Amazon at MRP 2019: Parsing Meaning Representations with Lexical and Phrasal Anchoring

Jie Cao[†]*, **Yi Zhang**[‡], **Adel Youssef**[‡], **Vivek Srikumar**[†]
[†]School of Computing, University of Utah
[‡]AWS AI, Amazon
{jcao, svivek}@cs.utah.edu, {yizhngn, adel}@amazon.com

Abstract

This paper describes the system submission of our team *Amazon* to the shared task on Cross Framework Meaning Representation Parsing (MRP) at the 2019 Conference for Computational Language Learning (CoNLL). Via extensive analysis of implicit alignments in AMR, we recategorize five meaning representations (MRs) into two classes: Lexical-Anchoring and Phrasal-Anchoring. Then we propose a unified graph-based parsing framework for the lexical-anchoring MRs, and a phrase-structure parsing for one of the phrasal-anchoring MRs, UCCA. Our system submission ranked 1st in the AMR subtask, and later improvements shows promising results on other frameworks as well.

1 Introduction

The design and implementation of broad-coverage and linguistically motivated meaning representation frameworks for natural language is attracting growing attention in recent years. With the advent of deep neural network-based machine learning techniques, we have made significant progress to automatically parse sentences intro structured meaning representation (Oepen et al., 2014, 2015; May, 2016; Hershcovich et al., 2019). Moreover, the differences between various representation frameworks has a significant impact on the design and performance of the parsing systems.

Due to the abstract nature of semantics, there is a diverse set of meaning representation frameworks in the literature (Abend and Rappoport, 2017). In some application scenario, tasks-specific formal representations such as database queries and arithmetic formula have also been proposed. However, primarily the study in computational semantics focuses on frameworks that are theoretically grounded on formal semantic theories, and sometimes also with assumptions on underlying syntactic structures.

Anchoring is crucial in graph-based meaning representation parsing. Training a statistical parser typically starts with a conjectured alignment between tokens/spans and the semantic graph nodes to help to factorize the supervision of graph structure into nodes and edges. In our paper, with evidence from previous research on AMR alignments (Pourdamghani et al., 2014; Flanigan et al., 2014; Wang and Xue, 2017; Chen and Palmer, 2017; Szubert et al., 2018; Lyu and Titov, 2018), we propose a uniform handling of three meaning representations from `Flavor-0` (DM, PSD) and `Flavor-2` (AMR) into a new group referred to as the **lexical-anchoring** MRs. It supports both explicit and implicit anchoring of semantic concepts to tokens. The other two meaning representations from `Flavor-1` (EDS, UCCA) is referred to the group of **phrasal-anchoring** MRs where the semantic concepts are anchored to phrases as well.

To support the simplified taxonomy, we named our parser as LAPA (**L**exical-**A**nchoring and **P**hrasal-**A**nchoring)[1]. We proposed a graph-based parsing framework with a latent-alignment mechanism to support both explicit and implicit lexicon anchoring. According to official evaluation results, our submission for this group ranked 1st in the AMR subtask, 6th on PSD, and 7th on DM respectively, among 16 participating teams. For phrasal-anchoring, we proposed a CKY-based constituent tree parsing algorithm to resolve the anchor in UCCA, and our post-evaluation submission ranked 5th on UCCA subtask.

2 Anchoring in Meaning Representation

The 2019 Conference on Computational Language Learning (CoNLL) hosted a shared task on

*Work done when Jie Cao was an intern at AWS AI

[1]The code is available online at https://github.com/utahnlp/lapa-mrp

Proceedings of the Shared Task on Cross-Framework Meaning Representation Parsing at the 2019 CoNLL, pages 138–148

https://doi.org/10.18653/v1/D19-6801

Cross-Framework Meaning Representation Parsing (MRP 2019, Oepen et al., 2019), which encourage participants in building a parser for five different meaning representations in three distinct flavors. `Flavor-0` includes the DELPHIN MRS Bi-lexical Dependencies (DM, Ivanova et al., 2012) and Prague Semantic Dependencies (PSD, Hajic et al., 2012; Miyao et al., 2014). Both frameworks under this representation have a syntactic backbone that is (either natively or by-proxy) based on bi-lexical dependency structures. As a result, the semantic concepts in these meaning representations can be anchored to the individual lexical units of the sentence. `Flavor-1` includes Elementary Dependency Structures (EDS, Oepen and Lønning, 2006) and Universal Conceptual Cognitive Annotation framework (UCCA, Abend and Rappoport, 2013), which shows an explicit, many-to-many anchoring of semantic concepts onto sub-strings of the underlying sentence. Finally, `Flavor-2` includes Abstract Meaning Representation (AMR, Banarescu et al., 2013), which is designed to abstract the meaning representation away from its surface token. But it leaves open the question of how these are derived. Previous studies have shown that the nodes in AMR graphs are predominantly aligned with the surface lexical units, although explicit anchoring is absent from the AMR representation. In this section, we review the related work supporting the claim of the implicit anchoring in AMR is actually lexical-anchoring, which can be merged into `Flavor-0` when we consider the parsing methods on it.

2.1 Implicit Anchoring in AMR

AMR tries to abstract the meaning representation away from the surface token. The absense of explicit anchoring can present difficulties for parsing. In this section, by extensive analysis on previous work AMR alignments, we show that AMR nodes can be implicitly aligned to the leixical tokens in a sentence.

AMR-to-String Alignments A straightforward solution to find the missing anchoring in an AMR Graph is to align it with a sentence; We denote it as AMR-to-String alignment.

