Peking at MRP 2019: Factorization- and Composition-Based Parsing for Elementary Dependency Structures

Yufei Chen[♠], **Yajie Ye**[♠], **Weiwei Sun**[♠♡]
[♠] Wangxuan Institute of Computer Technology, Peking University
[♠] The MOE Key Laboratory of Computational Linguistics, Peking University
[♡] Center for Chinese Linguistics, Peking University
{yufei.chen,yeyajie,ws}@pku.edu.cn

Abstract

We design, implement and evaluate two semantic parsers, which represent factorization- and composition-based approaches respectively, for Elementary Dependency Structures (EDS) at the *CoNLL 2019 Shared Task on Cross-Framework Meaning Representation Parsing*. The detailed evaluation of the two parsers gives us a new perception about parsing into linguistically enriched meaning representations: current neural EDS parsers are able to reach an accuracy at the inter-annotator agreement level in the same-epoch-and-domain setup.

1 Introduction

For the *CoNLL 2019 Shared Task on Cross-Framework Meaning Representation Parsing* (MRP; Oepen et al., 2019), we concentrate on Elementary Dependency Structures (EDS; Oepen and Lønning, 2006), the graph-based meaning representations derived from English Resource Semantics[1] (ERS; Flickinger et al., 2014b) that is the richly detailed semantic annotation associated to English Resource Grammar (ERG; Flickinger, 2000), a domain-independent, linguistically deep and broad-coverage HPSG grammar. The full ERS and EDS annotations include not only basic predicate–argument structures, but also information about quantifiers and scopal operators, e.g. negation, as well as analyses of linguistically complex phenomena such as time and date expressions, conditionals, and comparatives.

Following Koller et al. (2019)'s practice, we divide existing work on string-to-semantic-graph parsing into four types, namely factorization-, composition-, transition- and translation-based approaches. Our previous studies (Chen et al., 2018b; Cao et al., 2019) as well as other investigations on other graph banks indicate that the factorization- and composition-based approaches obtain currently superior accuracies. In this paper, we fine-tune our factorization- and composition-based parsers and present a detailed evaluation on the MRP data.

Our factorization-based system obtains an overall accuracy of 94.47 in terms of the official MRP evaluation metrics, and out-performs other submission systems by a large margin with respect to the prediction for labels, properties, anchors and edges. We highlight a new perception: Current neural parsers are able to reach an accuracy at the inter-annotator agreement level (Bender et al., 2015) for the linguistically enriched EDS representations in the same-epoch-and-domain setup. Given the information depth of ERS, we think many NLP applications may benefit from a *revisit* of classic discrete semantic representations. The composition-based system reaches a score of 91.84. We do not think the performance gap suggests a weakness of the latter approach, but take it a reflection of the fact that a composition-based parser involves more individual modules that have not been fully optimized yet.

2 Parsing to Semantic Graphs

In this section, we present a summary of factorization-, composition-, transition- and translation-based parsing approaches.

Factorization-Based Approach. This type of approach is inspired by the successful design of graph-based dependency tree parsing (McDonald, 2006). A factorization-based parser explicitly models the target semantic structures by defining a score function that is able to evaluate the *goodness* of any candidate graph. Usually, the set of possible graphs that can be assigned to an input sentence is extremely large. Therefore, a parser also needs to know how to find the highest-scoring graphs from

[1] http://moin.delph-in.net/ErgSemantics

Proceedings of the Shared Task on Cross-Framework Meaning Representation Parsing at the 2019 CoNLL, pages 166–176

https://doi.org/10.18653/v1/K19-2016

a large set.

To the best of our knowledge, McDonald and Pereira (2006) present the first graph-based syntactic dependency parsing algorithm that removes the tree-shape constraint. In the scenario of semantic dependency parsing, Kuhlmann and Jonsson (2015) generalize the graph-based framework (aka Maximum Spanning Tree parsing) and propose Maximum Subgraph parsing. Given a directed graph $G = (V, E)$ that corresponds to an input sentence $x = w_0, \ldots w_{n-1}$ and a score function SCOREG. The string-to-graph parsing is formulated as a problem of searching for a subset $E' \subseteq E$ with the maximum score. Formally, we have the following optimization problem:

$$(V, E') = \underset{G^*=(V,E^* \subseteq E)}{\arg\max} \ \textsc{ScoreG}(G^*) \quad (1)$$

For semantic dependency parsing, V is the set of surface tokens, and G is, usually, the corresponding complete graph.

It is relatively straightforward to extend Kuhlmann and Jonsson's framework to cover more types of semantic graphs as follows,

$$G' = \underset{G^* \in \textsc{Gen}(x)}{\arg\max} \ \textsc{ScoreG}(G^*) \quad (2)$$

where $\textsc{Gen}(x)$ denotes all plausible semantic graphs that can be assigned to x.

