Unleashing the True Potential of Sequence-to-Sequence Models for Sequence Tagging and Structure Parsing

Han He
Department of Computer Science
Emory University
Atlanta, GA 30322 USA
han.he@emory.edu

Jinho D. Choi
Department of Computer Science
Emory University
Atlanta, GA 30322 USA
jinho.choi@emory.edu

Abstract

Sequence-to-Sequence (S2S) models have achieved remarkable success on various text generation tasks. However, learning complex structures with S2S models remains challenging as external neural modules and additional lexicons are often supplemented to predict non-textual outputs. We present a systematic study of S2S modeling using contained decoding on four core tasks: part-of-speech tagging, named entity recognition, constituency, and dependency parsing, to develop efficient exploitation methods costing zero extra parameters. In particular, 3 lexically diverse linearization schemas and corresponding constrained decoding methods are designed and evaluated. Experiments show that although more lexicalized schemas yield longer output sequences that require heavier training, their sequences being closer to natural language makes them easier to learn. Moreover, S2S models using our constrained decoding outperform other S2S approaches using external resources. Our best models perform better than or comparably to the state-of-the-art for all 4 tasks, lighting a promise for S2S models to generate non-sequential structures.

1 Introduction

Sequence-to-Sequence (S2S) models pretrained for language modeling (PLM) and denoising objectives have been successful on a wide range of NLP tasks where both inputs and outputs are sequences (Radford et al., 2019; Raffel et al., 2020; Lewis et al., 2020; Brown et al., 2020). However, for non-sequential outputs like trees and graphs, a procedure called linearization is often required to flatten them into ordinary sequences (Li et al., 2018; Fernández-González and Gómez-Rodríguez, 2020; Yan et al., 2021; Bevilacqua et al., 2021; He and Choi, 2021a), where labels in non-sequential structures are mapped heuristically as individual tokens in sequences, and numerical properties like indices are either predicted using an external decoder such as Pointer Networks (Vinyals et al., 2015a) or cast to additional tokens in the vocabulary. While these methods are found to be effective, we hypothesize that S2S models can learn complex structures without adapting such patches.

To challenge the limit of S2S modeling, BART (Lewis et al., 2020) is finetuned on four tasks without extra decoders: part-of-speech tagging (POS), named entity recognition (NER), constituency parsing (CON), and dependency parsing (DEP). Three novel linearization schemas are introduced for each task: label sequence (LS), label with text (LT), and prompt (PT). LS to PT feature an increasing number of lexicons and a decreasing number of labels, which are not in the vocabulary (Section 3). Every schema is equipped with a constrained decoding algorithm searching over valid sequences (Section 4).

Our experiments on three popular datasets depict that S2S models can learn these linguistic structures without external resources such as index tokens or Pointer Networks. Our best models perform on par with or better than the other state-of-the-art models for all four tasks (Section 5). Finally, a detailed analysis is provided to compare the distinctive natures of our proposed schemas (Section 6).[1]

2 Related Work

S2S (Sutskever et al., 2014) architectures have been effective on many sequential modeling tasks. Conventionally, S2S is implemented as an encoder and decoder pair, where the encoder learns input representations used to generate the output

[1]All our resources including source codes are publicly available: https://github.com/emorynlp/seq2seq-corenlp.

Transactions of the Association for Computational Linguistics, vol. 11, pp. 582–599, 2023. https://doi.org/10.1162/tacl_a_00557
Action Editor: Emily Pitler. Submission batch: 11/2022; Revision batch: 1/2023; Published 6/2023.

sequence via the decoder. Since the input sequence can be very long, attention mechanisms (Bahdanau et al., 2015; Vaswani et al., 2017) focusing on particular positions are often augmented to the basic architecture. With transfer-learning, S2S models pretrained on large unlabeled corpora have risen to a diversity of new approaches that convert language problems into a text-to-text format (Akbik et al., 2018; Lewis et al., 2020; Radford et al., 2019; Raffel et al., 2020; Brown et al., 2020). Among them, tasks most related to our work are linguistic structure predictions using S2S, POS, NER, DEP, and CON.

POS has been commonly tackled as a sequence tagging task, where the input and output sequences have equal lengths. S2S, on the other hand, does not enjoy such constraints as the output sequence can be arbitrarily long. Therefore, S2S is not as popular as sequence tagging for POS. Prevailing neural architectures for POS are often built on top of a neural sequence tagger with rich embeddings (Bohnet et al., 2018; Akbik et al., 2018) and Conditional Random Fields (Lafferty et al., 2001).