ISI alignments (Pourdamghani et al., 2014) first linearizes the AMR graph into a sequence, and then use IBM word alignment model (Brown et al., 1993) to align the linearized sequence of concepts and relations with tokens in the sentence. According to the AMR annotation guidelines and error analysis of ISI aligner, some of the nodes or relations are evoked by subwords, e.g., the whole graph fragment `(p/possible-01 :polarity -)` is evoked by word "impossible", where the subword `"im-"` actually evoked the relation polarity and concept `"-"`; On the other side, sometimes concepts are evoked by multiple words, e.g., named entities, `(c/city :name (n/name :op1 "New":op2 "York"))`, which also happens in explict anchoring of DM and PSD. Hence, aligning and parsing with recategorized graph fragments are a natural solution in aligners and parsers. JAMR aligner (Flanigan et al., 2014) uses a set of rules to greedily align single tokens, special entities and a set of multiple word expression to AMR graph fragments, which is widely used in previous AMR parsers (*e.g.* Flanigan et al., 2014; Wang et al., 2015; Artzi et al., 2015; Pust et al., 2015; Peng et al., 2015; Konstas et al., 2017; Wang and Xue, 2017).

Other AMR-to-String Alignments exists, such as the extended HMM-based aligner. To consider more structure info in the linearized AMR concepts, Wang and Xue (2017) proposed a Hidden Markov Model (HMM)-based alignment method with a novel graph distance. All of them report over 90% F-score on their own hand-aligned datasets, which shows that AMR-to-String alignments are almost token-level anchoring.

AMR-to-Dependency Alignments Chen and Palmer (2017) first tries to align an AMR graph with a syntactic dependency tree. Szubert et al. (2018) conducted further analysis on dependency tree and AMR interface. It showed 97% of AMR edges can be evoked by words or the syntactic dependency edges between words. Those nodes in the dependency graph are anchored to each lexical token in the original sentence. Hence, this observation indirectly shows that AMR nodes can be aligned to the lexical tokens in the sentence.

Both AMR-to-String and AMR-to-dependency alignments shows that AMR nodes, including recategorized AMR graph fragements, do have implicit lexical anchoring. Based on this, Lyu and Titov (2018) propose to treat token-node alignments as discrete and exclusive alignment matrix and learn the latent alignment jointly with parsing. Recently, attention-based seq2graph model also

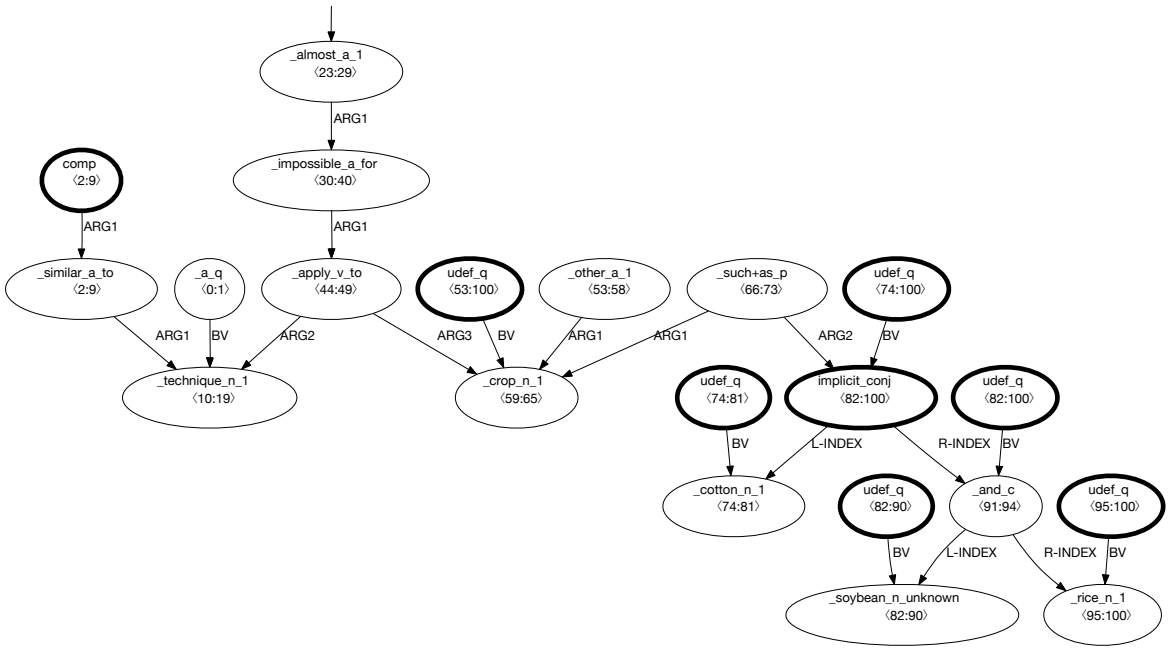


Figure 1: Phrasal-anchoring in EDS[wsj#0209013], for the sentence "`A similar technique is almost impossible to apply to other crops, such as cotton, soybeans and rice.`". Bold nodes are similar to the non-terminal nodes in UCCA, which are anchored multiple tokens, thus overlapping with the anchors of other nodes.

achieved the state-of-the-art accuracy on AMR parsing (Zhang et al., 2019). However, whether the attention weights can be explained as AMR alignments needs more investigation in future.

2.2 Taxonomy of Anchroing

Given the above analysis on implicit alignments in AMR, in this section, we further discuss the taxonomy of anchoring of the five meaning representations in this shared task.

