# Efficient AMR parsing with CLAP: Compact Linearization with an Adaptable Parser

**Abelardo Carlos Martinez Lorenzo and Roberto Navigli**
Sapienza University of Rome
Sapienza NLP Group
{martinezlorenzo, navigli}@diag.uniroma1.it

**Abstract**

Sequence-to-sequence models have become the de facto standard for Abstract Meaning Representation (AMR) parsing due to their high-quality performance. However, these systems face efficiency challenges because of their large model size and computational time, which limit their accessibility within the research community. This paper aims to break down these barriers by introducing a novel linearization and system that significantly enhances the efficiency and accessibility of previous AMR parsers. First, we propose our novel Compact linearization that simplifies encoding, thereby reducing the number of tokens by between 40% and 50%. Second, we present CLAP, an innovative modular system that maintains the model's high performance while achieving remarkable 80% reduction in training and inference times. Furthermore, CLAP is compatible with multiple autoregressive Language Models (LM) and tokenizers, such as BART, T5, and others. These advancements underscore the importance of optimizing sequence-to-sequence models in AMR parsing, thus democratizing access to high-quality semantic analysis. Our code is publicly available at `https://github.com/SapienzaNLP/clap/`.

**Keywords:** Semantic Parsing, Efficiency, Abstract Meaning Representation, AMR parsing

## 1. Introduction

Semantic Parsing is the subfield of Natural Language Understanding that aims to encode the meaning of natural language into a machine-interpretable structure. In the last decade, one of the formalisms that has attracted the most attention is Abstract Meaning Representation (Banarescu et al., 2013, AMR). AMR embeds the semantics of a sentence in a directed acyclic graph, where concepts are represented by nodes, semantic relations between concepts by edges, and co-references by reentrant nodes. Since AMR can act as an interface that is easy to read for humans and easy to interpret for machines, it has been applied to multiple NLP areas, including Question Answering (Lim et al., 2020; Bonial et al., 2020b; Kapanipathi et al., 2021), Text Summarization (Hardy and Vlachos, 2018; Liao et al., 2018), Human-Robot Interaction (Bonial et al., 2020a), Information Extraction (Rao et al., 2017), Machine Translation (Song et al., 2019), and has also been extended to non-English languages (Anchiêta and Pardo, 2020; Wein et al., 2022; Linh and Nguyen, 2019; Azin and Eryiğit, 2019; Blloshmi et al., 2020; Navigli et al., 2022; Martínez Lorenzo et al., 2022).

State-of-the-art AMR parsers currently depend on transformer sequence-to-sequence architectures, as introduced into the task by Bevilacqua et al. (2021) through SPRING, an autoregressive model based on BART (Lewis et al., 2020). SPRING converts texts into linearized representations of AMR graphs (see linearization in Figure 1). In the post-processing, the linearization is converted into an AMR graph.

```
(z0 / move-01
  :ARG0 (z1 / person
    :name (z2 / name
      :op1 "Pablo"
      :op2 "Picasso" ) )
  :ARG1 (z3 / hometown
      :poss z1
    :ARG0-of (z4 / have-location-91
      :ARG1 (z5 / city
        :name (z6 / name
           :op1 "Malaga" ) ) ) )
  :ARG2 (z7 / city
    :name (z8 / name
      :op1 "A"
      :op2 "Coruña" ) )
  :time (z9 / date-entity
    :year 1891 ) )
```

Figure 1: AMR graph of: "In 1891, Pablo Picasso moved from his hometown Málaga to A Coruña".

In an attempt to push SMATCH (Cai and Knight, 2013) performance, there has been a recent emphasis towards the creation of larger models with more parameters and more complex training pipelines, such as pre-training on structural graph information (Bai et al., 2022), incorporating shallow semantic information (Chen et al., 2022), modifying ancestor information during decoding (Yu and Gildea, 2022), enabling cross-lingual alignment extraction (Martínez Lorenzo et al., 2023b), adding a structural graph information task during training (Cheng et al., 2022; Vasylenko et al., 2023), or even ensemble AMR graphs (Martínez Lorenzo et al., 2023a). This trend has resulted in significant efficiency challenges, whereby the extensive training times and high computational costs involved in such initiatives hinder their widespread adoption and accessibility for researchers with limited resources.

