Core Semantic First: A Top-down Approach for AMR Parsing[*]

Deng Cai
The Chinese University of Hong Kong
thisisjcykcd@gmail.com

Wai Lam
The Chinese University of Hong Kong
wlam@se.cuhk.edu.hk

Abstract

We introduce a novel scheme for parsing a piece of text into its Abstract Meaning Representation (AMR): Graph Spanning based Parsing (GSP). One novel characteristic of GSP is that it constructs a parse graph incrementally in a top-down fashion. Starting from the root, at each step, a new node and its connections to existing nodes will be jointly predicted. The output graph spans the nodes by the distance to the root, following the intuition of first grasping the main ideas then digging into more details. The *core semantic first* principle emphasizes capturing the main ideas of a sentence, which is of great interest. We evaluate our model on the latest AMR sembank and achieve the state-of-the-art performance in the sense that no heuristic graph re-categorization is adopted. More importantly, the experiments show that our parser is especially good at obtaining the core semantics.

1 Introduction

Abstract Meaning Representation (AMR) (Banarescu et al., 2013) is a semantic formalism that encodes the meaning of a sentence as a rooted labeled directed graph. As illustrated by an example in Figure 1, AMR abstracts away from the surface forms in text, where the root serves as a rudimentary representation of the overall focus while the details are elaborated as the depth of the graph increases. AMR has been proved useful for many downstream NLP tasks, including text summarization (Liu et al., 2015; Hardy and Vlachos, 2018) and question answering (Mitra and Baral, 2016).

The task of AMR parsing is to map natural language strings to AMR semantic graphs automatically. Compared to constituent parsing (Zhang
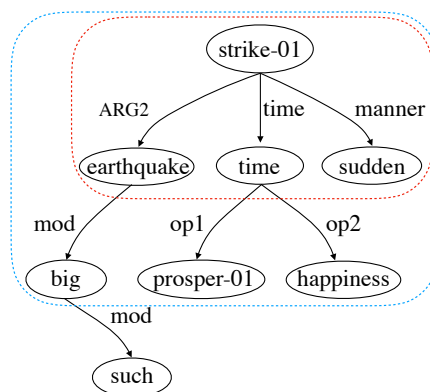

Figure 1: AMR for the sentence "*During a time of prosperity and happiness, such a big earthquake suddenly struck.*", where the subgraphs close to the root represent the core semantics.

and Clark, 2009) and dependency parsing (Kübler et al., 2009), AMR parsing is considered more challenging due to the following characteristics: (1) The nodes in AMR have no explicit alignment to text tokens; (2) The graph structure is more complicated because of frequent reentrancies and non-projective arcs; (3) There is a large and sparse vocabulary of possible node types (concepts).

Many methods for AMR parsing have been developed in the past years, which can be categorized into three main classes: Graph-based parsing (Flanigan et al., 2014; Lyu and Titov, 2018) uses a pipeline design for concept identification and relation prediction. Transition-based parsing (Wang et al., 2016; Damonte et al., 2017; Ballesteros and Al-Onaizan, 2017; Guo and Lu, 2018; Liu et al., 2018; Wang and Xue, 2017) processes a sentence from left-to-right and constructs the graph incrementally. The third class is seq2seq-based parsing (Barzdins and Gosko, 2016; Konstas et al., 2017; van Noord and Bos, 2017), which views parsing as sequence-to-sequence transduction by a linearization (depth-first traversal) of the AMR graph.

While existing graph-based models cannot sufficiently model the interactions between individual decisions, the autoregressive nature of transition-based and seq2seq-based models makes them suffer from error propagation, where later decisions can easily go awry, especially given the complexity of AMR. Since capturing the core semantics of a sentence is arguably more important and useful in practice, it is desirable for a parser to have a global view and a priority for capturing the main ideas first. In fact, AMR graphs are organized in a hierarchy that the core semantics stay closely to the root, for which a top-down parsing scheme can fulfill the desiderata. For example, in Figure 1, the subgraph in the red box already conveys the core meaning "*an earthquake suddenly struck at a particular time*", and the subgraph in the blue box further informs that "*the earthquake was big*" and "*the time was of prosperity and happiness*".

We propose a novel framework for AMR parsing known as Graph Spanning based Parsing (GSP). One novel characteristic of GSP is that, to our knowledge, it is the first top-down AMR parser.[1] GSP performs parsing in an incremental, root-to-leaf fashion, but still maintains a global view of the sentence and the previously derived graph. At each step, it generates the connecting arcs between the existing nodes and the coming new node, upon which the type of the new node (concept) is jointly decided. The output graph spans the nodes by the distance to the root, following the intuition of first grasping the main ideas then digging into more details. Compared to previous graph-based methods, our model is capable of capturing more complicated intra-graph interactions, while reducing the number of parsing steps to be linear in the sentence length.[2] Compared to transition-based methods, our model removes the left-to-right restriction and avoids sophisticated oracle design for handling the complexity of AMR graphs.