Lexical-Anchoring According to the bi-lexical dependency structures of DM and PSD, and implicit lexical token anchoring on AMR, the nodes/-categorized graph fragments of DM, PSD, and AMR are anchored to surface lexical units in an explicit or implict way. Especially, those lexical units do not overlap with each other, and most of them are just single tokens, multiple word expression, or named entities. In other words, when parsing a sentence into DM, PSD, AMR graphs, tokens in the original sentence can be merged by looking up a lexicon dict when preprocessing and then may be considered as a single token for aligning or parsing.

Phrasal-Anchoring However, different from the lexical anchoring without overlapping, nodes in EDS and UCCA may align to larger overlapped word spans which involves syntactic or semantic pharsal structure. Nodes in UCCA do not have node labels or node properties, but all the nodes are anchored to the spans of the underlying sentence. Furthermore, the nodes in UCCA are linked into a hierarchical structure, with edges going between parent and child nodes. With certain exceptions (e.g. remote edges), the majority of the UCCA graphs are tree-like structures. According to the position as well as the anchoring style, nodes in UCCA can be classified into the following two types:

1. **Terminal nodes** are the leaf semantic concepts anchored to individual lexical units in the sentence

2. **Non-terminal nodes** are usually anchored to a span with more than one lexical units, thus usually overlapped with the anchoring of terminal nodes.

The similar classification of anchoring nodes also applies to the nodes in EDS, although they do not regularly form a recursive tree like UCCA. As the running example in Figure 1, most of the nodes belongs to terminal nodes, which can be explicitly anchored to a single token in the original sentence. However, those bold non-teriminal nodes are an-

chored to a large span of words. For example, the node "undef_q" with span <53:100> is aligned to the whole substring starting from "other crops" to the end; The abstract node with label imp_conj are corresponding to the whole coordinate structure between soybeans and rice

In summary, by treating AMR as an implicitly lexically anchored MR, we propose a simplified taxonomy for parsing the five meaning representation in this shared task.

- Lexical-anchoring: DM, PSD, AMR
- Phrasal-anchoring: EDS, UCCA

3 Model

For the two groups of meaning representations defined in Section 2, in this section, we propose two parsing framework: a graph-based parsing framework with latent alignment for lexically anchored MRs, and a minimal span-based CKY parser for one of the phrasally anchored MRs, UCCA.[2]

3.1 Graph-based Parsing Framework with Latent Alignment

Before formulating the graph-based model into a probabilistic model as Equation 1, we denote some notations: C, R are sets of concepts (nodes) and relations (edges) in the graph, and w is a sequence of tokens. $a \in \mathbb{Z}^m$ as the alignment matrix, each a_i is the index of aligned token where ith node aligned to. When modeling the negative log likelihood loss (NLL), with independence assumption between each node and edge, we decompose it into node- and edge-identification pipelines.

$$
\begin{aligned}
&NLL(P(C, R \mid w)) \\
&= -\log(P(C, R \mid w)) \\
&= -\log(\sum_a P(a)P(C, R \mid w, a)) \\
&= -\log\left(\sum_a P(a)P(R \mid w, a, c)P(c \mid w, a)\right) \\
&= -\log\left(\sum_a P(a)\prod_i^m P(c_i \mid h_{a_i}) \right. \\
&\qquad \left. \cdot \prod_{i,j=1}^m P(r_{ij} \mid h_{a_i}, c_i, h_{a_j}, c_j)\right)
\end{aligned} \tag{1}
$$

[2] After the CKY parser gets the related phrasal spans, graph-based parser can also be used to predict the relations between nodes.

In DM, PSD, and AMR, every token will only be aligned once. Hence, we train a joint model to maximize the above probability for both node identification $P(c_i \mid h_{a_i})$ and edge identification $P(r_{ij} \mid h_{a_i,c_i,h_{a_j},c_j})$, and we need to marginalize out the discrete alignment variable a.

3.1.1 Alignment Model

The above model can support both explicit alignments for DM, PSD, and implicit alignments for AMR.

Explicit Alignments For DM, PSD, with explicit alignments $a*$, we can use $P(a^*) = 1.0$ and other alignments $P(a|a \neq a^*) = 0.0$

Implicit Alignments For AMR, without gold alignments, one requires to compute all the valid alignments and then condition the node- and edge-identification methods on the alignments.

$$
\begin{aligned}
&\log(P(C, R \mid w)) \geq \\
&E_Q[\log(P_\theta(c \mid w, a)P_\Phi(R \mid w, a, c))] \\
&- D_{KL}(Q_\Psi(a \mid c, R, w) \mid\mid P(a))
\end{aligned} \tag{2}
$$

However, it is computationally intractable to enumerate all alignments. We estimate posterior alignments model Q as Equation 3, please refer to Lyu and Titov (2018) for more details.

- Applying variational inference to reduce it into Evidence Lower Bound (ELBO, Kingma and Welling, 2013)
- The denominator Z_Ψ in Q can be estimated by Perturb-and-Max(MAP) (Papandreou and Yuille, 2011)

$$
Q_\Psi(a \mid c, R, w) = \frac{\exp(\sum_{i=1}^n \phi(g_i, h_{a_i}))}{Z_\Psi(c, w)} \tag{3}
$$

  Where $\phi(g_i, h_{a_i})$ score each alignment link between node i and the corresponding words, g_i is node encoding, and h_{a_i} is encoding for the aligned token.

