential of current pre-trained neural models, we propose a two-stage approach consisting of generating a masked surface realization and then infilling the masked tokens using a pre-trained model. Figure 2 shows an example of the whole process.

3.2.1 Masked Surface Realisation

The first stage involves generating a sentence corresponding to an AMR graph in which some tokens are masked. The idea behind this is that some tokens can be more difficult to be predicted. This way, we can mask them and let the next stage complete the masked tokens.

To decide what tokens should be masked, we use Part-of-Speech-based criteria. This way, we group all Part-of-Speech (POS) classes into main classes according to their function. For example, pronouns, nouns, and proper nouns are usually actors/places in a sentence, while verbs represent relations. This way, the main classes are: "*substantivos*" (nouns), "*verbos*" (verbs), "*qualificadores*" (qualifiers), and "*outros*" (others). Table 1 shows the main and POS classes included in each.

Main Class	Part-of-Speech
Substantivos (nouns)	pronoun, noun, proper noun
Verbos (verbs)	auxiliary verb, verb
Qualificadores (qualifiers)	adverb, adjective
Outros (others)	other Part-of-Speech

Table 1: Main and POS classes used in experiments

We train a model for each main class separately. Besides, we train a model for all main classes together. The input consists of a prefix and an AMR graph in the PENMAN notation (eliminating the frameset numbers). We use the expression "*mascarar **X** desde amr:*" ("Mask **X** from amr:") as prefix for each instance, where "***X***" is an specific main class. The output is the corresponding sentence, but words that belong to the target main class are masked. Box 1 from Figure 2 shows an example of this sub-task.

For experiments, we fine-tune the Portuguese T5 (PTT5) (Carmo et al., 2020)[4] on our corpus. Among the hyperparameters, we use AdamW optimizer with a learning rate of 5e-4, a max source and target length of 120 and 80 tokens, a batch size of 8, and a gradient accumulation of 4. The model trains by 12 epochs and is evaluated after each epoch. We use perplexity as evaluation criteria, and the training is halted if the model does not improve after 4 epochs.

3.2.2 Word Infilling

The second stage in the pipeline consists of infilling the masked tokens. In general, the task can be defined as follows: given an AMR graph in a similar format to the one used at the previous stage and a masked sentence, the model predicts the masked words.

Each instance in the corpus is formatted as follows: a prefix, the AMR graph similar to the one used in the previous stage, the word "*contexto*:" (context), and the masked sentence. Box 2 from Figure 2 shows an example of the input and output. We use the expression "*preencher amr:*" ("fill amr:") as prefix and train a model for each main class separately and another model for all main classes together.

Similar to the previous stage, we fine-tune PTT5 on our task. The main reason use PTT5 is that it was pre-trained for a similar task ('filling-in-the-blank') (Carmo et al., 2020; Raffel et al., 2020). This way, we aim to leverage the learned knowledge in our use case. We use the same hyperparameters as the used ones in the first stage; however, we modify the source length to 200 tokens.

4 Results and Discussion

Table 2 shows the overall results for all the trained models on test set in terms of BLEU (Papineni et al., 2002), METEOR (Lavie and Agarwal, 2007), chrF++ (Popović, 2017), and BERTScore (Zhang et al., 2020) evaluation metrics[5][6]. In addition, we report the results for a baseline that generates sentences with no masked tokens. This baseline is obtained by fine-tuning PPT5 on our task. However, the input consists of a prefix "*gerar texto desde amr:*" ("generate text from amr:"), followed

[4]Available at https://huggingface.co/unicamp-dl/ptt5-base-portuguese-vocab.

[5]We execute 4 runs for each experiment and show the mean and standard deviation.

[6]Metrics are calculated by using the code available at https://github.com/WebNLG/GenerationEval.