Notably, most existing methods including the state-the-of-art parsers often rely on heavy graph re-categorization for reducing the complexity of the original AMR graphs. For graph re-categorization, specific subgraphs of AMR are grouped together and assigned to a single node with a new compound category (Werling et al.,

---

[1]Depth-first traversal in seq2seq models does not produce a strictly top-down order due to the reentrancies in AMR.

[2]Since the size of AMR graph is approximately linear in the length of sentence.

2015; Wang and Xue, 2017; Foland and Martin, 2017; Lyu and Titov, 2018; Groschwitz et al., 2018; Guo and Lu, 2018). The hand-crafted rules for re-categorization are often non-trivial, requiring exhaustive screening and expert-level manual efforts. For instance, in the re-categorization system of Lyu and Titov (2018), the graph fragment "`temporal-quantity` $\xrightarrow{:ARG3-of}$ `rate-entity-91` $\xrightarrow{:unit}$ `year` $\xrightarrow{:quant}$ `1`" will be replaced by one single nested node "`rate-entity-3(annual-01)`". There are hundreds of such manual heuristic rules. This kind of re-categorization has been shown to have considerable effects on the performance (Wang and Xue, 2017; Guo and Lu, 2018). However, one issue is that the precise set of re-categorization rules differs among different models, making it difficult to distinguish the performance improvement from model optimization or carefully designed rules. In fact, some work will become totally infeasible when removing this re-categorization step. For example, the parser of Lyu and Titov (2018) requires tight integration with this step as it is built on the assumption that an injective alignment exists between sentence tokens and graph nodes.

We evaluate our parser on the latest AMR sembank and achieve competitive results to the state-of-the-art models. The result is remarkable since our parser directly operates on the original AMR graphs and requires no manual efforts for graph re-categorization. The contributions of our work are summarized as follows:

- We propose a new method for learning AMR parsing that produces high-quality core semantics.

- Without the help of heuristic graph re-categorization which requires expensive expert-level manual efforts for designing re-categorization rules, our method achieves state-of-the-art performance.

2 Related Work

Currently, most AMR parsers can be categorized into three classes: (1) *Graph-based* methods (Flanigan et al., 2014, 2016; Werling et al., 2015; Foland and Martin, 2017; Lyu and Titov, 2018; Zhang et al., 2019) adopt a pipeline approach for graph construction. It first maps continuous text spans into AMR concepts, then calculates the scores of possible edges and uses a max-

imum spanning connected subgraph algorithm to select the final graph. The major deficiency is that the concept identification and relation prediction are strictly performed in order, yet the interactions between them should benefit both sides (Zhou et al., 2016). In addition, for computational efficacy, usually only first-order information is considered for edge scoring. (2) *Transition-based* methods (Wang et al., 2016; Damonte et al., 2017; Wang and Xue, 2017; Ballesteros and Al-Onaizan, 2017; Liu et al., 2018; Peng et al., 2018; Guo and Lu, 2018; Naseem et al., 2019) borrow techniques from shift-reduce dependency parsing. Yet the non-trivial nature of AMR graphs (e.g., reentrancies and non-projective arcs) makes the transition system even more complicated and difficult to train (Guo and Lu, 2018). (3) *Seq2seq-based* methods (Barzdins and Gosko, 2016; Peng et al., 2017; Konstas et al., 2017; van Noord and Bos, 2017) treat AMR parsing as sequence-to-sequence problem by linearizing AMR graphs, thus existing seq2seq models (Bahdanau et al., 2014; Luong et al., 2015) can be readily utilized. Despite its simplicity, the performance of the current seq2seq models lag behind when the training data is limited. The first reason is that seq2seq models are often not as effective on smaller datasets. The second reason is that the linearized AMRs add the challenges of making use of the graph structure information.

There are also some notable exceptions. Peng et al. (2015) introduce a synchronous hyperedge replacement grammar solution. Pust et al. (2015) regard the task as a machine translation problem, while Artzi et al. (2015) adapt combinatory categorical grammar. Groschwitz et al. (2018); Lindemann et al. (2019) view AMR graphs as the structure AM algebra.

Most AMR parsers require an explicit alignment between tokens in the sentences and nodes in the AMR graph during training. Since such information is not annotated, a pre-trained aligner (Flanigan et al., 2014; Pourdamghani et al., 2014; Liu et al., 2018) is often required. More recently, Lyu and Titov (2018) demonstrate that the alignments can be treated as latent variables in a joint probabilistic model.