- Discrete argmax of a permutation can be estimated by Gumbel-Softmax Sinkhorn Networks (Mena et al., 2018; Lyu and Titov, 2018)

3.1.2 Node Identification

Node Identification predicts a concept c given a word. A concept can be either *NULL* (when there is no semantic node anchoring to that word, e.g.,

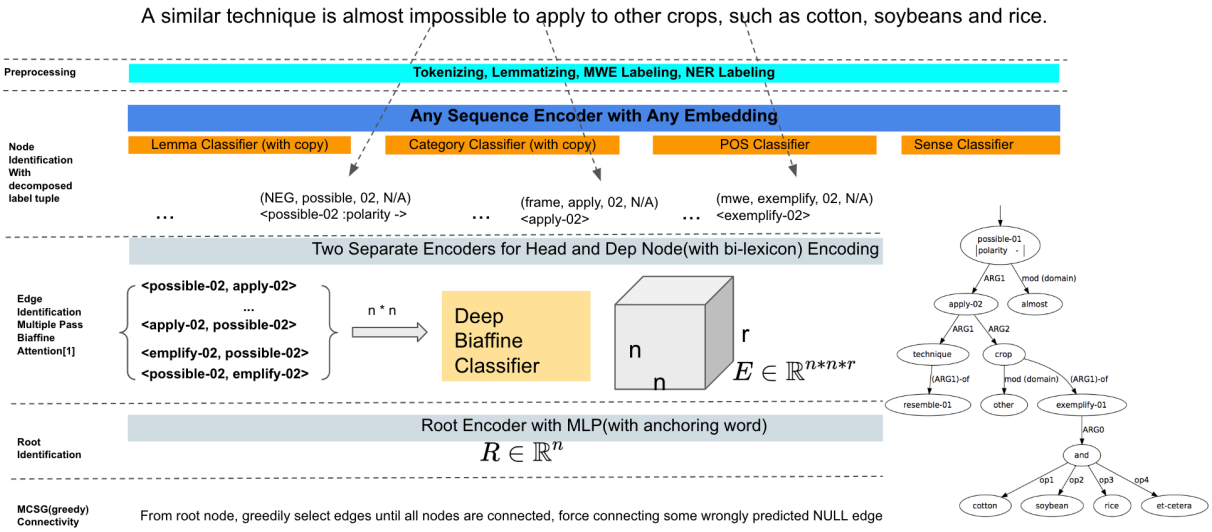


Figure 2: Architecture of graph-based model and inference, for running exmaple [wsj#0209013]

the word is dropped), or a node label (e.g., lemma, sense, POS, name value in AMR, frame value in PSD), or other node properties. One challenge in node identification is the data sparsity issue. Many of the labels are from open sets derived from the input token, e.g., its lemma. Moreover, some labels are constrained by a deterministic label set given the word. Hence, we designed a copy mechanism (Luong et al., 2014) in our neural network architecture to decide whether to copying deterministic label given a word or estimate a classification probability from a fixed label set.

3.1.3 Edge Identification

By assuming the independence of each edge, we model the edges probabilites independently. Given two nodes and their underlying tokens, we predict the edge label as the semantic relation between the two concepts with a bi-affine classifier (Dozat and Manning, 2016).

3.1.4 Inference

In our two-stage graph-based parsing, after nodes are identified, edge identification only output a probility distribution over all the relations between identified nodes. However, we need to an inference algorithm to search for the maximum spanning connected graph from all the relations. We use Flanigan et al. (MSCG, 2014) to greedily select the most valuable edges from the identified nodes and their relations connecting them. As shown in Figure 2, an input sentence goes through preprocessing, node identification, edge identification, root identification, and MCSG to generate a final connected graph as structured output.

3.2 Minimal Span-based CKY Parsing Framework

Let us now see our phrasal-anchoring parser for UCCA. We introduce the transformation we used to reduce UCCA parsing into a consituent parsing task, and finally introduce the detailed CKY model for the constituent parsing.

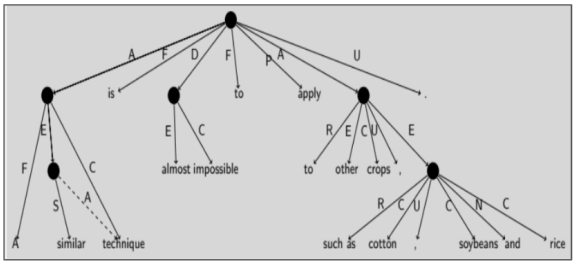
3.2.1 Graph-to-CT Transformation

We propose to transform a graph into a constituent tree structure for parsing, which is also used in recent work (Jiang et al., 2019). Figure 3 shows an example of transforming a UCCA graph into a constituent tree. The primary transformation assigns the original label of an edge to its child node. Then to make it compatible with parsers for standard PennTree Bank format, we add some auxiliary nodes such as special non-terminal nodes, TOP, HEAD, and special terminal nodes TOKEN and MWE. We remove all the "remote" annotation in UCCA since the constituent tree structure does not support reentrance. A fully compatible transformation should support both graph-to-tree and tree-to-graph transformation.

In our case, due to time constraints, we remove those remote edges and reentrance edges during training. Besides that, we also noticed that for multi-word expressions, the children of a parent node might not be in a continuous span (i.e., discontinuous constituent), which is also not supported by our constituent tree parser. Hence, when training the tree parser, by reattaching the dis-

continuous tokens to its nearest continuous parent nodes, we force every sub span are continuous in the transformed trees. We leave the postprocessing to recover those discontinuous as future work.

