Cognitive Computation (2024) 16:2846–2862
https://doi.org/10.1007/s12559-024-10339-4

Shift-Reduce Task-Oriented Semantic Parsing with Stack-Transformers

Daniel Fernández-González[1]

Received: 28 June 2023 / Accepted: 30 July 2024 / Published online: 22 August 2024

Abstract
Intelligent voice assistants, such as Apple Siri and Amazon Alexa, are widely used nowadays. These task-oriented dialogue systems require a semantic parsing module in order to process user utterances and understand the action to be performed. This semantic parsing component was initially implemented by rule-based or statistical slot-filling approaches for processing simple queries; however, the appearance of more complex utterances demanded the application of shift-reduce parsers or sequence-to-sequence models. Although shift-reduce approaches were initially considered the most promising option, the emergence of sequence-to-sequence neural systems has propelled them to the forefront as the highest-performing method for this particular task. In this article, we advance the research on shift-reduce semantic parsing for task-oriented dialogue. We implement novel shift-reduce parsers that rely on Stack-Transformers. This framework allows to adequately model transition systems on the transformer neural architecture, notably boosting shift-reduce parsing performance. Furthermore, our approach goes beyond the conventional top-down algorithm: we incorporate alternative bottom-up and in-order transition systems derived from constituency parsing into the realm of task-oriented parsing. We extensively test our approach on multiple domains from the Facebook TOP benchmark, improving over existing shift-reduce parsers and state-of-the-art sequence-to-sequence models in both high-resource and low-resource settings. We also empirically prove that the in-order algorithm substantially outperforms the commonly used top-down strategy. Through the creation of innovative transition systems and harnessing the capabilities of a robust neural architecture, our study showcases the superiority of shift-reduce parsers over leading sequence-to-sequence methods on the main benchmark.

Keywords Natural language understanding · Computational linguistics · Semantic parsing · Task-oriented dialogue · Neural network · Voice assistants

Introduction

The research community and industry have directed significant attention towards the advancement of intelligent personal assistants such as Apple Siri, Amazon Alexa, and Google Assistant. These systems, known as *task-oriented dialogue* systems, streamline task completion and information retrieval via natural language interactions within defined domains such as media playback, weather inquiries, or restaurant reservations. The increasing adoption of these voice assistants by users has not only transformed individuals' lives but also impacted real-world businesses.

Humans effortlessly understand language, deriving meaning from sentences and extracting relevant information. *Semantic parsing* attempts to emulate this process by understanding the meaning of natural language expressions and translating them into a structured representation that can be interpreted by computational systems. Therefore, a crucial component of any voice assistant is a *semantic parser* in charge of natural language understanding. Its purpose is to process user dialogue by converting each input utterance into an unequivocal task representation understandable and executable by a machine. Specifically, these parsers identify the user's requested task intent (e.g., play music) as well as pertinent entities needed to further refine the task (e.g., which playlist?).

Traditional commercial voice assistants conventionally handle user utterances by conducting *intent* detection and *slot* extraction tasks separately. For example, given the utterance *Play Paradise by Coldplay*, the semantic parsing module processes it in two stages: *a)* initially determining the user's intent as `IN:PLAY_MUSIC`, and then *b)* recognizing task-specific named entities *Paradise* and

✉ Daniel Fernández-González
danifg@uvigo.gal

[1] Departamento de Informática, Universidade de Vigo, Campus As Lagoas s/n, 32004 Ourense, Spain

Coldplay, respectively tagging these elements (*slots*) as `SL:MUSIC_TRACK_TITLE` and `SL:MUSIC_ARTIST_NAME`. Intent detection has traditionally been approached as text classification, where the entire utterance serves as input, while slot recognition has been formulated as a sequence tagging challenge [1–3].

Annotations generated by these traditional semantic parsers only support a single intent per utterance and a list of non-overlapping slots exclusively composed by tokens from the input. While this flat semantic representation suffices for handling straightforward utterances, it falls short in adequately representing user queries that involve *compositional* requests. For instance, the query *How long will it take to drive from my apartment to San Diego?* necessitates first identifying *my apartment* (`IN:GET_LOCATION_HOME`) before estimating the duration to the destination (`IN:GET_ESTIMATED_DURATION`). Hence, there is a requirement for a semantic representation capable of managing multiple intents per utterance, where slots encapsulate nested intents.

In order to represent more complex utterances, Gupta et al. [4] introduced the *task-oriented parsing* (TOP) formalism: a hierarchical annotation scheme expressive enough to describe the task-specific semantics of nested intents and model compositional queries. In Fig. 1, we illustrate how the intents and slots of utterances mentioned in the previous examples can be represented using the TOP annotation.

An advantage of the TOP representation is its ease of annotation and parsing compared to more intricate semantic formalisms such as *logical forms* [5] or *abstract meaning representations* (AMR) [6]. In fact, its similarity to a *syntactic constituency* tree enables the adaptation of algorithms from the *constituency parsing* literature to process task-oriented requests. This was the driving force behind [4] initial proposal to modify the *shift-reduce* constituency parser introduced by [7] for generating TOP annotations.

Alternatively, Gupta et al. [4] also proposed the application of different sequence-to-sequence models [8–10] for parsing compositional queries. Sequence-to-sequence models comprise a specific neural architecture tasked with predicting a sequence of output tokens based on an input sequence of items. After conducting empirical comparisons between the shift-reduce technique and various sequence-to-sequence models for parsing compositional queries, they determined that the shift-reduce parser surpassed other methods and was the only approach capable of guaranteeing that the output representation adhered to a well-formed TOP tree. This superiority can be largely attributed to the fact that, unlike sequence-to-sequence models, shift-reduce algorithms adhere to grammar constraints throughout the parsing process and exhibit an inductive bias towards tree structures, resulting in enhanced performance.

Although sequence-to-sequence approaches may produce flawed representations, recent advancements [11, 12] have substantially enhanced their performance by leveraging *Transformer* neural networks [10] in conjunction with pre-trained language models such as RoBERTa [13] or BART [14]. Consequently, they have emerged as the most accurate approach to date for generating TOP tree structures.

This article presents further advancements in the realm of shift-reduce semantic parsing for natural language understanding. Specifically, we enhance the initial framework introduced by [4], which relied on the *top-down* transition

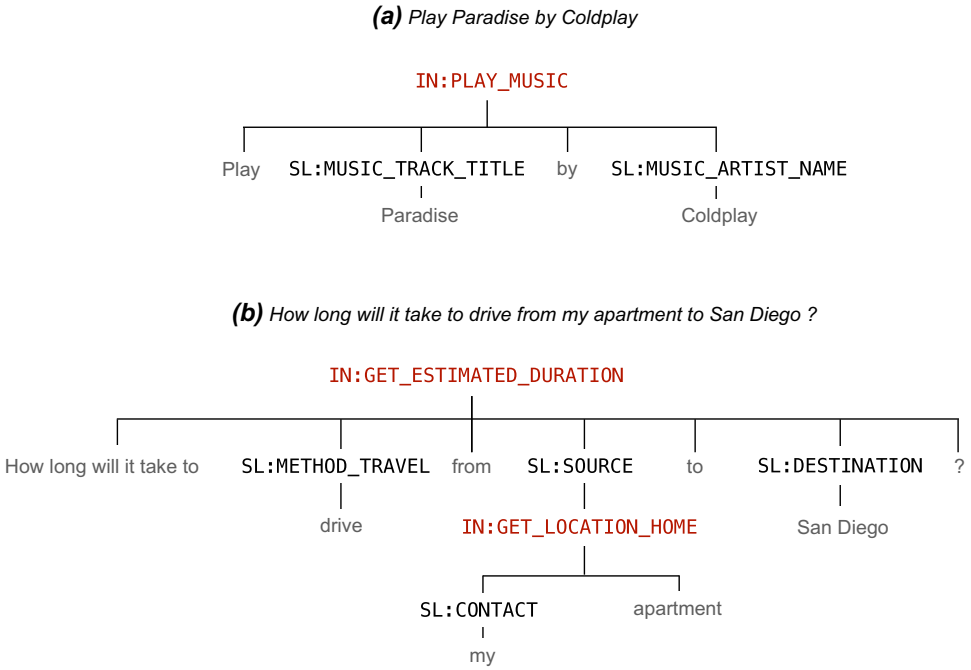

Fig. 1 Flat and compositional TOP annotations of utterances from music and navigation domains, respectively. Note that intents and slots are respectively prefixed with `IN:` and `SL:`

system [7] and a *Stack-LSTM*-based neural architecture [15]. Firstly, we implement a more robust neural model based on *Stack-Transformers* [16], enabling the accurate modeling of shift-reduce systems within a Transformer-based neural architecture. Secondly, we adapt the *bottom-up* and *in-order* transition systems [17, 18] from the constituency parsing literature to task-oriented semantic parsing. Lastly, we empirically evaluate these alternatives, along with the top-down algorithm, on our neural architecture. Our findings demonstrate that the in-order transition system achieves the highest accuracy on the Facebook TOP benchmark [4, 19], even outperforming the most robust sequence-to-sequence baselines.

In summary, our contributions in this article are as follows:

- We develop innovative shift-reduce semantic parsers for task-oriented dialogues utilizing Stack-Transformers and *deep contextualized word embeddings* derived from RoBERTa.
- We adapt various transition systems from the constituency parsing literature to handle TOP annotations and conduct a comprehensive comparison against the original top-down approach, demonstrating the superiority of the in-order algorithm across all scenarios.
- We evaluate our approach on both low-resource and high-resource settings of the Facebook TOP datasets, pushing the boundaries of the state of the art in task-oriented parsing and narrowing the divide with sequence-to-sequence models.
- Upon acceptance, we will make our system's source code freely available for public use.

The remainder of this article is organized as follows: In "Related Work," we provide an overview of prior research on semantic parsing for task-oriented compositional queries. "Methodology" outlines our proposed approach, beginning with an exposition of the transition-based algorithms adapted from constituency parsing, followed by a detailed description of the Stack-Transformer-based neural model. "Experiments" presents the experiments conducted with the three transition systems using the proposed neural architecture as a testing platform, along with a comprehensive analysis of their performance. Finally, concluding remarks are presented in "Conclusions."