3 Background and Overview

3.1 Background of Multi-head Attention

The multi-head attention mechanism introduced by Vaswani et al. (2017) is used as a basic building block in our framework. The multi-head attention consists of H attention heads, and each of which learns a distinct attention function. Given a query vector x and a set of vectors $\{y_1, y_2, \ldots, y_m\}$ or in short $y_{1:m}$, for each attention head, we project x and $y_{1:m}$ into distinct query, key, and value representations $q \in \mathbb{R}^d$, $K \in \mathbb{R}^{m \times d}$ and $V \in \mathbb{R}^{m \times d}$ respectively, where d is the dimension of the vector space. Then we perform scaled dot-product attention (Vaswani et al., 2017):

$$a = \text{softmax}\frac{(Kq)}{\sqrt{d}}$$
$$attn = aV$$

where $a \in \mathbb{R}^m$ is the attention vector (a distribution over all input $y_{1:m}$) and $attn$ is the weighted sum of the value vectors. Finally, the outputs of all attention heads are concatenated and projected to the original dimension of x. For brevity, we will denote the whole attention procedure described above as a function $T(x, y_{1:m})$.

Based on the multi-head attention, the Transformer encoder (Vaswani et al., 2017) uses self-attention for context information aggregation when given a set of vectors (e.g., word embeddings in a sentence or node embeddings in a graph).

3.2 Overview

Figure 2 depicts the major neural components in our proposed framework: The Sentence Encoder component and the Graph Encoder component are designed for token-level sentence representation and node-level graph representation respectively. Given an input sentence $\mathbf{w} = (w_1, w_2, \ldots, w_n)$, where n is the sentence length, the Sentence Encoder component will first read the whole sentence and encode each word w_i into the hidden state s_i. The initial graph G_0 is always initialized with one dummy node d^* and a previously generated concept c_j is encoded into the hidden state v_j by the Graph Encoder component.

At each time step t, the Focus Selection component reads both the sentence representation $s_{1:n}$ and the graph representation $v_{0:t-1}$ of G_{t-1} repeatedly, generates the initial parser state h_t. The
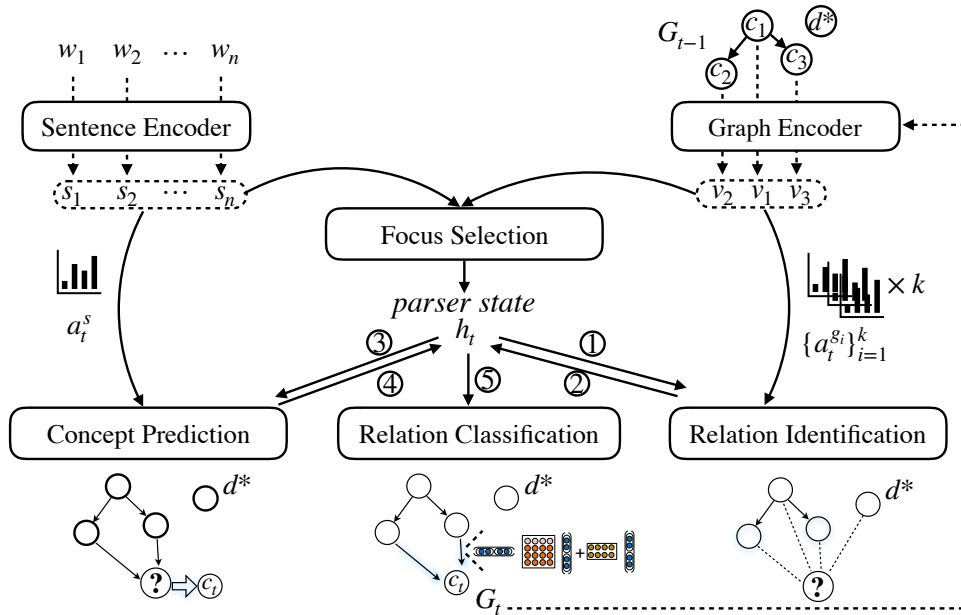
Figure 2: Model architecture of GSP, together with the decoding procedure at the time step t, where the read and write operations around the parser state h_t follow the order ① → ② → ③ → ④ → ⑤.

parser state carries the most useful information and serves as a writable memory during the expansion step. Next, the Relation Identification component decides which specific head nodes to expand by computing the multiple attention scores $\{a_t^{g_i}\}_{i=1}^k$ over the existing nodes. New arcs are generated according to the attention scores. Then the Concept Prediction component updates the parser state h_t with arc information, computes the attention vector a_t^s over the sentence and accordingly chooses a specific part to generate the new concept c_t. Finally, the Relation Classification component is used to predict the relation labels between the newly generated concept and its predecessors. Consequently an updated graph G_t is produced and G_t will be processed for the next time step. The whole decoding procedure is terminated if the newly generated concept is the special stop concept $\oslash$.

Our method expands the graph in a root-to-leaf fashion, nodes with shorter distances to the root will be introduced first. It follows a similar way that humans grasp the meaning: first seeking the main concepts then proceeding to the substructures governed by certain head concepts (Banarescu et al., 2013).

During training, we use breadth-first search to decide the order of nodes. However, for nodes with multiple children, there still exist multiple valid selections. In order to define a deterministic decoding process, we sort sibling nodes by their relations to the head node. We will present more discussions on the choice of sibling order in § 5.3.