To make the above combinatorial optimization problems solvable, people usually employ a **factorization** strategy, i.e. defining a decomposable score function that enumerates all sub-parts of a candidate graph. This view matches a classic solution to structured prediction which captures elemental and structural information through part-wise factorization. For example, the following formula defines a first-order factorization model for semantic dependency parsing,

$$G' = \underset{G^*=(V,E^* \subseteq E)}{\arg\max} \sum_{e \in E^*} \textsc{ScoreEdge}(e) \quad (3)$$

The essential computational module in this architecture is the score function, which is usually induced based on moderate-sized annotated sentences. Various deep learning models together with vector-based encodings induced from large-scale raw texts have been making advances in shaping a score function significantly (Dozat and Manning, 2018). We will detail our factorization-based parser in §3.

Composition-Based Approach. Compositionality is a cornerstone for many formal semantic theories. Following a principle of compositionality, a semantic graph can be viewed as the result of a derivation process, in which a set of lexical and syntactico-semantic rules are iteratively applied and evaluated. On the linguistic side, such rules extensively encode explicit knowledge about natural languages. On the computational side, such rules must be governed by a well-defined grammar formalism. In particular, to manipulate graph construction in a principled way, Hyperedge Replacement Grammar (HRG; Drewes et al., 1997) and AM Algebra (Groschwitz et al., 2017) have been applied to build semantic parsers for various graph banks (Chen et al., 2018b; Groschwitz et al., 2018; Lindemann et al., 2019).

A composition-based parser explicitly models derivations that yield semantic graphs by defining a score function SCORED. Assume a derivation $D = r_1, r_2, \ldots, r_m$ is a sequence of rules. Formally, we have the following optimization problem:

$$G' = \underset{G^* \in \textsc{Gen}(x)}{\arg\max} \sum_{D \in \textsc{Deriv}(G^*)} \textsc{ScoreD}(D) \quad (4)$$

To make the above problem solvable, people usually employ a **decomposition** strategy, i.e. summing over local scores that correspond to individual derivation steps:

$$\textsc{ScoreD}(D) = \sum_{i=1}^{m} \textsc{ScoreRule}(r_i) \quad (5)$$

Again, this matches many structured prediction models. Deep learning has been shown very powerful to associate scores to individual rule applications, and thus to provide great models for evaluating a derivation. The general form of (4) is a very complex combinatorial optimization problem. The approximating strategy to search for the best derivation instead has been shown practical yet effective for ERS parsing (Chen et al., 2018b). Formally, we solve the below problem,

$$D' = \underset{\substack{D^* \in \textsc{Gen}_{\textsc{Deriv}}(x) \\ D^* = r_1 r_2 \cdots r_m}}{\arg\max} \sum_{i=1}^{m} \textsc{ScoreRule}(r_i) \quad (6)$$

where $\textsc{Gen}_{\textsc{Deriv}}(x)$ denotes all sound derivations that yield x. Then we get a target graph by evaluating D'. We will detail our composition-based parser in §4.

Transition-Based Approach. This type of approach is inspired by the successful design of transition-based dependency tree parsing (Yamada and Matsumoto, 2003; Nivre, 2008). To the best of our knowledge, Sagae and Tsujii (2008) firstly apply this type of approach to predict predicate–argument structures grounded in HPSG (Miyao et al., 2005). A number of new transition systems and disambiguation models have been discussed for parsing into different graphs (Wang et al., 2015; Zhang et al., 2016; Buys and Blunsom, 2017; Gildea et al., 2018; Sun et al., 2019)

Translation-Based Approach. This type of approach is inspired by the success of sequence-to-sequence (seq2seq for short) models that are the heart of modern Neural Machine Translation. A translation-based parser takes a family of semantic graphs as a foreign language, in that a semantic graph is encoded and then viewed as a string from *another* language (Peng et al., 2017b; Konstas et al., 2017; Buys and Blunsom, 2017). A parser knows how to linearize a graph. Data augmentation has been shown very helpful (Konstas et al., 2017), partially reflecting the *data-hungry nature* of seq2seq models.

Simple application of seq2seq models is not sucessful. However, some basic models can be integrated with other types of approaches. Peng et al. (2018) propose to combine the translation- and transition-based approaches. Zhang et al. (2018) combined the translation- and factorization-based approaches.

3 The Factorization-Based Parser

3.1 Elements in EDS Graphs

The key idea underlying the factorization-based approach is to explicitly model what are expected as elements in target structures. Therefore before introducing the technical details of our parser, we roughly sketch key elements in EDS graphs. Refer to Flickinger et al. (2014a) for more information about the design of ERS.

We distinguish three kinds of elements: (1) labeled nodes, (2) node properties and (3) labeled edges. Nodes are sometimes called concepts[2], where their labels reflect conceptual meaning. The node labels can be divided into two classes: (1) *surface concepts* that are exclusively introduced by lexical entries, whose orthography is the source form of a core part of a concept symbol, and (2) *abstract concepts* that are used to represent the semantic contribution of grammatical constructions or more specialized lexical entries. Take the output structure in Figure 1 for example: _go_v_1 and _want_v_1 indicate surface concepts, while proper_q and named indicate abstract concepts.