Related Work

The hierarchical semantic representation introduced by [4] to address compositional queries spurred the adaptation of parsing algorithms initially developed for constituency parsing, such as the Stack-LSTM-based shift-reduce parser [7]. Additionally, Gupta et al. [4] proposed sequence-to-sequence models for this task, including those based on *convolutional neural networks* (CNNs) [9], *long short-term memory* (LSTM) neural networks [8], and transformers [10]. Although sequence-to-sequence methods were originally devised for machine translation [20], they were also adapted to constituency parsing by first linearizing the tree structure [21].

Given that the shift-reduce parser initially emerged as the leading method for generating TOP representations, Einolghozati et al. [22] opted to enhance the original system by incorporating an ensemble of seven parsers, contextualized word embeddings extracted from ELMo [23], and a language model ranker. Concurrently, Pasupat et al. [24] modified the span-based constituency parser proposed by [25] to process utterances into TOP trees, achieving promising results without the use of deep contextualized word embeddings.

While sequence-to-sequence models initially lagged behind all available semantic parsing methods, recent advancements have substantially improved their performance in constructing TOP representations. Notably, Rongali et al. [11] devised a sequence-to-sequence architecture bolstered by a Pointer Generator Network [26] and a RoBERTa-based encoder [13]. This neural architecture emerged as the state of the art in task-oriented semantic parsing and has since been adopted and extended by subsequent studies. Among them, Aghajanyan et al. [12] and Chen et al. [19] proposed simplifying the target sequence by eliminating input tokens that are not slot values, while also initializing both the encoder and the decoder with the pre-trained sequence-to-sequence model BART [14]. Furthermore, non-autoregressive variants of the sequence-to-sequence architecture introduced by [11] have been presented as well [27–30]. Additionally, Shrivastava et al. [31] recently enhanced sequence-to-sequence models with a scenario-based approach, where incomplete intent-slot templates are available in advance and can be retrieved after identifying the utterance's scenario. Meanwhile, Wang et al. [32] chose to enhance the efficiency of sequence-to-sequence models by generating subtrees as output tokens at each decoding step.

Diverging from the current mainstream trends, we push forward the boundaries of research in shift-reduce task-oriented parsing by crafting a novel approach grounded in a more accurate transition system and implemented on a more robust neural architecture. As a result, our system surpasses even the strongest sequence-to-sequence baselines.

Simultaneously with our research, Do et al. [33] have developed a two-staged approach that demonstrates remarkable results. Initially, they enhance standard pre-trained language models through fine-tuning, incorporating additional hierarchical semantic information. Subsequently, the resulting model is integrated with a recursive insertion-based mechanism [34], constrained by grammar information. Specifically, grammar rules extracted from the training dataset are employed to prune unpromising predictions dur-

ing the parsing process [35]. It is worth noting that these contributions are orthogonal to our approach and could certainly enhance its performance.

Methodology

This section outlines our proposed approach. Specifically, we elaborate on the transition-based algorithms adapted from the constituency parsing literature to handle task-oriented utterances, as well as the neural architecture serving as the foundation of our system.

Transition Systems for Task-Oriented Semantic Parsing

In task-oriented semantic parsing, the objective is to transform an input utterance comprising n words, denoted as $X = w_1, \ldots, w_n$, into a semantic representation—in our case, a TOP tree Y. Similar to syntactic constituency representations, Y is a rooted tree consisting of tokens $w_1, \ldots w_n$ as its leaves and a collection of internal nodes (referred to as *constituents*) hierarchically structured above them. These constituents are denoted as tuples (N, W), where W represents the set of tokens covered by its span, and N denotes the *non-terminal* label. For example, (`SL:SOURCE`, {*my*, *apartment*}) and (`SL:DESTINATION`, {*San*, *Diego*}) are constituents extracted from the TOP tree depicted in Fig. 1b. Additionally, in our specific scenario, two distinct types of constituents emerge: *intents* and *slots*, with non-terminal labels respectively prefixed with `IN:` and `SL:`. Finally, tree structures must adhere to certain constraints to be deemed a valid TOP representation:

- The root constituent, which encompasses the entire utterance, must be an intent node.
- Only tokens and/or slot constituents can serve as child nodes of an intent node.
- A slot node may have either words (one or several) or a single intent constituent as child nodes.

To process the input utterance, we employ shift-reduce parsers, initially introduced for dependency and constituency parsing [36, 37]. These parsers construct the target tree incrementally by executing a sequence of actions that analyze the input utterance from left to right. Specifically, shift-reduce parsers are characterized by a non-deterministic *transition system*, which defines the necessary data structures and the set of operations required to complete the parsing process; and an *oracle*, which selects one of these actions deterministically at each stage of the parsing process. Formally, a transition system is represented as a quadruple $S = (C, c_0, C_f, T)$, where:

- C denotes the set of possible *state configurations*, defining the data structures necessary for the parser.
- c_0 represents the *initial configuration* of the parsing process.
- C_f is the set of *final configurations* reached at the end of the parsing process.
- T signifies the set of available *transitions* (or *actions*) that can be applied to transition the parser from one state configuration to another.

Moreover, during training, a rule-based oracle o, given the gold parse tree Y_g, selects action a_t for each state configuration c_t at each time step t: $a_t = o(c_t, Y_g)$. Once the model is trained, it approximates the oracle during decoding.

We can utilize a transition system S along with an oracle o to parse the utterance X: commencing from the initial configuration c_0, a sequence of transitions $a_0, \ldots, a_{m-1}$ (determined by the oracle at each time step t) guides the system through a series of state configurations $c_0, \ldots, c_m$ until a final configuration is reached ($c_m \in C_f$). At this stage, the utterance will have been fully processed, and the parser will generate a valid TOP tree Y. Various transition systems exist in the literature on constituency parsing. In addition to the algorithm employed by [4], we have adapted two other transition systems for task-oriented semantic parsing, which we elaborate on in the subsequent sections.

Top-Down Transition System Initially conceived by [7] for constructing constituency trees in a top-to-bottom fashion, this transition system was later adapted by [4] to accommodate TOP tree representations. The top-down transition system comprises the following components:

- State configurations within C are structured as $c = \langle \Sigma, B \rangle$, where Σ denotes a *stack* (responsible for storing non-terminal symbols, constituents, and partially processed tokens), and B represents a *buffer* (containing unprocessed tokens to be read from the input).
- At the initial configuration c_0, the buffer B encompasses all tokens from the input utterance, while the stack Σ remains empty.
- Final configurations within C_f are structured as $c = \langle [I], \emptyset \rangle$, where the buffer is empty (indicating that all words have been processed), and the stack contains a single item I. This item represents an intent constituent spanning the entire utterance, as the root node of a valid TOP tree must be an intent.
- The set of available transitions T consists of three actions:
  - The NON- TERMINAL- L transition involves pushing a non-terminal node labeled L onto the stack, transitioning the system from state configurations of the form $\langle \Sigma, B \rangle$ to $\langle \Sigma | L, B \rangle$ (where $\Sigma | L$ denotes a stack with item L placed on top and Σ as the tail). Unlike

in constituency parsing, this transition can generate intent and slot non-terminals (with labels L prefixed with `IN:` or `SL:`, respectively). Therefore, it must adhere to specific constraints to produce a well-formed TOP tree:

  - Since the root node must be an intent constituent, the first NON- TERMINAL- L transition must introduce an intent non-terminal onto the stack.
  - A NON- TERMINAL- L transition that inserts an intent non-terminal onto the stack is permissible only if the last pushed non-terminal was a slot, performed in the preceding state configuration. This condition ensures that the resulting intent constituent from this transition becomes the sole child node of that preceding slot, as required by the TOP formalism.
  - A NON- TERMINAL- L transition adding a slot non-terminal to the stack is allowed only if the last inserted non-terminal was an intent.

- A SHIFT action is employed to retrieve tokens from the input by transferring words from the buffer to the stack. This operation transitions the parser from state configurations $\langle \Sigma, w_i|B\rangle$ to $\langle \Sigma|w_i, B\rangle$ (where $w_i|B$ denotes a buffer with token w_i on top and B as the tail, and conversely, $\Sigma|w_i$ represents a stack with Σ as the tail and w_i as the top). This transition is permissible only if the buffer is not empty. Specifically for task-oriented semantic parsing, this action will not be available in state configurations where the last non-terminal added to the stack was a slot, and an intent constituent was already created as its first child node. This constraint ensures that slot constituents have only one intent as their child node.
- Additionally, a REDUCE transition is necessary to construct a new constituent by removing all items (including tokens and constituents) from the stack until a non-terminal symbol is encountered, then grouping them as child nodes of that non-terminal. This results in a new constituent placed on top of the stack, transitioning the parser from configurations $\langle \Sigma|L|e_k|\ldots|e_0, B\rangle$ to $\langle \Sigma|L_{e_k\ldots e_0}, B\rangle$ (where $L_{e_k\ldots e_0}$ denotes a constituent with non-terminal label L and child nodes $e_k \ldots e_0$). This transition can be executed only if there is at least one non-terminal symbol and one item (token or constituent) in the stack.

Please note that the original work by [4] did not provide specific transition constraints tailored to generating valid TOP representations. Therefore, we undertook a complete redesign of the original top-down algorithm [7] for task-oriented semantic parsing to incorporate these task-specific transition constraints.

Finally, Table 1 illustrates how the top-down algorithm parses the utterance depicted in Fig. 1a. It demonstrates the step-by-step construction of each constituent, which involves defining the non-terminal label, reading and/or creating all corresponding child nodes, and then reducing all items within its span.

