ORIGINAL ARTICLE

From text to graph: a general transition-based AMR parsing using neural network

Min Gu[1] · Yanhui Gu[1] · Weilan Luo[2] · Guandong Xu[3] · Zhenglu Yang[4] · Junsheng Zhou[1] · Weiguang Qu[1]

Received: 12 March 2020 / Accepted: 18 September 2020 / Published online: 15 October 2020

Abstract
Semantic understanding is an essential research issue for many applications, such as social network analysis, collective intelligence and content computing, which tells the inner meaning of language form. Recently, Abstract Meaning Representation (AMR) is attracted by many researchers for its semantic representation ability on an entire sentence. However, due to the non-projectivity and reentrancy properties of AMR graphs, they lose some important semantic information in parsing from sentences. In this paper, we propose a general AMR parsing model which utilizes a two-stack-based transition algorithm for both Chinese and English datasets. It can incrementally parse sentences to AMR graphs in linear time. Experimental results demonstrate that it is superior in recovering reentrancy and handling arcs while is competitive with other transition-based neural network models on both English and Chinese datasets.

Keywords Semantic analysis · AMR parsing · Two-stack-based transition algorithm · Neural network

1 Introduction

Meaning Representation of natural language is an important issue for massive data in the real world. How to achieve complete semantic understanding of natural language sentences in real-world data has been attracted by researchers [1–6]. When applying natural language processing technologies to mine semantic information and understand the real meaning of the data, semantic representation is the carrier of semantic information.

In recent years, graph representation-based strategies show their excited performance in expressing information in complex condition, such as social network representation [3, 7–10], chemical molecule representation [11–15] and so forth. Because of the characteristics of real-world data and the properties of graphs which can handle much more complex information, we intend to harness the wisdom of graphs to represent the semantic information in sentences. Because of the ambiguity and polysemy of semantics, it is a great challenge to realize the semantic understanding of natural language sentences. Traditional sentence semantic understanding research usually designs a formal meaning representation form for a specific domain [16, 17]. However, the real world is a multi-domain hybrid environment where people could consume, produce and share information easily. Semantic representation needs a domain-independent general representation method to mine semantic information in multi-domain natural language sentences.

To tackle this issue, we propose an automatic semantic parsing model based on a representative semantic representation method, i.e., Abstract Meaning Representation (AMR). We represent semantics in natural language sentences by semantic graphs to express the deep semantic

✉ Yanhui Gu
gu@njnu.edu.cn

✉ Zhenglu Yang
yangzl@nankai.edu.cn

[1] School of Computer Science and Technology, Nanjing Normal University, Nanjing, China

[2] School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing, China

[3] Advanced Analytics Institute, University of Technology Sydney, Sydney, Australia

[4] Institute of Big Data, College of Computer Science, Nankai University, Tianjin, China

information in sentences. AMR is a domain-independent semantic representation language which represents an entire sentence as a rooted, directed, acyclic graph [18]. Due to its semantic representation ability for entire sentence, AMR can assist in various semantic-based tasks, such as Text Summarization, Machine Translation, Event Extraction and so forth [19–24].

For a given sentence "*Iranian law states the death penalty for drug trafficking.*", its AMR graph is illustrated in Fig. 1. A word may correspond to a concept fragment formed by a single concept (e.g., "state-01") or multiple concepts (e.g., "country + name + "Iran""). In an AMR graph, concepts are represented as labeled nodes and relations are represented as labeled and directed arcs. Some AMR graphs are non-projective (having crossing arcs) or have reentrant nodes (e.g., node "law" has two parent nodes: "state-01" and "penalize-01" in Fig. 1. "他" has two parent nodes: "想-02" and "吃-01" in Fig. 2a), which make the generation of AMR graphs more difficult. AMR uses graph variables and reentrancy to express coreference. The reentrancy prevents the AMR graph from being a tree structure [25].

When transforming natural language sentences to AMR graphs, transition-based models [26] and graph-based models [27] are two common strategies. Previous transition-based methods mainly have two methods. One is utilizing dependency tree as an intermediary and performing tree-to-graph transformation [26, 28–32]. The other takes sentences as input and directly parses sentences to graphs [33–36]. Transition-based model predicts transition actions and constructs an AMR graph based on predicted actions. However, traditional transition-based decoding methods only fit dependency trees which are projective and do not perform well in crossing arcs and reentrancy situations. Thus, previous methods try to design complex actions for graph construction [37] or introduce extra classifiers for reentrancy arcs generation [33], but the reentrancy classifier does not perform well.

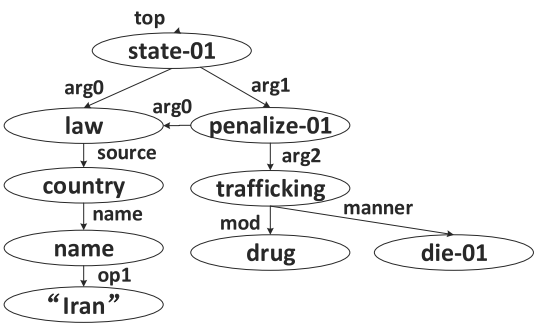


Fig. 1 AMR graph of English sentence "*Iranian law states the death penalty for drug trafficking.*" from LDC2014T12 dataset

In our automatic semantic parsing strategy, we propose a two-stack-based transition algorithm to realize the generation of complete AMR graphs incrementally. The decoding algorithm can cover more AMR graphs with simple actions. With the decoding algorithm, a general transition-based AMR parser using neural network is constructed for both English and Chinese data. The experimental evaluation demonstrates the effectiveness of our proposed model on both English and Chinese datasets.

Our contributions are listed as follows:

(1) We survey the state-of-the-art AMR semantic parsing models and analyze them from different aspects.
(2) We propose an effective general strategy for transforming English or Chinese natural language sentences to semantic graphs.
(3) We directly parse sentences to AMR graphs without transformation of dependency trees. It reduces information loss and error propagation.

The remainder of this paper is organized as follows: Sect. 2 mainly introduces the related work of semantic representation and AMR semantic parsing. Section 3 describes the proposed model. Section 4 shows the experiments and discusses the experimental results. Finally, we conclude this paper in Sect. 5.

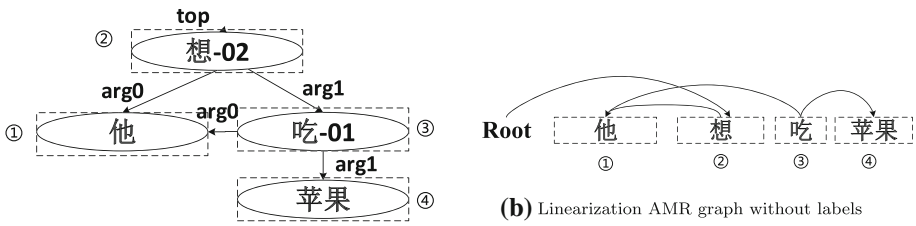
2 Related work

2.1 Semantic representation by applying graph construction

In this paper, we select AMR as the graph representation method. AMR is a domain-independent semantic representation language for sentences. It abstracts the semantics of a sentence into a single directed acyclic graph. Compared with English AMR annotation, the research of Chinese AMR annotation started late [39]. Based on the framework of English AMR annotation, Li et al. introduce AMR semantic representation system into Chinese [40]. They focus on solving the issues of AMR concept and word alignment and initially propose Chinese AMR annotation specification.

Compared with dependency trees, AMR graphs have more complicated structure. Sometimes, it is non-projective and may have reentrant nodes. Nodes indicate concepts in AMR graph, which can be either English or Chinese words (e.g., law, 他, 苹果) or PropBank framesets [41] (e.g., want-01, 想 -02, state-01).

Based on AMR annotation specification and datasets, the researchers attempt to automatically convert sentences to AMR graphs through different AMR transformation



(a) AMR graph (他: he, 想-02: want-02, 吃: eat-01, 苹果: apples)

(b) Linearization AMR graph without labels

Fig. 2 AMR graph of Chinese sentence "他想吃苹果 *(He wants to eat apples)*"

algorithms. AMR transformation is designed to parse sentences into corresponding AMR graphs. In an AMR graph, the label of a directed arc represents the relationship between two concepts. As shown in Fig. 1 "*Iranian law states the death penalty for drug trafficking.*", the arc between concept "law" and concept "state-01" is marked with ":arg0" relation label, indicating the subject–predicate relationship. The arc between concept "他 -02" and concept "想 " is also labeled with ":arg0" in Fig. 2a.

AMR contains more than one hundred semantic relations, which can be summarized into five types. Table 1 gives some examples of the five types. The collection of all concepts and relationships in AMR enables it to abstractly represent the semantics of sentences in a reasonable and consistent way.

AMR graphs have the following characteristics:

Concept: It is the basic component of AMR graphs in English and Chinese AMR graphs. A node in a graph indicates a concept. A concept or a concept fragment can be aligned to words in sentences. For example, the word "想" in sentence "他想吃苹果*(He wants to eat apples.)*" is aligned to concept "想-02."

Non-projectivity: It means that there are crossing arcs in AMR graphs if the arcs are drawn in sentence order, e.g., the arc between "他" and "吃-01" is crossing with another arc which is illustrated in Fig. 2b.

Reentrancy: It indicates that the node in an AMR graph has multiple parent nodes. We call the node "**Reentrant Node**" and the arc linked to the node "**Reentrant Arc**." For example, Reentrant Node "law" has two parent nodes "state-01" and "penalize-01" in Fig. 1. Reentrant Node "他 " also has two parent nodes " 想 -02" and " 吃 -01" which is shown in Fig. 2a.

In 200 manually aligned AMR graphs, 41% of them have reentrant nodes and 51% are non-projective graphs [33]. These unique characteristics of AMR graphs make the prediction of transition actions in transition-based AMR transformation become more complicated.