Therefore, the objective of this paper is to make AMR parsing more efficient and make it more accessible to the research community. This is achieved through the following contributions:

- Introducing a Compact Linearization, which reduces the total number of tokens by approximately 40-50%.
- Presenting the CLAP model, which is an efficient and modular AMR parser, leading to a reduction of 80% of train and inference time compared to previous parsers.
- Conducting a comparative analysis of the performance of popular autoregressive models, such as T5 and BART, in AMR parsing.

## 2. Linearization

The autoregressive nature of AMR parsers makes decoding challenging because longer predictions slow down model inference. To address this issue, we introduce Compact linearization, a technique that maximizes compression.

### 2.1. Penman

Initially, AMR parsers followed the Penman (PM) linearization, which is the encoding used in the release files of the official dataset AMR 3.0 (Knight et al., 2020) (see in Figure 1 with 83 tokens). However, LMs and their tokenizers struggle with this linearization due to: i) the presence of redundant tokens, such as slashes, ii) hard-to-process tokens, like quotation marks; and iii) the lack of special tokens for node identifiers, which are merely letters followed by numbers (e.g. z0).

### 2.2. DFS

Bevilacqua et al. (2021) introduced a Depth-First Search (DFS)–based linearization which encodes AMR graphs to make them readable for autoregressive models and isomorphic in order to revert them to canonical AMR graphs. This is achieved by introducing special tokens for node identifiers (<p:0>), and for identifying literals (<l>). Figure 2 exemplifies DFS-based linearization (71 tokens).

```
( <p:0> move-01
  :ARG0 ( <p:1> person
    :name ( <p:2> name
      :op1 <l> Pablo <\l>
      :op2 <l> Picasso <\l> ) )
  :ARG1 ( <p:3> hometown
    :poss <p:1>
    :ARG0-of (<p:4> have-location-91
      :ARG1 ( <p:5> city
         :name (<p:6> name
         :op1 <l> Malaga <\l>))))
  :ARG2 ( <p:7> city
    :name ( <p:8> name
      :op1 <l> A <\l>
      :op2 <l> Coruña <\l> ) )
  :time ( <p:9> date-entity
    :year 1891 ) )
```

Figure 2: DFS-based linearization of Figure 1.

```
move-01
    :ARG0 person
        :name Pablo Picasso )
        :ARG1 hometown
            :poss person
            :location city
                :name Malaga )1
    :ARG2 city#2
        :name A Coruña )
    :time date-entity
        :year 1891 )
```

Figure 3: Our Compact linearization of Figure 1.

### 2.3. Compact

Nevertheless, the DFS-based linearization still exhibits redundancies and complex structures that could be simplified, such as in named entity structures (see the Pablo Picasso representation in Figure 2). To address these issues, we propose our novel Compact linearization. Firstly, we eliminate redundant tokens like slashes and parentheses, retaining only the closing parenthesis when returning to a parent node. Additionally, we remove wiki tags from the raw AMR graphs, as the parser's primary objective is to establish the underlying structure. The incorporation of wikilinks is deferred until later stages using BLINK (Wu et al., 2020). Subsequently, we streamline redundant structures, including named and URL entities, and dereify node structures like *have-location-91*. Lastly, we eliminate the variables (<*p:X*>) within the graphs, preserving solely the original node words. A unique identifier is appended to the new concept when the graphs contain duplicate concepts. In this way, we reduced the total amount of tokens for our example to 28 (see Figure 3).

## 3. CLAP Model

The latest autoregressive AMR parsers are built on top of SPRING (Bevilacqua et al., 2021), an autoregressive model that converts texts into linearized representations of AMR graphs. However, the SPRING system exhibits significant inefficiencies in training and inference, and the specialized tokenizer and rule-based post-processing also hinder its adaptability for leveraging other LMs. As a solution, we propose our Compact Linearization with an Adaptable Parser (CLAP), which integrates the previous Compact Linearization, simplifies the model tokenizer and post-processing and leverages the latest NLP frameworks in order to enhance the training phase.