Bottom-Up Transition System In contrast to the top-down approach, shift-reduce algorithms traditionally perform constituency parsing by building trees from bottom to top. Therefore, we have also adapted the bottom-up transition system developed by [17] for task-oriented semantic parsing. Unlike classic bottom-up constituency parsing algorithms [37, 38], this transition system does not require prior binarization of the gold tree during training or subsequent recovery

Table 1 Top-down transition sequence and state configurations (represented by the stack and the buffer) for producing the TOP tree in Fig. 1a

Transition	Stack	Buffer
	[ ]	[Play, Paradise, by, Coldplay]
NT- `IN:PLAY_MUSIC`	[`IN:PLAY_MUSIC`]	[Play, Paradise, by, Coldplay]
SHIFT	[`IN:PLAY_MUSIC`, Play]	[Paradise, by, Coldplay]
NT- `SL:TITLE`	[`IN:PLAY_MUSIC`, Play, `SL:TITLE`]	[Paradise, by, Coldplay]
SHIFT	[`IN:PLAY_MUSIC`, Play, `SL:TITLE`, Paradise]	[by, Coldplay]
REDUCE	[`IN:PLAY_MUSIC`, Play, $\texttt{SL:TITLE}_{Paradise}$]	[by, Coldplay]
SHIFT	[`IN:PLAY_MUSIC`, Play, $\texttt{SL:TITLE}_{Paradise}$, by]	[Coldplay]
NT- `SL:ARTIST`	[`IN:PLAY_MUSIC`, Play, $\texttt{SL:TITLE}_{Paradise}$, by, `SL:ARTIST`]	[Coldplay]
SHIFT	[`IN:PLAY_MUSIC`, Play, $\texttt{SL:TITLE}_{Paradise}$, by, `SL:ARTIST`, Coldplay]	[ ]
REDUCE	[`IN:PLAY_MUSIC`, Play, $\texttt{SL:TITLE}_{Paradise}$, by, $\texttt{SL:ARTIST}_{Coldplay}$]	[ ]
REDUCE	[$\texttt{IN:PLAY\_MUSIC}_{Play\ \texttt{SL:TITLE}\ by\ \texttt{SL:ARTIST}}$]	[ ]

NT- ι stands for NON- TERMINAL- L and slot labels have been abbreviated from `SL:MUSIC_TRACK_TITLE` and `SL:MUSIC_ARTIST_NAME` to `SL:TITLE` and `SL:ARTIST`, respectively

of the non-binary structure after decoding. Specifically, the non-binary bottom-up transition system comprises the following:

- State configurations have the form $c = \langle \Sigma, B, f \rangle$, where Σ is a *stack*, B is a *buffer*, as described for the top-down algorithm, and f is a boolean variable indicating whether a state configuration is terminal or not.
- In the initial configuration c_0, the buffer contains the entire user utterance, the stack is empty, and f is *false*.
- Final configurations in C_f have the form $c = \langle [I], \emptyset, true \rangle$, where the stack holds a single intent constituent, the buffer is empty, and f is *true*. Following a bottom-up algorithm, we can continue building constituents on top of a single intent node in the stack, even when it spans the whole input utterance. To avoid that, this transition system requires the inclusion of variable f in state configurations to indicate the end of the parsing process.
- Actions provided by this bottom-up algorithm are as follows:
  - Similar to the top-down approach, a SHIFT action moves tokens from the buffer to the stack, transitioning the parser from state configurations $\langle \Sigma, w_i | B, false \rangle$ to $\langle \Sigma | w_i, B, false \rangle$. This operation is not permissible under the following conditions:
    - When the buffer is empty and there are no more words to read.
    - When the top item on the stack is an intent node and, since slots must have only one intent child node, the parser needs to build a slot constituent on top of it before shifting more input tokens.
  - A REDUCE#K- L transition (parameterized with the non-terminal label L and an integer k) is used to create a new constituent by popping k items from the stack and combining them into a new constituent on top of the stack. This transitions the parser from state configurations $\langle \Sigma | e_{k-1} | \ldots | e_0, B, false \rangle$ to $\langle \Sigma | L_{e_{k-1} \ldots e_0}, B, false \rangle$. To ensure a valid TOP representation, this transition can only be applied under the following conditions:
    - When the REDUCE#K- L action creates an intent constituent (i.e., L is prefixed with `IN:`), it is permissible only if there are no intent nodes among the k items popped from the stack (as an intent constituent cannot have other intents as child nodes).
    - When the REDUCE#K- L transition builds a slot node (i.e., L is prefixed with `SL:`), it is allowed only if there are no slot constituents among the k elements affected by this operation (as slots cannot have other slots as child nodes). Additionally, if the item on top of the stack is an intent node, only the REDUCE action with k equal to 1 is permissible (since slots can only contain a single intent constituent as a child node).
  - Lastly, a FINISH action is used to signal the end of the parsing process by changing the value of f, transitioning the system from configurations $\langle \Sigma, B, false \rangle$ to final configurations $\langle \Sigma, B, true \rangle$. This operation is only allowed if the stack contains a single intent constituent and the buffer is empty.

Finally, Table 2 illustrates how this shift-reduce parser processes the utterance in Fig. 1a, constructing each constituent from bottom to top by assigning the non-terminal label after all child nodes are fully assembled in the stack.

In-Order Transition System

Alternatively to the top-down and bottom-up strategies, Liu and Zhang [18] introduced the *in-order* transition system for constituency parsing. We have tailored this algorithm for parsing task-oriented utterances. Specifically, the proposed in-order transition system consists of the following:

Table 2 Transition sequence and state configurations (represented by the stack, buffer, and variable f) for building the TOP semantic representation in Fig. 1a following a non-binary bottom-up approach

Transition	Stack	Buffer	f
	[ ]	[Play, Paradise, by, Coldplay]	*false*
SHIFT	[Play]	[Paradise, by, Coldplay]	*false*
SHIFT	[Play, Paradise]	[by, Coldplay]	*false*
RE#1- SL:TITLE	[Play, $\texttt{SL:TITLE}_{Paradise}$]	[by, Coldplay]	*false*
SHIFT	[Play, $\texttt{SL:TITLE}_{Paradise}$, by]	[Coldplay]	*false*
SHIFT	[Play, $\texttt{SL:TITLE}_{Paradise}$, by, Coldplay]	[ ]	*false*
RE#1- SL:ARTIST	[Play, $\texttt{SL:TITLE}_{Paradise}$, by, $\texttt{SL:ARTIST}_{Coldplay}$]	[ ]	*false*
RE#4- IN:PLAY_MUSIC	[$\texttt{IN:PLAY\_MUSIC}_{Play\ \texttt{SL:TITLE}\ by\ \texttt{SL:ARTIST}}$]	[ ]	*false*
FINISH	[$\texttt{IN:PLAY\_MUSIC}_{Play\ \texttt{SL:TITLE}\ by\ \texttt{SL:ARTIST}}$]	[ ]	*true*

RE#K- L stands for REDUCE#K- L

- Configurations maintain the same format as the bottom-up algorithm (*i.e.,* $c = \langle \Sigma, B, f \rangle$).
- In the initial configuration c_0, the buffer contains the entire user utterance, the stack is empty, and the value of f is *false*.
- Final configurations take the form $c = \langle [I], \emptyset, true \rangle$. Similar to the bottom-up approach, the in-order algorithm may continue creating additional constituents above the intent node left on the stack indefinitely. Hence, a flag is necessary to indicate the completion of the parsing process.
- The available transitions are adopted from both top-down and bottom-up algorithms, but some of them exhibit different behaviors or are applied in a different order:
  - A NON- TERMINAL- L transition involves pushing a non-terminal symbol L onto the stack, transitioning the system from state configurations represented as $\langle \Sigma, B, false \rangle$ to $\langle \Sigma | L, B, false \rangle$. However, unlike the top-down algorithm, this transition can only occur if the initial child node of the upcoming constituent is fully constructed on top of the stack. Furthermore, it must meet other task-specific constraints to generate valid TOP representations:
    - A NON- TERMINAL- L transition that introduces an intent non-terminal to the stack (i.e., L prefixed with `IN:`) is valid only if its first child node atop the stack is not an intent constituent.
    - A NON- TERMINAL- L transition that places a slot non-terminal on the stack (i.e., L prefixed with `SL:`) is permissible only if the fully-created item atop the stack is not a slot node.
  - Similarly to other transition systems, a SHIFT operation is used to retrieve tokens from the buffer. However, unlike those algorithms, this action is restricted if the upcoming constituent has already been labeled as a slot (by a non-terminal previously added to the stack) and its first child node is an intent constituent already present in the stack. This condition aims to prevent slot constituents from having more than one child node when the item at the top of the stack is an intent.
  - A REDUCE transition is employed to generate intent or slot constituents. Specifically, it removes all elements from the stack until a non-terminal symbol is encountered, which is simultaneously replaced by the preceding item to form a new constituent at the top of the stack. Consequently, it guides the parser from state configurations represented as $\langle \Sigma | e_k | L | e_{k-1} | \dots | e_0, B, false \rangle$ to $\langle \Sigma | L_{e_k \dots e_0}, B, false \rangle$. This transition is only applicable if there is a non-terminal in the stack (preceded by its first child constituent according to the in-order algorithm). Additionally, this transition must comply with specific constraints for task-oriented semantic parsing:
    - When the REDUCE operation results in an intent constituent (as determined by the last non-terminal label added to the stack), it is permissible only if there are no intent nodes among the preceding $k-1$ items (since the first child e_k already adheres to the TOP formalism, as verified during the application of the NON-TERMINAL- L transition).
    - When the REDUCE transition produces a slot constituent, it is allowed only if there are no other slot nodes within the preceding $k-1$ elements that will be removed by this operation. This condition also encompasses scenarios where the initial child node e_k of the upcoming slot constituent is an intent and, since the SHIFT transition is not permitted under such circumstances, only the REDUCE action can construct a slot with a single intent.
  - Lastly, akin to the bottom-up approach, a FINISH action is utilized to finalize the parsing process. This action is only permissible if the stack contains a single intent constituent and the buffer is empty.