Table 1 Examples of AMR relations

Relation types	Examples
PropBank framesets	:arg0, :arg1, :arg2
General	:age, :location, :name
Date	:day, :month, :time
List	:op1, :op2, :op3
Number	:quant, :unit, :scale

2.2 Automatic semantic representation transformation

The traditional English semantic AMR graph transformation based on statistical learning model can be divided into four different types: graph-based model [27, 42–45], transition-based model [28–35, 46], Combinatory Categorial Grammar-based model [47–49] and Machine Translation-based model [50]. The transformation process is also named as parsing. Among them, graph-based parsing models and transition-based parsing models are two most common models. Flanigan et al. propose the first AMR parser: JAMR [42]. It is a graph-based model which divides the AMR parsing task into two subtasks: concept recognition task and relation recognition task. Concept recognition is the process of mapping a word or a word string in an input sentence to a concept or concept fragment in AMR graphs. It applies the semi-Markov model to label the concepts. Based on the concept sequence, a Maximum Spanning Connected Subgraph (MSCG) algorithm is used to search all the relationships between concepts to find the subgraph with the highest score. The state-of-the-art graph-based model is an attention-based model that treats AMR parsing as sequence-to-graph transduction [51]. Transition-based model has two types. One is represented by CAMR [26], which generates AMR graphs from dependency trees. The other directly parses sentences to graphs through transition algorithms [33].

Traditional statistical learning-based models rely on artificial feature engineering to obtain complex features, and the combined features may make parameter space of the model become too large and result in low efficiency on time and space. Considering the representation learning ability of neural networks, AMR parsing models based on neural network models were proposed for English datasets

[33]. The application of neural network models in AMR parsing can be divided into two types: (1) learning feature representation through neural network model, using graph-based or transition-based decoding method to generate AMR graphs, and (2) utilizing sequence-to-sequence model to generate serialized AMR graphs directly from sentences [52].

On the basis of CAMR, Puzikov et al. use the Feed-forward Neural Network to predict transition actions [46]. With the combination of numerical features and word embeddings, a multilayer Feedforward Neural Network classifier is applied to predict actions. Ballesteros et al. propose utilization of Stack-LSTM for AMR state representation learning, which can obtain feature representations without external resources [35].

Barzdins and Gosko first use the sequence-to-sequence (Seq2Seq) model in AMR parsing. They apply the depth-first algorithm to serialize AMR graphs, encode the intermediate representation through LSTM and then obtain the serialized AMR representation by decoding [53]. However, due to the sparse of data, its accuracy is lower than the statistical learning-based model. On the basis of this, Konstas et al. utilize stacked LSTM to optimize representation learning with large-scale unlabeled datasets as external resources for self-training and improve the AMR parsing performance of sequence-to-sequence-based models [54]. Guo and Lu also apply stacked LSTM with nine types of actions to construct AMR graphs. They design a representation called compact AMR graph to simplify concepts and relations of an AMR graph. Their model achieves Smatch F1 score of 0.683 on LDC2014T12 dataset with optimized aligner and compact AMR graph [37].

Similar to English AMR parsing based on the statistical learning model, Wang et al. propose a method for transforming dependency trees to Chinese AMR graphs [36]. The model consists of two steps. First, an existing dependent parser is used to generate the corresponding dependency trees; then, the transition algorithm is applied to convert dependency trees to AMR graphs. The model designs nine transition actions. By predicting transition actions, the greedy decoding algorithm selects the action with the highest score from the action set to perform actions on the corresponding dependency trees, thus transforming the dependency tree to the AMR graph. However, this model relies on dependency tree as an intermediary, and errors in dependency parsing are directly propagated to AMR parsing.

Considering the researches on English and Chinese AMR parsing and the issues in AMR parsing, this paper proposes an extended Shift/Reduce decoding algorithm based on two stacks for AMR parsing. It is a transition-based model which can incrementally transform sentences to AMR graphs. It also integrates word, POS tags and dependency information for parsing. It can reduce loss of information in transformation and cover more AMR graphs in training.

2.3 Sentence modeling and feature learning

Text modeling is the basis of NLP tasks. Researchers apply models, such as RNN, CNN and pre-training models, to learn word, character and fragment information for NLP researches [38]. According to feature extraction methods, AMR parsing models can be divided into (a) combined feature learning model, (b) RNN-based feature learning model and (c) CNN-based feature learning model.

(a) Combined feature learning model

In traditional graph decoding dependency parsing models, the features are expressed as high-dimensional and sparse vectors, which are independent. This feature representation method cannot capture the commonalities between features and also suffers from data sparse problem. In graph decoding dependency parsing models which are based on neural network, features are mapped to low-dimensional and dense vectors. Features with close syntactic and semantic properties are embedded in adjacent positions, which shows commonality or similarity in semantic space. In order to capture the nonlinear relationship between features and goals of models, it is necessary to utilize combined features, that is, combine multiple basic features as new features. Puzikov et al. use a combination of numerical features and embedding features [46] to connect features through hidden layers to form a combined vector representation. Damonte et al. apply numerical features (such as the depth of the current concept subgraph, the number of child nodes and the number of parent nodes), word embeddings (words in the sentence corresponding to the concept subgraph), part-of-speech tag embeddings, and dependency tags embeddings to represent a certain configuration in the transition system [33]. Though combined features are effective in AMR parsing models, the introduction of combined features increases the difficulty of feature engineering. People need to try and select effective features from a large number of combined features.

(b) RNN-based feature learning model

Barzdins and Gosko utilize the Seq2seq model for the first time in AMR analysis. They use a depth-first algorithm to serialize the AMR graph, but due to data sparsity, its accuracy is much lower than the model based on statistical learning [53]. Foland et al. also propose introducing RNN into AMR parsing [44]. However, due to data sparseness and the limitation of AMR graph structure, its application in AMR analysis is insufficient.

(c) CNN-based feature learning model

The affixes and roots of English words contain rich semantic information. For example, in the word "unprecedented" the prefix "un" needs to be marked as a ": polarity" relationship in AMR. Although word embeddings can describe the information of a word, the information contained in the affix and root may not be clear. Therefore, Wang et al. propose a model which encodes character feature by CNN [32] to obtain character-level features for the recognition of AMR concepts.

3 Proposed model

We propose a general automatic semantic representation model based on the two-stack-based transition algorithm for transforming natural language sentences to semantic graphs. In the model, the transition decoding model is a model which incrementally parses sentences by performing actions such as Shift and Reduce based on defined configuration. Nivre proposes two transition-based algorithms for dependency parsing: arc-standard [55] and arc-eager [56]. The goal of dependency parsing task is to parse the input sentence to a dependency tree. Dependency trees can represent the syntactic relationships between words in sentences. The nodes in a tree represent the words in a sentence, and the arcs represent the dependency syntax relationship between two words. Dependency trees and AMR graphs are similar in structure, their nodes both represent a lexical content, and arcs all represent semantic relationships. In addition, there is a strong correlation between AMR parsing task and dependency parsing task. The input of both tasks is a sentence, and the output is an abstract representation structure corresponding to the sentence. Therefore, appropriate improvements on traditional dependency parsing algorithms can be applied to AMR parsing tasks. Inspired by the algorithm based on transition-based decoding for dependency parsing: arc-eager and its improvement in directed acyclic graphs proposed by Sagae et al. [57], this paper proposes an extended two-stack-based transition-based AMR parsing model for both English and Chinese AMR. It consists of two Feedforward Neural Network models. It can incrementally parse sentences to AMR graphs from left to right in linear time. The baseline strategy of this paper is a parsing model based on AMR-eager, which is similar to Damonte et al. [33]. We also apply and modify the proposed model in [33] on the Chinese dataset.

3.1 Transition-based algorithms for automatic semantic representation transformation

Transition-based algorithms utilize configurations or states to formalize parsing process. A triple (σ, β, A) is used to represent a configuration. It contains a stack σ for nodes in partially constructed AMR graph; a buffer β for words in a sentence; and a list A for constructed arcs. When a parsing procedure starts, it calculates scores for taking transition actions to construct a new configuration according to current configuration. Transition actions are diverse in different models. Transition actions with the highest score will be chosen to update current configuration. It processes sentences in a left–right order until the stack and buffer are both empty. The most popular transition-based algorithms are arc-standard and arc-eager.

3.1.1 Arc-standard algorithm

For a sequence of words $w = \{w_1, w_2, \ldots, w_n\}$, $w_i \rightarrow w_j$ indicates that there is an arc from w_i to w_j. Arc-standard contains three transition actions:

(1) Left-reduce(l): Add an arc with the label l from the first element in the buffer to the top element of the stack, and remove the top element from the stack;
(2) Right-reduce(l): Add an arc labeled l from the top element of the stack to the first element of the buffer, and replace the first element of the buffer with the top element of the stack and remove the top element from the stack;
(3) Shift: The first element of the buffer is moved to the stack, which is used to determine whether a word in the buffer needs to be pushed into the stack.

Transition actions' definitions are stated in Table 2.

3.1.2 Arc-eager algorithm

The arc-eager algorithm uses the same triple to store parsing configuration. It modifies transition actions in arc-standard. Arc-eager contains four transition actions: Shift, Reduce, LArc(l) and RArc(l). Shift and LArc(l) are the same as those in arc-standard algorithm. Reduce indicates removing the top element in the stack. RArc(l) means adding an arc with label l from the top element in stack to the first element in the buffer and removing the first element to the stack. Table 3 gives the formal definitions of these four transition actions and their prerequisites.