NER has been cast to a neural sequence tagging task using the IOB notation (Lample et al., 2016) over the years, which benefits most from contextual word embeddings (Devlin et al., 2019; Wang et al., 2021). Early S2S-based works cast NER to a text-to-IOB transduction problem (Chen and Moschitti, 2018; Straková et al., 2019; Zhu et al., 2020), which is included as a baseline schema in Section 3.2. Yan et al. (2021) augment Pointer Networks to generate numerical entity spans, which we refrain to use because the focus of this work is purely on the S2S itself. Most recently, Cui et al. (2021) propose the first template prompting to query all possible spans against a S2S language model, which is highly simplified into a one-pass generation in our PT schema. Instead of directly prompting for the entity type, Chen et al. (2022) propose to generate its concepts first then its type later. Their two-step generation is tailored for few-shot learning, orthogonal to our approach. Moreover, our prompt approach does not rely on non-textual tokens as they do.

CON is a more established task for S2S models since the bracketed constituency tree is naturally a linearized sequence. Top-down tree linearizations based on brackets (Vinyals et al., 2015b) or shift-reduce actions (Sagae and Lavie, 2005) rely on a strong encoder over the sentence while bottom-up ones (Zhu et al., 2013; Ma et al., 2017) can utilize rich features from readily built partial parses. Recently, the in-order traversal has proved superior to bottom-up and top-down in both transition (Liu and Zhang, 2017) and S2S (Fernández-González and Gómez-Rodríguez, 2020) constituency parsing. Most recently, a Pointer Networks augmented approach (Yang and Tu, 2022) is ranked top among S2S approaches. Since we are interested in the potential of S2S models without patches, a naive bottom-up baseline and its novel upgrades are studied in Section 3.3.

DEP has been underexplored as S2S due to the linearization complexity. The first S2S work maps a sentence to a sequence of source sentence words interleaved with the arc-standard, reduce-actions in its parse (Wiseman and Rush, 2016), which is adopted as our LT baseline in Section 3.4. Zhang et al. (2017) introduce a stack-based multi-layer attention mechanism to leverage structural linguistics information from the decoding stack in arc-standard parsing. Arc-standard is also used in our LS baseline, however, we use no such extra layers. Apart from transition parsing, Li et al. (2018) directly predict the relative head position instead of the transition. This schema is later extended to multilingual and multitasking by Choudhary and O'riordan (2021). Their encoder and decoder use different vocabularies, while in our PT setting, we re-use the vocabulary in the S2S language model.

S2S appears to be more prevailing for semantic parsing due to two reasons. First, synchronous context-free grammar bridges the gap between natural text and meaning representation for S2S. It has been employed to obtain silver annotations (Jia and Liang, 2016), and to generate canonical natural language paraphrases that are easier to learn for S2S (Shin et al., 2021). This trend of insights viewing semantic parsing as prompt-guided generation (Hashimoto et al., 2018) and paraphrasing (Berant and Liang, 2014) has also inspired our design of PT. Second, the flexible input/output format of S2S facilitates joint learning of semantic parsing and generation. Latent variable sharing (Tseng et al., 2020) and unified pretraining (Bai et al., 2022) are two representative joint modeling approaches, which could be augmented with our idea of PT schema as a potentially more effective linearization.

Our finding that core NLP tasks can be solved using LT overlaps with the Translation between Augmented Natural Languages (Paolini et al.,

2021). However, we take one step further to study the impacts of textual tokens in schema design choices. Our constrained decoding is similar to existing work (Hokamp and Liu, 2017; Deutsch et al., 2019; Shin et al., 2021). We craft constrained decoding algorithms for our proposed schemas and provide a systematic ablation study in Section 6.1.

3 Schemas

This section presents our output schemas for POS, NER, CON, and DEP in Table 1. For each task, 3 lexically diverse schemas are designed as follows to explore the best practice for structure learning. First, Label Sequence (LS) is defined as a sequence of labels consisting of a finite set of task-related labels, that are merged into the S2S vocabulary, with zero text. Second, Label with Text (LT) includes tokens from the input text on top of the labels such that it has a medium number of labels and text. Third, PromptT (PT) gives a list of sentences describing the linguistic structure in natural language with no label. We hypothesize that the closer the output is to natural language, the more advantage the S2S takes from the PLM.

3.1 Part-of-Speech Tagging (POS)

LS LS defines the output as a sequence of POS tags. Formally, given an input sentence of n tokens $\mathbf{x} = \{x_1, x_2, \cdots, x_n\}$, its output is a tag sequence of the same length $\mathbf{y}^{\text{LS}} = \{y_1, y_2, \cdots, y_n\}$. Distinguished from sequence tagging, any LS output sequence is terminated by the ''end-of-sequence'' (EOS) token, which is omitted from $\mathbf{y}^{\text{LS}}$ for simplicity. Predicting POS tags often depends on their neighbor contexts. We challenge that the autoregressive decoder of a S2S model can capture this dependency through self-attention.