In Table 3, we illustrate how the in-order strategy parses the user utterance depicted in Fig. 1a. While the top-down and bottom-up approaches can be respectively regarded as a *pre-order* and *post-order* traversal over the tree, this transition system constructs the constituency structure following an *in-order* traversal, addressing the drawbacks of the other two alternatives. The in-order strategy creates each constituent by determining the non-terminal label after its first child is completed in the stack, and then processing the remaining child nodes. Unlike the top-down approach, which assigns the non-terminal label before reading the tokens composing its span, the in-order algorithm can utilize information from the first child node to make a better choice regarding the non-terminal label. On the other hand, the non-binary bottom-up strategy must simultaneously determine the non-terminal symbol and the left span boundary of the future constituent once all child nodes are completed in the stack. Despite having local information about already-built subtrees, the bottom-up strategy lacks global guidance from top-down parsing, which is essential for selecting the correct non-terminal label. Additionally, determining span boundaries can be challenging when the target constituent has a long span, as REDUCE#K- L transitions with a high k value are less frequent in the training data and thus harder to learn. The in-order approach avoids these drawbacks by predicting the non-terminal label and marking the left span boundary after creating its first child. In "Experiments," we will empirically demonstrate that, in practice,

Table 3 In-order transition sequence and state configurations for generating the TOP representation in Fig. 1(a)

Transition	Stack	Buffer	f
	[ ]	[Play, Paradise, by, Coldplay]	*false*
SHIFT	[Play]	[Paradise, by, Coldplay]	*false*
NT- IN:PLAY_MUSIC	[Play, IN:PLAY_MUSIC]	[Paradise, by, Coldplay]	*false*
SHIFT	[Play, IN:PLAY_MUSIC, Paradise]	[by, Coldplay]	*false*
NT- SL:TITLE	[Play, IN:PLAY_MUSIC, Paradise, SL:TITLE]	[by, Coldplay]	*false*
REDUCE	[Play, IN:PLAY_MUSIC, $\text{SL:TITLE}_{Paradise}$]	[by, Coldplay]	*false*
SHIFT	[Play, IN:PLAY_MUSIC, $\text{SL:TITLE}_{Paradise}$, by]	[Coldplay]	*false*
SHIFT	[Play, IN:PLAY_MUSIC, $\text{SL:TITLE}_{Paradise}$, by, Coldplay]	[ ]	*false*
NT- SL:ARTIST	[Play, IN:PLAY_MUSIC, ..., Coldplay, SL:ARTIST]	[ ]	*false*
REDUCE	[Play, IN:PLAY_MUSIC, ..., by, $\text{SL:ARTIST}_{Coldplay}$]	[ ]	*false*
REDUCE	[$\text{IN:PLAY\_MUSIC}_{Play\ \text{SL:TITLE}\ by\ \text{SL:ARTIST}}$]	[ ]	*false*
FINISH	[$\text{IN:PLAY\_MUSIC}_{Play\ \text{SL:TITLE}\ by\ \text{SL:ARTIST}}$]	[ ]	*true*

NT-ι stands for NON-TERMINAL-L and slot labels have been respectively abbreviated from SL:MUSIC_TRACK_TITLE and SL:MUSIC_ARTIST_NAME to SL:TITLE and SL:ARTIST

the advantages of the in-order transition system result in substantial accuracy improvements compared to the other two alternatives.

Neural Parsing Model

Earlier shift-reduce systems in dependency parsing [15], constituency parsing [7], AMR parsing [39], and task-oriented semantic parsing [4, 22] relied on *Stack-LSTMs* for modeling state configurations. These architectures are grounded in LSTM recurrent neural networks [40], which dominated the natural language processing community until [10] introduced *Transformers*. This neural architecture offers a cutting-edge attention mechanism [41] that outperforms LSTM-based systems and, unlike recurrent neural networks, can be easily parallelized. This motivated [16] to design *Stack-Transformers*. In particular, they use Stack-Transformers to replace Stack-LSTMs in shift-reduce dependency and AMR parsing, achieving remarkable gains in accuracy.

In our research, we leverage Stack-Transformers to represent the buffer and stack structures of the described transition systems, employing them to construct innovative shift-reduce task-oriented parsers. Specifically, we implement the following encoder-decoder architecture:

Encoder Top-performing sequence-to-sequence approaches [11, 27] directly use pre-trained models like RoBERTa [13] as the encoder in their neural architectures, conducting a task-specific fine-tuning during training. RoBERTa, short for "Robustly optimized BERT pretraining approach," employs the same transformer architecture as BERT [43] and was pre-trained on masked word prediction using a large dataset.

Unlike strong sequence-to-sequence techniques, we adopt a less resource-consuming and greener strategy: we extract fixed weights from the pre-trained language model $\text{RoBERTa}_{\text{LARGE}}$[1] to initialize word embeddings, which remain frozen throughout the training process. Specifically, we use mean pooling (i.e., averaging the weights from wordpieces) to generate a word representation e_i for each token w_i in the input utterance $X = w_1, \ldots, w_n$, resulting in the sequence $E = e_1, \ldots, e_n$.

Next, we define the *encoder* using a 6-layer transformer with a hidden size of 256. Transformers utilize a *multi-head self-attention layer* with multiple attention heads (four in our case) to assess the relevance of each input token relative to the other words in the utterance. The output of this layer is fed into a feed-forward layer, ultimately producing an encoder hidden state h_i for each input word (represented as e_i). Therefore, given the sequence of word representations E, the encoding process yields the sequence of *encoder hidden states* $H = h_1, \ldots, h_n$. Figure 2 illustrates the transformer neural architecture.

Decoder with Stack-Transformers The decoder is responsible for generating the sequence of target actions $A = a_0, \ldots, a_{m-1}$ to parse the input utterance X according to a specific transition system S.

We use Stack-Transformers (with 6 layers, a hidden size of 256, and 4 attention heads) to effectively model the stack and buffer structures at each state configuration of the shift-reduce parsing process. In the original transformer decoder model, a *cross-attention layer* employs multiple *attention heads* to attend to all input tokens and compute their compatibility with the last *decoder hidden state* q_t (which encodes the transition history). However, Stack-Transformers specialize one attention head to focus exclusively on tokens in the

[1] https://huggingface.co/roberta-large

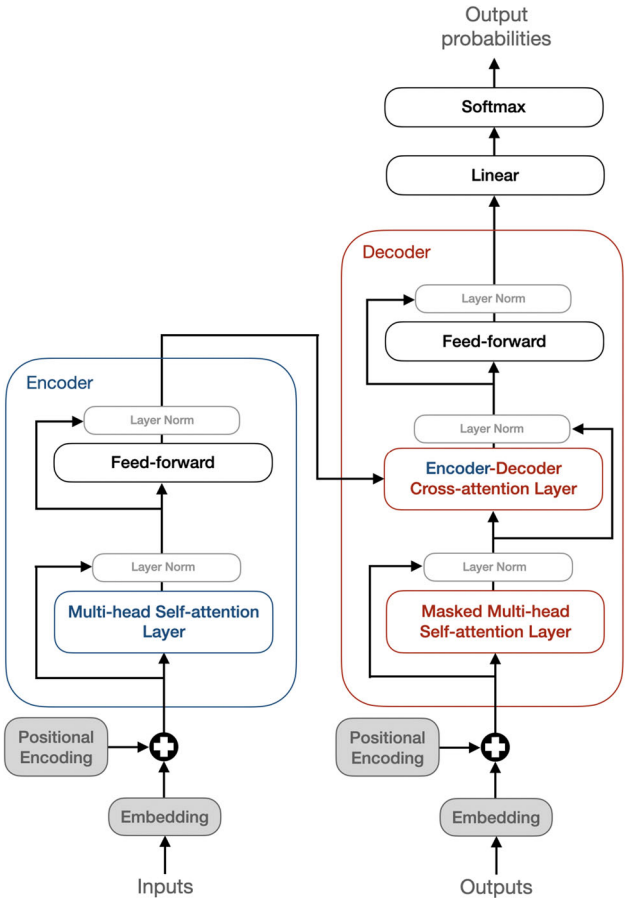

Fig. 2 Transformer neural architecture introduced by [10]. Note that this neural network requires the incorporation of *positional encoding* for each input token to maintain sequential order, and *Layer Norm* refers to the layer normalization technique proposed by [42]

stack at state configuration c_t and another head solely on the remaining words in the buffer at c_t. This specialization allows the transformer to represent stack and buffer structures.

In practice, these dedicated stack and buffer attention heads are implemented using masks m^{stack} and m^{buffer} over the input. After applying the transition a_{t-1} to state configuration c_{t-1}, these masks must be updated at time step t to accurately represent the stack and buffer contents in the current state configuration c_t. To achieve this, we define how these masks are specifically modified for each transition system described in "Methodology":

- If the action a_{t-1} is a SHIFT transition, the first token in m^{buffer} will be masked out and added to m^{stack}. This applies to all proposed transition systems, as the SHIFT transition behaves consistently across them.
- When a NON- TERMINAL- L transition is applied, it affects the stack structure in c_t but has no effect on m^{stack}. This is because attention heads only attend to input tokens, and non-terminals are artificial symbols not present in the user utterance.
- For a REDUCE transition (including the REDUCE#K- L action from the non-binary bottom-up transition system), all tokens in m^{stack} that form the upcoming constituent will be masked out, except for the initial word representing the resultant constituent (since artificial non-terminals cannot be considered by the attention heads).

In Fig. 3, we illustrate how these masks represent the content of the buffer and stack structures and how they are adjusted as the parser transitions from state configurations c_{t-1} to c_t.

After encoding the stack and buffer in state configuration c_t into masks m_t^{stack} and m_t^{buffer} (both represented as vectors with values of $-\infty$ or 0), the attention head z_t^{stack} (focused exclusively on the stack) is computed as follows:

$$z_t^{stack} = \sum_{i=1}^{n} \alpha_{ti}(h_i W_d^V), \alpha_{ti} = \frac{\exp(\beta_{ti})}{\sum_{k=1}^{n} \exp(\beta_{tk})},$$
$$\beta_{ti} = \frac{(q_t W_d^Q)(h_i W_d^K)^T}{\sqrt{d}} + m_{ti}^{stack} \quad (1)$$

where W_d^K, W_d^Q, and W_d^V are parameter matrices unique to each attention head, d is the dimension of the resulting attention vector z_t^{stack}, and β_{ti} is a compatibility function that measures the interaction between the decoder hidden state q_t and each input token w_i (represented by h_i). By introducing the mask m_t^{stack} into the original equation to compute β_{ti}, this scoring function will only affect the words that are in the stack at time step t.

Similarly, the attention vector z_t^{buffer} (which only affects input tokens in the buffer in c_t) is calculated as follows:

$$z_t^{buffer} = \sum_{i=1}^{n} \alpha_{ti}(h_i W_d^V), \alpha_{ti} = \frac{\exp(\beta_{ti})}{\sum_{k=1}^{n} \exp(\beta_{tk})},$$
$$\beta_{ti} = \frac{(q_t W_d^Q)(h_i W_d^K)^T}{\sqrt{d}} + m_{ti}^{buffer} \quad (2)$$

The other two regular attention heads z_t are computed as originally described in [10]. All resulting attention vectors are combined and passed through subsequent linear and SOFTMAX layers (as depicted in Fig. 2) to ultimately select the next action a_t from the permitted transitions in state configuration c_t, according to a specific transition system S.