4 Framework Description

4.1 Sentence & Graph Representation

Transformer encoder architecture is employed for both the Sentence Encoder and the Graph encoder components. For sentence encoding, a special token ($\triangleright$) is prepended to the input word sequence, whose final hidden state s_0 is regarded as an aggregated summary of the whole sentence and used as the initial state in parsing steps.

The Graph Encoder component takes previously generated concept sequence $(c_0, c_1, \ldots, c_{t-1})$ (c_0 is the dummy node d^*) as input. For computation efficiency and reducing error propagation, instead of encoding the edge information explicitly, we use the Transformer encoder to capture the interactions between nodes. Finally, the encoder outputs a sequence of node representations $(v_0, v_1, \ldots, v_{t-1})$.

4.2 Focus Selection

At each time step t, the Focus Selection component will read the sentence and the partially constructed graph repeatedly for gradually locating and collecting the most relevant information for the next expansion. We simulate the repeated reading by multiple levels of attention. Formally,

the following recurrence is applied by L times:

$$x_{t,1}^{l+1} = \text{LN}(h_t^l + T_1^{l+1}(h_t^l, s_{1:n}))$$
$$x_{t,2}^{l+1} = \text{LN}(x_{t,1}^{l+1} + T_2^{l+1}(x_{t,1}^{l+1}, v_{0:t-1}))$$
$$h_t^{l+1} = \max(x_{t,2}^{l+1}W_1^{l+1} + b_1^{l+1})W_2^{l+1} + b_2^{l+1}$$

where $T(\cdot, \cdot)$ is the multi-head attention function. LN is the layer normalization (Ba et al., 2016) and h_t^0 is always initialized with s_0. For clarity, we denote the last hidden state h_t^L as h_t, as the parser state at the time step t. We now proceed to present the details of each decision stage of one parsing step, which is also illustrated in Figure 2.

4.3 Relation Identification

Our Relation Identification component is inspired by a recent attempt of exposing auxiliary supervision on attention mechanism (Strubell et al., 2018). It can be considered as another attention layer over the existing graph, yet the attention weights explicitly indicate the likelihood of the new node being attached to a specific node. In other words, its aim is to answer the question of where to expand. Since a node can be attached to multiple nodes by playing different semantic roles, we utilize multi-head attention and take the maximum over different heads as the final arc probabilities.

Formally, through a multi-head attention mechanism taking h_t and $v_{0:t-1}$ as input, we obtain a set of attention weights $\{a_t^{g_i}\}_{i=1}^k$, where k is the number of attention heads and $a_t^{g_i}$ is the i-th probability vector. The probability of the arc between the new node and the node v_j is then computed by $a_{t,j}^g = max_i(a_{t,j}^{g_i})$. Intuitively, each head is in charge of a set of possible relations (though not explicitly specified). If certain relations do not exist between the new node and any existing node, the probability mass will be assigned to the dummy node d^*. The maximum pooling reflects that the arc should be built once one relation is activated.[3]

The attention mechanism passes the arc decisions to later layers by the update of the parser state as follows:

$$h_t = LN(h_t + W^{\text{arc}} \sum_{j=0}^{t-1} a_{t,j}^g v_j)$$

4.4 Concept Prediction

Our Concept Prediction component uses a soft alignment between words and the new concept. Concretely, a single-head attention a_t^s is computed based on the parser state h_t and the sentence representation $s_{1:n}$, where $a_{t,i}^s$ denotes the attention weight of the word w_i in the current time step. This component then updates the parser state with the alignment information via the following equation:

$$h_t = LN(h_t + W^{\text{conc}} \sum_{i=1}^n a_{t,i}^s s_i)$$

The probability of generating a specific concept c from the concept vocabulary $\mathcal{V}$ is calculated as $gen(c|h_t) = \exp(x_c^T h_t)/\sum_{c' \in \mathcal{V}} \exp(x_{c'}^T h_t)$, where x_c (for $c \in \mathcal{V}$) denotes the model parameters. To address the data sparsity issue in concept prediction, we introduce a copy mechanism in similar spirit to Gu et al. (2016). Besides generation, our model can either directly copy an input token w_i (e.g, for entity names) or map w_i to one concept $m(w_i)$ according to the alignment statistics[4] in the training data (e.g., for "went", it would propose go). Formally, the prediction probability of a concept c is given by:

$$P(c|h_t) = P(copy|h_t) \sum_{i=1}^n a_{t,i}^s [[w_i = c]]$$
$$+ P(map|h_t) \sum_{i=1}^n a_{t,i}^s [[m(w_i) = c]]$$
$$+ P(gen|h_t)gen(c|h_t)$$

where $[[\ldots]]$ is the indicator function. $P(copy|h_t)$, $P(map|h_t)$ and $P(gen|h_t)$ are the probabilities of three prediction modes respectively, computed by a single layer neural network with softmax activation.