To avoid proliferation of concepts, some concepts are parameterized. The parameters can be viewed as properties of nodes. For example, named("*Tom*") is a named concept with a CARG property of "*Tom*". For every EDS graph, there exists a top concept, which relates to the top handle in its original ERS annotation. In Figure 1, for example, _want_v_1 is the top. In this paper, we practically treat whether a node is *top* as a property whose value can be either *true* or *false*.

Edges are called relations. An edge links exactly two nodes and mainly reflects predicate–argument relations. Edges are assigned with a small, fixed inventory of role labels (e.g. ARG1, ARG2, ...).

3.2 The Architecture

We employ a four-stage pipeline to incrementally construct an EDS graph. Figure 1 illustrates the four steps with a simple sentence. The core idea is to identify concepts from surface strings, and then detect the relations between them.

3.3 Tokenization

Automatic tokenization for English has been widely viewed as a solved problem for quite a long time. Taking the risk of oversimplifying the situation, tokenization does not have a significant impact on downstream NLP tasks, e.g. POS tagging and syntactic parsing. When we consider semantic parsing, however, it is still a controversial issue which unit is the most basic one that triggers conceptual meaning and semantic construction. Therefore, we need to rethink the tokenization problem in which *tokens* may not be fully consistent with their traditional definitions. Moreover, when we consider other languages like German or Chinese, tokenization brings other issues.

In this paper, we take the most basic **word-level units**[3] as strings that are separated by whitespaces

[2] Considering the original design and especially the logic foundation of ERS, the seemly more standard name is *predicate*. In this paper, we call them *concepts*, mainly because we want to follow the new tradition of graph-based meaning representations (Kuhlmann and Oepen, 2016).

[3] We purposely avoid using *words* here. But when we in-

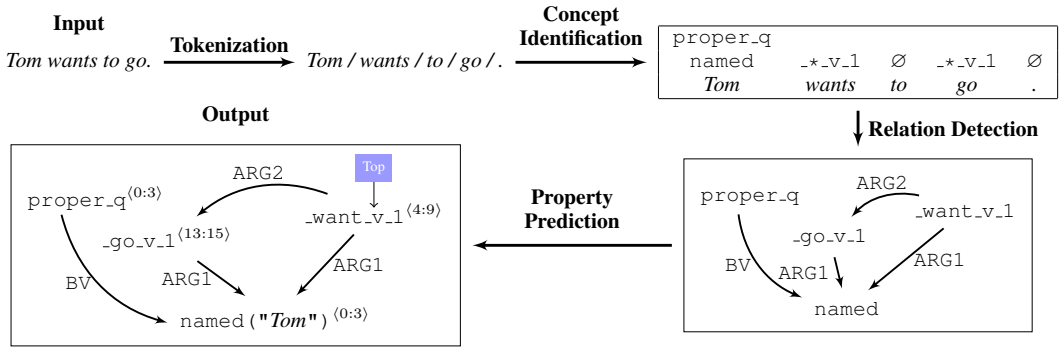

Figure 1: The workflow of our factorization-based parser. Tokenization: To separate an input sentence into semantic parsing-oriented tokens. Concept Identification: To generate concepts with a sequence labeling model. Relation Detection: To link concepts with a semantic dependency parsing model. Property Prediction: To predict node properties by classification.

Input string	*Assets of these short-term funds surged more than $5.5 billion in September.*
RegEx match	*Assets of these short - term funds surged more than $ 5 . 5 billion in September .*
Classification	*B B B B B B B I B B I I B B B B*
Our tokens	*Assets of these short - term funds surged more␣than $ 5␣␣5 billion in September .*
PTB tokens	*Assets of these short␣-␣term funds surged more than $ 5␣␣5 billion in September .*

Table 1: A tokenization example. Row "**Our tokens**" shows the result of our tokenizer, while Row "**PTB tokens**" shows the tokenization results defined by the Penn TreeBank (PTB; Marcus et al., 1993).

Concept	Type	String
_more+than_p	multi-unit	*more␣than*
_asset_n_1	single unit	*Assets*
_short_a_of	sub-unit	*short-term*
_term_n_of	sub-unit	*short-term*
_mis-_a_error	sub-unit	*misinterpreted*
_interpret_v_1	sub-unit	*misinterpreted*

Table 2: Examples to illustrate the relationships between surface concepts and word-level units. '␣' is an escape character for whitespace.

and punctuation markers. In an EDS graph, a surface concept may be aligned with a sub-unit, a single unit or multiple units. Table 2 shows some examples. During concept identification (§3.4), there should exist a surjection from surface concepts to the input tokens. Therefore, tokenization is important for obtaining a reasonable alignment between concepts and input tokens.

We adopt the character-based word segmentation approach for Chinese (Sun, 2010) to find suitable tokens. We first split an input sentence into a sequence of basic elements with simply defined regular expressions. The core part of our tokenizer is a sequence labeling model over this sequence. In particular, each element is assigned with a positional label that indicates token boundaries. The labels can be either **B**, which means the unit is at the begining of a target token, or **I**, which means the unit is inside a token. For sequential classification, we utilize a multi-layer BiLSTM network. Tokens can be retrieved from the predicted labels. See Figure 1 for an example. Note that, Dridan and Oepen (2012) showed that regular expressions are quite powerful to deal with the tokenizaiton problem for different styles.

troduce our neural parsing models, we still use *word* to relate the units to *word embeddings*.