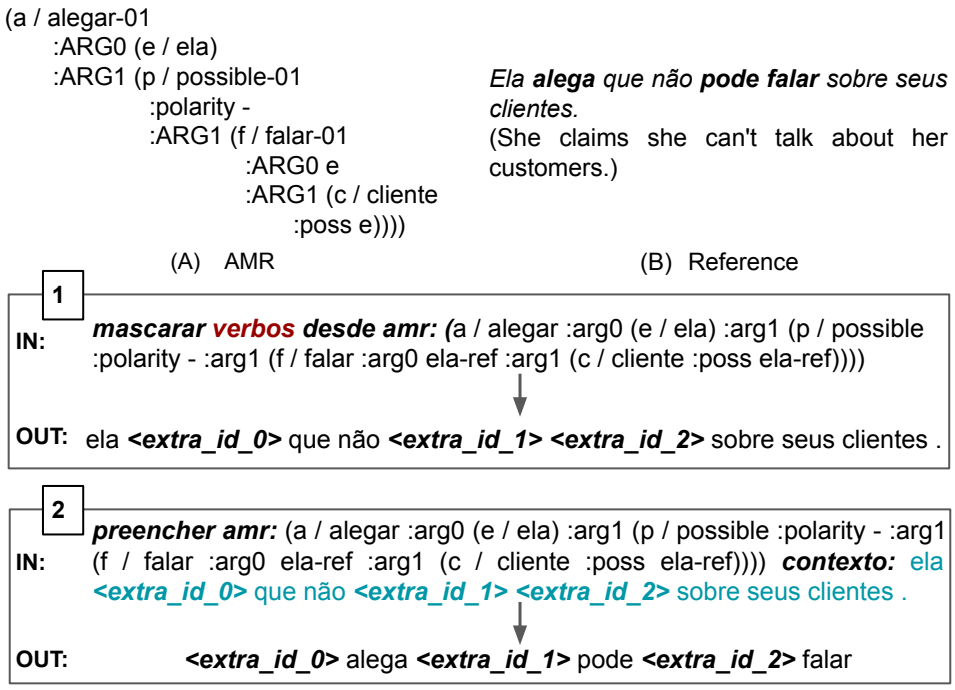

Figure 2: Pipeline Example. Box 1 describes the input and output for the masked surface realization module, and Box 2 illustrates the input and output for the word infilling module.

by an AMR graph represented by the PENMAN notation in a similar way as all already mentioned models, and the output is the original sentence.

Overall, results show a slight improvement over the baseline when we use the model trained on all main classes, mainly in BLEU (+1.63) and METEOR (+0.02) scores. Moreover, the best main classes to be masked seem to be “verb" and “qualifier". On the other hand, masking nouns and other POS classes harm the decoding performance. We might interpret this result as the characters in a sentence, and some connections between chunks are the most important in the realization of a sentence.

Another point to note is that it is better to train models on all main classes together instead of separately. A possible explanation is that more data can lead to better results. Also, examples from other main classes serve as negative examples for a specific main class, and it helps to improve its performance.

In order to verify which stage of the pipeline is affecting the overall performance, we evaluate each module separately. Table 3 and 4 shows the performance on both modules in terms of BLEU, METEOR and chrF++. However, for word infilling, we only evaluate BLEU-2 and BLEU-3, as the number of tokens to be predicted is three as most. In addition, we evaluate METEOR.

Concerning the Mask Surface Realization task, Table 3 indicates that verb masking leads to the best performance. A possible explanation for this result is that, as mentioned before, participants, situations, or locations in a sentence and connections between chunks are the most important and the easiest classes to predict during decoding. Also, it is worth noting that the verbs and qualifiers are less frequent in our dataset, as we can find 1.37-1.47 verbs/qualifiers per sentence. Therefore, it can make decoding easier than nouns (2.23 nouns per sentence).

Table 4 shows the opposite result, as the verb infilling is the most challenging task. However, we note that the values for BLEU-3 in the case of nouns and others are small. This way, it can confuse the infilling order in sentences with more tokens belonging to these classes. Moreover, we note that METEOR score for verbs less penalizes the performance (in comparison with BLEU), suggesting that the model can predict a different conjugation of the expected word.

It is worth noting that, in general, the bottleneck of the whole pipeline is the masked surface realization task, as values are similar to the overall performance. Even the verb-focused decoding,

