ÚFAL at MRP 2020: Permutation-invariant Semantic Parsing in PERIN

David Samuel and **Milan Straka**
Charles University
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics
{samuel,straka}@ufal.mff.cuni.cz

Abstract

We present PERIN, a novel permutation-invariant approach to sentence-to-graph semantic parsing. PERIN is a versatile, cross-framework and language independent architecture for universal modeling of semantic structures. Our system participated in the CoNLL 2020 shared task, Cross-Framework Meaning Representation Parsing (MRP 2020), where it was evaluated on five different frameworks (AMR, DRG, EDS, PTG and UCCA) across four languages. PERIN was one of the winners of the shared task. The source code and pretrained models are available at `http://www.github.com/ufal/perin`.

1 Introduction

The aim of the CoNLL 2020 shared task, Cross-Framework Meaning Representation Parsing (MRP 2020; Oepen et al., 2020), is to translate plain text sentences into their corresponding graph-structured meaning representation.[1] MRP 2020 features five formally and linguistically different frameworks with varying degrees of linguistic and structural complexity:

- **AMR:** Abstract Meaning Representation (Banarescu et al., 2013);
- **DRG:** Discourse Representation Graphs (Abzianidze et al., 2017) provide a graph encoding of Discourse Representation Structure (Van der Sandt, 1992);
- **EDS:** Elementary Dependency Structures (Oepen and Lønning, 2006);
- **PTG:** Prague Tectogrammatical Graphs (Hajic et al., 2012);
- **UCCA:** Universal Conceptual Cognitive Annotation (Abend and Rappoport, 2013).

These frameworks constitute the *cross-framework* track of MRP 2020, while the separate *cross-lingual* track introduces one additional language for four out of the five frameworks: Chinese AMR (Li et al., 2016), German DRG, Czech PTG and German UCCA (Hershcovich et al., 2019).

In agreement with the shared task objective to advance uniform meaning representation parsing across diverse semantic graph frameworks and languages, we propose a language and structure agnostic sentence-to-graph neural network architecture modeling semantic representations from input sequences.

The main characteristics of our approach are:

- **Permutation-invariant model:** PERIN is, to our best knowledge, the first graph-based semantic parser that predicts all nodes at once in parallel and trains them with a permutation-invariant loss function. Semantic graphs are naturally *orderless*, so constraining them to an artificial node ordering creates an unfounded restriction; furthermore, our approach is more expressive and more efficient than *order-based* auto-regressive models.
- **Relative encoding:** We present a substantial improvement of relative encodings of node labels, which map anchored tokens onto label strings (Straka and Straková, 2019). Our novel formulation allows using a richer set of encoding rules.
- **Universal architecture:** Our work presents a general sentence-to-graph pipeline adaptable for specific frameworks only by adjusting pre-processing and post-processing steps.

Our model was ranked among the two winning systems in both the *cross-framework* and the *cross-lingual* tracks of MRP 2020 and significantly advanced the accuracy of semantic parsing from the last year's MRP 2019.

[1] See `http://mrp.nlpl.eu/2020/` for more details.

2 Related Work

Examples of general, formalism-independent semantic parsers are scarce in the literature. Hershcovich et al. (2018) propose a universal transition-based parser for directed, acyclic graphs, capable of parsing multiple conceptually and formally different schemes. Furthermore, several participants of MRP 2019 presented universal parsers. Che et al. (2019) improved uniform transition-based parsing and used a different set of actions for each framework. Lai et al. (2019) submitted a transition-based parser with shared actions across treebanks, but failed to match the performance of the other parsers. Straka and Straková (2019) presented a general graph-based parser, where the meaning representation graphs are created by repeatedly adding nodes and edges.

Graph-based parsers (McDonald and Pereira, 2006; Peng et al., 2017; Dozat and Manning, 2018; Cai and Lam, 2020) usually predict nodes in a sequential, auto-regressive manner and then connect them with a biaffine classifier. Unlike these approaches, our model infers all nodes in parallel while allowing the creation of rich intermediate representations by node-to-node self-attention.

Machine learning tools able to efficiently process unordered sets are gaining more attention in recent years. Qi et al. (2017) and particularly Zhang et al. (2019b) proposed permutation-invariant neural networks for point clouds, which are of great relevance to our system. Our work was further inspired by Carion et al. (2020), who utilize permutation invariance for object detection in a similar fashion to our sentence-to-graph generation.

3 Methods

3.1 Graph Representation

All five semantic formalisms share the same representation via directed labeled multigraphs in the graph interchange format proposed by Kuhlmann and Oepen (2016). Universally, the semantic units are represented by nodes and the semantic relationships by labeled edges. Each node can be anchored to a (possibly empty) set of input characters, and can contain a (possibly empty) list of properties, each being an attribute-value pair.

We simplify this graph structure by turning the properties into graph nodes: every property $\{$`attribute : value`$\}$ of node n is removed and a new node with label `value` is connected to the parent node n by an edge labeled with `attribute`; the anchors of the new node are the same as of its parent.[2] Figure 4 illustrates this transformation together with other pre-processing steps (specific for each framework) explained in detail in Section 3.7.

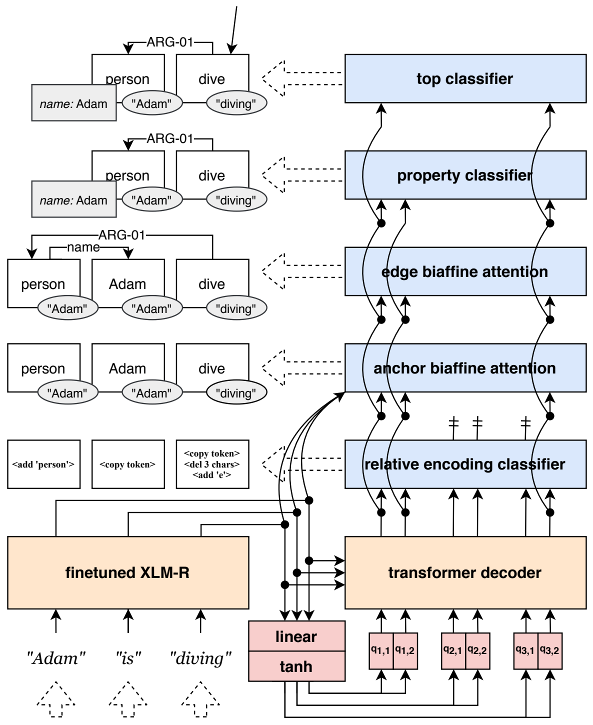

Figure 1: Data flow through PERIN during inference. Every input token is processed by an encoder and transformed into multiple queries, which are further refined by a decoder. Each query is either *denied* or *accepted*, and the accepted ones are then gradually processed into the final semantic graph.