Table 2 Transition actions' definitions and prerequisites in arc-standard algorithm

Actions	Definitions	Prerequisites
Shift	$(\sigma, \beta_0\|\beta, A) \rightarrow (\sigma\|\beta_0, \beta, A)$	$\|\beta\| \geqslant 1$
Left-reduce(l)	$(\sigma\|\sigma_0, \beta_0\|\beta, A) \rightarrow (\sigma, \beta_0\|\beta, A \cup \{<\beta_0, l, \sigma_0>\})$	$\|\sigma\| \geqslant 2$ and $\|\beta\| \geqslant 1$ and σ_0 is not root and does not contain root node
Right-reduce(l)	$(\sigma\|\sigma_0, \beta_0\|\beta, A) \rightarrow (\sigma, \sigma_0\|\beta, A \cup \{<\sigma_0, l, \beta_0>\})$	$\|\sigma\| \geqslant 2$ and $\|\beta\| \geqslant 1$ and β_0 is not root

Table 3 Transition actions' definitions and prerequisites in arc-eager algorithm

Actions	Definitions	Prerequisites
Shift	$(\sigma, \beta_0\|\beta, A) \rightarrow (\sigma\|\beta_0, \beta, A)$	$\|\beta\| \geqslant 1$
Reduce	$(\sigma\|\sigma_0, \beta, A) \rightarrow (\sigma, \beta, A)$	$\|\sigma\| \geqslant 1$
LArc(l)	$(\sigma\|\sigma_0, \beta_0\|\beta, A) \rightarrow (\sigma, \beta_0\|\beta, A \cup \{<\beta_0, l, \sigma_0>\})$	$\|\sigma\| \geqslant 2$ and $\|\beta\| \geqslant 1$ and σ_0 is not root and does not contain root node
RArc(l)	$(\sigma, \beta_0\|\beta, A) \rightarrow (\sigma\|\beta_0, \beta, A \cup \{<\sigma_0, l, \beta_0>\})$	$\|\sigma\| \geqslant 2$ and $\|\beta\| \geqslant 1$ and β_0 does not contain root node

3.2 General automatic semantic parsing model based on two-stack-based transition algorithm

The baseline strategy of this paper is based on the arc-eager algorithm. It applies heuristic search method for concept recognition. In order to modify the transition algorithm to suit AMR transformation, we introduce a two-stack-based transition algorithm to solve the problem in AMR graph generation. Inspired by [58], the two-stack-based transition algorithm is an extended Shift/Reduce decoding algorithm based on two stacks. We modify the transition actions in [58]. At the same time, we apply more appropriate feature representation to enrich feature representation learning in the prediction of transition actions. This model can be divided into six parts, which are illustrated in Fig. 3. Among them, pre-training, AMR concept annotation and named-entity recognition are pre-processing processes. Based on external datasets, we train word embeddings using Word2vec model as the input of this model. The AMR concept annotation aligns words to concepts and constructs an alignment table through an aligner [42]. This table is the input for the concept recognition module. In addition, we utilize Corenlp to label named entities in dataset as features for the concept recognition module and transition-based AMR parsing module.

After pre-processing, the heuristic search-based concept recognition module uses the alignment table to train the concept recognition model. The concept recognition model returns concepts or concept fragments corresponding to words.

Based on concept recognition results, we train two Feedforward Neural Network classifiers for predicting transition actions and arc labels in AMR parsing model based on the extended Shift/Reduce algorithm. We design two classifiers for English and Chinese AMR parsing: (a) transition action classifier and (b) label classifier. According to concept recognition result and prediction results from the classifiers, the model can construct a single root, directed and acyclic AMR graph. Finally, evaluation is performed on generated AMR graphs and gold AMR graphs.

3.2.1 Concept recognition algorithm

We define the set {1,2,...,n} as $[n]$. An AMR graph is defined as a triple (G, x, π), where s represents sentence $x = \{x_1, x_2, \ldots, n_n\}$. $x_i (i \in [n])$ is the word in the sentence. $G = (V, E)$ is a directed graph, where V is the node set and E stores arcs. $\pi : V \rightarrow [n]$ is the mapping from concepts in AMR graphs to word positions in sentences. A node v in an AMR graph is abstracted from word $x_{\pi(v)}$. We define the mapping from word at position i in a sentence to AMR concept as:

$$\begin{aligned} \overleftarrow{\pi}(i) &= (\pi^{-1}(i), E \cap (\pi^{-1}(i) \times \pi^{-1}(i))), \\ \pi^{-1}(i) &= \{v | v \in V, \pi(v) = i\} \end{aligned} \tag{1}$$

Based on alignment annotations, we can obtain an alignment table of word–concept pairs. According to alignment result, this model utilizes heuristic search algorithm to select the concept with the highest score corresponding to the current word. The concept acts as input for the transition-based module.

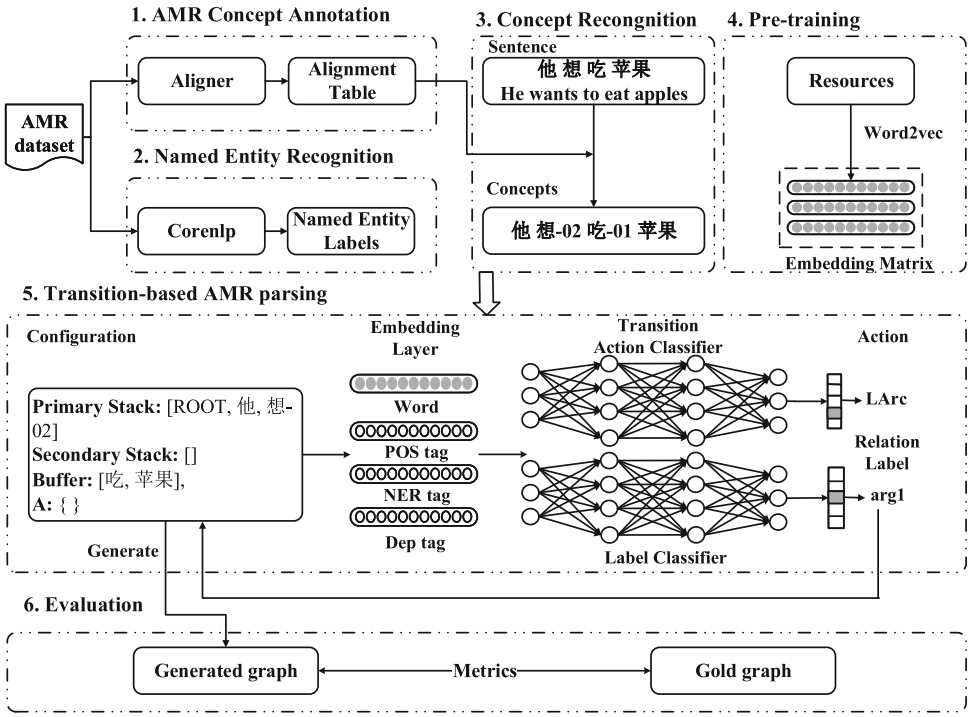


Fig. 3 General automatic semantic parsing model based on two-stack-based transition algorithm

For better comprehension, we define the mapping function as $\overleftarrow{\pi}'$. For $i \in [n]$, the result of $\overleftarrow{\pi}'$ is a subgraph $G_i = (V_i, E_i)$. The subgraph has only one root node, denoted as $root(G_i)$. $\overleftarrow{\pi}'$ allows a word map to a complex concept subgraph or none.

3.2.2 Two-stack-based transition algorithm

Similar to two-stack-based transition-based dependency parsing [58], we propose a two-stack-based transition-based AMR parser. Different transition actions, including **SHIFT**, **REDUCE**, **LEFT_ARC**(l), **RIGHT_ARC**(l) and **MEM**, are applied to update configurations and construct AMR graphs. In the two-stack-based transition algorithm, a configuration is defined as $\{\sigma, \sigma', \beta, A\}$. It contains a primary stack σ for nodes in partially constructed AMR graph; a secondary stack σ' as a cache for nodes; a buffer β for words; and a list A for constructed arcs. When a parsing procedure starts, we calculate scores for taking transition actions to construct a new configuration according to current configuration. In the five transition actions, LEFT_ARC(l), RIGHT_ARC(l), SHIFT and REDUCE are similar to those in the arc-eager algorithm. They only operate on the primary stack. The fifth action MEM pushes the top element in the primary stack to the secondary stack to handle crossing arcs and reentrancy issues.

Table 4 gives the formal definitions of these five transition actions and their prerequisites. (1)**SHIFT**: remove the first node in β and push nodes in σ' to σ. (2)**LEFT/RIGHT_ARC**(l): update a configuration by adding an arc (i, l, j) to A, where i is the top of σ, and j is the first node in β. (3)**REDUCE**: update a configuration by popping the top of σ. (4)**MEM**: pop the top element from σ and push it into σ'.