3.4 Concept Identification

Surface concepts (e.g. quantifier _some_q) and some of the abstract concepts (e.g. named entity named) have a more transparent connection to surface forms and are relatively easier to identify. We call such concepts *lexicalized concepts*, which include all but are not limited to surface concepts. We cast identification of lexicalized concepts as a token-based tagging problem. The lexicalized concepts usually include lemma information in its

label. For example, _boy_n_1 consists of a lemma (boy) and a *type* , denoted as _*_n_1. As lemmas are much more easily to analyze, our concept identifier targets the type part only.

Some of the rest of abstract concepts are triggered by phrasal constructions. For example, compound is associated to the combination of multiple words. In this case, a concept is originally aligned to a sequence of continuous words. Considering that this type of concepts is a small portion, we propose to handle them in a word-level tagger. To this end, we re-align them to specific tokens with a small set of heuristic rules. For example, compound is re-aligned to the first word of a compound. Re-aligning these concepts means discarding their original anchors. To fully fit the MRP goals, we treat anchors as properties of concepts, and recover them by predicting the start/end boundaries with a classification model, as to be described in §3.6.

We employ a neural sequence labeling model to predict concepts. A multi-layer BiLSTM is utilized to encode tokens and another two softmax layers to predict concept-related labels: One for lexicalized concepts and the other for the rest. We also use recently widely-used contextualized word representation models, including ELMo (Peters et al., 2018) and BERT (Devlin et al., 2018). Figure 2 shows the neural network for concept identication.

3.5 Relation Detection

After finding a set of concepts, the next step is to link them together. Each semantic dependency is treated independently. We use integers as indices to mention concepts nodes. For any two nodes i and j, we give a score $\textsc{ScoreEdge}(i, j)$ to the possible arc $i \rightarrow j$. An arc is included to the final graph if and only if its score is greater than 0. We use a first-order model as described in Eq. (3). Figure 2 briefly summarizes the neural network for relation detection.

Following Dozat and Manning (2016, 2018), we use a deep biaffine attention to evaluate a candidate edge:

$$\begin{aligned}\textsc{ScoreEdge}(i, j) &= \textsc{Biaffine}(\boldsymbol{c}_i, \boldsymbol{c}_j) \\ &= \boldsymbol{c}_i^T U \boldsymbol{c}_j + W(\boldsymbol{c}_i + \boldsymbol{c}_j) + \boldsymbol{b}\end{aligned}$$

where $\boldsymbol{c}_i/\boldsymbol{c}_j$ is the vector associated to i/j. We consider two information sources to calculate $\boldsymbol{c}$: a textual part $\boldsymbol{r}_{\text{c2w}(i)}$ and a conceptual part $\boldsymbol{n}_i$, as following,

$$\boldsymbol{c}_i = \boldsymbol{r}_{\text{c2w}(i)} \oplus \boldsymbol{n}_i$$

Due to our concept identification method, we have a function "c2w" that takes as input the index of a node and returns as output the index of its anchored word. $\boldsymbol{r}_{\text{c2w}(i)}$ is the contextual vector of the word aligned to i, which is calculated by the word embedding layer and the encoder layers. $\boldsymbol{n}_i$ is the randomly-initialized embedding of i's concept type, e.g. _*_v_1. We also use the deep biaffine attention function to calculate each edge's scores for all labels, according to which we select the *best* label that achieves the maximum.

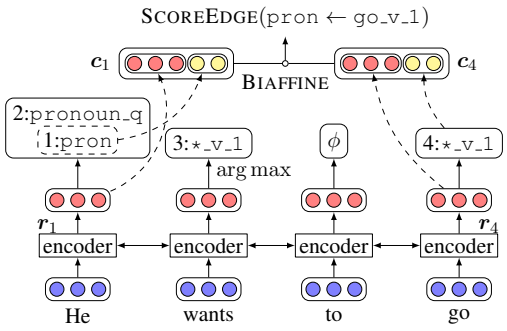

Figure 2: The network architecture for our concept identification and relation detection models which share the same architecture in word embedding and contextual encoder layers but with the same sets of parameters. A softmax layer is used for concept identification. To determine whether the dependency pron ← _go_v_1 exists, i.e. unlabeled dependency parsing, the corresponding embeddings $\boldsymbol{c}_1$ and $\boldsymbol{c}_4$, which are the concatenation of textual embeddings (in the red color) and the conceptual embeddings (in the yellow color), are biaffinely transformed into a score.

Following, for training, we use a margin-based approach to compute loss from the gold graph G^* and the best predicted $\hat{G}$ according to current model parameters. We define the *loss* term as:

$$\begin{aligned}loss = \max(0, \Delta(G^*, \hat{G}) \\ - \textsc{ScoreG}(G^*) + \textsc{ScoreG}(\hat{G}))\end{aligned} \quad (7)$$

The margin objective Δ measures the similarity between G^* and $\hat{G}$. Following Peng et al. (2017a), we define Δ as weighted Hamming to trade off between precision and recall.