Finally, note that this neural architecture is flexible enough to implement not only the transition systems described in "Methodology," but also any shift-reduce parser for task-oriented semantic parsing.

Training Objective Each shift-reduce parser is trained through the minimization of the overall log loss (implemented as a cross-entropy loss) when selecting the correct sequence of

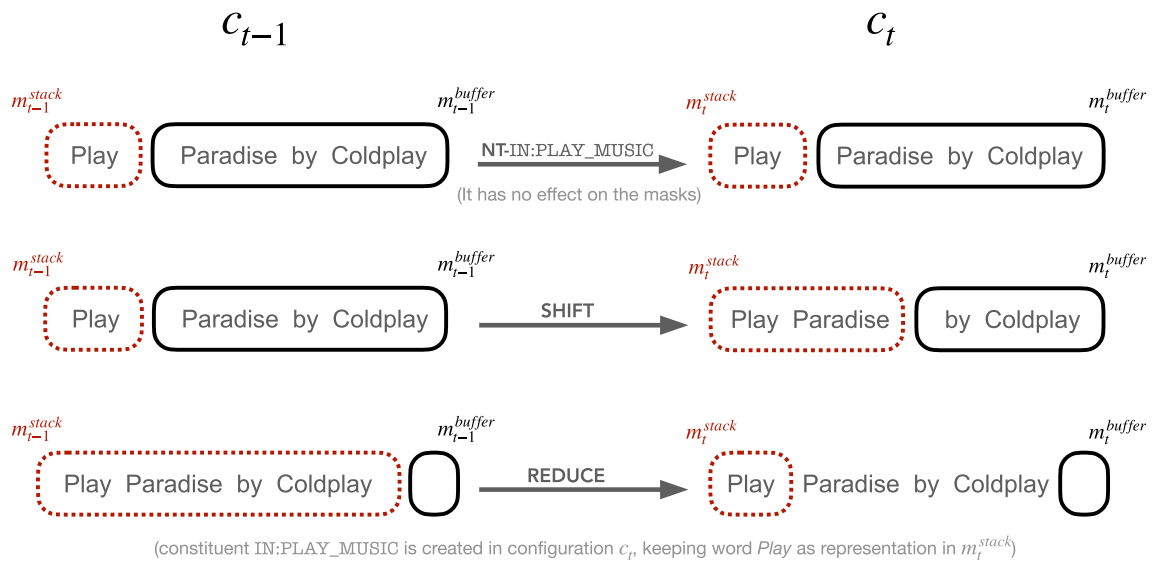

Fig. 3 Updates to the masks m_t^{stack} and m_t^{buffer} that reflect the effects of certain in-order transitions on the stack and buffer during the shift-reduce parsing process illustrated in Table 3

transitions $A = a_0, \ldots, a_{m-1}$ to generate the gold TOP tree Y_g for the user utterance X:

$$\mathcal{L}(\theta) = -\sum_{t=0}^{T} \log P_\theta(a_t | a_{<t}, X) \tag{3}$$

where the transition a_t (predicted in time step t) is conditioned by previous action predictions ($a_{<t}$).

Experiments

Setup

Data We conduct experiments on the main benchmark for task-oriented semantic parsing of compositional queries: the Facebook TOP datasets. The initial version (TOP)[2] was introduced by [4], who annotated utterances with multiple nested intents across two domains: *event* and *navigation*. This was further extended by [19] in the second version (TOPv2),[3] which added six additional domains: *alarm*, *messaging*, *music*, *reminder*, *timer*, and *weather*. While the first version presents user queries with a high degree of compositionality, the extension TOPv2 introduced some domains (such as *music* and *weather*) where all utterances can be parsed with flat trees. Table 4 provides some statistics of the TOP and TOPv2 datasets.

Furthermore, TOPv2 offers specific splits designed to evaluate task-oriented semantic parsers in a *low-resource domain adaptation* scenario. The conventional approach involves utilizing some samples from the *reminder* and *weather* domains as target domains, while considering the remaining six full domains (including *event* and *navigation* from TOP) as source domains if necessary. Moreover, instead of selecting a fixed number of training samples per target domain, TOPv2 adopts a SPIS (samples per intent and slot) strategy. For example, a 25 SPIS strategy entails randomly selecting the necessary number of samples to ensure at least 25 training instances for each intent and slot of the target domain. To facilitate a fair comparison, we evaluate our approach on the training, test, and validation splits at both 25 SPIS and 500 SPIS for the target domains *reminder* and *weather*, as provided in TOPv2. Additionally, following the methodology proposed by [19], we employ a joint training strategy in the 25 SPIS setting, wherein the training data from the source domain is combined with the training split from the target domain.

Finally, we further evaluate our shift-reduce parsers on a variant of the TOPv2 dataset (referred to as TOPv2*). This variant comprises domains with a high percentage of hierarchical structures: *alarm*, *messaging*, and *reminder*. Our aim is to rigorously test the three proposed transition systems on complex compositional queries, excluding those domains that can be fully parsed with flat trees, which are more easily handled by traditional slot-filling methods.

Evaluation We use the official TOP scoring script for performance evaluation, which reports three different metrics:

[2] http://fb.me/semanticparsingdialog

[3] https://fb.me/TOPv2Dataset

Table 4 Data statistics for the Facebook TOP benchmark

Dataset	Domain	Training	Valid	Test	Intents	Slots	%Compos
TOP	Event	9170	1336	2654	11	17	20%
	Navigation	20,998	2971	6075	17	33	43%
TOPv2	Alarm	20,430	2935	7123	8	9	16%
	Messaging	10,018	1536	3048	12	27	16%
	Music	11,563	1573	4184	15	9	0%
	Reminder	17,840	2526	5767	19	32	21%
	Timer	11,524	1616	4252	11	5	4%
	Weather	23,054	2667	5682	7	11	0%

We provide the number of queries in the training, validation, and test splits, along with the number of intents and slots. Additionally, we include the percentage of compositional queries (i.e., utterances parsed by non-flat trees with depth > 2)

- *Exact match* accuracy (EM), which measures the percentage of full trees correctly built.
- *Labeled bracketing* F_1 score (F_1), which compares the non-terminal label and span of each predicted constituent against the gold standard. This is similar to the scoring method provided by the EVALB script[4] for constituency parsing [44], but it also includes pre-terminal nodes in the evaluation.
- *Tree-labeled* F_1 score (TF_1), which evaluates the subtree structure of each predicted constituent against the gold tree.

Recent research often reports only the EM accuracy; however, in line with [4], we also include F_1 and TF_1 scores to provide a more comprehensive comparison of the proposed transition systems. Lastly, for each experiment, we present the average score and standard deviation across three runs with random initialization.

Implementation Details Our neural architecture was built upon the Stack-Transformer framework developed by [16] using the FAIRSEQ toolkit [45]. We maintained consistent hyperparameters across all experiments, based on those specified by [16], with minor adjustments. Specifically, we used the Adam optimizer [46] with $\beta_1 = 0.9$ and $\beta_2 = 0.98$, and a batch size of 3584 tokens. The learning rate was linearly increased for the first 4000 training steps from $1e^{-7}$ to $5e^{-4}$, followed by a decrease using the `inverse-sqrt` scheduling scheme, with a minimum of $1e^{-9}$ [10]. Additionally, we applied a label smoothing rate of 0.01, a dropout rate of 0.3, and trained for 90 epochs. Furthermore, we averaged the weights from the three best checkpoints based on the validation split using greedy decoding and employed a beam size of 10 for evaluation on the test set. All models were trained and tested on a single Nvidia TESLA P40 GPU with 24 GB of memory.

[4] https://nlp.cs.nyu.edu/evalb/

Baselines In addition to evaluating the three proposed transition systems, we compare them against the leading shift-reduce parser for task-oriented dialogue: the system developed by [22]. This model builds upon the system by [4], which uses a top-down transition system and a Stack-LSTM-based architecture, and enhances it with ELMo-based word embeddings, a majority-vote ensemble of seven parsers, and an SVM language model ranker. We also include current top-performing sequence-to-sequence models in our comparison [11, 12, 27, 29, 31]. For low-resource domain adaptation, we compare our models with the enhanced implementation by [19], which is based on [11] and specifically tested on the low-resource TOPv2 splits. Lastly, we incorporate the recent state-of-the-art approach by [33], which employs a language model enhanced with semantic structured information, into both high-resource and low-resource comparisons.

Results

High-Resource Setting We first present the evaluation results of the three described transition systems with Stack-Transformers on the TOP and TOPv2* datasets in Table 5. Regardless of the metric, the in-order algorithm consistently outperforms the other two alternatives on both datasets. Although the TOP dataset contains a higher percentage of compositional queries than TOPv2*, the in-order parser shows a more significant accuracy advantage over the top-down parser on TOP (0.49 EM accuracy points) compared to TOPv2* (0.13 EM accuracy points). The bottom-up approach notably underperforms compared to the other transition systems on both datasets.

In Table 6, we compare our shift-reduce parsers to strong baselines on the TOP dataset. Using frozen RoBERTa-based word embeddings, the in-order shift-reduce parser outperforms all existing methods under similar conditions, including sequence-to-sequence models that fine-tune language models for task-oriented parsing. Specifically, it surpasses the single-model and ensemble variants of the shift-reduce parser by [22] by 3.22 and 0.89 EM accuracy points, respec-

Table 5 Average performance across 3 runs on TOP and TOPv2* test splits

Transition system	TOP			TOPv2*		
	EM	F_1	TF_1	EM	F_1	TF_1
Top-down	86.66±0.06	95.33±0.07	90.80±0.01	87.98±0.09	94.51±0.03	90.99±0.09
Bottom-up	85.89±0.10	94.86±0.06	90.33±0.05	86.27±0.03	93.56±0.06	89.57±0.02
In-order	**87.15**±0.01	**95.57**±0.15	**91.18**±0.13	**88.11**±0.07	**94.60**±0.04	**91.11**±0.07

Standard deviations are reported with ±. Best scores are marked in bold

tively. Additionally, our best transition system achieves improvements of 0.41 and 0.05 EM accuracy points over top-performing sequence-to-sequence baselines initialized with RoBERTa [27] and BART [12], respectively. The exceptions are the enhanced variant of [22] (which uses an ensemble of seven parsers and an SVM language model ranker) and the two-staged system by [33] that employs an augmented language model with hierarchical information, achieving the best accuracy to date on the TOP dataset.