### 3.1. The Model's Vocabulary

SPRING relies on a complex pipeline for tokenizing the DFS-based linearization, which involves adding over 3,000 new tokens to accommodate all possible pointers (e.g., <p:0>, <p:1>, etc.), relations (e.g., *:ARG0*, *:ARG1*, *:ARG0-of*, etc.), and PropBank (Palmer et al., 2005) frames (e.g., move-01). While this approach may reduce the total number of tokens by not splitting spans like "move-01" into separated tokens ("move" and "-01"), it also leads to an increase in model size, impacting memory usage and restricting the maximum number of tokens per batch. Furthermore, introducing special tokens for each edge, such as *:op1* and *:op2*, hampers the model's ability to learn sequential relationships. Notably, in the case of lengthy names, the model struggles to discern the order between *:opX* and *:op(X+1)*. To address these issues, our modified vocabulary includes only the 25 different relations of AMR, along with special tokens for node differentiation (e.g., *#1*, *#2*, etc.), a token for inverse relations (*-of*), and additional tokens to differentiate and preserve the meaning of frames through combinations (e.g., *-01*, *-02*, etc.). This approach results in a smaller vocabulary, enhancing the efficiency of the model.

### 3.2. Tokenizer

SPRING faces a significant challenge with its specialized tokenizer, which struggles to handle complex structures and hinders seamless integration with different autoregressive models. However, thanks to our Compact linearization method and small extra vocabulary, our system uses the original tokenizer of the selected language model without modifications. Consequently, our system can seamlessly integrate with other autoregressive models, such as mBART, T5, and Long-T5. The simplified tokenizer facilitates flexible model selection and opens doors for future advancements and explorations in the AMR parsing domain.

### 3.3. Decoding Graphs

After SPRING has generated the DFS-based linearized graph, it is transformed into PM graphs. However, this process involves multiple manual rules to address potential errors introduced by the model, such as hallucinations or corrupted predictions. These rules are tightly coupled with the tokenizer, limiting the model's portability across different model architectures. In contrast, our new approach eliminates the need to revert to PM linearization. Instead, we iterate over the predicted sequence, extracting the triplets directly. Then, we construct a new AMR graph based on the list of triplets. This strategy allows our system to adapt to other autoregressive models without needing manually created, rule-based corrections.

### 3.4. Efficient Training

To ensure optimal training efficiency, we leverage the latest version of PyTorch Lightning (Falcon and The PyTorch Lightning team, 2019). In addition, we utilize Adafactor (Shazeer and Stern, 2018) as our optimizer and implement a dynamic batch size strategy based on the number of tokens; we maximize GPU utilization and significantly expedite the training process. This adaptive approach ensures optimal resource allocation, resulting in a simplified system architecture that is adaptable and has higher training efficiency.

## 4. Experimental Setup

### 4.1. Dataset

We evaluate our model using the *AMR 3.0* corpus (LDC2020T02), which contains 55635, 1722, and 1898 sentence-AMRs in the training, validation, and test sets, respectively.

### 4.2. Metrics

Our work is evaluated across multiple dimensions: (i) comparing the total number of tokens in the training and test sets, assessing the efficiency of our novel linearization approach; (ii) measuring training time per epoch and inference time on the test set in order to evaluate the efficiency of our models; and (iii) employing the SMATCH metric (Cai and Knight, 2013) in order to assess the parsing performance, which quantifies the similarity between AMR graphs based on triplet overlapping.