4.5 Relation Classification

Lastly, the Relation Classification component employs a multi-class classifier for labeling the arcs detected in the Relation Identification component. The classifier uses a biaffine function to score each label, given the head concept representation v_i and the child vector h_t as input:

$$e_t^i = h_t^T W v_i + U^T h_t + V^T v_i + b$$

---

[3]We also found that there may exist more than one relation between two distinct nodes, however, it rarely happens.

[4]Based on the alignments provided by Liu et al. (2018), for each word, the most frequently aligned concept (or its lemma if it has empty alignment) is used for direct mapping.

where W, U, V, b are model parameters. As suggested by Dozat and Manning (2016), we project v_i and h_t to a lower dimension for reducing the computation cost and avoiding the overfitting of the model. The label probabilities are computed by a softmax function over all label scores.

4.6 Reentrancies

AMR reentrancy is employed when a node participates in multiple semantic relations (with multiple parent nodes), and that is why AMRs are graphs, rather than trees. The reentrancies are often hard to treat. While previous work often either remove them (Guo and Lu, 2018) or relies on rule-based restoration in the postprocessing stage (Lyu and Titov, 2018; van Noord and Bos, 2017), our model provides a new and principled way to deal with reentrancies. In our approach, when a new node is generated, all its connections to already existing nodes are determined by the multi-head attention. For example, for a node with k parent nodes, k different heads will point to the those parent nodes respectively. For a better understanding of our model, a pseudocode is presented in Algorithm 1.

4.7 Training and Inference

Our model is trained to maximize the log likelihood of the gold AMR graphs given sentences, i.e. $\log P(G|\mathbf{w})$, which can be factorized as:

$$
\begin{aligned}
\log P(G|\mathbf{w}) = \sum_{t=1}^{m} \Bigg(& \log P(c_t|G_{t-1}, \mathbf{w}) \\
& + \sum_{i \in pred(t)} \log P(arc_{it}|G_{t-1}, \mathbf{w}) \\
& + \sum_{i \in pred(t)} \log P(rel_{arc_{it}}|G_{t-1}, \mathbf{w}) \Bigg)
\end{aligned}
$$

where m is the total number of vertices. The set of predecessor nodes of c_t is denoted as $pred(t)$. arc_{it} denotes the arc between c_i and c_t, and $rel_{arc_{it}}$ indicates the arc label (relation type).

As mentioned, GSP is an autoregressive model, such as seq2seq models and transition models, but it factors the distribution according to a top-down graph structure rather than a depth-first traversal or a left-to-right chain. Meanwhile, GSP has a clear separation of node, arc and relation label probabilities, interacting in a more interpretable and tighten manner.

Algorithm 1 Graph Spanning based Parsing

Input: the input sentence $\mathbf{w} = (w_1, w_2, \ldots, w_n)$
Output: the AMR graph G corresponds to $\mathbf{w}$.
  ◇ Learning Sentence Representation
1: $\mathbf{w} = (w_0 = \triangleright) + (w_1, w_2, \ldots, w_n)$
2: $s_0, s_1, s_2, \ldots, s_n = \text{Transformer}(\mathbf{w})$
  ◇ Initialization
3: initialize the graph G_0 $(c_0 = d^*)$
4: initialize time step $t = 1$
  ◇ Entering Main Spanning Loop
5: **while** True **do**
6:    $h_0, \ldots, v_{t-1} = \text{Transformer}(c_{0:t-1})$
7:    $h_t = \text{Focus\_Selection}\ (s_0, v_{0:t-1}, s_{1:n})$
8:    $h_t = \text{Relation\_Identification}\ (h_t, v_{0:t-1})$
8:    *decide the parents nodes $pred(t)$ of c_t*
9:    $h_t = \text{Concept\_Prediction}\ (h_t, s_{1:n})$
9:    *decide the node type of c_t*
10:    **if** $c_t == \oslash$ **then**
11:       **break**
12:    **end if**
13:    **for** $i \in pred(t)$ **do**
14:       Relation_Classification (h_t, v_i)
14:       *decide the edge type between c_t and c_i*
15:    **end for**
16:    update G_{t-1} to G_t
17: **end while**
18: **return** G_{t-1}

At the operational or testing time, the prediction for the input $\mathbf{w}$ is obtained via $\hat{G} = \arg\max_{G'} P(G'|\mathbf{w})$. Rather than iterating over all possible graphs, we adopt a beam search to approximate the best graph. Specifically, for each partially constructed graph, we only consider the top-K concepts obtaining the best single-step probability (a product of the corresponding concept, arc, and relation label probability), where K is the beam size. Only the best K graphs at each time step are kept for the next expansion.

5 Experiments

5.1 Setup

We focus on the most recent LDC2017T10 dataset, as it is the largest AMR corpus. It consists of 36521, 1368, and 1371 sentences in the training, development, and testing sets respectively.