		BLEU	METEOR	chrF++	BERTScore
Baseline		10.39 ± 0.48	0.29 ± 0.01	0.41 ± 0.01	0.82 ± 0.00
SEP	Noun	5.32 ± 0.56	0.22 ± 0.01	0.35 ± 0.01	0.80 ± 0.01
	Verb	8.95 ± 1.46	0.27 ± 0.01	0.39 ± 0.02	0.81 ± 0.00
	Qualifier	9.44 ± 0.87	0.27 ± 0.01	0.39 ± 0.01	0.81 ± 0.00
	Other	8.21 ± 0.99	0.27 ± 0.01	0.39 ± 0.02	0.81 ± 0.01
ALL	Noun	8.87 ± 0.69	0.28 ± 0.01	0.40 ± 0.02	0.81 ± 0.01
	Verb	**12.02 ± 2.13**	**0.31 ± 0.03**	0.42 ± 0.03	0.83 ± 0.01
	Qualifier	10.34 ± 1.34	0.30 ± 0.02	0.42 ± 0.02	0.83 ± 0.01
	Other	7.74 ± 1.71	0.28 ± 0.01	0.42 ± 0.02	0.81 ± 0.00

Table 2: Overall Results on test set. Experiments in block “SEP" are the ones in which a model is trained on each main class separately, and “ALL" are the ones in which a model is trained on all main classes together, but we evaluate it individually.

having the worst performance on the word infilling task, achieves the highest performance because the previous task gets the best one. A possible explanation for this problem is how the generation is performed. We use an encoder-decoder architecture in which the generation of a token depends on the previously generated tokens. This way, adding mask tokens in training could make it more difficult as the pre-trained model never saw these tokens in a generation task (these were used for training the blank infilling task). Among the alternatives to solve this issue, we could explore other strategies to determine the less confident tokens in a generated sentence and mask them for the next stage. Also, we could try a non-autoregressive model that can overcome the problem of dependency mentioned before (Su et al., 2021).

		BLEU	METEOR	chrF++
SEP	Noun	6.90 ± 1.05	0.45 ± 0.02	0.48 ± 0.02
	Verb	10.91 ± 0.48	0.42 ± 0.02	0.47 ± 0.02
	Qualifier	8.43 ± 0.82	0.30 ± 0.01	0.39 ± 0.01
	Other	10.11 ± 0.74	0.53 ± 0.03	0.56 ± 0.03
ALL	Noun	9.41 ± 1.65	0.49 ± 0.03	0.51 ± 0.03
	Verb	12.31 ± 1.52	0.45 ± 0.03	0.49 ± 0.04
	Qualifier	10.31 ± 1.27	0.32 ± 0.03	0.40 ± 0.03
	Other	10.22 ± 2.67	0.54 ± 0.04	0.56 ± 0.04

Table 3: Results on Mask Surface Realisation on dev test.

5 Manual Revision

We conduct a manual revision of the outputs for each model in order to check the main and most common errors. In particular, we select the two best models in our experiments, i.e., the ones trained on all main classes but focusing on masking/filling verbs and qualifiers.

		BLEU-2	BLEU-3	METEOR
SEP	Noun	33.80 ± 3.83	11.45 ± 3.33	0.46 ± 0.03
	Verb	18.53 ± 2.22	-	0.41 ± 0.01
	Qualifier	44.98 ± 9.12	-	0.57 ± 0.01
	Other	40.35 ± 3.99	18.48 ± 4.02	0.52 ± 0.02
ALL	Noun	41.20 ± 3.07	22.20 ± 3.43	0.57 ± 0.02
	Verb	20.95 ± 3.77	-	0.50 ± 0.02
	Qualifier	39.05 ± 10.21	-	0.65 ± 0.01
	Other	40.90 ± 4.70	19.55 ± 4.13	0.53 ± 0.03

Table 4: Results on Word Infilling on dev set

We analyze 35 instances from the test set and classify the outputs into four classes: (1) Accurate (“Acc"), for accurate outputs,(2) Hallucination (“Hall"), for outputs that are not related to the reference, (3) Cut chunk, for outputs that contains only a portion of the reference, and (4) Small Changes, for outputs with slightly different from the reference (some tokens are different). Table 5 shows the frequency of each class for all evaluated models.

In general, the model trained on all main classes, but focusing on verbs got the best results. It is worth noting the high number of hallucinations in all models, mainly when longer sentences are evaluated. Also, the cut chunks happen in the same cases. Moreover, there are several instances where only changing a simple word (or two) would be necessary to make the output similar to the reference. This problem happens mainly with connectors such as “*em*" (“in" or “at") or “*de*" (“of") (words highlighted in red in Figure 3) and with bad conjugations in the case of the verbs.