Another change to the internal graph representation is the use of relative label encoding (discussed in Section 3.4), which substitutes the original node labels by lists of relative encoding rules.

3.2 Overall Architecture

A simplified illustration of the whole model can be seen in Figure 1. The input is tokenized, an encoder (Section 3.5) computes contextual embeddings of the tokens, and each embedded token $\mathbf{e}_i$ is then mapped onto Q queries by nonlinear transformations $\mathbf{q}_{i,t} = \tanh\left(\mathbf{W}_t\mathbf{e}_i + \mathbf{b}_t\right), t \in \{1, \ldots Q\}$, where $\mathbf{W}_t$ is a trainable weight matrix and $\mathbf{b}_t$

[2]"Nodeification" of properties was motivated by the nature of AMR graphs, where the properties are equivalent to instanced concepts/nodes (Banarescu et al., 2013). From a more practical viewpoint, it allows us to utilize a single classifier for both the node labels and the less-frequent properties, and to simplify the whole architecture.

is a trainable bias vector. After that, a decoder (Transformer with pre-norm residual connections (Nguyen and Salazar, 2019) and cross-attention into the contextual embeddings $\mathbf{e}_i$) processes the queries, obtaining their final feature vectors $\mathbf{h}_{i,t}$. These feature vectors are shared across all classification heads, each inferring specific aspects of the final meaning representation graph from them:

- **Relative encoding classifier** decides what node label should serve as the "answer" to each query; a query can also be *denied* (no node is created) when classified as "null". Relative label prediction is described in detail in Section 3.4.3.
- **Anchor biaffine classifier** uses deep biaffine attention (Dozat and Manning, 2017) to create anchors between nodes and surface tokens – to be more precise, the biaffine attention processes the latent vectors of queries $\mathbf{h}_{i,t}$ and tokens $\mathbf{e}_j$, and predicts the presence of an anchor between every pair of them as a binary classification task.
- **Edge biaffine classifier** uses three biaffine attention modules to predict whether an edge should exist between a pair of nodes (*presence* binary classification), what label(s) should it have (*label* multi-class or multi-label classification, depending on the framework) and what attribute should it have (*attribute* multi-class classification) – in essence, this module is a simple extension of the standard edge classifier by Dozat and Manning (2018).
- **Property classifier** uses a linear layer followed by a sigmoid nonlinearity to identify nodes that should be converted to properties.
- **Top classifier** uses a linear layer followed by a softmax nonlinearity (where the probabilities are normalized across nodes) to detect the *top* node.

This section described all modules capable of handling different characteristics of meaning representation graphs. Not all of them appear in each framework – for example, AMR graphs do not need edge attributes, while UCCA graphs do not contain any properties. More details about specific framework configurations are given in Section 3.7.

3.3 Permutation-invariant Graph Generation

Semantic graphs are *orderless*, so it is unnatural to constrain their generation by some artificial node ordering. Traditionally, graph nodes have been predicted by a sequence-to-sequence model (Peng et al., 2017), with the nodes being generated in some hardwired order (Zhang et al., 2019a). Demanding a fixed node ordering causes the *discontinuity issue* (Zhang et al., 2019b): even when correct items are predicted, they are viewed as completely wrong if not in the expected order. We avoid this issue by using such a loss function and such a model that produce the same outcome independently on the node ordering (Zaheer et al., 2017).

3.3.1 Permutation-equivariant Model

We transform the queries $\mathbf{q} = \{\mathbf{q}_i\}_{i=1}^N$ into hidden features $\mathbf{h} = \{\mathbf{h}_i\}_{i=1}^N$ in such manner that any permutation $\pi \in \mathfrak{S}_N$ of the input $\pi(\mathbf{q}) = \{\mathbf{q}_{\pi(i)}\}_{i=1}^N$ produces the same – but permuted – output $\pi(\mathbf{h}) = \{\mathbf{h}_{\pi(i)}\}_{i=1}^N$. The Transformer architecture (Vaswani et al., 2017) conveniently fulfills this requirement (assuming positional embeddings are not used). Furthermore, it can combine any pair of input items independently of their distance and in an efficient non-autoregressive way.

3.3.2 Permutation-invariant Loss

The hidden features $\mathbf{h}$ are further refined into predictions $\hat{\mathbf{y}} = f_{\boldsymbol{\theta}}(\mathbf{h})$ by the classification heads. In order to create a permutation-invariant loss function, i.e., a function $\mathcal{L}(\pi(\hat{\mathbf{y}}), \mathbf{y})$ giving the same result for every $\pi \in \mathfrak{S}_N$, we find a permutation $\pi^* \in \mathfrak{S}_N$ assigning each query to its most similar node. After permuting the targets according to π^*, the standard losses can be computed, because they are no longer dependent on the original ordering of $\hat{\mathbf{y}}$ and $\mathbf{y}$.[3]

To find the minimizing permutation π^*, we start by extending the (multi)set of target nodes $\mathbf{y}$ by "null" nodes (denoted as $\varnothing$) in order to fulfill $|\hat{\mathbf{y}}| = |\mathbf{y}|$. When classified as "null" during inference, the query is *denied* and omitted from further processing. The permutation π^* is then defined as

$$\pi^* = \underset{\pi \in \mathfrak{S}_N}{\arg\max} \sum_{i=1}^{N} p_{\text{match}}(\hat{\mathbf{y}}_i, \mathbf{y}_{\pi(i)}), \qquad (1)$$

where the matching score p_{match} is composed of a *label score* and the geometric mean (GM) of

[3]Unfortunately, the uniqueness of π^* is not always guaranteed: given that the proposed matching depends only on labels and anchors, there might be multiple equivalent nodes (considering only labels and anchors). We break ties between such nodes by also minimizing the likelihood of their edges across all their permutations.