|  |  | Model |  |  |  | Training |  |  |  | Inference |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
|  |  | name | Hyper. | Vocab. | Linear. | B. Size | B. Total | Tokens | Time (h) | B. Size | B. Total | Tokens | Time (s) | SMATCH |
| T5 | small | CLAP | 60M | 32.279 | PM | 15,000 | 946 | 7,149,589 | 7 | 10,000 | 34 | 308,682 | 240 | 76.3 |
|  |  | CLAP | 60M | 32,279 | DFS | 15,000 | 482 | 5,185,376 | 4 | 10,000 | 25 | 221,472 | 94 | 80.1 |
|  |  | CLAP | 60M | 32,279 | CMP | 15,000 | **260** | **2,529,214** | **2** | 10,000 | **13** | **107,699** | **37** | **80.2** |
| BART | base | CLAP | 140M | 50,459 | PM | 12,000 | 631 | 5,558,900 | 8 | 10,000 | 27 | 242,714 | 80 | 80.4 |
|  |  | CLAP | 140M | 50,459 | DFS | 12,000 | 452 | 4,376,811 | 4 | 10,000 | 24 | 187,947 | 65 | **81.4** |
|  |  | CLAP | 140M | 50,459 | CMP | 12,000 | **304** | **2,476,757** | **3** | 10,000 | **11** | **103,732** | **29** | 81.3 |
| T5 | base | CLAP | 220M | 32,279 | PM | 6,500 | 1,423 | 7,149,589 | 10 | 10,000 | 34 | 308,682 | 180 | 77.5 |
|  |  | CLAP | 220M | 32,279 | DFS | 6,500 | 1,046 | 5,185,376 | 6 | 10,000 | 25 | 221,472 | 149 | **82.5** |
|  |  | CLAP | 220M | 32,279 | CMP | 6,500 | **549** | **2,529,214** | **4** | 10,000 | **13** | **107,699** | **37** | **82.5** |
| BART | large | CLAP | 400M | 50,459 | PM | 6,000 | 1,206 | 5,558,900 | 17 | 10,000 | 27 | 242,714 | 147 | 81.5 |
|  |  | SPRING | 400M | 53,587 | DFS | 4,000 | 2,806 | 4,040,794 | 45 | 10,000 | 51 | 180,086 | 134 | 83.0 |
|  |  | CLAP | 400M | 50,459 | DFS | 6,000 | 1085 | 4,376,811 | 14 | 10,000 | 21 | 187,947 | 67 | 83.0 |
|  |  | CLAP | 400M | 50,459 | CMP | 6,000 | **567** | **2,476,757** | **8** | 10,000 | **11** | **103,732** | **29** | **83.1** |
| T5 | large | CLAP | 770M | 32,279 | PM | 2,000 | 4,477 | 7,149,589 | 37 | 10,000 | 34 | 308,682 | 403 | 83.0 |
|  |  | CLAP | 770M | 32,279 | DFS | 2,000 | 3,257 | 5,185,376 | 27 | 10,000 | 25 | 221,472 | 227 | **84.0** |
|  |  | CLAP | 770M | 32,279 | CMP | 2,000 | **1,670** | **2,529,214** | **12** | 10,000 | **13** | **107,699** | **98** | **84.0** |

Table 1: Results through different systems and linearizations. Block best performance in bold. Row Blocks: models by linearization. Column blocks: Model - model name, number of hyperparameters, vocab size, type of linearization; Training - max batch size, total number of batches, total number of tokens in the training dataset, time to train 50 epochs in hours; Inference - max batch size, total number of batches, total number of tokens in the test dataset, inference time of test dataset in seconds, and SMATCH score.

### 4.3. Models

We use SPRING as our baseline system, which is based on the BART (Lewis et al., 2020) model. In addition, we compare with our novel CLAP system based on BART base and large, and Flan-T5 (Chung et al., 2022) small, base and large. Furthermore, to assess the benefits of our Compact linearization, we train our system with three linearizations: penman (PM), DFS-based (DFS), and Compact (CMP). We highlight that we focus our comparisons on SPRING rather than on the latest models, since these subsequent approaches build on top of SPRING, just modifying the training strategies. Our main goal is to emphasize the distinction between different linearization techniques and different systems, rather than that between different training strategies. To ensure a fair comparison, we do not apply BLINK as a post-processing phase to improve the Wikifications, and we carefully allocate the maximum batch size for each model to fit the available GPU resources while maintaining stable training conditions. The hyperparameters for our experiments include using the Adafactor optimizer, 0.15 for Dropout, 0.0 for Attention Dropout, 0.01 for weight decay, and learning rates of 0.001 for T5 and 0.0001 for BART. We utilize the Inverse Square Root (sqrt) learning rate scheduler and employ a Beam size 5 for our experiments.

## 5. Results

Table 1 comprehensively compares the number of tokens processed and computation time between SPRING and our CLAP system based on BART and T5 and using PM, DFS and CMP linearization. When we consider SMATCH performance, we observe that our CMP linearization and CLAP system closely align with DFS and the original SPRING.

### 5.1. Linearization

The DFS-based linearization reduces the graph number of tokens by 25-30% compared to PM. However, our CMP linearization surpasses DFS compression, achieving 55-65% reduction against PM and around 45-55% with respect to DFS. This enables faster processing, as the models only generate half of the tokens to construct the graphs. Therefore, the CLAP model training time with CMP linearization is on average 60% faster than using PM and 50% faster than using DFS linearization, and 75% and 65% faster in inference time, respectively.