LT For LT, the token from the input is inserted before its corresponding tag. Formally, the output is defined $\mathbf{y}^{\text{LT}} = \{(x_1, y_1), (x_2, y_2), .., (x_n, y_n)\}$. Both $\mathbf{x}$ and $\mathbf{y}$ are part of the output and the S2S model is trained to generate each pair sequentially.

PT PT is a human-readable text describing the POS sequence. Specifically, we use a phrase $\mathbf{y}_i^{\text{PT}} =$''$x_i$ is y_i''' for the i-th token, where y_i' is the definition of a POS tag y_i, e.g., a noun. The final prompt is then the semicolon concatenation of all phrases: $\mathbf{y}^{\text{PT}} = \mathbf{y}_1^{\text{PT}}; \mathbf{y}_2^{\text{PT}}; \cdots; \mathbf{y}_n^{\text{PT}}$.

3.2 Named Entity Recognition (NER)

LS LT of an input sentence comprising n tokens $\mathbf{x} = \{x_1, x_2, \cdots, x_n\}$ is defined as the BIEOS tag sequence $\mathbf{y}^{\text{LS}} = \{y_1, y_2, \cdots, y_n\}$, which labels each token as the Beginning, Inside, End, Outside, or Single-token entity.

LT LT uses a pair of entity type labels to wrap each entity: $\mathbf{y}^{\text{LT}} = ..\text{B}\text{-}y_j, x_i, .., x_{i+k}, \text{E}\text{-}y_j, ..$, where y_j is the type label of the j-th entity consisting of k tokens.

PT PT is defined as a list of sentences describing each entity: $\mathbf{y}_i^{\text{PT}} =$''$x_i$ is y_i''', where y_i' is the definition of a NER tag y_i, e.g., a person. Different from the prior prompt work (Cui et al., 2021), our model generates all entities in one pass which is more efficient than their brute-force approach.

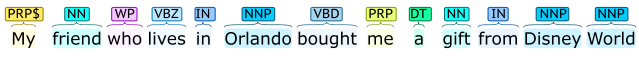
3.3 Constituency Parsing (CON)

Schemas for CON are developed on constituency trees pre-processed by removing the first level of non-terminals (POS tags) and rewiring their children (tokens) to parents, e.g., (NP (PRON My) (NOUN friend)) → (NP My friend).

LS LS is based on a top-down shift-reduce system consisting of a stack, a buffer, and a depth record d. Initially, the stack contains only the root constituent with label TOP and depth 0; the buffer contains all tokens from the input sentence; d is set to 1. A Node-X (N-X) transition creates a new depth-d non-terminal labeled with X, pushes it to the stack, and sets $d \leftarrow d + 1$. A Shift (SH) transition removes the first token from the buffer and pushes it to the stack as a new terminal with depth d. A Reduce (RE) pops all elements with the same depth d from the stack then make them the children of the top constituent of the stack, and it sets $d \leftarrow d - 1$. The linearization of a constituency tree using our LS schema can be obtained by applying 3 string substitutions: replace each left bracket and the label X following it with a Node-X, replace terminals with SH, replace right brackets with RE.

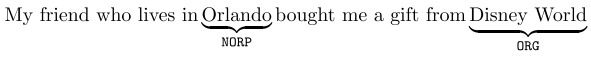
LT LT is derived by reverting all SH in LS back to the corresponding tokens so that tokens in LT effectively serves as SH in our transition system.

Part-of-Speech Tagging



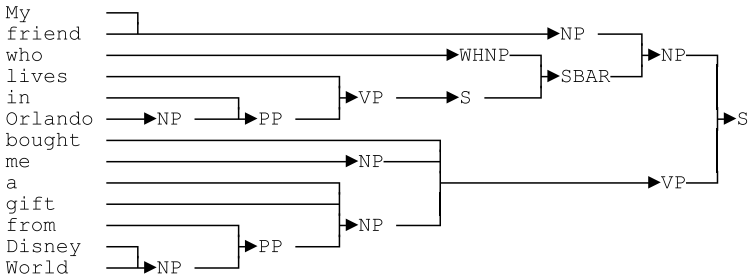
LS	PRP$ NN WP VBZ IN NNP VBD PRP DT NN IN NNP NNP
LT	My/PRP$ friend/NN who/WP lives/VBZ in/IN Orlando/NNP bought/VBD me/PRP a/DT gift/NN from/IN Disney/NNP World/NNP
PT	"My" is a *possessive pronoun*; "friend" is a *noun*; "who" is a *wh-pronoun*; "lives" is a *3rd-person present verb*; "in" is a *preposition*; "Orlando" is a *proper noun*; "bought" is a *past verb*; "me" is a *personal pronoun*; "a" is a *determiner*; "gift" is a *noun*; "from" is a *preposition*; "Disney" is a *proper noun*; "World" is a *proper noun*.