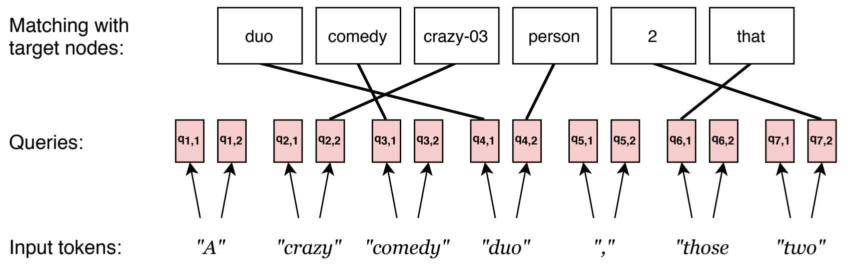

Figure 2: Example of a matching between queries and target nodes during training. Every input token is mapped onto Q (2 in this case) queries $\mathbf{q}_{i,j}$, which are decoded into node predictions $\hat{\mathbf{y}}_{i,j}$. These predictions are paired with the ground truth nodes $\mathbf{y}$, as in Equation 1. Then, the loss functions are computed with respect to the paired target nodes. Queries without any match should be classified as "null" nodes. When classified as "null" during inference, the query is not turned into any node (the query is *denied*).

anchor scores of all input tokens T. The *label score* of the i^{th} query and the j^{th} node is defined as the predicted probability of the target j^{th} label; the *anchor score* of the i^{th} query, j^{th} node and a token $t \in T$ is defined as the predicted probability of the actual (non)existence of an anchor between t and the j^{th} node:

$$
\begin{aligned}
p_{\text{match}} &= p_{\text{label}} \cdot \bar{p}_{\text{anchor}} \\
p_{\text{label}}(\hat{\mathbf{y}}_i, \mathbf{y}_j) &= \mathbb{1}_{\mathbf{y}_j^{\text{label}} \neq \varnothing} P\big(\mathbf{y}_j^{\text{label}} | \mathbf{h}_i; \boldsymbol{\theta}\big) \\
\bar{p}_{\text{anchor}}(\hat{\mathbf{y}}_i, \mathbf{y}_j) &= \underset{t \in \text{tokens}}{\text{GM}} \; P\big(\mathbb{1}_{t \in \mathbf{y}_j^{\text{anchors}}} | t, \mathbf{h}_i; \boldsymbol{\theta}\big).
\end{aligned}
$$

We use the geometric mean to keep the anchor score $\bar{p}_{\text{anchor}}$ magnitude independent of the number of tokens, and therefore have a similar weight as the label score p_{label}.

The optimal matching π^* can be efficiently computed by the Hungarian algorithm (Kuhn, 1955) in $O(n^3)$. As a result, every query is assigned either to a regular node or to a "null" node $\varnothing$. An illustration of a matching between queries and target nodes is presented in Figure 2.

The loss functions for the queries are computed with respect to the matched nodes. After finding π^*, we permute all target nodes and compute the classification losses in the standard "order-based" way (i.e., by minimizing the cross-entropy between the predictions and the corresponding targets). The losses of queries matched to the "null" nodes are ignored, except for their relative label loss ℓ_{label}, which pushes these queries to predict $\varnothing$ as their label. The label loss is further altered by the focal loss factor (Lin et al., 2017) to mitigate the imbalance of labels introduced by extending the targets with the "null" nodes.

3.3.3 Anchor Masking

During the early experiments with this architecture, we noticed that nodes tend to be generated from their anchored tokens (or more precisely from the queries of their anchored tokens), after the outputs stabilize during first epochs. We employ this observation to create an inductive bias by limiting the possible pairings to occur only between target nodes and predictions from their anchored tokens. Formally, this is achieved by setting

$$
\bar{p}_{\text{anchor}}\big(f_{\boldsymbol{\theta}}(\mathbf{h}_i), \mathbf{y}_j\big) = \varepsilon,
$$

if the j^{th} node is not anchored to the i^{th} token, with ε being a small positive constant close to 0.

3.4 Relative Label Encoding

Similarly to Straka et al. (2019); Straka and Straková (2019), we use *relative encodings* for the prediction of node labels: instead of direct classification of label strings, we utilize rules specifying how to transform anchored surface tokens into the semantic labels. For example, in Figure 1, the anchored token *"diving"* is transformed into *"dive"* by using a relative encoding rule deleting its last three characters and appending a character *"e"*. Such a rule could be also employed for predicting a node anchored to *"taking"* or *"giving"*. Relative encoding of labels is thus able to reduce the number of classification targets and generalize outside of the set of "absolute" label strings seen during training. Alternatively, the relative encoding can be seen as an extension of the pointer networks (Gu et al., 2016), which also decides how to post-process the copied tokens. Table 1 demonstrates how the relative encoding rules reduce the number of targets that need to be classified.

framework	language	# labels strings	# encoding rules
AMR	English	27,049	4,385
	Chinese	30,949	2,560
DRG	English	5,715	684
	German	1,905	918
EDS	English	31,933	1,322
PTG	English	39,336	529
	Czech	38,448	1,321

Table 1: The numbers of absolutely and relatively encoded node labels. Relative encodings lead to a significant reduction of classification targets in an order of magnitude across all frameworks. Note that node labels are the union of labels and property values (except for PTG), as described in Section 3.1.

3.4.1 Minimal Encoding Rule Set

Naturally, a label can be generated from anchored tokens in multiple ways. Unlike previous works that needed some heuristic to select a single rule from all suitable ones (Straka and Straková, 2019), we do not constraint the space of the possible rules much. Instead, we construct the final set of encoding rules to be the smallest possible one capable of encoding all labels.