Algorithm 1 Oracle algorithm

Input: Sentence $x = \{x_0, x_1, ..., x_n\}$ and gold AMR graph $G_g = (V_g, A_g)$
Output: Transition action sequence T

1: $T = [\,]$
2: **while** $index < size(x)$ **do**
3: $\quad concept \leftarrow getConcept(x[index])$
4: $\quad index \leftarrow index + 1$
5: **end while**
6: $C \leftarrow C_s(ConSet)$
7: **if** $\exists l[(\beta_0, l, \sigma_0) \in A_g]$ **then**
8: $\quad$ T.append(LEFT_ARC(l))
9: **else if** $\exists l[(\sigma_0, l, \beta_0) \in A_g]$ **then**
10: $\quad$ T.append(RIGHT_ARC(l))
11: **else if** $\exists i \in [1, n], l[(\sigma_0, l, \beta_i) \in A_g \vee (\beta_i, l, \sigma_0) \in A_g]$ **then**
12: $\quad$ T.append(MEM)
13: **else if** $T[-1] \neq MEM\ and \nexists i \in [1, n], l[(\sigma_0, l, \beta_i) \in A_g \vee (\beta_i, l, \sigma_0) \in A_g]$ **then**
14: $\quad$ T.append(REDUCE)
15: **else**
16: $\quad$ T.append(SHIFT)
17: **end if**

We utilize an oracle algorithm to obtain gold transition actions in parsing for training, as shown in Algorithm 1. In training procedure, the model processes each word in a sentence from left to right and obtains transition actions based on current configurations until the end of the sentence. The oracle algorithm returns a gold transition action sequence T. Firstly, we obtain AMR concept of each word in a sentence (lines 2 to 5) and initialize a configuration (line 6). Then, we add transition actions to action sequence T on the basis of current configurations. If there is a left arc

Table 4 Transition actions' definitions and prerequisites in the two-stack-based algorithm

Actions	Definitions	Prerequisites
LEFT_ARC(l)	$(\sigma\|\sigma_0,\sigma',\beta_0\|\beta,A) \rightarrow (\sigma\|\sigma_0,\sigma',\beta_0\|\beta,A \cup \{<\beta_0,l,\sigma_0>\})$	$\|\sigma\| \geqslant 1$and $\|\beta\| \geqslant 1$
RIGHT_ARC(l)	$(\sigma\|\sigma_0,\sigma',\beta_0\|\beta,A) \rightarrow (\sigma\|\sigma_0,\sigma',\beta_0\|\beta,A \cup \{<\sigma_0,l,\beta_0>\})$	$\|\sigma\| \geqslant 1$ and $\|\beta\| \geqslant 1$
SHIFT	$(\sigma,\sigma'\|\sigma_0',\beta,A) \rightarrow (\sigma\|\sigma_0,\sigma',\beta,A), then(\sigma,\sigma',\beta_0\|\beta,A) \rightarrow (\sigma\|root(a(\beta_0)),\sigma',\beta,A \cup E_a), where\, a(\beta_0) = (V_a,E_a)$	$\|\beta\| \geqslant 1$
REDUCE	$(\sigma\|\sigma_0,\sigma',\beta,A) \rightarrow (\sigma,\sigma',\beta,A)$	$\|\sigma\| \geqslant 2$
MEM	$(\sigma\|\sigma_0,\sigma',\beta,A) \rightarrow (\sigma,\sigma'\|\sigma_0,\beta,A)$	$\|\sigma\| \geqslant 2$

[1] $a(\beta_0) = (V_a,E_a)$ indicates the AMR subgraph which β_0 corresponds to

or right arc between the top node of stack σ and the first node in buffer β, action **LEFT_ARC** or **RIGHT_ARC** with an arc label l will be added to T (lines 7 to 10). If the top node of stack σ has arcs with others in buffer β except the first node, action **MEM** will be accepted (lines 11 to 12). If last action is not **MEM** and the top node in stack σ has no other arcs with nodes in buffer β, the node will be deleted from the stack (lines 13 to 14). Otherwise, action **SHIFT** will be taken (lines 15 to 16).

For example, the parsing process of sentence "他想吃苹果" in two-stack-based algorithm and arc-eager algorithm is shown in Tables 5 and 6. We also illustrate the graph generation procedure in two-stack-based algorithm in Fig. 4. For each sentence in the training dataset, given the corresponding standard concept sequence, using the transition action set in the baseline strategy does not construct a valid transition action sequence to generate the corresponding gold AMR graph. When the gold AMR graph corresponding to the sentence contains a reentrant node, the AMR graph parsed by the baseline strategy loses the reentrant arc, e.g., the arc <吃-01,:arg0,他> in gold AMR graph is missing in arc-eager in Table 6.

In view of the shortcomings of the baseline strategy, this paper redefines and expands the set of transfer actions in the baseline strategy. Therefore, it can cover the entire training dataset more effectively.

In transition decoding, we propose two classifiers based on Feedforward Neural Network: transition action classifier and label classifier. Transition action classifier predicts next transition action according to current configuration. The feature in this classifier is shown in Table 7. When the current action is LEFT_ARC or RIGHT_ARC, we need to tell the label of the new generated arc. The label classifier predicts labels based on the configuration after performing LEFT_ARC and RIGHT_ARC. Its features are shown in Table 8.

Compared with the proposed model, the baseline strategy utilizes three classifiers: transition action classifier, label classifier and reentrancy classifier. The former two classifiers are similar to our proposed model. The last classifier predicts whether the concept node and its sibling need to add a reentrancy arc. The features for transition action classifier, label classifier and reentrancy classifier are shown in Tables 9, 10 and 11.

In the proposed model, we add more dependency features for training, since the dependency relation indicates the semantic connection between words. We propose to utilize a multilayer Feedforward Neural Network to construct classifiers. Its structure is illustrated in Fig. 5.

For each classifier, its features are represented as:

$$e_i = [e_i^x; e_i^p; e_i^{NER}; e_i^d; e^n] \tag{2}$$

Table 5 Parsing process of sentence 他想吃苹果“ (He wants to eat apples.)” in the proposed model

No.	Action	σ	σ'	β	A
1	-	[•]	[ ]	[他, 想, 吃, 苹果]	{}
2	SHIFT	[•, 他]	[ ]	[想, 吃, 苹果]	{}
3	LEFT_ARC	[•, 他]	[ ]	[想,吃, 苹果]	{<想-02, :arg0, 他>} = A_1
4	MEM	[•]	[他]	[想,吃, 苹果]	A_1
4	RIGHT_ARC	[•]	[他]	[想,吃, 苹果]	$A_1 \cup$ {< •, :top, 想-02>} = A_2
5	SHIFT	[•, 他, 想-02]	[ ]	[吃, 苹果]	A_2
6	RIGHT_ARC	[•, 他, 想-02]	[ ]	[吃, 苹果]	$A_2 \cup$ {<想-02, :arg1, 吃-01>} = A_3
7	REDUCE	[•, 他]	[ ]	[吃, 苹果]	A_3
8	LEFT_ARC	[•, 他]	[ ]	[吃, 苹果]	$A_3 \cup$ {<吃-01, :arg0, 他>} = A_4
10	REDUCE	[•]	[ ]	[吃, 苹果]	A_4
11	SHIFT	[•, 吃-01]	[ ]	[苹果]	A_4
12	RIGHT_ARC	[•, 吃-01]	[ ]	[苹果]	$A_4 \cup$ {< 吃-01, :arg1, 苹果>} = A_5
13	REDUCE	[•]	[ ]	[苹果]	A_5
14	SHIFT	[•, 苹果]	[ ]	[ ]	A_5
15	REDUCE	[•]	[ ]	[ ]	A_5

[1] [•] indicates the Root node in the primary stack.

Table 6 Parsing process of sentence 他想吃苹果“ (He wants to eat apples.)” in arc-eager

No.	Action	σ	β	A
1	-	[•]	[他, 想, 吃, 苹果]	{}
2	Shift	[•, 他]	[想, 吃, 苹果]	{}
3	Shift	[•, 他, 想-02]	[吃, 苹果]	{}
4	LArc	[•, 想-02]	[吃, 苹果]	{< 想-02, :arg0, 他>} = A_1
5	RArc	[•, 想-02]	[吃, 苹果]	$A_1 \cup$ {< •, :top, 想-02>} = A_2
6	Shift	[•, 想-02, 吃-01]	[苹果]	A_2
7	RArc	[•, 想-02, 吃-01]	[苹果]	$A_2 \cup$ {<想-02, :arg1, 吃-01>} = A_3
8	Shift	[•, 想-02, 吃-01, 苹果]	[ ]	A_3
9	RArc	[•, 想-02, 吃-01, 苹果]	[ ]	$A_3 \cup$ {<吃-01, :arg1, 苹果>} = A_4
10	Reduce	[•, 想-02, 吃-01]	[ ]	A_4
11	Reduce	[•, 想-02]	[ ]	A_4
12	Reduce	[•]	[ ]	A_4

[1] [•] indicates the Root node in the primary stack.

e_i^x is word embedding; e_i^p is POS tag embedding; e_i^{NER} is NER tag embedding; e_i^d is named-entity label embedding; and e_i^n is numerical features.

Through multiple hidden layers, the representation is encoded as:

$$H_t = \sigma(W_t e_i + b_t) \tag{3}$$

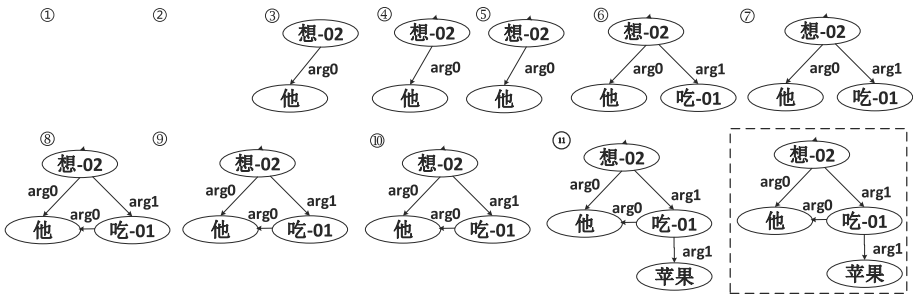


Fig. 4 AMR graph generation procedure of sentence "他想吃苹果 (He wants to eat apples.)"