For inference, given an input sentence, we first use the trained constituent tree parsing model to parse it into a tree, and then we transform a tree back into a directed graph by assigning the edge label as its child's node label, and deleting those auxiliary labels, adding anchors to every remaining node.




```
(TOP
    (HEAD
        (:A
            (:F   (TOK A))
            (:E   (:S (TOK similar)))
            (:C   (TOK technique)))
        (:F     (TOK is))
        (:D     (:E almost) (:C impossible))
        (:F     (TOK to))
        (:P     (TOK apply))
        (:A     (:R (TOK to))
                (:E (TOK other))
                (:C (TOK crops))
                (:U (TOK ,))
                (:C (TOK soybeans))
                (:N (TOK and))
                (:C (TOK rice))
        (:U     (TOK .))
    )
)
```

Figure 3: UCCA to Constituent Tree Transformation for [wsj#0209013]

3.2.2 CKY Parsing and Span Encoding

After transforming the UCCA graph into a constituent tree, we reduce the UCCA parsing into a constituent tree parsing problem. Similar to the previous work on UCCA constituent tree parsing (Jiang et al., 2019), we use a minimal span-based CKY parser for constituent tree parsing. The intuition is to use dynamic programming to recursively split the span of a sentence recursively, as shown in Figure 3. The entire sentence can be splitted from top to bottom until each span is a single unsplittable tokens. For each node, we also need to assign a label. Two simplified assumptions are made when predicting the hole tree given a sentence. However, different with previous work, we use 8-layers with 8 heads transformer encoder, which shows better performance than LSTM in Kitaev and Klein (2018).

Tree Factorization In the graph-to-tree transformation, we move the edge label to its child node. By assuming the labels for each node are independent, we factorize the tree structure prediction as independent span-label prediction as Equation 4. However, this assumption does not hold for UCCA. Please see more error analysis in §4.4

$$
\begin{aligned}
T^* &= \arg\max_{T} s(T) \\
s(T) &= \sum_{(i,j,l)\in T} s(i,j,l)
\end{aligned}
\tag{4}
$$

CKY Parsing By assuming the label prediction is independent of the splitting point, we can further factorize the whole tree as the following dynamic programming in Equation 5.

$$
\begin{aligned}
s_{\text{best}}(i,i+1) &= \max_{l} s(i,i+1,l) \\
s_{\text{best}}(i,j) &= \max_{l} s(i,j,l) \\
&\quad + \max_{k}[s_{\text{best}}(i,k) + s_{\text{best}}(k,j)]
\end{aligned}
\tag{5}
$$

Span Encoding For each span (i,j), we represent the span encoding vector $v_{(i,j)} = [\vec{y_j} - \vec{y_i}] \oplus [\overleftarrow{y_{j+1}} - \overleftarrow{y_{i+1}}]$. $\oplus$ denotes vector concatenation. Assuming a bidirectional sentence encoder, we use the forward and backward encodings $\vec{y_i}$ and $\overleftarrow{y_i}$ of i_{th} word. Following the previous work, and we also use the loss augmented inference training. More details about the network architecture are in the Section 4.2

3.3 Summary of Implementation

We summarize our implementation for five meaning representations as Table 1. As we mentioned in the previous sections, we use latent-alignment graph-based parsing for lexical anchoring MRs (DM, PSD, AMR), and use CKY-based constituent parsing phrasal anchoring in MRs (UCCA, EDS). This section gives information about various decision for our models.

Top The first row "Top" shows the numbers of root nodes in the graph. We can see that for PSD, 11.56% of graphs with more than 1 top nodes. In our system, we only predict one top node with a N (N is size of identified nodes) way classifier, and

	Lexicon Anchoring			Phrase Anchoring	
	DM	PSD	AMR	EDS	UCCA
Top	1	$\geq$ 1 (11.56%)	1	1	1
Node Label	Lemma	Lemma(*)	Lemma(*) + NeType(143+)	_lemma(*)_semi_sense	N/A
Node Properties	POS semi(160*)_args(25)	POS wordid_sense(25)	constant values polarity, Named entity	 carg: constant value	N/A N/A
Edge Label	(45)	(91)	(94+)	(45)	(15)
Edge Properties	N/A	N/A	N/A	N/A	"remote"
Connectivity	True	True	True	True	True
Training Data	35656	35656	57885	35656	6485
Test Data	3269	3269	1998	3269	1131

Table 1: Detailed classifiers in our model, round bracket means the number of ouput classes of our classify, * means copy mechanism is used in our classifier. At the end of shared task, EDS are not fully supported to get an official results, we leave it as our future work.

then fix this with a post-processing strategy. When our model predicts one node as the top node, and if we find additional coordination nodes with it, we add the coordination node also as the top node.

Node Except for UCCA, all other four MRs have labeled nodes, the row "Node Label" shows the templates of a node label. For DM and PSD, the node label is usually the lemma of its underlying token. But the lemma is neither the same as one in the given companion data nor the predicted by Stanford Lemma Annotators. One common challenge for predicting the node labels is the open label set problem. Usually, the lemma is one of the morphology derivations of the original word. But the derivation rule is not easy to create manually. In our experiment, we found that handcrafted rules for lemma prediction only works worse than classification with copy mechanism, except for DM.