Formally, let $\mathcal{S}$ be an arbitrary class of functions transforming a list of text strings (anchored tokens) into another string (node label), and let N be the set of all nodes from the training set. For $n \in \mathsf{N}$, denote n_t the anchored surface tokens and n_ℓ the target label string. Then the set of applicable rules for the node n is $\mathcal{S}_n = \{r \in \mathcal{S} | r(n_t) = n_\ell\}$. Our goal is to find the smallest subclass $\mathcal{S}^* \subseteq \mathcal{S}$ capable of encoding all node labels, in other words a subclass $\mathcal{S}^*$ satisfying

$$\forall n \in \mathsf{N} : \mathcal{S}^* \cap \mathcal{S}_n \neq \emptyset.$$

This formulation is equivalent to the minimal hitting set problem. Therefore, we can find the optimal solution of our problem by reducing it to a weighted MaxSAT formula in CNF: every $\mathcal{S}_n = \{r_1, r_2, \ldots, r_k\}$ becomes a hard clause $(r_1 \vee r_2 \vee \ldots \vee r_k)$ and every $r \in \mathcal{S}$ becomes a soft clause $(\neg r)$. We then submit this formula to the RC2 solver (Ignatiev et al., 2019) to obtain the minimal set of rules. Note that although solving this problem can take up to several hours, it needs to be done only once and then cached for all the training runs.

3.4.2 Space of Relative Rules

Our space of relative rules $\mathcal{S}$ consists of four disjoint subclasses:

1. *token rules* are represented by seven-tuples $(d_l, d_r, s, r_l, r_r, a_l, a_r)$ and process a list of anchored tokens n_t by first deleting the first d_l and the last d_r tokens, then by concatenating the remaining ones into one text string with the separator s inserted between them, followed by removing the first r_l and last r_r characters and finally by adding the prefix a_l and suffix a_r;[4]
2. *lemma rules* are created similarly to the token rules, but use the provided lemmas instead of tokens;
3. *number rules* transform word numerals into digits – for example, tokens [*"forty"*, *"two"*] become *"42"*;
4. *absolute rules* use the original label string n_ℓ, without taking into account any anchored tokens n_t; they serve as the fallback rules when no relative encodings are applicable.

3.4.3 Prediction of Relative Rules

Even with the minimal set of rules $\mathcal{S}^*$, multiple rules may be applicable to a single node. Therefore, the prediction of relative rules is a multi-label classification problem. The target distribution for a node n over all $r \in \mathcal{S}^*$ is defined as follows:[5]

$$P(r|n) = \begin{cases} \dfrac{1}{|\mathcal{S}^* \cap \mathcal{S}_n|}, & \text{if } r \in \mathcal{S}_n; \\ 0, & \text{otherwise.} \end{cases}$$

The label loss ℓ_{label} is then calculated as the cross-entropy between the target and the predicted distributions.

We use mixture of softmaxes (MoS) to mitigate the softmax bottleneck (Yang et al., 2018) that arises when multiple hypotheses can be correctly applied to a single input. MoS allows the model to consider K different hypotheses at the same time and weight them relatively to their plausibility.

Formally, let $\mathbf{h}_q$ be the final latent vector for query q and let $\mathbf{W}_k$, $\mathbf{b}_k$, $\mathbf{w}_k$, b_k, $\mathbf{w}_r$, b_r be the trainable weights. Then, the estimated MoS distribution

[4]To show a real example of a *token rule* from EDS, the rule `(0, 1, +, 0, 0, _, _a_1)` maps tokens ("`at`", "`the`", "`very`", "`least`", "`,`") into the label "`_at+the+very+least_a_1`".

[5]The target distribution is further modified by label smoothing (Szegedy et al., 2016) for better regularization.

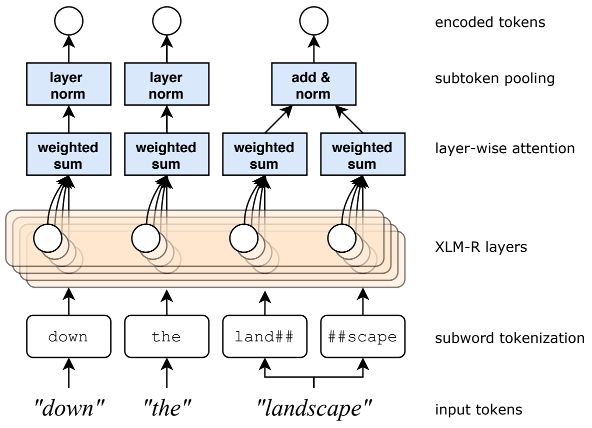

Figure 3: Architecture of the encoder with finetuned XLM-R. The input tokens are first tokenized into subwords, which are then processed into contextual embeddings by layer-wise attention on the XLM-R intermediate layers. Finally, the subword embeddings are pooled to obtain the encoded tokens.

of relative rules $P_{\boldsymbol{\theta}}(r|n)$ is defined as follows:

$$\begin{aligned}\mathbf{x}_k &= \tanh(\mathbf{W}_k\mathbf{h}_q + \mathbf{b}_k)\\ \pi_k &= \frac{\text{sigmoid}(\mathbf{h}_q^\top \mathbf{w}_k + b_k)}{\sum_{k'} \text{sigmoid}(\mathbf{h}_q^\top \mathbf{w}_{k'} + b_{k'})}\\ P_{\boldsymbol{\theta}}(r|n) &= \sum_{k=1}^{K} \pi_k \,\text{softmax}(\mathbf{x}_k^\top \mathbf{w}_r + b_r).\end{aligned}$$

3.5 Finetuning XLM-R

To obtain rich contextual embeddings for each input token, we finetune the pretrained multilingual model XLM-R (Conneau et al., 2020). The architecture of the encoder is presented in Figure 3.