We use Stanford CoreNLP (Manning et al., 2014) for text preprocessing, including tokenization, lemmatization, part-of-speech, and named-entity tagging. The input for sentence encoder

consists of the randomly initialized lemma, part-of-speech tag, and named-entity tag embeddings, as well as the output from a learnable CNN with character embeddings as inputs. The graph encoder uses randomly initialized concept embeddings and another char-level CNN. Model hyperparameters are chosen by experiments on the development set. The details of the hyper-parameter settings are provided in the Appendix. During testing, we use a beam size of 8 for generating graphs.[5]

Conventionally, the quality of AMR parsing results is evaluated using the Smatch tool (Cai and Knight, 2013), which seeks for the maximum number of overlaps between two AMR annotations after decomposing AMR graphs into triples. However, the ordinary Smatch metric treats all triples equally regardless of their roles in the composition of the whole sentence meaning. We refine the ordinary Smatch metric to take into consideration the notion of core semantics. Specifically, we compute:

- Smatch-weighted: This metric weights different triples by their importance of composing the core ideas. The root distance d of a triple is defined as the minimum root distance of its involving nodes, the weight of the triple is then computed as:

$$w = \min(-d + d_{thr}, 1)$$

In other words, the weight has a linear decay in root distance until d_{thr}. If two triples are matched, the minimum importance score of them is obtained. In our experiments, d_{thr} is set to 5.

- Smatch-core: This metric only compares the subgraphs representing the main meaning. Precisely, we cut down AMR graphs by setting a maximum root distance d_{max} and only keep the nodes and edges within the threshold. d_{max} is set to 4 in our experiments, of which the remaining subgraphs still have a broad coverage of the original meaning, as illustrated by the distribution of root distance in Figure 3.

Besides, we also evaluate the quality by computing the following metrics.

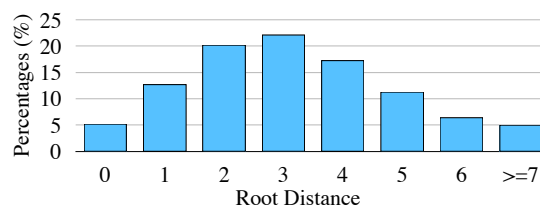⁵Our code can be found at https://github.com/jcyk/AMR-parser.

Figure 3: The distribution of root distance of concepts in the test set.

- complete-match (CM): This metric counts the number of parsing results that are completely correct.

- root-accuracy (RA): This metric measures the accuracy of the root concept identification.

5.2 Main Results and Case Study

The main result is presented in Table 1. We compare our method with the best-performing models in each category as discussed in § 2.

Concretely, van Noord and Bos (2017) is a character-level seq2seq model that achieves very competitive result. However, their model is very data demanding as it requires to train on additional 100K sentence-AMR pairs generated by other parsers. Guo and Lu (2018) is a transition-based parser with refined search space for AMR. Certain concepts and relations (e.g., reentrancies) are removed to reduce the burdens during training. Lyu and Titov (2018) is a graph-based method that achieves the best-reported result evaluated by the ordinary Smatch metric. Their parser uses different LSTMs for concept prediction, relation identification, and root identification sequentially. Also, the relation identification stage has the time complexity of $O(m^2 \log m)$ where m is the number of concepts. Groschwitz et al. (2018) views AMR as terms of the AM algebra (Groschwitz et al., 2017), which allows standard tree-based parsing techniques to be applicable. The complexity of their projective decoder is $O(m^5)$. Last but not least, all these models except for that of van Noord and Bos (2017) require hand-crafted heuristics for graph re-categorization.

We consider the Smatch-weighted metric as the most suitable metric for measuring the parser's quality on capturing core semantics. The comparison shows that our method significantly outperforms all other methods. The Smatch-core metric also demonstrates the advantage of our

3805

Model	Graph Re-ca.	Smatch(%)			RA(%)	CM(%)
		weighted	core	ordinary		
Buys and Blunsom (2017)	No	-	-	61.9	-	-
van Noord and Bos (2017) + 100K	No	68.8	67.6	71.0	75.8	10.2
Guo and Lu (2018)	Yes	63.5	62.3	69.8	63.6	9.4
Lyu and Titov (2018)	Yes	66.6	67.1	**74.4**	59.1	10.2
Groschwitz et al. (2018)	Yes	-	-	71.0	-	-
Ours	No	**71.3**	**70.2**	73.2	**76.9**	**11.6**

Table 1: Comparison with state-of-the-art methods (results on the test set). Results relying on heuristic rules for graph re-categorization are marked "Yes" in the Graph Re-ca. column.