For AMR and EDS, there are other components in the node labels beyond the lemma. Especially, the node label for AMR also contains more than 143 fine-grained named entity types; for EDS, it uses the full SEM-I entry as its node label, which requires extra classifiers for predicting the corresponding sense. In addition to the node label, the properties of the label also need to be predicted. Among them, node properties of DM are from the SEMI sense and arguments handler, while for PSD, senses are constrained the senses in the predefined the vallex lexicon.

Edge Edge predication is another challenge in our task because of its large label set (from 45 to 94) as shown in row "Edge Label", the round bracket means the number of output classes of our classifiers. For Lexical anchoring MRs, edges are usually connected between two tokens, while phrasal anchoring needs extra effort to figure out the corresponding span with that node. For example, in UCCA parsing, To predict edge labels, we first predicted the node spans, and then node labels based that span, and finally we transform back the node label into edge label.

Connectivity Beside the local label classification for nodes and edges, there are other global structure constraints for all five MRs: All the nodes and edges should eventually form a connected graph. For lexical anchoring, we use MSCG algorithm to find the maximum connected graph greedily; For phrasal anchoring, we use dynamic programming to decoding the constituent tree then deterministically transforming back to a connected UCCA Graph [3]

4 Experiments and Results

4.1 Dataset and Evaluation

For DM, PSD, EDS, we split the training set by taking WSJ section (00-19) as training, and section 20 as dev set. For other datasets, when developing and parameter tuning, we use splits with a ratio of 25:1:1. In our submitted model, we did not use multitask learning for training. Following the unified MRP metrics in the shared tasks, we train our model based on the development set and finally evaluate on the private test set. For more details of the metrics, please refer to the summarization of the MRP 2019 task (Oepen et al., 2019),

4.2 Model Setup

For lexical-anchoring model setup, our network mainly consists of node and edge prediction

[3]Due to time constraint, we ignored all the discontinuous span and remote edges in UCCA

model. For AMR, DM, and PSD, they all use one layer Bi-directional LSTM for input sentence encoder, and two layers Bi-directional LSTM for head or dependent node encoder in the bi-affine classifier. For every sentence encoder, it takes a sequence of word embedding as input (We use 300 dimension Glove here), and then their output will pass a softmax layer to predicting output distribution. For the latent AMR model, to model the posterior alignment, we use another Bi-LSTM for node sequence encoding. For phrasal-anchoring model setup, we follow the original model set up in Kitaev and Klein (2018), and we use 8-layers 8-headers transformer with position encoding to encode the input sentence.

For all sentence encoders, we also use the character-level CNN model as character-level embedding without any pre-trained deep contextualized embedding model. Equipping our model with Bert or multi-task learning is promising to get further improvement. We leave this as our future work.

Our models are trained with Adam (Kingma and Ba, 2014), using a batch size 64 for a graph-based model, and 250 for CKY-based model. Hyper-parameters were tuned on the development set, based on labeled F1 between two graphs. We exploit early-stopping to avoid over-fitting.

4.3 Results

At the time of official evaluation, we submitted three lexical anchoring parser, and then we submitted another phrasal-anchoring model for UCCA parsing during post-evaluation stage, and we leave EDS parsing as future work. The following sections are the official results and error breakdowns for lexical-anchoring and phrasal-anchoring respectively.

Official Results on Lexical Anchoring Table 2 shows the official results for our lexical-anchoring models on AMR, DM, PSD. By using our latent alignment based AMR parser, our system ranked top 1 in the AMR subtask, and outperformed the top 5 models in large margin. Our parser on PSD ranked 6, but only 0.02% worse then the top 5 model. However, official results on DM and PSD shows that there is still around 2.5 points performance gap between our model and the top 1 model.

Official Results on Phrasal Anchoring Table 3 shows that our span-based CKY model for UCCA can achieve 74.00 F1 score on official test set, and ranked 5th. When adding ELMo (Peters et al., 2018) into our model, it can further improve almost 3 points on it.

MR	Ours (P/R/F1)	Top 1/3/5 (F1)
AMR(1)	**75/71/73.38**	**73.38/71.97/71.72**
PSD(6)	89/89/**88.75**	90.76/89.91/**88.77**
DM(7)	93/92/92.14	94.76/94.32/93.74

Table 2: Official results overview on unified MRP metric, we selected the performance from top 1/3/5 system(s) for comparison

MR	Ours (P/R/F1)	Top 1/3/5 (F1)
UCCA(5)	80.83/73.42/**76.94**	81.67/77.80/73.22
EDS	N/A	94.47/90.75/89.10

Table 3: Official results overview on unified MRP metric, we selected the performance from top 1/3/5 system(s) for comparison. It shows our UCCA model for post-evluation can rank 5th

4.4 Error Breakdown

Table 4, 5, 6 and 7 shows the detailed error breakdown of AMR, DM, PSD and UCCA respectively. Each column in the table shows the F1 score of each subcomponent in a graph: top nodes, node lables, node properties, node anchors, edge labels, and overall F1 score. No anchors for AMR, and no node label and propertis for UCCA. We show the results of MRP metric on two datasets. "all" denotes all the examples for that specific MR, while lpps are a set of 100 sentences from `The Little Prince`, and annotated in all five meaning representations. To better understand the performance, we also reported the official results from two baseline models TUPA (Hershcovich and Arviv, 2019) and ERG (Oepen and Flickinger, 2019).

	data	tops	labels	prop	edges	all
TUPA single	all	63.95	57.20	22.31	36.41	44.73
	lpps	71.96	55.52	26.42	36.38	47.04
TUPA multi	all	61.30	39.80	27.70	27.35	33.75
	lpps	72.63	50.11	20.25	33.12	43.38
Ours(1)	all	65.92	82.86	**77.26**	63.57	**73.38**
	lpps	72.00	78.71	58.93	63.96	71.11
Top 2	all	78.15	82.51	71.33	63.21	72.94
	lpps	83.00	76.24	51.79	60.43	69.03