3.6 Property Prediction

The final stage is to predict properties for each concept that is generated in the previous stages. For the EDS representation at CoNLL2019, we

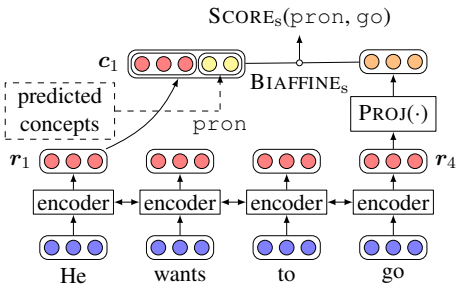

Figure 3: The network architecture for property prediction. The vector representations of concepts are obtained similarly to relation detection. The only difference is that the labels of concepts are provided by previous stages instead of being predicted by a softmax layer.

consider three types of properties and apply different strategies.

Anchors (spans). String anchors are treated as properties of concepts. For a given concept, a classification model is utilized to select two tokens over all input tokens as the start/end boundary of the concept respectively. We use exactly the same neural architecture in §3.5 to encode input tokens. See Figure 3 for a visualized illustration. The score of token j_w being the start/end boundary of node i can be computed by following equation:

$$\text{SCORE}_{\text{s/e}}(i, j_w) = \text{BIAFFINE}_{\text{s/e}}(\boldsymbol{c}_i, \text{PROJ}(\boldsymbol{r}_{j_w}))$$

Here PROJ$(\cdot)$ represents a feed-forward network with LEAKYRELU activation.

The anchors provided by training dataset are all character-based, so transformation is required before training this model. In the same manner, after retrieving the start/end word of a concept, we need to convert word-based anchors back to character-based anchors. Margin-based loss is used again when training this model and the total loss is the sum of losses for both boundaries.

The CARG property. Since the main function of the CARG attribute is to reduce the size of predicate names by parameterizing them with regularized surface strings, a rule-based system could be effective to predict the CARG information.

Firstly, we decide whether a concept has the CARG property according to its label. For example, `named`, `card` and `ord` need CARGs, but not `_the_q`.

Secondly, we use a dictionary which is extracted automatically from the training dataset. Entries of the dictionary are of the form $\langle \text{label}, \text{string}, \text{CARG} \rangle$. For example, a concept `named` whose *anchoring string* is *D.C.* will be mapped to *WashingtonDC*. Based on a close observation of the data, we introduce several heuristic rules if there is no applicable entry for a concept in the dictionary. For example, one widely applicable rule is to use `1` as the CARG value for concepts labeled `card` and aligned to a float number which is less than 1.

Finally, if no rule is available, we remove punctuation markers at left or right boundaries of anchoring strings and use the remaining part.

Top concept. We cast the precition for `top` as a binary classification problem over all nodes in a final graph. This strategy matches a recent research interest in graph neural networks (Li et al., 2015; Veličković et al., 2017; Defferrard et al., 2016; Chen et al., 2018a; Song et al., 2018), one goal of which is to associate vectors to graph nodes. Such vectors can be more easily to be integrated to neural networks for various purposes. We employ a Graph-based LSTM (Song et al., 2018) to encode an `EDS`graph and a multi-layer feed-forward network to determine whether a node is `top`. Similar as §3.5, margin-based approach is used to compute the *loss* term.

4 The Composition-Based Parser

Our composition-based parser is based on our previous work (Chen et al., 2018b). The core engine is a graph rewriting system that explicitly explores the syntactico-semantic recursive derivations that are governed by a synchronous `HRG` (`SHRG`). See Figure 4 for an example. Our parser constructs `EDS` graphs by explicitly modeling such derivations. In particular, it utilizes a constituent parser to build a syntactic derivation, and then selects semantic `HRG` rules associated to syntactic `CFG` rules to generate a graph. When multiple rules are applicable for a single phrase, a neural network is used to rank them.

One main difference between our submission parer and the parser introduced in Chen et al. (2018b) is that the syntactic parsing model is a re-implementation of Kitaev and Klein (2018). It utilizes transformer layers to capture words' contextual information, denoted as $\boldsymbol{r}_i$. After encoding an input sentence, a multiple-layer peceptron (MLP) is employed to get span scores. The score of span (i, j) with label L is calculated from its embedding $\boldsymbol{s}_{i,j}$, which is from the contextual vector of