3.5.1 Contextual Embedding Extraction

Different layers in BERT-like models represent varying levels of syntactic and semantic knowledge (van Aken et al., 2019), raising a question of which layer (or layers) should be used to extract the embeddings from. Following Kondratyuk and Straka (2019), we solve this problem by a purely data-driven approach and compute the weighted sum of all layers. Formally, let $\mathbf{e}_l$ be the intermediate output from the l^{th} layer and let w_l be a trainable scalar weight. The final contextual embedding is then calculated as

$$\mathbf{e} = \sum_{l=1}^{L} \text{softmax}(w_l)\mathbf{e}_l.$$

Note that each input token can be divided into multiple subwords by the XLM-R tokenizer. To obtain a single embedding for every token, we sum the embeddings of all its subwords. Finally, the contextual embeddings are normalized with layer normalization (Ba et al., 2016) to stabilize the training.[6]

3.5.2 Finetuning Stabilization

Given the large number of parameters in the pretrained XLM-R model, we employ several stabilization and regularization techniques in attempt to avoid overfitting.

We start by dividing the model parameters into two groups: the finetuned XLM-R and the rest of the network. Both groups are updated with AdamW optimizer (Loshchilov and Hutter, 2019) , and their learning rate follows the inverse square root schedule with warmup (Vaswani et al., 2017). The learning rate of the finetuned encoder is frozen for the first 2000 steps before the warmup phase starts (Howard and Ruder, 2018). The warmup is set to 6000 steps for both groups, while the learning rate peak is $6 \cdot 10^{-5}$ for the XLM-R and $6 \cdot 10^{-4}$ for the rest of the network. The weight decay for XLM-R, 10^{-2}, is considerably higher compared to $1.2 \cdot 10^{-6}$ used in the rest of the network (Devlin et al., 2019).

Dropout of entire intermediate XLM-R layers results in additional regularization – we drop each layer with 10% probability by replacing w_l with $-\infty$ during the final contextual embedding computation (Section 3.5.1). Inter-layer and attention dropout rates are the same as during the XLM-R pretraining.[7]

3.6 Balanced Loss Weights

Semantic parsing is an instance of multi-task learning, where each task $t \in \mathsf{T}$ can have conflicting needs and where the task losses ℓ_t can have different magnitudes. The overall loss function $\mathcal{L}$ to be optimized therefore consists of the weighted sum of partial losses ℓ_t:

$$\mathcal{L}(f_{\boldsymbol{\theta}}(\mathbf{x}), y) = \sum_{t \in \mathsf{T}} w_t \ell_t(f_{\boldsymbol{\theta}}(\mathbf{x}), y).$$

Finding optimal values for the loss weights w_t is extremely complicated. This issue is usually resolved either by (suboptimally) setting all weights equally to 1 or by a thorough grid search. However, the

[6] A side effect of the normalization step is that the subword summation is equal to the more common subword average (Zhang et al., 2019a).

[7] Due to the space constrains, all hyperparameters for each training configuration (together with the source code and pretrained models) are published at https://github.com/ufal/perin.

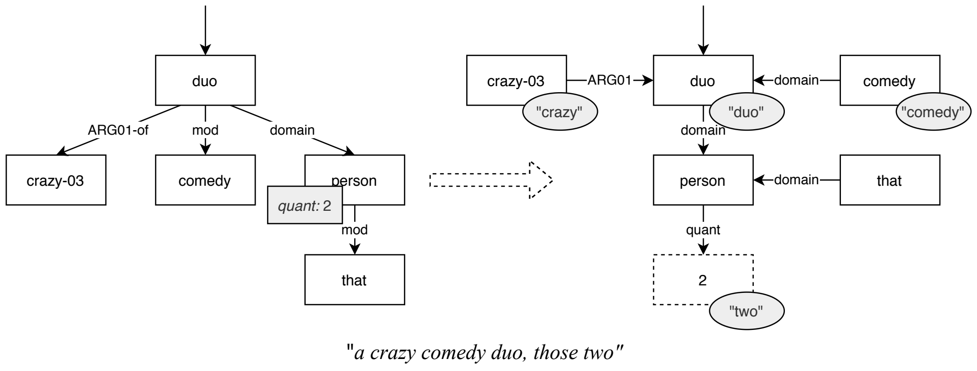

Figure 4: Visualization of AMR pre-processing (Section 3.7.1) for the sentence *"a crazy comedy duo, those two"*. The original graph is on the left and the transformed graph is shown on the right. Notice that the property `quant:2` of `person` is converted into a standalone node. The graph is normalized by reversing three inverted edges (note that `mod` is in fact `domain-of`) and some nodes get artificial anchors. Relative encoding rules are not included in this illustration for the sake of clarity, but it is worthwhile noting that nodes `person` and `that` contain only absolute label rules and are therefore not anchored.

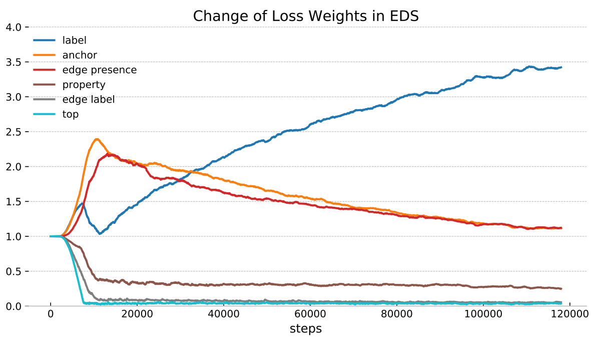

Figure 5: Change of the loss weights throughout the training of an EDS parser. The relative difficulty of edge and anchor predictions seems to be higher at the beginning of the training, but then gradually decreases, allowing the model to concentrate primarily on label prediction.

complexity of the grid search grows exponentially with $|\mathrm{T}|$ and would need to be performed independently for all nine combinations of frameworks and languages.

A more feasible solution is to set the weights adaptively according to a data-driven metric as in Kendall et al. (2018). We follow Chen et al. (2018), who balance the magnitudes of gradients $\|\nabla_{\boldsymbol{\theta_s}} w_i \ell_i\|_2$, where $\boldsymbol{\theta_s}$ are the weights of the shared part of the network. That magnitude is made proportional to the ratio of the current loss and its initial value: when ℓ_i decreases relatively quickly, its strength gets reduced to leave more space for the other tasks. Consequently, the loss weights w_t are not static, but change throughout the training to balance the individual gradient norms. Figure 5 shows an example of the balancing dynamics.