Figure 3 shows three examples. The first example shows that the model focused on verbs gets an accurate output (example 1). The second example shows that the outputs for models focused on verbs and qualifiers can generate paraphrases instead of

	Acc	Hall	Cut Chunk	Small Changes
Baseline	9	16	4	6
ALL-Verb	14	12	1	8
ALL-Qualifier	9	18	1	7

Table 5: Number of accurate outputs ("Acc") and errors in the human evaluation.

the same sentence; however, these are accurate too. Finally, the third example is a case in which the models generate hallucinations ("*eua investiram em 2010.*"), outputs with cut chunks ("*investir em os eua.*") or small changes.

Reference	*nada disso é criminoso .* none of this is criminal.
Baseline	*nada de isso .* none of that.
ALL-Verb	***nada de isso é criminoso .*** none of this is criminal.
ALL-Qualifier	*nada de criminoso .* nothing criminal.
Reference	*no vestiário , passou mal .* in the locker room , he felt sick .
Baseline	*passou mal .* he was feeling sick
ALL-Verb	***ele passou mal no vestiário .*** he got sick in the locker room.
ALL-Qualifier	***passou mal no vestiário .*** he got sick in the locker room.
Reference	*desde 2010 , o empresário investe nos eua .* since 2010, the businnessman invests in the usa.
Baseline	***eua investiram em 2010 .*** usa invested in 2010.
ALL-Verb	*investir em os eua .* invest in the usa.
ALL-Qualifier	***em** 2010 , o empresário **não** investiu no eua .* in 2010, the businessman did not invest in the usa.

Figure 3: Outputs comparison between the reference, the baseline, and the two best models in our experiments. The first lines for each model are the sentences generated in Brazilian Portuguese, and the next ones are the corresponding English translations.

6 Conclusion and Further Work

This work presents a simple two-stage approach to the low-resource AMR-to-text generation task. The approach consists of generating a masked surface realization in which some tokens are masked according to a POS class criteria and infilling the masked tokens according to the AMR graph and the previous masked surface realization.

Results show a slight improvement over the baseline, mainly in BLEU (1.63) and METEOR (0.02) scores. However, it is necessary to fine-tune the model on all the sub-corpus created together. Besides, we can note that verb masking seems to be the best strategy in this approach.

On the other hand, we note that the bottleneck of this approach is the masked surface realization model, as some generated tokens are different and unrelated to the original reference (hallucinations), and some tokens are omitted from the original reference. Some possible explanations for this problem are how the generation is performed -as each output word is conditioned on previously generated outputs-and the need to constrain the decoding process.

As further work, we plan to explore strategies to enforce the model to cover all the AMR concepts in the masked generated sentence and non-autoregressive text generation with pre-trained models, similar to Su et al. (2021). Besides, we plan to explore other strategies to mask tokens according to its confidence in decoding instead of using a POS-based one as the later can add more complexity to the task. Finally, we plan to extend this work to English AMR corpus, in order to make a better comparison in terms of generalization.

Limitations

This work tackles the AMR-to-Text generation task with a pipeline approach, and the results are similar to those obtained for previous work with the same amount of data (∼10-15 BLEU score). However, the performance could be different as the lengths of the sentences in our task are up to 23 tokens, and the sentences evaluated in works for English are longer.

Other limitation is related to the criteria used for masking some tokens as it can introduce more complexity, mainly for low-resource languages.

Acknowledgments

The authors of this work would like to thank the Center for Artificial Intelligence (C4AI-USP) and the support from the São Paulo Research Foundation (FAPESP grant 2019/07665-4) and from the IBM Corporation. Besides, this research carried out using the computational resources of the Center for Mathematical Sciences Applied to Industry (CeMEAI) funded by FAPESP (grant 2013/07375-0).

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, Sofia, Bulgaria. Association for Computational Linguistics.

Kris Cao and Stephen Clark. 2019. Factorising AMR generation through syntax. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2157–2163, Minneapolis, Minnesota. Association for Computational Linguistics.

Diedre Carmo, Marcos Piau, Israel Campiotti, Rodrigo Nogueira, and Roberto Lotufo. 2020. Ptt5: Pretraining and validating the t5 model on brazilian portuguese data. *arXiv preprint arXiv:2008.09144*.

Thiago Castro Ferreira, Chris van der Lee, Emiel van Miltenburg, and Emiel Krahmer. 2019. Neural data-to-text generation: A comparison between pipeline and end-to-end architectures. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 552–562, Hong Kong, China. Association for Computational Linguistics.