Table 7 Features in transition action classifier

Features	Templates
Number	The distance from σ_0 and $a(\beta_0)$ to the root
	The distance from σ_0 and $a(\beta_0)$ to the most left child node
	The number of child nodes of σ_0 and $a(\beta_0)$
	The depth of parent node of σ_0 and $a(\beta_0)$
Word	$w(\sigma_0)$, $w(\sigma_0^{'})$, $w(\beta_0)$, $w(p(\sigma_0))$, $w(p(a(\beta_0)))$, $w(c(\sigma_0))$, $w(c(a(\beta_0)))$, $w(c(p(\sigma_0)))$, $w(c(p(a(\beta_0))))$
POS tag	$pos(\sigma_1)$, $pos(\sigma_0)$, $pos(\beta_0)$, $pos(\beta_1)$
Named entity	$NE(\sigma_1)$, $NE(\sigma_0)$, $NE(\beta_0)$, $NE(\beta_1)$
Dependency	$\forall \in \{0,1\}$, $dep(\sigma_i, \beta_0)$ *and* $dep(\beta_0, \sigma_i)$
	$\forall \in \{1,2,3,4\}$, $dep(\beta_i, \beta_0)$ *and* $dep(\beta_0, \beta_i)$
	$\forall \in \{1,2,3,4\}$, $dep(\beta_i, \sigma_0)$ *and* $dep(\sigma_0, \beta_i)$
	$\forall \in \{1,2\}$, $dep(\sigma_i, \sigma_0)$ *and* $dep(\sigma_0, \sigma_i)$

[1] $p(a)$ means the most left parent node of a and $c(a)$ is the most left child node of a. $w(a)$ is the word embedding of a. $pos(a)$ indicates the POS tag of the word corresponding to a. $NE(a)$ is the named entity label of the word corresponding to a. $dep(a, b)$ means the dependency label between a and b

Table 8 Features in label classifier

Features	Templates
Number	The distance from σ_0 and $a(\beta_0)$ to the root
	The distance from σ_0 and $a(\beta_0)$ to the most left child node
	The number of child nodes of σ_0 and $a(\beta_0)$
	The depth of parent node of σ_0 and $a(\beta_0)$
Word	$w(\sigma_0)$, $w(\beta_0)$, $w(p(\sigma_0))$, $w(p(a(\beta_0)))$, $w(c(\sigma_0))$, $w(c(a(\beta_0)))$, $w(c(p(\sigma_0)))$, $w(c(p(a(\beta_0))))$
POS tag	$pos(\sigma_0)$, $pos(\beta_0)$
Named entity	$NE(\sigma_0)$, $NE(\beta_0)$
Dependency	$dep(\sigma_0, \beta_0)$ *and* $dep(\beta_0, \sigma_0)$

W_t is the parameter between layers, b_t is the bias item and σ is activation function.

The output of hidden layers is the input of the output layer:

$$O_t = \sigma(W_o H_t + b_o) \tag{4}$$

W_o is the parameter between layers, and b_o is the bias item.

Finally, softmax function is used to get the probability distribution of the predicted target:

$$y_t = softmax(W_s O_t + b_s) \tag{5}$$

We apply Adagrad algorithm to minimize the object function:

Table 9 Features in transition action classifier of the baseline strategy

Features	Templates
Number	The distance from σ_0 and σ_1 to the root and fromσ_0 and σ_1 to the most left child node
	The number of child nodes of σ_0 and σ_1
	The depth of parent node of σ_0 and σ_1
Word	$w(\sigma_0)$, $w(\sigma_1)$, $w(p(\sigma_0))$, $w(p(\sigma_1))$, $w(c(\sigma_0))$, $w(c(\sigma_1))$, $w(c(p(\sigma_0)))$, $w(c(p(\sigma_1)))$, $w(\beta_0)$, $w(\beta_1)$
POS tag	$pos(\sigma_1)$, $pos(\sigma_0)$, $pos(\beta_0)$, $pos(\beta_1)$
Named entity	$NE(\sigma_1)$, $NE(\sigma_0)$, $NE(\beta_0)$, $NE(\beta_1)$
Dependency	$dep(\sigma_0, \sigma_1)$, $dep(\sigma_1, \sigma_0)$
	$\forall \in \{0,1\}$, $dep(\sigma_i, \beta_0) and dep(\beta_0, \sigma_i)$
	$\forall \in \{1,2,3\}$, $dep(\beta_i, \beta_0) and dep(\beta_0, \beta_i)$
	$\forall \in \{1,2,3\}$, $dep(\beta_i, \sigma_0) and dep(\sigma_0, \beta_i)$

Table 10 Features in label classifier of the baseline strategy

Features	Templates
Number	The distance from σ_0 and σ_1 to the root
	The distance from σ_0 and σ_1 to the most left child node
	The number of child nodes of σ_0 and σ_1
	The depth of parent node of σ_0 and σ_1
Word	$w(\sigma_0)$, $w(\sigma_1)$, $w(p(\sigma_0))$, $w(p(\sigma_1))$, $w(c(\sigma_0))$, $w(c(\sigma_1))$, $w(c(p(\sigma_0)))$, $w(c(p(\sigma_1)))$
POS tag	$pos(\sigma_1)$, $pos(\sigma_0)$
Named entity	$NE(\sigma_1)$, $NE(\sigma_0)$
Dependency	$dep(\sigma_0, \sigma_1)$, $dep(\sigma_1, \sigma_0)$

Table 11 Features in reentrancy classifier of the baseline strategy

Features	Templates
Word	$w(\sigma_0)$, $w(s(\sigma_0))$, $w(ps(\sigma_0))$
POS tag	$pos(\sigma_0)$, $pos(s(\sigma_0))$, $pos(ps(\sigma_0))$
Dependency	$dep(\sigma_0, s(\sigma_0))$ *and* $dep(s(\sigma_0), \sigma_0)$
	$dep(\sigma_0, ps(\sigma_0))$ *and* $dep(ps(\sigma_0), \sigma_0)$
	$dep(s(\sigma_0), ps(\sigma_0))$ *and* $dep(ps(\sigma_0), s(\sigma_0))$

[1] $ps(a)$ is the parent node of a and sibling of a. $s(a)$ is the sibling of a

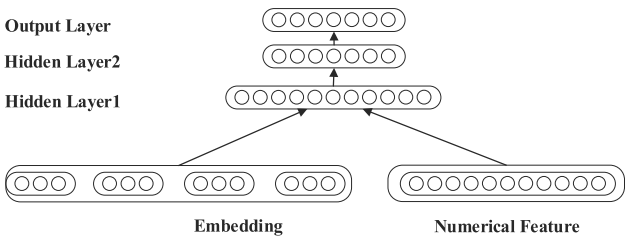


Fig. 5 Multilayer Feedforward Neural Network

$$L(\theta) = -\sum_{i=1}^{n} y_i log(\hat{y_i}) + (1 - y_i) log(1 - \hat{y_t}) + \lambda \|\theta\|_2^2 \tag{6}$$

θ is the parameters in the model, y_i is the gold output at time i, $\hat{y_t}$ is the real output, and $\lambda \|\theta\|_2^2$ is the L2 regularization item.

The algorithmic procedure of the proposed general automatic semantic parsing model based on two-stack-based transition algorithm is illustrated in Algorithm 2.

Algorithm 2 Algorithmic procedure of the proposed model

Input: Sentence $x = \{x_0, x_1, ..., x_n\}$
Output: AMR graph $G = \{V, A\}$

$C = [\sigma, \sigma', \beta, A]$
$\beta = x, \sigma = \emptyset, \sigma' = \emptyset, A = \emptyset, V = \emptyset$
//Encoder
for x_i *in* x **do**
  $G_{x_i} = \overleftarrow{\pi}'(x_i)$
  $\beta_0 = root(G_{x_i})$
  $e_i = [e_i^x; e_i^p; e_i^{NER}; e_i^d; e^n]$
  //Decoder
  while $\beta \neq \emptyset$ **do**
    $action_i = Action(e_i)$
    if $action_i == LEFT\_ARC$ *or* $action_i == RIGHT\_ARC$ **then**
      $e_j = [e_j^x; e_j^p; e_j^{NER}; e_j^d; e^n]$
      $label_i = Label(e_j)$
      $A.append((\sigma_0, label_i, \beta_0))$
      $V.append(\sigma_0), V.append(\beta_0)$
    else if $action_i == MEM$ **then**
      $\sigma'.append(\sigma_0)$
      $\sigma.pop()$
    else if $action_i == REDUCE$ *and* $last\_action \neq MEM$ **then**
      $\sigma.pop()$
    else
      $\sigma.extend(\sigma')$
      $\sigma' = \emptyset$
      $\sigma.push(\beta_0)$
      $\beta.pop()$
      $G.extend(G_{x_i})$
    end if
    $last\_action = action_i$
  end while
end for

The procedure of the proposed general automatic semantic parsing model can be divided into two modules: encoder and decoder. Given a sentence, the model initializes a new configuration C. Then, the encoder module encodes the features from the configuration. The embeddings are transferred to the decoder module for graph decoding. The AMR graph is constructed step by step based on the five transition actions and labels which are predicted by the action classifier and label classifier.

4 Experimental evaluation

4.1 Datasets and metrics

We evaluate our framework on both English dataset (LDC2014T12-proxy (14-p), LDC2014T12-all (14-all)) and Chinese dataset [59] with regard to the metrics of Smatch Precision (P), Recall (R) and F-measure (F1) [60]. The dataset contains discussion forums collected for the DARPA BOLT program, Wall Street Journal and translated Xinhua news texts, various newswire data from NIST OpenMT evaluations and weblog data used in the DARPA GALE program. Size of each dataset is shown in Table 12.

Metrics for evaluation are as follows:

$$Precision\ (\mathrm{P}) = \frac{\#\ correct\ predicted\ triples}{\#\ all\ predicted\ triples} \tag{7}$$

Table 12 Size of datasets

Datasets	Train	Dev	Test
Chinese dataset	7918	989	991
LDC2014T12	10,312	1,368	1371
LDC2014T12-proxy	6603	826	823

$$Recall\ (\mathrm{R}) = \frac{\#\ correct\ predicted\ triples}{\#\ all\ triples} \tag{8}$$

$$\mathrm{F1} = \frac{2 * P * R}{P + R} \tag{9}$$

4.2 Oracle evaluation

The baseline strategy utilizing arc-eager is called **AMR-eager** in this paper. We reproduce the model in [33] as our baseline and modify the model for the Chinese dataset. The proposed ***t**ransition-based **AMR** transformation using two-stack-based transition algorithm* is defined as **T-AMR**. TAMR applies the same dependency feature as those in **AMR-eager**. On the basis of TAMR, we add more appropriate features into the model for both English and Chinese datasets, especially dependency information. We call it ***G**eneral **A**utomatic **T**ransition-based **AMR** model* (**GAT-AMR**).