Table 4: Our parser on AMR ranked 1st. This table shows the error breakdown when comparing to the baseline TUPA model and top 2 (Che et al., 2019) in official results

	data	tops	labels	prop	anchors	edges	all
ERG	all	91.83	98.22	95.25	98.82	90.76	95.65
	lpps	95.00	97.32	97.75	99.46	92.71	97.03
Top 1	all	93.23	94.14	94.83	98.40	91.55	94.76
	lpps	96.48	91.85	94.36	99.04	93.28	94.64
Ours(7)	all	70.95	93.96	92.13	97.25	86.45	92.14
	lpps	84.00	90.55	91.91	97.96	87.24	91.82

Table 5: Our parser on DM ranked 7th. This table shows the error breakdown when comparing to the model ranked Top 1 (Li et al., 2019) in official results

	data	tops	labels	prop	anchors	edges	all
Top 1	all	93.45	94.68	91.78	98.35	77.79	90.76
	lpps	93.33	91.73	84.37	98.40	77.63	88.34
Ours(6)	all	82.01	94.18	91.28	96.94	72.40	88.75
	lpps	85.85	90.48	82.63	95.97	73.60	85.83

Table 6: Our parser on PSD ranked 6th. This table shows the error breakdown when comparing to the model ranked top 1 (Donatelli et al., 2019) in official results

4.4.1 Error Analysis on Lexical-Anchoring

As shown in Table 4, our AMR parser is good at predicting node properties and consistently perform better than other models in all subcomponent, except for top prediction. Node properties in AMR are usually named entities, negation, and some other quantity entities. In our system, we recategorize the graph fragements into a single node, which helps for both alignments and structured inference for those special graph fragments. We see that all our 3 models perform almost as good as the top 1 model of each subtask on node label prediction, but they perform worse on top and edge prediction. It indicates that our bi-affine relation classifier are main bottleneck to improve. Moreover, we found the performance gap between node labels and node anchors are almost consistent, it indicates that improving our model on predicting `NULL` nodes may further improve node label prediction as well. Moreover, we believe that multi-task learning and pre-trained deep models such as BERT (Devlin et al., 2018) may also boost the performance of our paser in future.

4.4.2 Error Analysis on Phrasal-Anchoring

According to Table 7, our model with ELMo works slightly better than the top 1 model on anchors prediction. It means our model is good at predicting the nodes in UCCA and we belive that it is also helpful for prediction phrasal anchoring nodes in EDS.

However, when predicting the edge and edge attributes, our model performs 7-8 points worse than the top 1 model. In UCCA, an edge label means the relation between a parent nodes and its children. In our UCCA transformation, we assign edge label as the node label of its child and then predict with only child span encoding. Thus it actually misses important information from the parent node. Hence, in future, more improvement can be done to use both child and parent span encoding for label prediction, or even using another span-based bi-affine classifier for edge prediction, or remote edge recovering.

	data	tops	anchors	edge	attr	all
TUPA single	all	78.73	69.17	16.96	15.18	27.56
	lpps	86.03	76.26	28.32	24.00	40.06
TUPA multi	all	84.92	65.74	12.99	9.07	23.65
	lpps	88.89	77.76	26.45	18.32	41.04
(Che et al., 2019)	all	1.00	95.36	72.66	61.98	81.67
	lpps	1.00	96.99	73.08	48.37	82.61
Ours(*5)	all	98.85	94.92	60.17	0.00	**74.00**
	lpps	96.00	96.75	60.20	0.00	75.17
Ours + ELMo	all	99.38	95.70	64.88	0.00	**76.94**
	lpps	98.00	96.84	66.63	0.00	78.77

Table 7: Our UCCA parser in post-evaluation ranked 5th according to the original official evaluation results. This table shows the error breakdown when comparing to the model ranked top 1 (Che et al., 2019) in official results. * denotes the ranking of post-evaluation results

5 Conclusion

In summary, by analyzing the AMR alignments, we show that implicit AMR anchoring is actually lexical-anchoring based. Thus we propose to regroup five meaning representations as two groups: lexical-anchoring and phrasal-anchoring. For lexical anchoring, we suggest to parse DM, PSD, and AMR in a unified latent-alignment based parsing framework. Our submission ranked top 1 in AMR sub-task, ranked 6th and 7th in PSD and DM tasks. For phrasal anchoring, by reducing UCCA graph into a constituent tree-like structure, and then use the span-based CKY parsing to parse their tree structure, our method would rank 5th in the original official evaluation results.

Acknowledgments

The authors wish to thank the anonymous reviewers and members of the Amazon LEX team for their valuable feedback.

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.

Omri Abend and Ari Rappoport. 2017. The state of the art in semantic representation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 77–89.

Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage CCG Semantic Parsing with AMR. *EMNLP*.

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. *LAW@ACL*.

Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311.

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 Natural Language Learning*, pages 76 – 85, Hong Kong, China.

Wei-Te Chen and Martha Palmer. 2017. Unsupervised amr-dependency parse alignment. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 558–567.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Lucia Donatelli, Meaghan Fowlie, Jonas Groschwitz, Alexander Koller, Matthias Lindemann, Mario Mina, and Pia Weienhorn. 2019. Saarland at MRP 2019: Compositional parsing across all graphbanks. In *Proceedings of the Shared Task on Cross-Framework Meaning Representation Parsing at the 2019 Conference on Natural Language Learning*, pages 66 – 75, Hong Kong, China.