Jeffrey Flanigan, Chris Dyer, Noah A. Smith, and Jaime Carbonell. 2016. Generation from abstract meaning representation using tree transducers. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 731–739, San Diego, California. Association for Computational Linguistics.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. Creating training corpora for NLG micro-planners. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 179–188, Vancouver, Canada. Association for Computational Linguistics.

Marcio Lima Inácio, Marco Antonio Sobrevilla Cabezudo, Renata Ramisch, Ariani Di Felippo, and Thiago Alexandre Salgueiro Pardo. 2022. The amr-pt corpus and the semantic annotation of challenging sentences from journalistic and opinion texts. *SciELO Preprints*.

Zdeněk Kasner and Ondřej Dušek. 2020. Data-to-text generation with iterative text editing. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 60–67, Dublin, Ireland. Association for Computational Linguistics.

Gerasimos Lampouras and Andreas Vlachos. 2017. Sheffield at SemEval-2017 task 9: Transition-based language generation from AMR. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 586–591, Vancouver, Canada. Association for Computational Linguistics.

Alon Lavie and Abhaya Agarwal. 2007. METEOR: An automatic metric for MT evaluation with high levels of correlation with human judgments. In *Proceedings of the 2nd Workshop on Statistical Machine Translation*, pages 228–231, Prague, Czech Republic. Association for Computational Linguistics.

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.

Kexin Liao, Logan Lebanoff, and Fei Liu. 2018. Abstract meaning representation for multi-document summarization. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1178–1190, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Shuming Ma, Pengcheng Yang, Tianyu Liu, Peng Li, Jie Zhou, and Xu Sun. 2019. Key fact as pivot: A two-stage model for low resource table-to-text generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2047–2057, Florence, Italy. 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.

Jonathan May and Jay Priyadarshi. 2017. SemEval-2017 task 9: Abstract Meaning Representation parsing and generation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 536–545, Vancouver, Canada. Association for Computational Linguistics.

Simon Mille, Roberto Carlini, Alicia Burga, and Leo Wanner. 2017. FORGe at SemEval-2017 task 9: Deep sentence generation based on a sequence of graph transducers. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 920–923, Vancouver, Canada. Association for Computational Linguistics.

Abelardo Vieira Mota, Ticiana Linhares Coelho da Silva, and José Antônio Fernandes De Macêdo. 2020. Template-based multi-solution approach for data-to-text generation. In *Advances in Databases and Information Systems: 24th European Conference, ADBIS 2020, Lyon, France, August 25–27, 2020, Proceedings*, page 157–170, Berlin, Heidelberg. Springer-Verlag.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Maja Popović. 2017. chrF++: words helping character n-grams. In *Proceedings of the Second Conference on Machine Translation*, pages 612–618, Copenhagen, Denmark. 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.

Ratish Puduppully and Mirella Lapata. 2021. Data-to-text Generation with Macro Planning. *Transactions of the Association for Computational Linguistics*, 9:510–527.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Ehud Reiter. 2018. Hallucination in neural NLG.

Leonardo F. R. Ribeiro, Martin Schmitt, Hinrich Schütze, and Iryna Gurevych. 2021a. 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.

Leonardo F. R. Ribeiro, Yue Zhang, and Iryna Gurevych. 2021b. Structural adapters in pretrained language models for AMR-to-Text generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4269–4282, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Marco A. Sobrevilla Cabezudo and Thiago A.S. Pardo. 2022. Low-resource amr-to-text generation: A study on brazilian portuguese. *Procesamiento del Lenguaje Natural*, 68.

Marco Antonio Sobrevilla Cabezudo and Thiago Pardo. 2019. Towards a general abstract meaning representation corpus for Brazilian Portuguese. In *Proceedings of the 13th Linguistic Annotation Workshop*, pages 236–244, Florence, Italy. 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.

Linfeng Song, Xiaochang Peng, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2017. AMR-to-text generation with synchronous node replacement grammar. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 7–13, Vancouver, Canada. Association for Computational Linguistics.

Yixuan Su, Deng Cai, Yan Wang, David Vandyke, Simon Baker, Piji Li, and Nigel Collier. 2021. Non-autoregressive text generation with pre-trained language models. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 234–243, Online. Association for Computational Linguistics.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.