3.7 Framework Specifics

3.7.1 AMR

AMR is a *Flavor 2* framework, which means its nodes are not anchored to the surface forms. We instead exploit the general algorithm for the minimal encoding rule set (Section 3.4.1) to create artificial anchors: considering all possible one-to-one anchors $a \in A_n$ for each node n, we infer all compatible rules $\mathcal{S}_n = \bigcup_{a \in A_n} \mathcal{S}_n^a$, and find the minimal set of rules $\mathcal{S}^*$. The artificial anchors of a node n are then defined as $\{a \in A_n | \mathcal{S}_n^a \cap \mathcal{S}^* \neq \emptyset\}$. Consequently, our parser does not need any approximate anchoring (because we instead compute an anchoring minimizing the number of relative rules).

On the other hand, Chinese AMR graphs contain anchors (they are actually of *Flavor 1*), therefore, the described procedure is applied only on English AMR.

AMR graphs also contains inverted edges that transform them into tree-like structures. The inverted edges are marked by modified edge labels (for example, `ARG0` becomes `ARG0-of`). We normalize the graphs back into their original non-inverted form, making them more uniform, simplifying edge prediction to become more local and independent of the global graph structure. An example of AMR pre-processing is shown in Figure 4.

Considering the fact that every node is artificially anchored to at most a single token, the anchor classifier is not needed, if anchor masking is used (Section 3.3.3). Finally, AMR parsing does not employ the edge attribute classifier.

System	AMR^{eng}	DRG^{eng}	EDS^{eng}	PTG^{eng}	UCCA^{eng}	Average
HUJI-KU (Arviv et al., 2020)	52.36%	62.75%	79.68%	53.76%	72.91%	64.29%
Hitachi (Ozaki et al., 2020)	**81.54%**	93.19%	**93.56%**	88.73%	75.07%	86.42%
HIT-SCIR (Dou et al., 2020)	69.80%	89.07%	87.40%	84.26%	74.76%	81.06%
ÚFAL PERIN	80.23%	**94.16%**	92.73%	88.44%	**76.40%**	86.39%
ÚFAL PERIN*	80.23%	**94.16%**	92.73%	**89.19%**	**76.40%**	**86.54%**

System	AMR^{zho}	DRG^{deu}	——	PTG^{ces}	UCCA^{deu}	Average
HUJI-KU (Arviv et al., 2020)	44.92%	62.33%	——	58.49%	74.72%	60.11%
Hitachi (Ozaki et al., 2020)	80.44%	**93.36%**	——	87.35%	79.04%	85.05%
HIT-SCIR (Dou et al., 2020)	49.39%	68.31%	——	77.93%	80.02%	68.91%
ÚFAL PERIN	78.17%	89.83%	——	91.27%	**81.01%**	85.07%
ÚFAL PERIN*	**80.52%**	89.83%	——	**92.24%**	**81.01%**	**85.90%**

Table 2: The **all** F1 scores and a macro-average total score of the shared task systems. PERIN is our official shared task submission and PERIN* is a post-competition submission with a fixed bug. The best results are typeset in **bold**. The top table contains the *cross-framework* scores on English treebanks, while the bottom table presents the *cross-lingual* ones.

3.7.2 DRG

Since the DRG graphs are also of *Flavor 2*, they are pre-processed similarly to English AMR. Additionally, we reduce all nodes representing binary relations into labeled edges between the corresponding discourse elements.

Nodes in German DRG graphs are labeled in English, which decreases the applicability of relative encoding. Therefore, we employ the `opus-mt-de-en` (Tiedemann and Thottingal, 2020) machine translation model from Huggingface's transformers package (Wolf et al., 2019) to translate the provided lemmas from German to English, before computing the relative encoding rules.

DRG parsing does not make use of anchor and edge attribute classifiers, just like AMR parsing.

3.7.3 EDS

EDS graphs are post-processed to contain a single continuous anchor for every node. The EDS parser contains all the classification modules described in Section 3.2, except for the edge attribute classifier.

3.7.4 PTG

Properties in PTG graphs are not converted into nodes as in other frameworks, but are directly predicted from latent vectors $\mathbf{h}_q$ by multi-class classifiers (one for each property type). Additionally, the `frame` properties are selected only from frames listed in CzEngVallex (Urešová et al., 2015).

We utilize all classification heads except for the top node classification, because PTG graphs contain special `<TOP>` nodes, which make the separate top prediction redundant.

3.7.5 UCCA

We augment the UCCA nodes by assigning them `leaf` and `inner` labels. Additionally, the `inner` nodes are anchored to the union of anchors of their children. Therefore, the nodes can be differentiated by the permutation-invariant loss (Section 3.3.2).

The UCCA parser does not have the property classifier and the top classifier, where the latter is not needed, because the top node can be inferred from the structure of the rooted UCCA graphs.

4 Results

We present the overall results of our system in Table 2 and Table 3. Both tables contain F1 scores obtained using the official MRP metric.[8] Table 2 shows the **all** F1 scores for the individual frameworks together with the overall averages for the *cross-framework* and *cross-lingual* tracks. Macro-averaged results (across all nine frameworks) for the different MRP metrics are displayed in Table 3.

Note that our original submission (denoted as PERIN) contained a bug in anchor prediction for Chinese AMR and both PTG frameworks. The bug caused the nodes to get anchored to at least one token. We submitted a fixed version called PERIN* in the post-competition evaluation and compare it with the original one in Table 2.

According to the official whole-percent-only **all** F1 score, our competition submission reached tied first place in both the *cross-lingual* and the *cross-framework* track, with its performance virtually

[8]Fine-grained results for each framework are available in the task overview by Oepen et al. (2020).

System	Tops	Labels	Properties	Anchors	Edges	Attributes	Average
HUJI-KU (Arviv et al., 2020)	85.85%	22.80%	29.48%	46.79%	61.35%	7.67%	62.43%
Hitachi (Ozaki et al., 2020)	**95.67%**	68.93%	48.89%	61.95%	**80.14%**	24.93%	85.81%
HIT-SCIR (Dou et al., 2020)	94.37%	61.84%	30.80%	52.18%	71.41%	22.51%	75.66%
ÚFAL PERIN*	94.20%	**70.36%**	**49.34%**	**63.45%**	79.68%	**27.07%**	**86.26%**

Table 3: Overall results for different MRP metrics, macro-averaged over all frameworks and languages. The best results are typeset in **bold**.