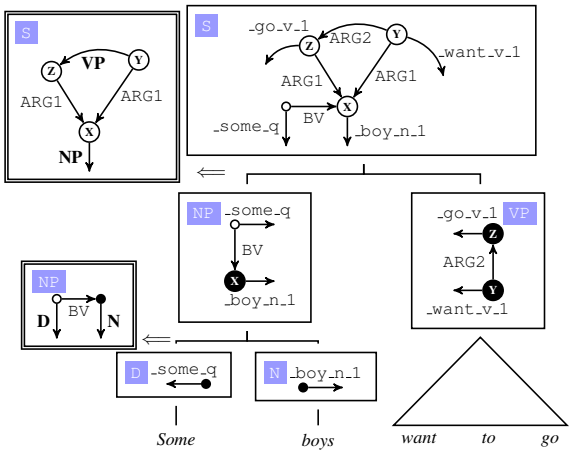

Figure 4: An HRG-based syntactico-semantic derivation. The derivation can be viewed as a syntactic tree enriched with semantic interpretation rules that are defined by an HRG. Each phrase in the syntactic tree is also assigned with a graph which corresonds to a subpart in the final semantic graph. Moreover, some particular nodes (filled nodes) in a sub-graph is marked as *communication channels* to other meaning parts in the same sentence. In HRG, these nodes are summarized as a hyperedge. Gluing two sub-graphs according to a construction rule follows the graph substitution principle of HRG. The application of the top rule that introduces a reentrancy structure is such an example. The "X" node in the graph of the left branching phrase is unified with the "X" node in the rule, and so do to the "Y" and "Z" nodes.

the two endpoints, $\boldsymbol{r}_i$ and $\boldsymbol{r}_{j-1}$:

$$\begin{aligned}\textsc{Sbcore}(i,j,L) &= \textsc{Mlp}(\boldsymbol{s}_{i,j})[L] \\ \boldsymbol{s}_{i,j} &= \boldsymbol{r}_i \oplus \boldsymbol{r}_{i-1} \\ \textsc{Mlp}(\boldsymbol{x}) &= \boldsymbol{W}_2\sigma(\boldsymbol{W}_1\boldsymbol{x}+\boldsymbol{b}_1)+\boldsymbol{b}_2\end{aligned}$$

The operator [] denotes index selection. We perform CKY decoding to retrieve the highest-scored constituent tree that agrees with the syntactic CFG grammar.

When a phrase structure tree is available, semantic interpretation can be regarded as translating this tree to the derivation of graph construction. As multiple subgraph correspondents in each node are available, the beam search strategy is used to balance the search complexity and quality.

To score subgraphs, we use two types of features. The first type is node feature. For a concept $\boldsymbol{n}$ aligned with span (i,j), we use the span embedding $\boldsymbol{s}_{i,j}$ as features, and score with non-linear transformation:

$$\textsc{ScorePart}_{\text{concept}}(i,j,p) = \textsc{Mlp}_{\text{concept}}(\boldsymbol{s}_{i,j})[p]$$

The second type is edge feature. Note that a semantic dependency with label L from conceptual node $\boldsymbol{n}_a$ to $\boldsymbol{n}_b$ are aligned to constituents (i_1,j_1) and (i_2,j_2) respectively. We calculate this part of score with non-linear transformation from the span embeddings $\boldsymbol{s}_{i_1,j_1}$, $\boldsymbol{s}_{i_2,j_2}$ and random initialized concept embeddings $\boldsymbol{n}_a$, $\boldsymbol{n}_b$:

$$\begin{aligned}&\textsc{ScorePart}_{\text{arc}}(i_1,j_1,i_2,j_2,\boldsymbol{p}_a,\boldsymbol{p}_b,L) \\ &\quad= \textsc{Mlp}_{\text{arc}}(\boldsymbol{s}_{i_1,j_1}\oplus\boldsymbol{s}_{i_2,j_2}\oplus\boldsymbol{p}_a\oplus\boldsymbol{p}_b)[L]\end{aligned}$$

For training, again, we use the margin-based loss.

5 Experiments

The MRP2019 training data consists of 35656 sentences in total. For convenience, the composition- and factorization-based parsers share the same tokenization model. Gold token position labels are extracted from DeepBank (Flickinger et al., 2012). For the composition-based parser, we leverage the syntactic information provided by DeepBank to extract synchronous grammars. Therefore, all sentences in the MRP2019 data that do not appear in DeepBank 1.1 are removed. Following the same preprocessing of semantic graphs in Chen et al. (2018b) and using the recommended setup in DeepBank, there are 33722 samples for training and 1689 samples for validation. The synchronous grammars are extracted from the training data using coarse-grained labels (Chen et al., 2018b). For factorization-based parser, we use heuristic rules to re-align the non-lexicalized concepts to input tokens. We remove all sentences that do not recieve results in this step from our training set. After re-alignment, 33580 sentences are left for training and 1689 for validation.

Table 3 shows the results of both parsers on the validation data using the official evalution tool—*mtool*[4]. Table 4 shows the intermediate results during parsing for both parsers.

For factorization-based parsing, we combine 4 models for concept identification and 5 models for relation detection. We ensemble models by averaging the score functions across all stand-alone models. These models use different initial random seeds, different pretraining methods (ELMo or BERT) or different encoder architectures (Transformer or BiLSTM). All these models achieve a similar performance respectively, but the ensemble one achieves a much better performance, as we can conclude from Table 3.