Timothy Dozat and Christopher D Manning. 2016. Deep biaffine attention for neural dependency parsing. *ICLR*.

Jeffrey Flanigan, Sam Thomson, Jaime G Carbonell, Chris Dyer, and Noah A Smith. 2014. A Discriminative Graph-Based Parser for the Abstract Meaning Representation. *ACL*.

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, 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.

Daniel Hershcovich and Ofir Arviv. 2019. TUPA at MRP 2019: A multi-task baseline system. In *Proceedings of the Shared Task on Cross-Framework Meaning Representation Parsing at the 2019 Conference on Natural Language Learning*, pages 28 – 39, Hong Kong, China.

Angelina Ivanova, Stephan Oepen, Lilja Øvrelid, and Dan Flickinger. 2012. Who did what to whom?: A contrastive study of syntacto-semantic dependencies. In *Proceedings of the sixth linguistic annotation workshop*, pages 2–11. Association for Computational Linguistics.

Wei Jiang, Yu Zhang, Zhenghua Li, and Min Zhang. 2019. Hlt@ suda at semeval 2019 task 1: Ucca graph parsing as constituent tree parsing. *arXiv preprint arXiv:1903.04153*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *ICLR*.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. *arXiv preprint arXiv:1805.01052*.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural AMR: Sequence-to-Sequence Models for Parsing and Generation.

Zuchao Li, Hai Zhao, Zhuosheng Zhang, Rui Wang, Masao Utiyama, and Eiichiro Sumita. 2019. SJTU–NICT at MRP 2019: Multi-task learning for end-to-end uniform semantic graph parsing. In *Proceedings of the Shared Task on Cross-Framework Meaning Representation Parsing at the 2019 Conference on Natural Language Learning*, pages 45 – 54, Hong Kong, China.

Minh-Thang Luong, Ilya Sutskever, Quoc V Le, Oriol Vinyals, and Wojciech Zaremba. 2014. Addressing the rare word problem in neural machine translation. *IJCNLP*.

Chunchuan Lyu and Ivan Titov. 2018. Amr parsing as graph prediction with latent alignment. *ACL*.

J May. 2016. SemEval-2016 Task 8: Meaning Representation Parsing. In *Proceedings of SemEval*, pages 1063–1073, Stroudsburg, PA, USA. Association for Computational Linguistics.

Gonzalo Mena, David Belanger, Scott Linderman, and Jasper Snoek. 2018. Learning latent permutations with gumbel-sinkhorn networks. *arXiv preprint arXiv:1802.08665*.

Yusuke Miyao, Stephan Oepen, and Daniel Zeman. 2014. In-house: An ensemble of pre-existing off-the-shelf parsers. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 335–340.

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 Natural Language Learning*, pages 1 – 27, Hong Kong, China.

Stephan Oepen and Dan Flickinger. 2019. The ERG at MRP 2019: Radically compositional semantic dependencies. In *Proceedings of the Shared Task on Cross-Framework Meaning Representation Parsing at the 2019 Conference on Natural Language Learning*, pages 40 – 44, Hong Kong, China.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinková, Dan Flickinger, Jan Hajič, and Zdeňka Urešová. 2015. SemEval 2015 Task 18. Broad-coverage semantic dependency parsing. In *Proceedings of the 9th International Workshop on Semantic Evaluation*, page 915 – 926, Bolder, CO, USA.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajič, Angelina Ivanova, and Yi Zhang. 2014. SemEval 2014 Task 8. Broad-coverage semantic dependency parsing. In *Proceedings of the 8th International Workshop on Semantic Evaluation*, page 63 – 72, Dublin, Ireland.

Stephan Oepen and Jan Tore Lønning. 2006. Discriminant-based mrs banking. In *LREC*, pages 1250–1255.

George Papandreou and Alan Yuille. 2011. Perturb-and-map random fields: Using discrete optimization to learn and sample from energy models–iccv 2011 paper supplementary material–.

Xiaochang Peng, Linfeng Song, and Daniel Gildea. 2015. A Synchronous Hyperedge Replacement Grammar based approach for AMR parsing. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 32–41, Stroudsburg, PA, USA. Association for Computational Linguistics.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Nima Pourdamghani, Yang Gao, Ulf Hermjakob, and Kevin Knight. 2014. Aligning English Strings with Abstract Meaning Representation Graphs. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 425–429, Stroudsburg, PA, USA. Association for Computational Linguistics.

Michael Pust, Ulf Hermjakob, Kevin Knight, Daniel Marcu, and Jonathan May. 2015. Using Syntax-Based Machine Translation to Parse English into Abstract Meaning Representation. *arXiv.org*.

Ida Szubert, Adam Lopez, and Nathan Schneider. 2018. A structured syntax-semantics interface for english-amr alignment. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1169–1180.

Chuan Wang and Nianwen Xue. 2017. Getting the most out of amr parsing. In *Proceedings of the 2017 conference on empirical methods in natural language processing*, pages 1257–1268.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015. Boosting Transition-based AMR Parsing with Refined Actions and Auxiliary Analyzers. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 1–6.

Sheng Zhang, Xutai Ma, Kevin Duh, and Benjamin Van Durme. 2019. AMR Parsing as Sequence-to-Graph Transduction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Florence, Italy. Association for Computational Linguistics.