Configuration	Tops	Labels	Properties	Anchors	Edges	Average
ÚFAL PERIN*	89.53%	93.45%	94.34%	93.40%	90.74%	92.73%
w/o MoS	88.04%	93.39%	93.79%	93.48%	90.76%	92.65%
w/o focal loss	89.08%	93.33%	93.59%	93.21%	90.46%	92.46%
BERT encoder	89.95%	92.97%	94.74%	92.92%	89.84%	92.27%
w/o balanced losses	89.23%	92.28%	94.46%	92.12%	89.19%	91.60%

Table 4: Ablation study showing MRP scores of different configurations on EDS. The top row contains the submitted configuration without any changes; then we report the results for 1) label classifier without the mixture of softmaxes (MoS); 2) label loss not multiplied by the focal loss coefficient; 3) encoder with finetuned BERT-large (English) instead of multilingual XLM-R and 4) constant loss weights, equally set to 1.0.

System	AMR^{eng}	EDS^{eng}	UCCA^{eng}
best from MRP 2019	73.11%	92.55%	82.61%
ÚFAL PERIN*	**78.43%**	**95.17%**	**82.71%**

Table 5: The last year's shared task had three frameworks – English AMR, EDS and UCCA – in common with MRP 2020. All parsers were evaluated on *The Little Prince* dataset, the first row shows the F1 scores of the best performing parser for each framework (Oepen et al., 2019).

identical to the system by Hitachi (Ozaki et al., 2020). Our bugfixed submission reached the first rank in both tracks, improving the *cross-lingual* score by nearly one percent point. Our system excels in label prediction, which might suggest the effectiveness of the relative label encoding. Furthermore, our system surpasses the best systems from the last year's semantic shared task, MRP 2019 (Oepen et al., 2019), by a wide margin – as can be seen in Table 5.

PERIN falls short in AMR^{eng} parsing by 1.31 %. On closer inspection, this follows from the inferior edge accuracy on this framework – the difference to Hitachi is 4.56 % on AMR^{eng} and 2.78 % on AMR^{zho}. Furthermore, Hitachi is better in all aspects of EDS^{eng} and DRG^{deu}. On the other hand PERIN consistently beats Hitachi in both PTG and both UCCA frameworks. We hope that combining the strengths of these two parsers will help to further advance the state of meaning representation parsing.

4.1 Ablation Experiments

We conducted several additional experiments to evaluate the effects of various components of our architecture. The results are summarized in Table 4. We have decided to use EDS for these experiments because – in our eyes – it represents the "average" framework without any significant irregularities.

The experiments show that using the mixture of softmaxes for label prediction does not have a substantial effect and can be potentially omitted to reduce the parameter count. On the other hand, the inferior results of the model with constant equal loss weights demonstrate the importance of balancing them.

5 Conclusion

We introduced a novel permutation-invariant sentence-to-graph semantic parser called PERIN. Given its state-of-the-art performance across a number of frameworks, we believe permutation-invariant node prediction might be the first step in a promising direction of semantic parsing and generally of graph generation.

Acknowledgments

This work has been supported by the Grant Agency of the Czech Republic, project EXPRO LUSyD (GX20-16819X). It has also been supported by the Ministry of Education, Youth and Sports of the Czech Republic, Project No. LM2018101 LINDAT/CLARIAH-CZ.

References

Omri Abend and Ari Rappoport. 2013. Universal Conceptual Cognitive Annotation (UCCA). In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 228–238, Sofia, Bulgaria. Association for Computational Linguistics.

Lasha Abzianidze, Johannes Bjerva, Kilian Evang, Hessel Haagsma, Rik van Noord, Pierre Ludmann, Duc-Duy Nguyen, and Johan Bos. 2017. The Parallel Meaning Bank: Towards a Multilingual Corpus of Translations Annotated with Compositional Meaning Representations. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 242–247, Valencia, Spain. Association for Computational Linguistics.

Betty van Aken, Benjamin Winter, Alexander Löser, and Felix A Gers. 2019. How does bert answer questions? a layer-wise analysis of transformer representations. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1823–1832.

Ofir Arviv, Ruixiang Cui, and Daniel Hershcovich. 2020. HUJI-KU at MRP 2020: Two Transition-based Neural Parsers. In *Proceedings of the CoNLL 2020 Shared Task: Cross-Framework Meaning Representation Parsing*, pages 73 – 82, Online.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

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.

Deng Cai and Wai Lam. 2020. AMR Parsing via Graph-Sequence Iterative Inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1290–1301, Online. Association for Computational Linguistics.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. 2020. End-to-End Object Detection with Transformers. *CoRR*, abs/2005.12872.

Wanxiang Che, Longxu Dou, Yang Xu, Yuxuan Wang, Yijia Liu, and Ting Liu. 2019. HIT-SCIR at MRP 2019: A Unified Pipeline for Meaning Representation Parsing via Efficient Training and Effective Encoding. In *Proceedings of the Shared Task on Cross-Framework Meaning Representation Parsing at the 2019 Conference on Computational Natural Language Learning*, pages 76 – 85, Hong Kong, China.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. 2018. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International Conference on Machine Learning*, pages 794–803.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised Cross-lingual Representation Learning at Scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. 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 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Longxu Dou, Yunlong Feng, Yuqiu Ji, Wanxiang Che, and Ting Liu. 2020. HIT-SCIR at MRP 2020: Transition-Based Parser and Iterative Inference Parser. In *Proceedings of the CoNLL 2020 Shared Task: Cross-Framework Meaning Representation Parsing*, pages 65 – 72, Online.

Timothy Dozat and Christopher D. Manning. 2017. Deep Biaffine Attention for Neural Dependency Parsing. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Timothy Dozat and Christopher D. Manning. 2018. Simpler but More Accurate Semantic Dependency Parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 484–490, Melbourne, Australia. Association for Computational Linguistics.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating Copying Mechanism in Sequence-to-Sequence Learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640, Berlin, Germany. Association for Computational Linguistics.