[4] https://github.com/cfmrp/mtool

#	Top	Label	Property	Anchor	Edge	DEVEL. P	DEVEL. R	DEVEL. F_1	TEST F_1
Factorization-based Parsing									
Gold tokenization	88.75	96.92	96.39	96.98	94.88	96.38	96.00	96.19	—
+ensemble stage2	89.23	97.37	97.46	97.16	95.18	96.80	96.26	96.53	—
+ensemble stage2 & 3	89.64	97.37	97.46	97.17	95.53	96.95	96.34	96.64	—
Full pipeline	88.87	96.76	96.03	96.72	94.69	96.20	95.78	95.99	—
+ensemble stage2	89.29	97.12	97.08	96.91	94.98	96.59	96.00	96.30	—
+ensemble stage2 & 3	89.52	97.12	97.10	96.93	95.34	96.74	96.10	96.42	94.47
Composition-based Parsing									
gold tokenization	88.63	95.73	97.37	96.85	93.00	95.38	95.04	95.21	—
Full pipeline	88.27	95.44	97.17	93.62	92.67	94.08	93.86	93.97	91.84

Table 3: Results on the development data set. The evaluation algorithm is Maximum Common Edge Subgraph Isomorphism (MRP). *Gold tokenization* means that the parser uses gold standard tokenization provided by DeepBank. *Full pipeline* means that all stages in the pipeline are based on automatic predictions. Columns in the middle block include F_1 scores with respect to basic evaluation items respectively. The right block shows overall precision, recall and F_1. All numbers are obtained by using *mtool*.

Factorization-based Parser	Concept Identification (F_1) Lexicalized	Non-lexicalized	Overall	Relation Detection Nodes	Edges	Overall
Gold tokenization	97.04	95.72	96.50	96.94	93.43	95.20
+ensemble stage2	97.40	96.20	96.94	97.30	93.85	95.60
+ensemble stage2 & 3	the same as above row			97.28	94.03	95.67

Composition-based Parser	Syntactic Parsing P	R	F_1	POS	Semantic Interpretation Nodes	Edges	Overall
Gold tokenization	92.16	92.16	92.16	95.01	95.63	91.43	93.56

Table 4: Results of each stage for both parsers on the development data. *Gold tokenization* has the same meaning in Table 3. Columns in the right block are the SMATCH scores ignoring all the node and edge properties for generated graphs. For factorization-based parser, columns in the middle block include the F_1 scores of concept identification with respect to *lexicalized*, *non-lexicalized* and all concepts respectively. For composition-based parser, columns in the middle block are the syntactic parsing results using standard metric and *POS* concerns the prediction of preterminals.

Our factorization-based parser achieves relatively satisfactory performance in all basic evaluation items except `top`. In the in-domain evalution, its performace nearly reaches the inter-annotator agreement reported in Bender et al. (2015). To find *top* concepts, our model encodes the semantic graphs and ignores the input sentences. We take the unsatisfactory result as a confirmation of the challenge to encode complex discrete structures into vectors.

The evalution results of our composition-based parser are not as good as the factorization-based one. We believe that the disagreement between our `SHRG` grammar and the original ERG leads to a major part of the performance gap.

6 Conclusion

Current neural `ERS` parsers work rapidly and reliably, with an MRP accuracy of over 94% in the same-epoch-and-domain setup. It is comparable to the inter-annotator agreement (in Elementary Dependency Match) reported in Bender et al. (2015). As `ERS` parsers become more and more accurate, efficient and robust, they have extensive application prospects in downstream deep language understanding-related tasks.

Acknowledgement

We are grateful for the great work of Sheng Huang, Sheng Xu and Xihao Wang on UCCA and AMR parsing tasks. This paper and research behind it would not have been possible without their help.

References

Emily M. Bender, Dan Flickinger, Stephan Oepen, Woodley Packard, and Ann A. Copestake. 2015. Layers of interpretation: On grammar and compositionality. In *Proceedings of the 11th International Conference on Computational Semantics, IWCS 2015, 15-17 April, 2015, Queen Mary University of London, London, UK*, pages 239–249.

Jan Buys and Phil Blunsom. 2017. Robust incremental neural semantic graph parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1215–1226, Vancouver, Canada. Association for Computational Linguistics.

Junjie Cao, Zi Lin, Weiwei Sun, and Xiaojun Wan. 2019. A comparative analysis of knowledge-intensive and data-intensive semantic parsers. *CoRR*, abs/1907.02298.

Jie Chen, Tengfei Ma, and Cao Xiao. 2018a. Fastgcn: fast learning with graph convolutional networks via importance sampling. *arXiv preprint arXiv:1801.10247*.

Yufei Chen, Weiwei Sun, and Xiaojun Wan. 2018b. Accurate shrg-based semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 408–418. Association for Computational Linguistics.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pages 3844–3852.

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*.

Timothy Dozat and Christopher D. Manning. 2016. Deep biaffine attention for neural dependency parsing. *CoRR*, abs/1611.01734.

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.

F. Drewes, H.-J. Kreowski, and A. Habel. 1997. Hyperedge Replacement Graph Grammars. In Grzegorz Rozenberg, editor, *Handbook of Graph Grammars and Computing by Graph Transformation*, pages 95–162. World Scientific Publishing Co., Inc., River Edge, NJ, USA.