As mentioned in Sect. 3, we apply the oracle algorithm to obtain gold actions for training. The performance of oracle algorithm indicates the performance of transition system. We compare our oracle algorithm with **AMR-eager** according to Smatch score of AMR graphs constructed utilizing gold actions on training dataset. As shown in Table 13, our algorithm performs the best with Smatch F1 score being 0.90 on the English datasets and 0.81 on the Chinese dataset. Experiment results show that our extended Shift/Reduce algorithm is better than arc-eager algorithm in AMR graph generation. It can cover more graphs than arc-eager and produce more suitable transition action sequences for training.

Table 13 Oracle evaluation of models

Datasets		P	R	F1
Chinese dataset	AMR-eager	0.82	0.73	0.77
	GAT-AMR	0.84	0.77	0.81
LDC2014T12	AMR-eager	0.86	0.78	0.81
	GAT-AMR	0.94	0.86	0.90

4.3 Parameter setting

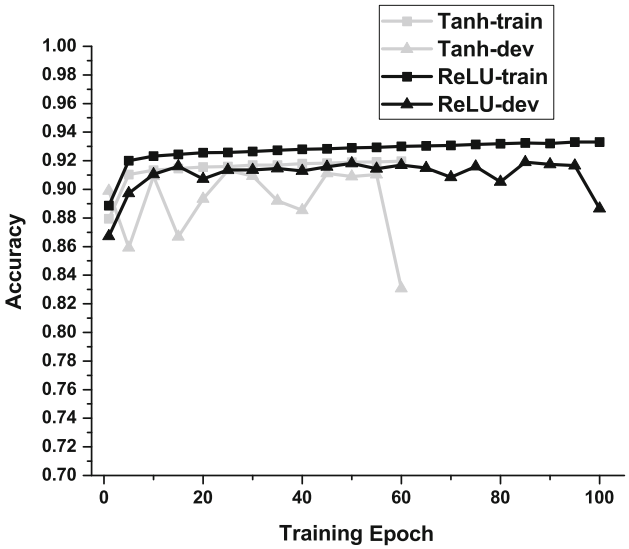
Early stopping strategy is applied in training. We calculate dev accuracy after each epoch and stop training when the accuracy does not improve in 30 rounds. The embeddings for Chinese words are pre-trained on CTB5.0 which has 507,222 words. The embeddings for English words and POS tags were pre-trained on a large unannotated dataset consisting of the first one billion characters from Wikipedia.8.

We conducted many experimental evaluations on both English and Chinese datasets and found that the data which have been trained from Chinese one can present the performance clearly and intuitively. Therefore, in this subsection we took the Chinese dataset as a representative to illustrate the training performance with different activation functions and batch sizes. We compare the influence of activation function and batch size in training of **GAT-AMR**. Figure 6 illustrates the accuracy of transition action classifier and label classifier with activation function: Tanh and ReLU on the Chinese dataset. The batch size is 32, and learning rate is 0.1.

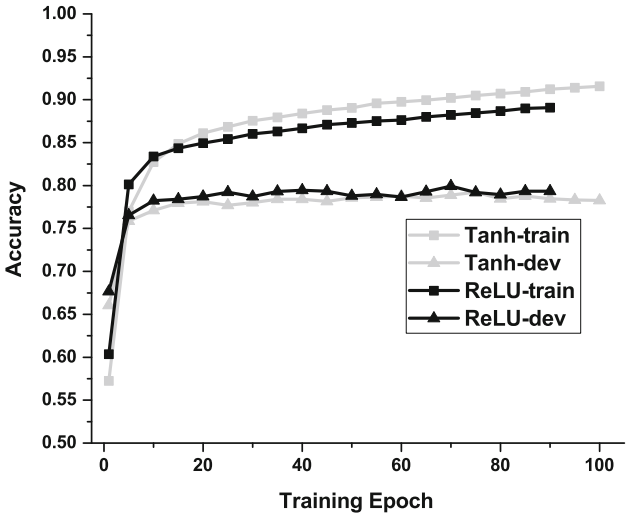
As shown in Fig. 6, the ReLU activation function performs better than Tanh. The label classifier of Tanh activation function is a little over-fitting. So, ReLU is more stable and suitable for AMR parser training. Because of the early stopping strategy, the classifier with Tanh activation function stops training at Epoch 60 when the classifier's performance on the dev dataset got worse in the next ten epochs. It can prevent the model from being over-fitting.

Figure 7 illustrates the dev accuracy of different learning rates in the label classifier on the Chinese dataset when learning rate is 0.1.

As we can see from Fig. 7, the dev accuracy increases rapidly and reaches a higher level with the increase in batch size. It indicates that the AMR parser training needs a larger batch size. The classifier stops training at Epoch 50, 60 and 80 because of the early stopping strategy.



(a) Transition Action Classifier



(b) Label Classifier

Fig. 6 Accuracy of different activation functions in two classifiers on the Chinese dataset

4.4 Parser evaluation

We compare the Smatch score on the English and Chinese datasets. The evaluation results are shown in Table 14. Bold represents the maximum value per row or column.

Compared with the baseline strategy, **GAT-AMR** outperforms **AMR-eager** without applying an extra classifier for reentrancy nodes by 1% on the Chinese dataset and 1% on LDC2014T12-p. Considering **AMR-eager** and **T-AMR**, **T-AMR** applies the two-stack-based transition algorithm, which performs better precision score than that of arc-eager in non-projectivity and reentrancy situations. The **GAT-AMR** model is a general model for both English and Chinese. The dependency feature provides more information which is needed in training.

The Chinese dataset used in this paper is randomly divided, and the dataset used by Wang et al. does not [36]. The Smatch F1 score is 0.58 in [30]. Since the code of Wang et al. is not released, it is not considered as a comparison object in this paper and we only compare transition-based models. As mentioned in Sect. 2.2, Guo and Lu achieve Smatch F1 score of 0.683 on LDC2014T12 dataset with optimized aligner and compact AMR graph in concept recognition [37]. Utilizing the same aligner and without compact AMR graph, their model's Smatch F1 score is only 0.639. Compared with the model, our model can

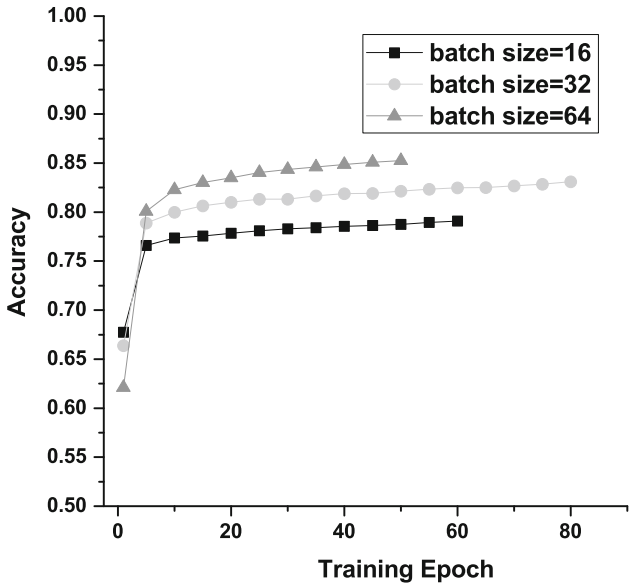


Fig. 7 Dev accuracy of different learning rates in label classifier on the Chinese dataset

achieve competitive performance as the state-of-the-art transition-based model in arc generation with less transition actions. Our model is also more generative than the model as we can construct both English and Chinese graphs.

We also evaluate **AMR-eager** and **GAT-AMR** on individual evaluation metrics, including: Unlabeled (evaluation on unlabeled arcs), No WSD (evaluation on concepts without PropBank suffix), Named Ent. (evaluation on named entities), Negations (evaluation on polarity arcs), Reentrancies (evaluation on reentrancy arcs), Concepts (evaluation on concepts), and SRL (evaluation on arcs with arg label). The results on the English dataset and Chinese dataset are shown in Tables 15 and 16.

As shown in Tables 15 and 16, **GAT-AMR** outperforms **AMR-eager** on all individual evaluation metrics in the Chinese dataset, especially on Unlabeled. It also outperforms **AMR-eager** on the English dataset except Concepts and Named Ent. These two metrics are related to concept recognition. Overall, experiments indicate that the transition algorithm in **GAT-AMR** is better than that in **AMR-eager** on arc prediction. It can perform well on non-projective graphs. The Reentrancy scores of both datasets also improve compared to **AMR-eager**. It shows that **GAT-AMR** can deal with reentrancy well without extra classifier.

Table 15 Individual evaluation on the Chinese dataset

Metrics	AMR-eager			GTAMR		
	P	R	F1	P	R	F1
Unlabeled	0.66	0.61	0.63	0.63	0.69	**0.66**
No WSD	0.58	0.54	0.56	0.56	0.62	**0.59**
Named Ent.	0.72	0.57	0.63	0.65	0.76	**0.70**
Negations	0.58	0.61	0.59	0.66	0.63	**0.64**
Concepts	0.79	0.71	0.75	0.73	0.82	**0.78**
Reentrancies	0.32	0.33	0.33	0.31	0.38	**0.34**
SRL	0.58	0.45	0.51	0.47	0.62	**0.53**

Table 16 Individual evaluation on the English dataset (LDC2014T12)

Metrics	AMR-eager			GTAMR		
	P	R	F1	P	R	F1
Unlabeled	0.71	0.7	0.71	0.71	0.73	**0.72**
No WSD	0.66	0.66	0.66	0.66	0.7	**0.68**
Named Ent.	0.74	0.95	**0.83**	0.75	0.90	0.82
Negations	0.39	0.56	0.46	0.40	0.58	**0.47**
Concepts	0.84	0.88	**0.86**	0.84	0.87	0.85
Reentrancies	0.47	0.34	0.4	0.34	0.55	**0.42**
SRL	0.56	0.54	0.55	0.56	0.68	**0.61**

4.5 Ablation study

We also conduct ablation study of the proposed **GAT-AMR** model. The proposed model without additional features is called **GAT-AMR–feature**, and the proposed model without two-stacked-based transition algorithm and additional features is called **GAT-AMR–trans**. We evaluate the three models on all individual evaluation metrics. The evaluation results on Chinese dataset are illustrated in Fig. 8.