Input: *The solution is functional human patterns and a balance between transport system capacity and land use generated travel demand.*
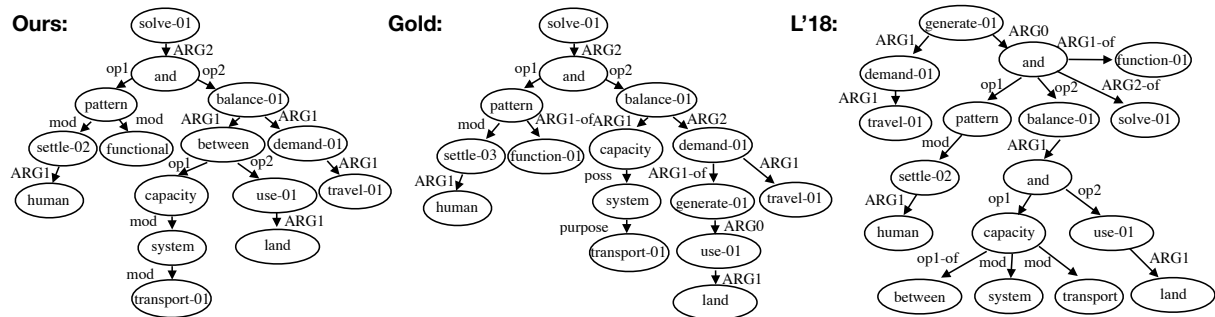

Figure 4: Case study.

Figure 5: Smatch scores with different root distances. vN'17 is van Noord and Bos (2017)'s parser with 100K additional training pairs. GL'18 is Guo and Lu (2018)'s parser. L'18 is Lyu and Titov (2018)'s parser.

method in capturing the core ideas. Besides, our model achieves the highest root-accuracy (RA) and complete-match (CM), which further confirms the usefulness of a global view and the core semantic first principle.

Even evaluated by the ordinary Smatch metric, our model yields better results than all previously reported models with the exception of Lyu and Titov (2018), which relies on a tremendous amount of manual heuristics for designing rules for graph re-categorization and adopts a pipeline approach. Note that our parser constructs the AMR graph in an end-to-end fashion with a better (quadratic) time complexity.

We present a case study in Figure 4 with comparison to the output of Lyu and Titov (2018)'s parser. As seen, both parsers make some mistakes. Specifically, our method fails to identify the concept generated-01. While Lyu and Titov (2018)'s parser successfully identifies it, their parser mistakenly treats it as the root of the whole AMR. It leads to a serious drawback of making the sentence meaning be interpreted in a wrong way. In contrast, our method shows a strong capacity in capturing the main idea "*the solution is about some p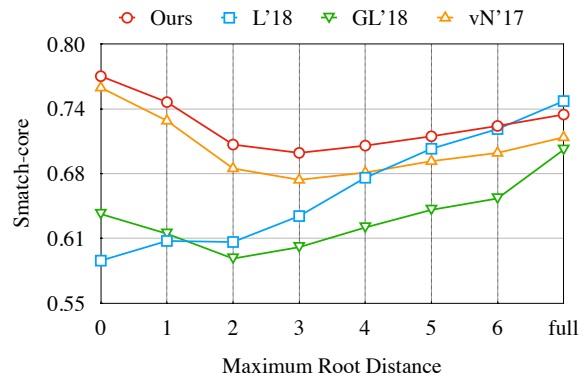atterns and a balance*". However, on the ordinary Smatch metric, their graph obtains a higher score (68% vs. 66%), which indicates that the ordinary Smatch is not a proper metric for evaluating the quality of capturing core semantics. If we adopt the Smatch-weighted metric, our method achieves a better score i.e. 74% vs. 61%.

5.3 More Results

To reveal our parser's ability for grasping meanings at different levels of granularity, we plot the Smatch-core scores in Figure 5 by varying

3806

the maximum root distance d_{max}, compared with several strong baselines and the state-of-the-art model. It demonstrates that our method is better at abstracting the core ideas of a sentence.

As discussed in § 3, there could be multiple valid generation orders for sibling nodes in an AMR graph. We experiment with the following traversal variants: (1) *random*, which sorts the sibling nodes in completely random order. (2) *relation freq.*, which sorts the sibling nodes according to their relations to the head node. We assign higher priorities to relations that occur more frequently, which drives our parser always to seek for the most common relation first. (3) *combined*, which combines the above two strategies by using *random* and *relation freq.* with equal chance. As seen in Table 2, the deterministic order strategy for training (*relation freq.*) achieves better performance than random order. Interestingly, the combined strategy significantly boosts the performance.[6] The reason is that the random order potentially produces a larger set of training pairs since each random order strategy can be considered as a different training pair. On the other hand, the deterministic order stabilizes the maximum likelihood estimate training. Therefore, the combined strategy benefits from both worlds.

6   Conclusion and Future Work

We presented the first top-down AMR parser. Our proposed parser builds a AMR graph incrementally in a root-to-leaf manner. Experiments show that our method has a better capability of capturing the core semantics in a sentence compared with previous state-of-the-art methods. In addition, we overcome the need of heuristics for graph recategorization employed in most previous work, which makes our method much more transferable to other semantic representations or languages.