Rebecca Dridan and Stephan Oepen. 2012. Tokenization: Returning to a long solved problem — a survey, contrastive experiment, recommendations, and toolkit —. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 378–382, Jeju Island, Korea. Association for Computational Linguistics.

Dan Flickinger. 2000. On building a more efficient grammar by exploiting types. *Natural Language Engineering*, 6(1):15–28.

Dan Flickinger, Emily M. Bender, and Stephan Oepen. 2014a. ERG semantic documentation. Accessed on 2019-09-04.

Dan Flickinger, Emily M. Bender, and Stephan Oepen. 2014b. Towards an encyclopedia of compositional semantics: Documenting the interface of the English Resource Grammar. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*, pages 875–881. European Language Resources Association (ELRA).

Daniel Flickinger, Yi Zhang, and Valia Kordoni. 2012. Deepbank: A dynamically annotated treebank of the wall street journal. In *Proceedings of the Eleventh International Workshop on Treebanks and Linguistic Theories*, pages 85–96.

Daniel Gildea, Giorgio Satta, and Xiaochang Peng. 2018. Cache transition systems for graph parsing. 44(1).

Jonas Groschwitz, Meaghan Fowlie, Mark Johnson, and Alexander Koller. 2017. A constrained graph algebra for semantic parsing with AMRs. In *IWCS 2017 - 12th International Conference on Computational Semantics - Long papers*.

Jonas Groschwitz, Matthias Lindemann, Meaghan Fowlie, Mark Johnson, and Alexander Koller. 2018. AMR dependency parsing with a typed semantic algebra. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1831–1841. Association for Computational Linguistics.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2676–2686. Association for Computational Linguistics.

Alexander Koller, Stephan Oepen, and Weiwei Sun. 2019. Graph-based meaning representations: Design and processing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*, pages 6–11, Florence, Italy. Association for Computational Linguistics.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural AMR. Sequence-to-sequence models for parsing and generation.

Marco Kuhlmann and Peter Jonsson. 2015. Parsing to noncrossing dependency graphs. *Transactions of the Association for Computational Linguistics*, 3:559–570.

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

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. 2015. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*.

Matthias Lindemann, Jonas Groschwitz, and Alexander Koller. 2019. Compositional semantic parsing across graphbanks.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: the penn treebank. *Computational Linguistics*, 19(2):313–330.

Ryan McDonald. 2006. *Discriminative learning and spanning tree algorithms for dependency parsing*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA, USA.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-2006))*, volume 6, pages 81–88.

Yusuke Miyao, Takashi Ninomiya, and Jun'ichi Tsujii. 2005. Corpus-oriented grammar development for acquiring a head-driven phrase structure grammar from the penn treebank. In *Proceedings of the Second International Joint Conference on Natural Language Processing*, pages 684–693.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34:513–553.

Stephan Oepen, Omri Abend, Jan Hajič, Daniel Hershcovich, Marco Kuhlmann, Tim O'Gorman, Nianwen Xue, and Milan Straka. 2019. MRP 2019: Cross-framework Meaning Representation Parsing. pages 1–26.

Stephan Oepen and Jan Tore Lønning. 2006. Discriminant-based mrs banking. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC-2006)*, Genoa, Italy. European Language Resources Association (ELRA). ACL Anthology Identifier: L06-1214.

Hao Peng, Sam Thomson, and Noah A. Smith. 2017a. Deep multitask learning for semantic dependency parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2037–2048, Vancouver, Canada. Association for Computational Linguistics.

Xiaochang Peng, Linfeng Song, Daniel Gildea, and Giorgio Satta. 2018. Sequence-to-sequence models for cache transition systems. pages 1842–1852.

Xiaochang Peng, Chuan Wang, Daniel Gildea, and Nianwen Xue. 2017b. Addressing the data sparsity issue in neural AMR parsing.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 2227–2237.

Kenji Sagae and Jun'ichi Tsujii. 2008. Shift-reduce dependency DAG parsing. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 753–760, Manchester, UK.

Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2018. A graph-to-sequence model for amr-to-text generation. *arXiv preprint arXiv:1805.02473*.

Weiwei Sun. 2010. Word-based and character-based word segmentation models: Comparison and combination. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 1211–1219, Beijing, China. Coling 2010 Organizing Committee.

Weiwei Sun, Yufei Chen, Xiaojun Wan, and Meichun Liu. 2019. Parsing Chinese sentences with grammatical relations. 45(1).

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015. A transition-based algorithm for AMR parsing.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *The 8th International Workshop of Parsing Technologies (IWPT2003)*, pages 195–206.

Sheng Zhang, Xutai Ma, Rachel Rudinger, Kevin Duh, and Benjamin Van Durme. 2018. Cross-lingual decompositional semantic parsing. pages 1664–1675.

Xun Zhang, Yantao Du, Weiwei Sun, and Xiaojun Wan. 2016. Transition-based parsing for deep dependency structures. *Computational Linguistics*, 42(3):353–389.