### 5.2. Systems

Comparing our DFS-based CLAP built on BART-large against SPRING, our system generates more tokens since its vocabulary is not specialized to DFS-based linearization (e.g., CLAP does not have special tokens for the node identifiers, so they are split). However, our system manages the batches more efficiently, making them three times smaller than SPRING, reducing the training time by 69%. Additionally, the inference time decreases by 50%. Furthermore, comparing SPRING against our CLAP system based on BART-large using CMP, we reduce the training time by 82% and inference time by 78%.

### 5.3. Compatibility

In contrast to SPRING, CLAP demonstrates enhanced modularity by adapting to different T5 and BART-based models. This adaptability underscores a noteworthy observation: BART-based models inherently generate fewer tokens than their T5 counterparts. This difference in tokenization arises from their inherent tokenization behaviour. While T5 relies on Unigram, BART relies on Byte Pair Encoding (BPE), resulting in vocabulary sizes of 32,279 and 50,459, respectively. Consequently, the larger vocabulary allocation of BART splits words into fewer tokens. Nevertheless, our CMP approach effectively mitigates this tokenization disparity, resulting in a consistent number of tokens between models.

## 6. Conclusion

Through our Compact linearization and our efficient CLAP system, we enhance AMR parsing by boosting its efficiency and flexibility. The Compact linearization reduces the number of tokens by 50%, and our CLAP system cuts training and inference times by 80%, while maintaining parsing performance. Our modular system ensures easy transitions between different LMs, enhancing adaptability in AMR parsing, and paving the way for new research and practical applications. We make our code publicly available for the community in the following GitHub repository `https://github.com/SapienzaNLP/clap/`.

## Acknowledgments

The first author gratefully acknowledges the support of the European Union's Horizon 2020 research project *Knowledge Graphs at Scale* (KnowGraphs) under the Marie Marie Skłodowska-Curie grant agreement No 860801.

The authors also gratefully acknowledge the support of the CREATIVE project (CRoss-modal understanding and gEnerATIon of Visual and tExtual content), which is funded by the MUR Progetti di Rilevante Interesse Nazionale programme (PRIN 2020).

## 7. Bibliographical References

Rafael Anchiêta and Thiago Pardo. 2020. Semantically inspired AMR alignment for the Portuguese language. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1595–1600, Online. Association for Computational Linguistics.

Zahra Azin and Gülşen Eryiğit. 2019. Towards Turkish Abstract Meaning Representation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 43–47, Florence, Italy. Association for Computational Linguistics.

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, Sofia, Bulgaria. Association for Computational Linguistics.

Michele Bevilacqua, Rexhina Blloshmi, and Roberto Navigli. 2021. One SPRING to Rule Them Both: Symmetric AMR semantic Parsing and Generation without a Complex Pipeline. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(14):12564–12573.

Rexhina Blloshmi, Rocco Tripodi, and Roberto Navigli. 2020. XL-AMR: Enabling cross-lingual AMR parsing with transfer learning techniques. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2487–2500, Online. Association for Computational Linguistics.

Claire Bonial, Lucia Donatelli, Mitchell Abrams, Stephanie M. Lukin, Stephen Tratz, Matthew Marge, Ron Artstein, David Traum, and Clare Voss. 2020a. Dialogue-AMR: Abstract Meaning Representation for dialogue. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 684–695, Marseille, France. European Language Resources Association.

Claire Bonial, Stephanie M. Lukin, David Doughty, Steven Hill, and Clare Voss. 2020b. InfoForager: Leveraging semantic search with AMR for COVID-19 research. In *Proceedings of the Second International Workshop on Designing Meaning Representations*, pages 67–77, Barcelona Spain (online). Association for Computational Linguistics.

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.

Liang Chen, Peiyi Wang, Runxin Xu, Tianyu Liu, Zhifang Sui, and Baobao Chang. 2022. ATP: AMRize then parse! enhancing AMR parsing with PseudoAMRs. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 2482–2496, Seattle, United States. Association for Computational Linguistics.