We can see that the **GAT-AMR–feature** performs better than **GAT-AMR–trans** on the unlabeled evaluation

Table 14 Smatch evaluation of models on the English and Chinese datasets

Models	LDC2014T12-p			LDC2014T12			Chinese		
	P	R	F1	P	R	F1	P	R	F1
AMR-eager	0.66	0.65	0.66	0.64	0.59	0.62	**0.61**	0.55	0.58
T-AMR	0.67	0.65	0.65	0.65	0.59	0.62	0.59	0.57	0.58
GAT-AMR	**0.69**	**0.66**	**0.67**	**0.66**	**0.60**	**0.62**	0.56	**0.62**	**0.59**

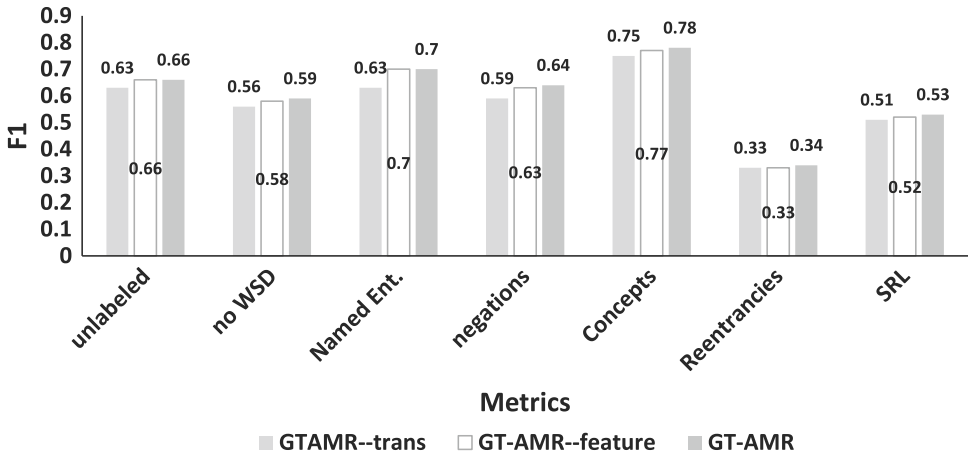


Fig. 8 Ablation study of proposed models

metric, which indicates that it can construct more complete AMR graphs. With additional features, the **GAT-AMR** model outperforms on all the evaluation metrics except unlabeled and Named Ent.

4.6 Error analysis

Figure 9a shows the correct AMR graph of sentence "地理学帮了我很大的忙(Geography helps me a lot.)," and Fig. 9b shows the AMR graph generated by our proposed model. The concept "帮忙-01(help)" cannot be recognized in our proposed model. We identify concepts in a sentence from left to right, but the words mapping to concept "帮忙-01(help)" are separated in the sentence. So, the proposed concept recognition algorithm cannot recognize the correct concept.

Figure 9c and d shows another example of the correct AMR graph and generated AMR graph for sentence "Hope this helps.". Since the word "i" is omitted in the sentence, the proposed model cannot complement the omitted part.

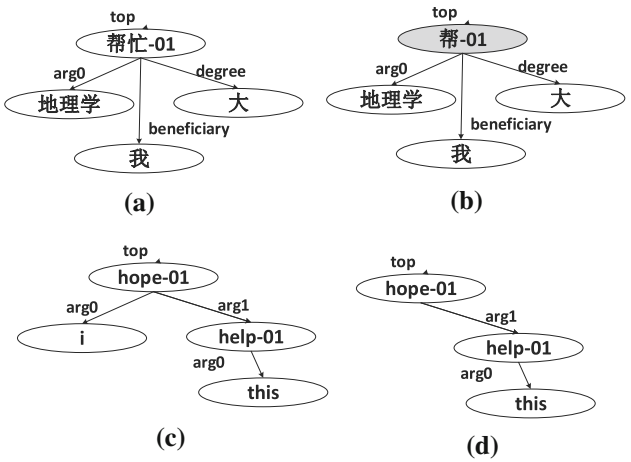


Fig. 9 Examples of correct AMR graphs and incorrectly predicted AMR graphs

These special linguistic phenomena, such as omission and separable words, are challenges for AMR parsing models.

5 Conclusion

Semantic representation in the real world needs a domain-independent general representation method to mine semantic information in multi-domain natural language sentences. We present a general automatic semantic representation model based on two-stack-based transition algorithm for multilingual data, which can represent semantic information in texts with graphs. The proposed two-stack-based transition algorithm can handle reentrancy and crossing arcs in AMR graphs better, and the feature templates in the model can enrich feature representation while training. Experiments demonstrate that our strategy is superior to the baseline strategy. In the future, we will try to apply sequence labeling methods to improve concept recognition and optimize the AMR parser for some special linguistic phenomena, such as omission and separable words.

Acknowledgements This work was supported by the Ministry of Education of Humanities and Social Science project under Grant 16YJC790123 and National Natural Science Foundation of China under Grant 61772278.

Compliance with ethical standards

Conflicts of interest The authors declare that they have no conflict of interest.

References

1. Zong Y, Xu G, Dolog P, Zhang Y, Liu R (2010) Co-clustering for Weblogs in Semantic Space. In: the 11th international conference of web information systems engineering, WISE 2010, Hong Kong, China, December 12–14. pp 120–127

2. Xu G, Yu JX, Lee W (2013) Social networks and social web mining. World Wide Web Internet Web Inf Syst 16(5–6):541–544
3. Li C, Goldwasser D (2019) Encoding social information with graph convolutional networks for political perspective detection in news media. In: the 57th conference of the association for computational linguistics, ACL 2019, Florence, Italy, July 28–August 2, pp 2594–2604
4. Islam J, Mercer RE, Xiao L (2019) Multi-Channel convolutional neural network for twitter emotion and sentiment recognition. In: the 2019 conference of the north American chapter of the association for computational linguistics: human language technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2–7, pp 1355–1365
5. Gupta I, Joshi N (2020) Enhanced twitter sentiment analysis using hybrid approach and by accounting local contextual semantic. J Intell Syst 29(1):1611–1625
6. Rudnik C, Ehrhart T, Ferret O, Teyssou D, Troncy R,Tannier X (2019) Searching news articles using an event knowledge graph leveraged by Wikidata. In: Companion of the 2019 world wide web conference, WWW 2019, San Francisco, CA, USA, May 13–17, pp 1232–1239
7. Wang H, Xu T, Liu Q, Lian D, Chen E, Du D, Wu H, Su W (2019) Mcne: an end-to-end framework for learning multiple conditional network representations of social network. In: the 25th ACM SIGKDD international conference on knowledge discovery & data mining, KDD 2019, Anchorage, AK, USA, August 4–8, pp 1064–1072
8. Yang C, Zhang J, Wang H, Li S, Kim M, Walker M, Han J (2020) Relation learning on social networks with multi-modal graph edge variational autoencoders. In: The thirteenth ACM international conference on web search and data mining, WSDM 2020, Houston, TX, USA, February 3–7, pp 2328–2337
9. Wang Y, Sun H, Zhao Y, Zhou W, Zhu S (2019) A heterogeneous graph embedding framework for location-based social network analysis in smart cities. IEEE Trans Ind Inf 16(4):2747–2755
10. Zhang J, Tan L, Tao X, Wang D, Ying JJ, Wang X (2019) Learning relational fractals for deep knowledge graph embedding in online social networks. In: 20th international conference on web information systems engineering, WISE 2019, Hong Kong, China, November 26–30, pp 660–674
11. Xie T, France-Lanord A, Wang Y, Shao-Horn Y, Grossman JC (2019) Graph dynamical networks for unsupervised learning of atomic scale dynamics in materials. Nature Commun 10(1):2667
12. Jin W, Barzilay R, Jaakkola TS (2018) Junction tree variational autoencoder for molecular graph generation. In: the 35th International conference on machine learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10–15, pp 1–13
13. Jin W, Yang KK, Barzilay R, Jaakkola TS (2019) Learning Multimodal Graph-to-Graph Translation for Molecule Optimization. In: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6–9, pp 2328–2337
14. Wang S, Li Z, Zhang S, Jiang M, Wang X, Wei Z (2020) Molecular property prediction based on a multichannel substructure graph. IEEE Access 8:18601–18614
15. Samanta B, Abir DE, Jana G, Chattaraj PK, Ganguly N, Rodriguez MG (2018) A deep generative model for molecular graphs. In: The Thirty-Third AAAI conference on artificial intelligence, AAAI 2019, Honolulu, Hawaii, USA, January 27–February 1, pp 2328–2337
16. Zettlemoyer LS, Collins M (2005) Learning to map sentences to logical form: structured classification with probabilistic categorial grammars. In: the 21st conference in uncertainty in artificial intelligence, UAI 2005, Edinburgh, Scotland, July 26–29, pp 658–666
17. Wong YW, Mooney RJ (2006) Learning for semantic parsing with statistical machine translation. In: Human language technology conference of the North American chapter of the association of computational linguistics, New York, USA, June 4–9, pp 439–446
18. Banarescu L, Bonial C, Cai S, Georgescu M, Griffitt K, Hermjakob U, Knight K, Koehn P, Palmer M, Schneider N (2013) Abstract meaning representation for sembanking. In: the 7th linguistic annotation workshop and interoperability with discourse, LAW-ID@ACL 2013, Sofia, Bulgaria, August 8–9, pp 178–186
19. Liu F, Flanigan J, Thomson S, Sadeh NM, Smith NA (2015) Toward abstractive summarization using semantic representations. In: the 2015 conference of the North American chapter of the association for computational linguistics: human language technologies, , NAACL-HLT 2015, Denver, Colorado, USA, May 31–June 5, pp 1077–1086
20. Garg S, Galstyan A, Hermjakob U, Marcu D (2016) Extracting biomolecular interactions using semantic parsing of biomedical text. In: The Thirtieth AAAI conference on artificial intelligence, AAAI 2016, Phoenix, Arizona, USA, February 12–17, pp 2718–2726
21. Sachan M, Xing EP (2016) Machine comprehension using rich semantic representations. In: the 54th annual meeting of the association for computational linguistics, ACL 2016, Berlin, Germany, August 7–12, pp 486–492
22. Huang L, Ji H, Cho K, Dagan I, Riedel S, Voss CR (2018) Zero-shot transfer learning for event extraction. In: the 56th annual meeting of the association for computational linguistics, ACL 2018, Melbourne, Australia, July 15–20, pp 2160–2170
23. Song L, Gildea D, Zhang Y, Wang Z, Su J (2019) Semantic neural machine translation using AMR. Trans Assoc Comput Linguistics 7:19–31
24. Takase S, Suzuki J, Okazaki N, Hirao T, Nagata M (2016) Neural headline generation on abstract meaning representation. In: the 2016 conference on empirical methods in natural language processing, EMNLP 2016, Austin, Texas, USA, November 1–4. pp 1054–1059
25. Chen WT (2015) Learning to map dependency parses to abstract meaning representations. In: the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing of the Asian federation of natural language processing, ACL 2015, Beijing, China, July 26–31. pp 41–46
26. Wang C, Pradhan S, Pan XM, Ji H, Xue NW (2016) CAMR at SemEval-2016 task 8: An extended transition-based AMR parser. In: the 10th international workshop on semantic evaluation, SemEval@NAACL-HLT 2016, San Diego, CA, USA, June 16–17, pp 1173–1178
27. Titov I, Lyu C (2018) AMR parsing as graph prediction with latent alignment. In: the 56th annual meeting of the association for computational linguistics, ACL 2018, Melbourne, Australia, July 15–20. pp 397–407
28. Wang C, Xue NW, Pradhan S (2015) Boosting transition-based AMR parsing with refined actions and auxiliary analyzers. In: the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing of the Asian federation of natural language processing, ACL 2015, Beijing, China, July 26–31, pp 857–862
29. Goodman J, Vlachos A, Naradowsky J (2016) Noise reduction and targeted exploration in imitation learning for abstract meaning representation parsing. In: the 54th annual meeting of the association for computational linguistics, ACL 2016, Berlin, Germany, August 7–12, pp 1–11
30. Wang C, Xue NW, Pradhan S (2015) A transition-based algorithm for AMR parsing. In: Human language technology