Lastly, our top-down parser with Stack-Transformers achieves accuracy comparable to the strongest sequence-to-sequence models using RoBERTa-based encoders [11, 27], and surpasses the single-model top-down shift-reduce baseline by [22] by a wide margin (2.73 EM accuracy points).

Low-Resource Setting Table 7 presents the performance of our approach on low-resource domain adaptation. Across all SPIS settings, the in-order strategy consistently achieves the highest scores, not only among shift-reduce parsers but also compared to top-performing sequence-to-sequence models. Specifically, the in-order algorithm outperforms the BART-based sequence-to-sequence model by 3.5 and 2.4 EM accuracy points in the 25 SPIS setting of the *reminder* and *weather* domains, respectively. In the 500 SPIS setting, our best shift-reduce parser achieves accuracy gains of 7.9 and 0.5 EM points on the *reminder* and *weather* domains over the strongest sequence-to-sequence baseline. Notably, while the *reminder* domain poses greater challenges due to the presence of compositional queries, our approach exhibits higher performance improvements in this domain compared to the *weather* domain, which exclusively contains flat queries. Additionally, we include the state-of-the-art scores achieved by the system developed by [33] by incorporating semantic structured information into the language model fine-tuning.

Table 6 Comparison of exact match performance among state-of-the-art task-oriented parsers on the TOP test set

Parser	EM
(Sequence-to-sequence models)	
Rongali et al. [11] + $\text{RoBERTa}_{\text{FINE- TUNED}}$	86.67
Aghajanyan et al. [12] + $\text{RoBERTa}_{\text{FINE- TUNED}}$	84.52
Aghajanyan et al. [12] + $\text{BART}_{\text{FINE- TUNED}}$	87.10
Zhu et al. [27] + $\text{RoBERTa}_{\text{FINE- TUNED}}$	86.74
Shrivastava et al. [29] + $\text{RoBERTa}_{\text{FINE- TUNED}}$	85.07
Oh et al. [30] + $\text{BERT}_{\text{FINE- TUNED}}$	86.00
Shrivastava et al. [31] + $\text{RoBERTa}_{\text{FINE- TUNED}}$	86.14
(Shift-reduce models)	
Einolghozati et al. [22] + ELMo	83.93
Top-down shift-reduce parser + RoBERTa	86.66
Bottom-up shift-reduce parser + RoBERTa	85.89
In-order shift-reduce parser + RoBERTa	**87.15**
Einolghozati et al. [22] + ELMo + ensemble	86.26
Einolghozati et al. [22] + ELMo + ensemble + SVM-Rank	87.25
Do et al. [33] + $\text{RoBERTa}^{+\ \text{HIERARCHICAL INFORMATION}}_{\text{FINE- TUNED}}$	**88.18**

The first block encompasses sequence-to-sequence models and shift-reduce parsers. In the second block, we additionally present the results of [22] with ensembling (+ *ensemble*) and language model re-ranking (+ *SVM-Rank*), along with a novel approach that fine-tunes a standard RoBERTa language model by integrating additional semantic structured information (+ HIERARCHICAL INFORMATION). Bold scores denote the best EM accuracy of each block. Lastly, we indicate with FINE- TUNED those approaches that utilize pretrained language models directly as encoders and undergo fine-tuning for adaptation to task-oriented parsing

Table 7 Comparison of exact match performance among top-performing task-oriented parsers on the test splits of *reminder* and *weather* domains within a low-resource setting

Parser	Reminder		Weather	
	25 SPIS	500 SPIS	25 SPIS	500 SPIS
(Sequence-to-sequence models)				
Chen et al. [19] + $\text{RoBERTa}_{\text{FINE- TUNED}}$	–	71.9	–	83.5
Chen et al. [19] + $\text{BART}_{\text{FINE- TUNED}}$	57.1	71.9	71.0	84.9
(Shift-reduce models)				
Top-down S-R parser + RoBERTa	57.39±0.27	**79.79**±0.19	71.22±1.03	83.19±0.20
Bottom-up S-R parser + RoBERTa	40.45±0.88	68.65±0.39	68.58±0.99	74.13±0.35
In-order S-R parser + RoBERTa	**60.56**±0.12	**79.79**±0.27	**73.36**±0.08	**85.44**±0.21
Do et al. [33]+$\text{RoBERTa}^{\text{+HIERAR. INFORM.}}_{\text{FINE- TUNED}}$	**72.12**	**82.28**	**77.96**	**88.08**

The first block compiles sequence-to-sequence models and shift-reduce (S-R) parsers. In the second block, we additionally present the results of the novel approach by [33], which involves fine-tuning a standard RoBERTa language model by integrating additional semantic structured information (+ HIERAR. INFORM.). Bold scores denote the best EM performance of each block on each dataset. Lastly, we indicate with FINE- TUNED those approaches that utilize pre-trained language models directly as encoders and undergo fine-tuning for adaptation to task-oriented parsing

Discussion Overall, our top-down and in-order shift-reduce parsers deliver competitive accuracies on the main Facebook TOP benchmark, surpassing the state of the art in both high-resource and low-resource settings in most cases. Furthermore, shift-reduce parsers ensure that the resulting structure is a well-formed tree in any setting, whereas sequence-to-sequence models may produce invalid trees due to the absence of grammar constraints during parsing. For instance, Rongali et al. [11] reported that 2% of generated trees for the TOP test split were not well-formed. Although [19] did not document this information, we anticipate a significant increase in invalid trees in the low-resource setting. Finally, it is worth mentioning that techniques such as ensembling, re-ranking, or fine-tuning pre-trained language models are orthogonal to our approach and, while they may consume more resources, they can be directly implemented to further enhance performance.

Analysis

To comprehend the variations in performance among the proposed transition systems, we conduct an error analysis focusing on utterance length and structural factors using the validation split of the TOP dataset.

Utterance Length In Fig. 4 a and b, we present the EM and labeled bracketing F_1 scores achieved by each transition system across various utterance length cutoffs. It can be seen that the bottom-up algorithm yields higher EM accuracy for the shortest utterances (≤ 5), but experiences a notable decline in accuracy for longer queries. While less pronounced than in the bottom-up strategy, both the in-order and top-down algorithms also exhibit a clear decrease in accuracy as utterance length increases. This outcome is anticipated as shift-reduce parsers are prone to *error propagation*: earlier mistakes in the transition sequence can lead the parser into suboptimal state configurations, resulting in further erroneous decisions later on. Notably, the in-order approach consistently outperforms the top-down baseline across all length cutoffs.

Query Compositionality Figure 4 c and d depict the EM and labeled F_1 scores achieved by each algorithm on queries with varying numbers of intents per utterance. We observe that the in-order transition system attains higher EM accuracy on utterances with fewer than 3 intents. However, its performance on more complex queries is surpassed by the top-down approach. A similar trend is evident when evaluating performance using the labeled F_1 score: the top-down strategy outperforms the in-order algorithm on queries with 4 intents. While this might suggest that a purely top-down approach is preferable for processing utterances with a compositionality exceeding 3 intents, it is essential to note that the number of queries with 4 intents in the validation split is relatively low (just 20 utterances with 4 intents, compared to 2525, 1179, and 307 utterances with 1, 2, and 3 intents, respectively). Consequently, its impact on the overall performance is limited. Finally, both plots also indicate that the bottom-up approach consistently underperforms the other two alternatives, except on queries with 3 intents, where it surpasses the in-order strategy in EM accuracy.

Span Length and Non-terminal Prediction Figure 4e illustrates the performance achieved by each transition system on span identification relative to different lengths, while Fig. 4f demonstrates the accuracy obtained by each algorithm on labeling constituents with the most frequent non-terminals (including the average span length in brackets). In Fig. 4e, we observe that error propagation affects span identification, as accuracy decreases on longer spans, which require a longer