Our methods follows the intuition that humans tend to grasp the core meaning of a sentence first. However, some cognitive theories (Langacker, 2008) also suggest that human language understanding is often presented as a circular, abductive process (hermeneutic circle). It is interesting to explore the use of some revision mechanisms when the initial steps go wrong.

---

[6]We note there are many other ways to generate a deterministic order. For example, van Noord and Bos (2017) uses the order of aligned words in the sentence. However, we use the relation frequency method for its simplicity and not relying on external resources (e.g, an aligner).

Order	Smatch		
	weighted	core	ordinary
random	68.2	67.4	70.4
relation freq.	69.9	68.3	70.9
combined	71.3	70.2	73.2

Table 2: The effect of different sibling orders.

References

Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage ccg semantic parsing with amr. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1699–1710.

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

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Miguel Ballesteros and Yaser Al-Onaizan. 2017. AMR parsing using stack-LSTMs. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1269–1275.

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.

Guntis Barzdins and Didzis Gosko. 2016. RIGA at SemEval-2016 task 8: Impact of Smatch extensions and character-level neural translation on AMR parsing accuracy. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1143–1147.

Jan Buys and Phil Blunsom. 2017. Oxford at semeval-2017 task 9: Neural amr parsing with pointer-augmented attention. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 914–919.

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 748–752.

Marco Damonte, Shay B. Cohen, and Giorgio Satta. 2017. An incremental parser for abstract meaning representation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 536–546.

Timothy Dozat and Christopher D Manning. 2016. Deep biaffine attention for neural dependency parsing. *arXiv preprint arXiv:1611.01734*.

Jeffrey Flanigan, Chris Dyer, Noah A Smith, and Jaime Carbonell. 2016. Cmu at semeval-2016 task 8: Graph-based amr parsing with infinite ramp loss. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1202–1206.

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A Smith. 2014. A discriminative graph-based parser for the abstract meaning representation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1426–1436.

William Foland and James H Martin. 2017. Abstract meaning representation parsing using lstm recurrent neural networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 463–472.

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.

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.

Zhijiang Guo and Wei Lu. 2018. Better transition-based amr parsing with refined search space. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1712–1722.

Hardy Hardy and Andreas Vlachos. 2018. Guided neural language generation for abstractive summarization using abstract meaning representation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 768–773.

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

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural AMR: Sequence-to-sequence models for parsing and generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 146–157.

Sandra Kübler, Ryan McDonald, and Joakim Nivre. 2009. Dependency parsing. *Synthesis Lectures on Human Language Technologies*, 1(1):1–127.

Ronald W Langacker. 2008. *Cognitive Grammar: A Basic Introduction*. Oxford University Press.

Matthias Lindemann, Jonas Groschwitz, and Alexander Koller. 2019. Compositional semantic parsing across graphbanks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4576–4585.

Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman Sadeh, and Noah A. Smith. 2015. Toward abstractive summarization using semantic representations. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1077–1086.

Yijia Liu, Wanxiang Che, Bo Zheng, Bing Qin, and Ting Liu. 2018. An AMR aligner tuned by transition-based parser. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2422–2430.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Chunchuan Lyu and Ivan Titov. 2018. AMR parsing as graph prediction with latent alignment. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 397–407.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60.

Arindam Mitra and Chitta Baral. 2016. Addressing a question answering challenge by combining statistical methods with inductive rule learning and reasoning. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Tahira Naseem, Abhishek Shah, Hui Wan, Radu Florian, Salim Roukos, and Miguel Ballesteros. 2019. Rewarding Smatch: Transition-based AMR parsing with reinforcement learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4586–4592.

3808

Rik van Noord and Johan Bos. 2017. Neural semantic parsing by character-based translation: Experiments with abstract meaning representations. *arXiv preprint arXiv:1705.09980*.

Xiaochang Peng, Daniel Gildea, and Giorgio Satta. 2018. Amr parsing with cache transition systems. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

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.

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.

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*, pages 425–429.

Michael Pust, Ulf Hermjakob, Kevin Knight, Daniel Marcu, and Jonathan May. 2015. Parsing english into abstract meaning representation using syntax-based machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1143–1154.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5027–5038.

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.

Chuan Wang, Sameer Pradhan, Xiaoman Pan, Heng Ji, and Nianwen Xue. 2016. Camr at semeval-2016 task 8: An extended transition-based amr parser. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1173–1178.

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.

Keenon Werling, Gabor Angeli, and Christopher D. Manning. 2015. Robust subgraph generation improves abstract meaning representation parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 982–991.

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*, pages 80–94.

Yue Zhang and Stephen Clark. 2009. Transition-based parsing of the chinese treebank using a global discriminative model. In *Proceedings of the 11th International Conference on Parsing Technologies*, pages 162–171.

Junsheng Zhou, Feiyu Xu, Hans Uszkoreit, Weiguang Qu, Ran Li, and Yanhui Gu. 2016. AMR parsing with an incremental joint model. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 680–689.