Jan Hajic, Eva Hajicová, Jarmila Panevová, Petr Sgall, Ondrej Bojar, Silvie Cinková, Eva Fucíková, Marie Mikulová, Petr Pajas, Jan Popelka, et al. 2012. Announcing Prague Czech-English Dependency Treebank 2.0. In *LREC*, pages 3153–3160.

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2018. Multitask Parsing Across Semantic Representations. In *Proceedings of the 56th Annual Meet-*

ing of the Association for Computational Linguistics (Volume 1: Long Papers), pages 373–385, Melbourne, Australia. Association for Computational Linguistics.

Daniel Hershcovich, Zohar Aizenbud, Leshem Choshen, Elior Sulem, Ari Rappoport, and Omri Abend. 2019. SemEval-2019 Task 1: Cross-lingual Semantic Parsing with UCCA. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 1–10, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

Jeremy Howard and Sebastian Ruder. 2018. Universal Language Model Fine-tuning for Text Classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia. Association for Computational Linguistics.

Alexey Ignatiev, Antonio Morgado, and Joao Marques-Silva. 2019. RC2: An efficient MaxSAT solver. *Journal on Satisfiability, Boolean Modeling and Computation*, 11(1):53–64.

Alex Kendall, Yarin Gal, and Roberto Cipolla. 2018. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7482–7491.

Dan Kondratyuk and Milan Straka. 2019. 75 Languages, 1 Model: Parsing Universal Dependencies Universally. 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 2779–2795, Hong Kong, China. Association for Computational Linguistics.

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

Harold W Kuhn. 1955. The Hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97.

Sunny Lai, Chun Hei Lo, Kwong Sak Leung, and Yee Leung. 2019. CUHK at MRP 2019: Transition-Based Parser with Cross-Framework Variable-Arity Resolve Action. In *Proceedings of the Shared Task on Cross-Framework Meaning Representation Parsing at the 2019 Conference on Computational Natural Language Learning*, pages 104 – 113, Hong Kong, China.

Bin Li, Yuan Wen, Weiguang Qu, Lijun Bu, and Nianwen Xue. 2016. Annotating the little prince with chinese amrs. In *Proceedings of the 10th Linguistic Annotation Workshop held in conjunction with ACL 2016 (LAW-X 2016)*, pages 7–15.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *11th Conference of the European Chapter of the Association for Computational Linguistics*.

Toan Q. Nguyen and Julian Salazar. 2019. Transformers without Tears: Improving the Normalization of Self-Attention. In *Proc. Workshop on Spoken Language Translation*.

Stephan Oepen, Omri Abend, Lasha Abzianidze, Johan Bos, Jan Hajič, Daniel Hershcovich, Bin Li, Tim O'Gorman, Nianwen Xue, and Daniel Zeman. 2020. MRP 2020: The Second Shared Task on Cross-Framework and Cross-Lingual Meaning Representation Parsing. In *Proceedings of the CoNLL 2020 Shared Task: Cross-Framework Meaning Representation Parsing*, pages 1 – 22, Online.

Stephan Oepen, Omri Abend, Jan Hajič, Daniel Hershcovich, Marco Kuhlmann, Tim O'Gorman, Nianwen Xue, Jayeol Chun, Milan Straka, and Zdeňka Urešová. 2019. MRP 2019: Cross-Framework Meaning Representation Parsing. In *Proceedings of the Shared Task on Cross-Framework Meaning Representation Parsing at the 2019 Conference on Computational Natural Language Learning*, pages 1 – 27, Hong Kong, China.

Stephan Oepen and Jan Tore Lønning. 2006. Discriminant-Based MRS Banking. In *LREC*, pages 1250–1255.

Hiroaki Ozaki, Gaku Morio, Yuta Koreeda, Terufumi Morishita, and Toshinori Miyoshi. 2020. Hitachi at MRP 2020: Text-to-Graph-Notation Transducer. In *Proceedings of the CoNLL 2020 Shared Task: Cross-Framework Meaning Representation Parsing*, pages 40 – 52, Online.

Xiaochang Peng, Chuan Wang, Daniel Gildea, and Nianwen Xue. 2017. Addressing the Data Sparsity Issue in Neural AMR Parsing. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 366–375, Valencia, Spain. Association for Computational Linguistics.

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. 2017. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660.

Rob A Van der Sandt. 1992. Presupposition projection as anaphora resolution. *Journal of semantics*, 9(4):333–377.

Milan Straka and Jana Straková. 2019. ÚFAL MRPipe at MRP 2019: UDPipe Goes Semantic in the Meaning Representation Parsing Shared Task. In *Proceedings of the Shared Task on Cross-Framework Meaning Representation Parsing at the 2019 Conference on Computational Natural Language Learning*, pages 127 – 137, Hong Kong, China.

Milan Straka, Jana Straková, and Jan Hajic. 2019. UDPipe at SIGMORPHON 2019: Contextualized Embeddings, Regularization with Morphological Categories, Corpora Merging. In *Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 95–103, Florence, Italy. Association for Computational Linguistics.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

Jörg Tiedemann and Santhosh Thottingal. 2020. OPUS-MT — Building open translation services for the World. In *Proceedings of the 22nd Annual Conferenec of the European Association for Machine Translation (EAMT)*, Lisbon, Portugal.

Zdeňka Urešová, Eva Fučíková, Jan Hajič, and Jana Šindlerová. 2015. CzEngVallex. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2019. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *ArXiv*, abs/1910.03771.

Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W. Cohen. 2018. Breaking the Softmax Bottleneck: A High-Rank RNN Language Model. In *International Conference on Learning Representations*.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. 2017. Deep sets. In *Advances in neural information processing systems*, pages 3391–3401.

Sheng Zhang, Xutai Ma, Kevin Duh, and Benjamin Van Durme. 2019a. AMR Parsing as Sequence-to-Graph Transduction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 80–94, Florence, Italy. Association for Computational Linguistics.

Yan Zhang, Jonathon Hare, and Adam Prugel-Bennett. 2019b. Deep set prediction networks. In *Advances in Neural Information Processing Systems*, pages 3212–3222.