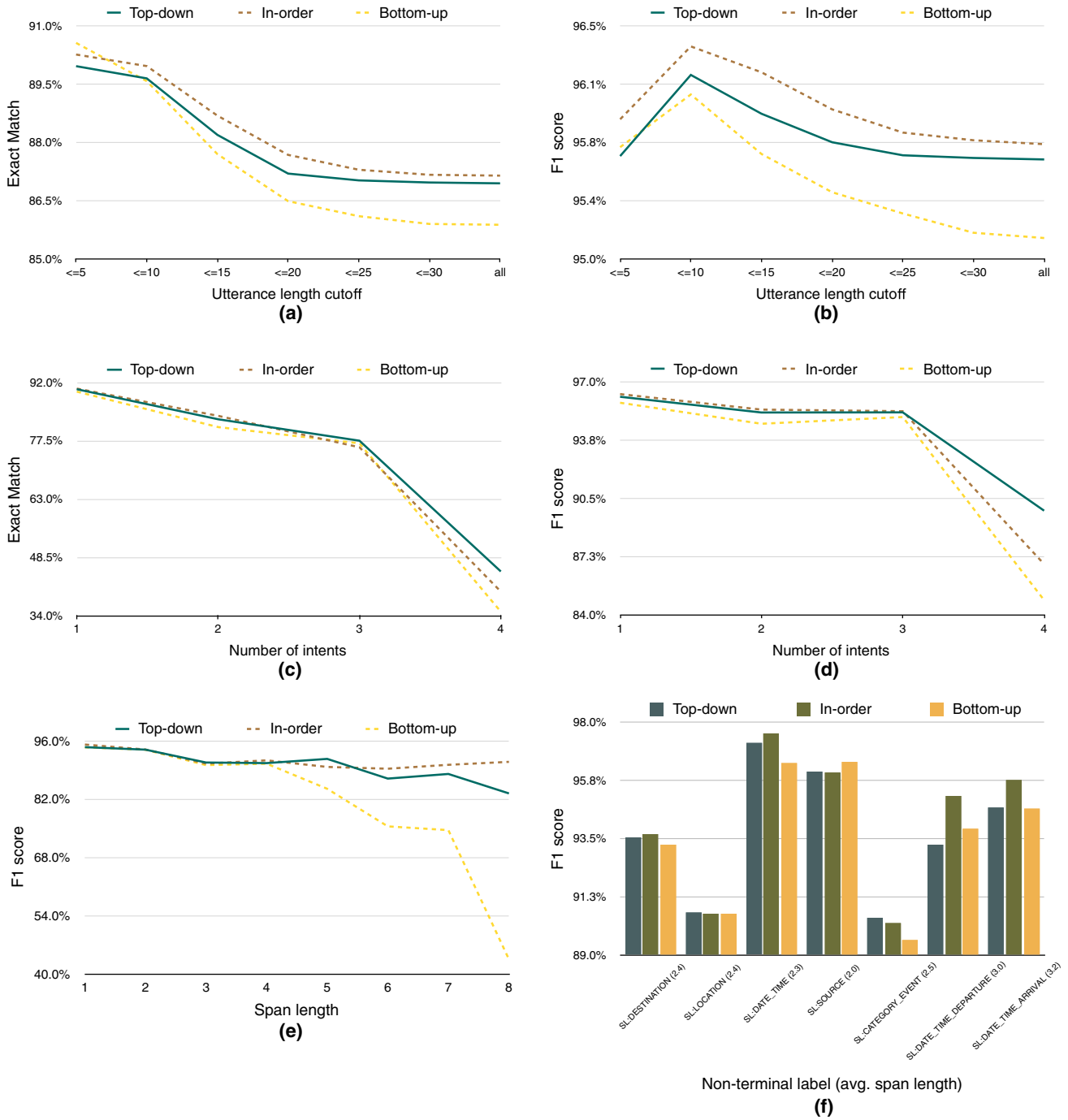

Fig. 4 Performance comparison of the three transition systems relative to utterance length and structural factors

transition sequence to be constructed and are thus more susceptible to error propagation. Additionally, the bottom-up transition system exhibits significant accuracy losses in producing constituents with longer spans. This can be attributed to the fact that, while the other two alternatives use a NON-TERMINAL-L action to mark the beginning of the future constituent, the bottom-up strategy determines the entire span with a single REDUCE#K-L transition at the end of the constituent creation. This approach, being more susceptible to error propagation, struggles with REDUCE#K-L transitions with higher k values, which are less frequent in the training data and hence more challenging to learn. Regarding the in-order algorithm, it appears to be more robust than the top-down transition system on constituents with the longest

span, indicating that the advantages of the in-order strategy (explained in "Transition Systems for Task-Oriented Semantic Parsing") mitigate the impact of error propagation. Moreover, in Fig. 4f, we observe that the in-order parser outperforms the other methods in predicting frequent non-terminal labels in nearly all cases, resulting in significant differences in accuracy in building slot constituents with the longest span, such as `SL:DATE_TIME_DEPARTURE` and `SL:DATE_TIME_ARRIVAL`. The only exceptions are slot constituents `SL:SOURCE` and `SL:CATEGORY_EVENT`, where the bottom-up and top-down algorithms respectively achieve higher accuracy.

Conclusions

In this paper, we introduce innovative shift-reduce semantic parsers tailored for processing task-oriented dialogue utterances. In addition to the commonly used top-down algorithm for this task, we adapt the bottom-up and in-order transition systems from constituency parsing to generate well-formed TOP trees. Moreover, we devise a more robust neural architecture that, unlike previous shift-reduce approaches, leverages Stack-Transformers and RoBERTa-based contextualized word embeddings.

We extensively evaluate the three proposed algorithms across high-resource and low-resource settings, as well as multiple domains of the widely used Facebook TOP benchmark. This marks the first evaluation of a shift-reduce approach in low-resource task-oriented parsing, to the best of our knowledge. Through these experiments, we demonstrate that the in-order transition system emerges as the most accurate alternative, surpassing all existing shift-reduce parsers not enhanced with re-ranking. Furthermore, it advances the state of the art in both high-resource and low-resource settings, surpassing all top-performing sequence-to-sequence baselines, including those employing larger pre-trained language models like BART.

Additionally, it is worth noting that our approach holds potential for further enhancement through techniques such as ensemble parsing with a ranker, as developed by [22], or by specifically fine-tuning a RoBERTa-based encoder for task-oriented semantic parsing, as employed by the strongest sequence-to-sequence models. Finally, incorporating hierarchical semantic information, as successfully implemented by [33], is another avenue for improving our approach.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature. We acknowledge the European Research Council (ERC), which has funded this research under the European Union's Horizon 2020 research and innovation programme (FASTPARSE, grant agreement No 714150), ERDF/MICINN-AEI (PID2020-113230RB-C21, PID2020-113230RB-C22 and PID2023-147129OB-C22), Xunta de Galicia (ED431C 2020/11), and Centro de Investigación de Galicia "CITIC", funded by Xunta de Galicia and the European Union (ERDF - Galicia 2014-2020 Program), by grant ED431G 2019/01

Data Availability The TOP and TOPv2 datasets used during the current research work are available in the Facebook repositories: http://fb.me/semanticparsingdialog and https://fb.me/TOPv2Dataset.

Declarations

Conflict of Interest The author declares no competing interests.

References

1. Mesnil G, Dauphin Y, Yao K, Bengio Y, Deng L, Hakkani-Tur D, He X, Heck L, Tur G, Yu D, Zweig G. Using recurrent neural networks for slot filling in spoken language understanding. IEEE/ACM Trans Audio Speech Lang Process. 2015;23(3):530–9. https://doi.org/10.1109/TASLP.2014.2383614.
2. Liu B, Lane IR. Attention-based recurrent neural network models for joint intent detection and slot filling. In: INTERSPEECH. 2016.
3. Goo C-W, Gao G, Hsu Y-K, Huo C-L, Chen T-C, Hsu K-W, Chen Y-N. Slot-gated modeling for joint slot filling and intent prediction. In: Proceedings of the 2018 conference of the North American chapter of the association for computational linguistics: human language technologies, vol 2 (Short papers). New Orleans: Louisiana Association for Computational Linguistics; 2018. p. 753–757. https://doi.org/10.18653/v1/N18-2118. https://www.aclweb.org/anthology/N18-2118.
4. Gupta S, Shah R, Mohit M, Kumar A, Lewis M. Semantic parsing for task oriented dialog using hierarchical representations. In: Proceedings of the 2018 conference on empirical methods in natural language processing. Brussels, Belgium: Association for Computational Linguistics; 2018. p. 2787–2792. https://doi.org/10.18653/v1/D18-1300 . https://www.aclweb.org/anthology/D18-1300.
5. Zelle JM, Mooney RJ. Learning to parse database queries using inductive logic programming. In: Proceedings of the thirteenth national conference on artificial intelligence - vol 2. AAAI'96. AAAI Press; 1996. p. 1050–1055.
6. Banarescu L, Bonial C, Cai S, Georgescu M, Griffitt K, Hermjakob U, Knight K, Koehn P, Palmer M, Schneider N. Abstract meaning representation for sembanking. In: Proceedings of the 7th linguistic annotation workshop and interoperability with discourse. Sofia, Bulgaria: Association for Computational Linguistics; 2013. p. 178–186. https://www.aclweb.org/anthology/W13-2322.
7. Dyer C, Kuncoro A, Ballesteros M, Smith NA. Recurrent neural network grammars. In: Proceedings of the 2016 conference of the north american chapter of the association for computational linguis-