conference of the north American chapter of the association of computational linguistics, NAACL-HLT 2015, Denver, Colorado, USA, May 31–June 5, pp 366–375
31. Werling K, Angeli G, Manning C (2015) Robust subgraph generation improves abstract meaning representation parsing. In: the 53rh annual meeting of the association for computational linguistics, ACL 2015, Beijing, China, July 26–31, pp 982–991
32. Wang C, Xue NW (2017) Getting the most out of AMR parsing. In: the 2017 conference on empirical methods in natural language processing, EMNLP 2017, Copenhagen, Denmark, September 9–11, pp 1257–1268
33. Damonte M, Cohen SB, Satta G (2017) An incremental parser for abstract meaning representation. In: the 15th conference of the European chapter of the association for computational linguistics, EACL 2017, Valencia, Spain, April 3–7, pp 536–546
34. Buys J, Blunsom P (2017) Robust incremental neural semantic graph parsing. In: the 55th annual meeting of the association for computational linguistics, ACL 2017, Vancouver, Canada, July 30–August 4, pp 1215–1226
35. Ballesteros M, Onaizan Y (2017) AMR parsing using stack-LSTMs. In: the 2017 conference on empirical methods in natural language processing, EMNLP 2017, Copenhagen, Denmark, September 9–11, pp 1269–1275
36. Wang C, Li B, Xue NW (2018) Transition-based chinese AMR parsing. In: Human language technology conference of the north American chapter of the association of computational linguistics, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1–6, pp 247–252
37. Guo Z, Lu W (2018) Better transition-based AMR parsing with refined search space. In: the 2018 conference on empirical methods in natural language processing, EMNLP 2018, Brussels, Belgium, October 31–November 4, pp 1712–1722
38. Zhang H, Li J, Ji Y, Yue H (2017) Understanding subtitles by character-level sequence-to-sequence learning. IEEE Trans Ind Inf 13(2):616–624
39. Qu WG, Zhou JS, Wu XD, Dai R, Gu M, Gu Y (2017) Survey on abstract meaning representation. J Data Acquis Process 32(1):26–36
40. Li B, Wen Y, Qu WG, Bu LJ, Xue NW (2016) Annotating the little prince with Chinese AMRs. In: the 10th linguistic annotation workshop and interoperability with discourse, LAW 2016, Berlin, Germany, August 11, pp 7–15
41. Kingsbury P, Palmer M (2002) From Treebank to Propbank. In: The third international conference on language resources and evaluation, LREC 2002, Las Palmas, Canary Islands, Spain, May 29–31, pp 1989–1993
42. Flanigan J, Thomson S, Carbonell JG, Dyer C, Smith NA (2014) A discriminative graph-based parser for the abstract meaning representation. In: the 52nd annual meeting of the association for computational linguistics, ACL 2014, Baltimore, MD, USA, June 22–27, pp 1426–1436
43. Zhou JS, Xu FY, Uszkoreit H, Qu WG, Li R, Gu YH (2016) AMR parsing with an incremental joint model. In: the 2016 conference on empirical methods in natural language processing, EMNLP 2016, Austin, Texas, USA, November 1–4, pp 680–689
44. Foland W, Martin JH (2016) CU-NLP at SemEval-2016 Task 8: AMR parsing using LSTM based recurrent neural networks. In: the 10th international workshop on semantic evaluation, SemEval@NAACL-HLT 2016, San Diego, CA, USA, June 16–17, pp 1197–1201
45. Foland W, Martin JH (2016) Abstract meaning representation parsing using LSTM recurrent neural networks. In: the 55th annual meeting of the association for computational linguistics, ACL 2017, Vancouver, Canada, July 30–August 4, pp 463–472
46. Puzikov Y, Kawahara D, Kurohashi S (2016) M2L at SemEval-2016 Task 8: AMR parsing with neural networks. In: International workshop on semantic evaluations, SemEval@NAACL-HLT 2016, San Diego, CA, USA, June 16–17, pp 1154–1159
47. Artzi Y, Lee K, Zettlemoyer L (2015) Broad-coverage CCG semantic parsing with AMR. In: Empirical methods in natural language processing, EMNLP 2015, Lisbon, Portugal, September 17–21, pp 1699–1710
48. Kwiatkowski T, Zettlemoyer L, Goldwater S, Steedman M (2011) Lexical generalization in CCG grammar induction for semantic parsing. In: Empirical methods in natural language processing, EMNLP 2011, Edinburgh, UK, July 27–31, pp 127–129
49. Misra DK, Artzi Y (2016) Neural shift-reduce CCG semantic parsing. In: the 2016 conference on empirical methods in natural language processing, EMNLP 2016, Austin, Texas, USA, November 1–4, pp 1775–1778
50. Pust M, Hermjakob U, Knight K et al (2015) Using syntax-based machine translation to parse english into abstract meaning representation. CoRR abs/1504.06665
51. Zhang S, Ma X, Duh K, Van Durme B (2019) AMR parsing as sequence-to-graph transduction. In: the 57th conference of the association for computational linguistics, ACL 2019, Florence, Italy, July 28–August 2, pp 80–94
52. Gildea D, Xue NW, Peng XC, Wang C (2017) Addressing the data sparsity issue in neural AMR parsing. In: the 15th conference of the European chapter of the association for computational linguistics, EACL 2017, Valencia, Spain, April 3–7, pp 366–375
53. Barzdins G, Gosko D (2016) RIGA at SemEval-2016 Task 8: Impact of smatch extensions and character-level neural translation on AMR parsing accuracy. In: International workshop on semantic evaluations, SemEval@NAACL-HLT 2016, San Diego, CA, USA, June 16–17, pp 1143–1147
54. Konstas I, Iyer S, Yatskar M, et al (2017) Neural AMR: sequence-to-sequence models for parsing and generation. In: the 55th annual meeting of the association for computational linguistics, ACL 2017, Vancouver, Canada, July 30–August 4, pp 146–157
55. Nivre J (2004) Incrementality in Deterministic dependency parsing. In: the ACL workshop incremental parsing: bringing engineering and cognition together, ACL 2004, Barcelona, Spain, July 21–26, pp 50–57
56. Nivre J (2008) Algorithms for deterministic incremental dependency parsing. Comput Linguist 34(5):513–553
57. Sagae K, Tsujii J (2008) Shift-reduce dependency DAG parsing. In: the 22nd international conference on computational linguistics, COLING 2008, Manchester, UK, August 18–22, pp 753–760
58. Zhang X, Du Y, Sun W et al (2016) Transition-based parsing for deep dependency structures. Comput Linguist 42(3):353–389
59. Li B, Wen Y, Song L, Bu LJ, Qu WG, Xue NW (2017) Construction of Chinese abstract meaning representation corpus with concept-to-word alignment. J Chin Inf Process 31(6):93–102
60. Cai S, Knight K (2013) Smatch: an evaluation metric for semantic feature structures. In: the 51st annual meeting of the association for computational linguistics, ACL 2013, Sofia, Bulgaria, August 4–9, pp 748–752