Ziming Cheng, Zuchao Li, and Hai Zhao. 2022. BiBL: AMR parsing and generation with bidirectional Bayesian learning. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 5461–5475, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models.

William Falcon and The PyTorch Lightning team. 2019. PyTorch Lightning. Technical report.

Hardy Hardy and Andreas Vlachos. 2018. Guided neural language generation for abstractive summarization using Abstract Meaning Representation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 768–773, Brussels, Belgium. Association for Computational Linguistics.

Pavan Kapanipathi, Ibrahim Abdelaziz, Srinivas Ravishankar, Salim Roukos, Alexander Gray, Ramón Fernandez Astudillo, Maria Chang, Cristina Cornelio, Saswati Dana, Achille Fokoue, Dinesh Garg, Alfio Gliozzo, Sairam Gurajada, Hima Karanam, Naweed Khan, Dinesh Khandelwal, Young-Suk Lee, Yunyao Li, Francois Luus, Ndivhuwo Makondo, Nandana Mihindukulasooriya, Tahira Naseem, Sumit Neelam, Lucian Popa, Revanth Gangi Reddy, Ryan Riegel, Gaetano Rossiello, Udit Sharma, G P Shrivatsa Bhargav, and Mo Yu. 2021. Leveraging Abstract Meaning Representation for knowledge base question answering. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3884–3894, Online. Association for Computational Linguistics.

Kevin Knight, Bianca Badarau, Laura Baranescu, Claire Bonial, Madalina Bardocz, Kira Griffitt, Ulf Hermjakob, Daniel Marcu, Martha Palmer, Tim O'Gorman, et al. 2020. Abstract Meaning Representation (AMR) Annotation Release 3.0 (LDC2020T02). Philadelphia. Linguistic Data Consortium.

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.

Jungwoo Lim, Dongsuk Oh, Yoonna Jang, Kisu Yang, and Heuiseok Lim. 2020. I know what you asked: Graph path learning using AMR for commonsense reasoning. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2459–2471, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Ha Linh and Huyen Nguyen. 2019. A case study on meaning representation for Vietnamese. In *Proceedings of the First International Workshop on Designing Meaning Representations*, pages 148–153, Florence, Italy. Association for Computational Linguistics.

Abelardo Carlos Martínez Lorenzo, Pere Lluís Huguet Cabot, and Roberto Navigli. 2023a.

AMRs assemble! learning to ensemble with autoregressive models for AMR parsing. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1595–1605, Toronto, Canada. Association for Computational Linguistics.

Abelardo Carlos Martínez Lorenzo, Pere Lluís Huguet Cabot, and Roberto Navigli. 2023b. Cross-lingual AMR aligner: Paying attention to cross-attention. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1726–1742, Toronto, Canada. Association for Computational Linguistics.

Abelardo Carlos Martínez Lorenzo, Marco Maru, and Roberto Navigli. 2022. Fully-Semantic Parsing and Generation: the BabelNet Meaning Representation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1727–1741, Dublin, Ireland. Association for Computational Linguistics.

Roberto Navigli, Rexhina Blloshmi, and Abelardo Carlos Martinez Lorenzo. 2022. BabelNet Meaning Representation: A Fully Semantic Formalism to Overcome Language Barriers. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36.

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

Sudha Rao, Daniel Marcu, Kevin Knight, and Hal Daumé III. 2017. Biomedical event extraction using Abstract Meaning Representation. In *BioNLP 2017*, pages 126–135, Vancouver, Canada,. Association for Computational Linguistics.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 4603–4611. PMLR.

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.

Pavlo Vasylenko, Pere Lluís Huguet Cabot, Abelardo Carlos Martínez Lorenzo, and Roberto Navigli. 2023. Incorporating graph information in transformer-based AMR parsing. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1995–2011, Toronto, Canada. Association for Computational Linguistics.

Shira Wein, Lucia Donatelli, Ethan Ricker, Calvin Engstrom, Alex Nelson, Leonie Harter, and Nathan Schneider. 2022. Spanish Abstract Meaning Representation: Annotation of a general corpus. In *Northern European Journal of Language Technology, Volume 8*, Copenhagen, Denmark. Northern European Association of Language Technology.

Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. 2020. Scalable zero-shot entity linking with dense entity retrieval. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6397–6407, Online. 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.