tics: human language technologies. San Diego, California: Association for Computational Linguistics; 2016. p. 199–209 https://doi.org/10.18653/v1/N16-1024. https://aclanthology.org/N16-1024.
8. Wiseman S, Rush AM. Sequence-to-sequence learning as beam-search optimization. In: Proceedings of the 2016 conference on empirical methods in natural language processing. Austin, Texas: Association for Computational Linguistics; 2016. p. 1296–1306. https://doi.org/10.18653/v1/D16-1137. https://www.aclweb.org/anthology/D16-1137.
9. Gehring J, Auli M, Grangier D, Yarats D, Dauphin YN. Convolutional sequence to sequence learning. In: Precup D, Teh YW, editors. Proceedings of the 34th International conference on machine learning. Proceedings of machine learning research, vol. 70. PMLR; 2017. p. 1243–1252. https://proceedings.mlr.press/v70/gehring17a.html.
10. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser U, Polosukhin I. Attention is all you need. In: Proceedings of the 31st international conference on neural information processing systems. NIPS'17. Red Hook, NY, USA: Curran Associates Inc.; 2017. p. 6000–6010.
11. Rongali S, Soldaini L, Monti E, Hamza W. Don't parse, generate! a sequence to sequence architecture for task-oriented semantic parsing. New York, USA: Association for Computing Machinery; 2020. p. 2962–2968. https://doi.org/10.1145/3366423.3380064.
12. Aghajanyan A, Maillard J, Shrivastava A, Diedrick K, Haeger M, Li H, Mehdad Y, Stoyanov V, Kumar A, Lewis M, Gupta S. Conversational semantic parsing. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). Association for Computational Linguistics, Online; 2020. p. 5026–5035. https://doi.org/10.18653/v1/2020.emnlp-main.408. https://aclanthology.org/2020.emnlp-main.408.
13. Liu Y, Ott M, Goyal N, Du J, Joshi M, Chen D, Levy O, Lewis M, Zettlemoyer L, Stoyanov V. Roberta: a robustly optimized bert pretraining approach. 2019. arXiv:1907.11692
14. Lewis M, Liu Y, Goyal N, Ghazvininejad M, Mohamed A, Levy O, Stoyanov V, Zettlemoyer L. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In: Proceedings of the 58th annual meeting of the association for computational linguistics. Association for Computational Linguistics, Online; 2020. p. 7871–7880. https://doi.org/10.18653/v1/2020.acl-main.703. https://aclanthology.org/2020.acl-main.703.
15. Dyer C, Ballesteros M, Ling W, Matthews A, Smith NA. Transition-based dependency parsing with stack long short-term memory. In: Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (vol 1: Long Papers). Beijing, China: Association for Computational Linguistics; 2015. p. 334–343. https://doi.org/10.3115/v1/P15-1033. https://www.aclweb.org/anthology/P15-1033.
16. Fernandez Astudillo R, Ballesteros M, Naseem T, Blodgett A, Florian R. Transition-based parsing with stack-transformers. In: Findings of the association for computational linguistics: EMNLP 2020. Online: Association for Computational Linguistics; 2020. p. 1001–1007. https://doi.org/10.18653/v1/2020.findings-emnlp.89. https://www.aclweb.org/anthology/2020.findings-emnlp.89.
17. Fernández-González D, Gómez-Rodríguez C. Faster shift-reduce constituent parsing with a non-binary, bottom-up strategy. Artif Intell. 2019;275:559–74. https://doi.org/10.1016/j.artint.2019.07.006.
18. Liu J, Zhang Y. In-order transition-based constituent parsing. Trans Assoc Comput Linguist. 2017;5:413–24.
19. Chen X, Ghoshal A, Mehdad Y, Zettlemoyer L, Gupta S. Low-resource domain adaptation for compositional task-oriented semantic parsing. In: Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP). Online: Association for Computational Linguistics; 2020. p. 5090–5100. https://doi.org/10.18653/v1/2020.emnlp-main.413. https://aclanthology.org/2020.emnlp-main.413.
20. Sutskever I, Vinyals O, Le QV. Sequence to sequence learning with neural networks. In: Proceedings of the 27th International conference on neural information processing systems - vol 2. NIPS'14. Cambridge, MA, USA: MIT Press; 2014. p. 3104–3112.
21. Vinyals O, Kaiser L, Koo T, Petrov S, Sutskever I, Hinton G. Grammar as a foreign language. In: Proceedings of the 28th International conference on neural information processing systems - vol 2. NIPS'15. Cambridge, MA, USA: MIT Press; 2015. p. 2773–2781. http://dl.acm.org/citation.cfm?id=2969442.2969550.
22. Einolghozati A, Pasupat P, Gupta S, Shah R, Mohit M, Lewis M, Zettlemoyer L. Improving semantic parsing for task oriented dialog. 2019. arXiv:1902.06000.
23. Peters M, Neumann M, Iyyer M, Gardner M, Clark C, Lee K, Zettlemoyer L. Deep contextualized word representations. In: Proceedings of the 2018 conference of the North American Chapter of the association for computational linguistics: human language technologies, vol 1 (Long papers). New Orleans, Louisiana: Association for Computational Linguistics; 2018. p. 2227–2237. https://doi.org/10.18653/v1/N18-1202. https://www.aclweb.org/anthology/N18-1202.
24. Pasupat P, Gupta S, Mandyam K, Shah R, Lewis M, Zettlemoyer L. Span-based hierarchical semantic parsing for task-oriented dialog. In: Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). Hong Kong, China: Association for Computational Linguistics; 2019. p. 1520–1526. https://doi.org/10.18653/v1/D19-1163. https://aclanthology.org/D19-1163.
25. Stern M, Andreas J, Klein D. A minimal span-based neural constituency parser. In: Proceedings of the 55th annual meeting of the association for computational linguistics (vol 1: Long Papers). Vancouver, Canada: Association for Computational Linguistics; 2017. p. 818–827. https://doi.org/10.18653/v1/P17-1076. https://www.aclweb.org/anthology/P17-1076.
26. See A, Liu PJ, Manning CD. Get to the point: summarization with pointer-generator networks. In: Proceedings of the 55th annual meeting of the association for computational linguistics (vol 1: Long Papers). Vancouver, Canada: Association for Computational Linguistics; 2017. p. 1073–1083. https://doi.org/10.18653/v1/P17-1099. https://www.aclweb.org/anthology/P17-1099.
27. Zhu Q, Khan H, Soltan S, Rawls S, Hamza W. Don't parse, insert: multilingual semantic parsing with insertion based decoding. In: Proceedings of the 24th conference on computational natural language learning. Online: Association for Computational Linguistics; 2020. p. 496–506. https://doi.org/10.18653/v1/2020.conll-1.40. https://aclanthology.org/2020.conll-1.40.
28. Babu A, Shrivastava A, Aghajanyan A, Aly A, Fan A, Ghazvininejad M. Non-autoregressive semantic parsing for compositional task-oriented dialog. In: Proceedings of the 2021 Conference of the North American chapter of the association for computational linguistics: human language technologies. Online: Association for Computational Linguistics; 2021. p. 2969–2978. https://doi.org/10.18653/v1/2021.naacl-main.236. https://aclanthology.org/2021.naacl-main.236.
29. Shrivastava A, Chuang P, Babu A, Desai S, Arora A, Zotov A, Aly A. Span pointer networks for non-autoregressive task-oriented semantic parsing. In: Findings of the Association for Computational Linguistics: EMNLP 2021. Punta Cana, Dominican Republic: Association for Computational Linguistics; 2021. p. 1873–1886. https://doi.org/10.18653/v1/2021.findings-emnlp.161. https://aclanthology.org/2021.findings-emnlp.161.
30. Oh G, Goel R, Hidey C, Paul S, Gupta A, Shah P, Shah R. Improving top-k decoding for non-autoregressive semantic parsing via

intent conditioning. In: Calzolari N, Huang C-R, Kim H, Pustejovsky J, Wanner L, Choi K-S, Ryu P-M, Chen H-H, Donatelli L, Ji H, Kurohashi S, Paggio P, Xue N, Kim S, Hahm Y, He Z, Lee TK, Santus E, Bond F, Na S-H, editors. Proceedings of the 29th International conference on computational linguistics. Gyeongju, Republic of Korea: International Committee on Computational Linguistics; 2022. p. 310–322. https://aclanthology.org/2022.coling-1.24.

31. Shrivastava A, Desai S, Gupta A, Elkahky A, Livshits A, Zotov A, Aly A. Retrieve-and-fill for scenario-based task-oriented semantic parsing. In: Vlachos, A., Augenstein, I. (eds.) Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, pp. 430–447. Dubrovnik, Croatia: Association for Computational Linguistics; 2023. https://doi.org/10.18653/v1/2023.eacl-main.32. https://aclanthology.org/2023.eacl-main.32.
32. Wang S, Shrivastava A, Livshits A. Treepiece: Faster semantic parsing via tree tokenization. In: Bouamor H, Pino J, Bali K editors. Findings of the association for computational linguistics: EMNLP 2023. Singapore: Association for Computational Linguistics; 2023. p. 11082–11092. https://doi.org/10.18653/v1/2023.findings-emnlp.740. https://aclanthology.org/2023.findings-emnlp.740.
33. Do T, Nguyen P, Nguyen M. StructSP: Efficient fine-tuning of task-oriented dialog system by using structure-aware boosting and grammar constraints. In: Rogers A, Boyd-Graber J, Okazaki N editors. Findings of the association for computational linguistics: ACL 2023. Toronto, Canada: Association for Computational Linguistics; 2023. p. 10206–10220. https://doi.org/10.18653/v1/2023.findings-acl.648. https://aclanthology.org/2023.findings-acl.648.
34. Mansimov E, Zhang Y. Semantic parsing in task-oriented dialog with recursive insertion-based encoder. 2021. arXiv:2109.04500
35. Do D-T, Nguyen M-P, Nguyen L-M. Gram: grammar-based refined-label representing mechanism in the hierarchical semantic parsing task. In: Métais E, Meziane F, Sugumaran V, Manning W, Reiff-Marganiec S, editors. Natural language processing and information systems. Cham: Springer; 2023. p. 339–51.
36. Yamada H, Matsumoto Y. Statistical dependency analysis with support vector machines. In: Proceedings of the 8th International Workshop on Parsing Technologies (IWPT). 2003. p. 195–206.
37. Sagae K, Lavie A. A classifier-based parser with linear run-time complexity. In: Proceedings of the 9th International Workshop on Parsing Technologies (IWPT). 2005. p. 125–132.
38. Zhu M, Zhang Y, Chen W, Zhang M, Zhu J. Fast and accurate shift-reduce constituent parsing. In: Proceedings of the 51st annual meeting of the association for computational linguistics (vol 1: Long Papers). Sofia, Bulgaria: Association for Computational Linguistics; 2013. p. 434–443. https://www.aclweb.org/anthology/P13-1043.
39. Ballesteros M, Al-Onaizan Y. AMR parsing using stack-LSTMs. In: Proceedings of the 2017 conference on empirical methods in natural language processing. Copenhagen, Denmark: Association for Computational Linguistics; 2017. pp. 1269–1275. https://doi.org/10.18653/v1/D17-1130. https://www.aclweb.org/anthology/D17-1130.
40. Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput. 1997;9(8):1735–80. https://doi.org/10.1162/neco.1997.9.8.1735.
41. Bahdanau D, Cho K, Bengio Y. Neural machine translation by jointly learning to align and translate. In: Bengio Y, LeCun Y, editorss. 3rd International conference on learning representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings; 2015. http://arxiv.org/abs/1409.0473.
42. Ba JL, Kiros JR, Hinton GE. Layer normalization. 2016. https://doi.org/10.48550/ARXIV.1607.06450. https://arxiv.org/abs/1607.06450
43. Devlin J, Chang M-W, Lee K, Toutanova K. BERT: pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, vol 1 (Long and Short Papers). Minneapolis, Minnesota: Association for Computational Linguistics; 2019. p. 4171–4186. https://doi.org/10.18653/v1/N19-1423. https://www.aclweb.org/anthology/N19-1423.
44. Black E, Abney S, Flickinger D, Gdaniec C, Grishman R, Harrison P, Hindle D, Ingria R, Jelinek F, Klavans J, Liberman M, Roukos S, Santorini B, Strzalkowski T. A procedure for quantitatively comparing the syntactic coverage of English grammars. In: Proceedings of the 4th DARPA Speech and Natural Language Workshop. 1991. p. 306–311
45. Ott M, Edunov S, Baevski A, Fan A, Gross S, Ng N, Grangier D, Auli M. fairseq: a fast, extensible toolkit for sequence modeling. In: Proceedings of NAACL-HLT 2019: Demonstrations; 2019.
46. Kingma DP, Ba J. Adam: a method for stochastic optimization. Published as a conference paper at the 3rd international conference for learning representations 2015. San Diego; 2014.