Named Entity Recognition



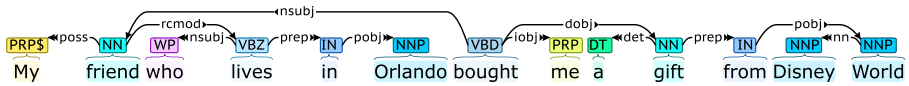
LS	O O O O O S-NORP O O O O O B-ORG E-ORG
LT	My friend who lives in <NORP>Orlando</NORP> bought me a gift from <ORG>Disney World</ORG>
PT	"Orlando" is a *geopolitical entity*; "Disney World" is an *organization*.

Constituency Parsing



LS	N-S N-NP N-NP SH SH RE N-SBAR N-WHNP SH RE N-S N-VP SH N-PP SH N-NP SH RE RE RE RE RE RE N-VP SH N-NP SH RE N-NP SH SH N-PP SH N-NP SH SH RE RE RE RE RE
LT	(S (NP (NP My friend) (SBAR (WHNP who) (S (VP lives (PP in (NP Orlando))))))) (VP bought (NP me) (NP a gift (PP from (NP Disney World))))
PT	The *sentence* has a *noun phrase* and a *verb phrase*; The *noun phrase* has the *noun phrase* "My friend" and the *subordinating clause*, which has the *wh-noun phrase* "who" and the *clause*, which has the *verb phrase*, which has "lives" and the *preposition phrase*, which has "in" and the *noun phrase* "Orlando"; the *verb phrase* has "bought" and the *noun phrase* "me" and the *noun phrase* "a gift" and the *preposition phrase*, which has "from" and the *noun phrase* "Disney World".

Dependency Parsing



LS	SH SH <poss SH SH <nsubj SH SH <case >obl >relcl SH <nsubj SH >iobj SH SH <det SH SH SH <compound <case >nmod >obj
LT	My friend <poss who lives <nsubj in Orlando <case >obl >relcl bought <nsubj me >iobj a gift <det from Disney World <compound <case >nmod >obj
PT	"My" is a *possessive modifier* of "friend"; "who" is a *nominal subject* of "lives"; "in" is a *case marker* of "Orlando"; "lives" has an *oblique* "Orlando"; "friend" has a *relative clause* "lives"; "friend" is a *nominal subject* of "bought"; "bought" has an *indirect object* "me"; "a" is a *determiner* of "gift"; "Disney" is a *compound word* of "World"; "from" is a *case marker* of "World"; "gift" has a *nominal modifier* "World"; "bought" has an *object* "gift".

Table 1: Schemas for the sentence ''My friend who lives in Orlando bought me a gift from Disney World''.

PT PT is also based on a top-down linearization, although it describes a constituent using templates: ''p_i has $\{c_j\}$'', where p_i is a constituent and c_j-s are its children. To describe a constituent, the indefinite article ''*a*'' is used to denote a new constituent (e.g., ''. . . *has* ***a*** *noun phrase*''). The definite article ''*the*'' is used for referring to an existing constituent mentioned before (e.g.,

‘‘***the*** *noun phrase has ...*’’), or describing a constituent whose children are all terminals (e.g., ‘‘*. . . has* ***the*** *noun phrase ‘My friend’*’’). When describing a constituent that directly follows its mention, the determiner ‘‘*which*’’ is used instead of repeating itself multiple times e.g., ‘‘(*. . . and the subordinating clause,* ***which*** *has ...*’’). Sentences are joined with a semicolon ‘‘**;**’’ as the final prompt.

3.4 Dependency Parsing (DEP)

LS LS uses three transitions from the arc-standard system (Nivre, 2004): shift (SH), left arc (<), and right arc (>).

LT LT for DEP is obtained by replacing each SH in a LS with its corresponding token.

PT PT is derived from its LS sequence by removing all SH. Then, for each left arc creating an arc from x_j to x_i with dependency relation r (e.g., a possessive modifier), a sentence is created by applying the template ‘‘x_i ***is*** r *of* x_j’’. For each right arc creating an arc from x_i to x_j with the dependency relation r, a sentence is created with another template ‘‘x_i ***has*** r x_j’’. The prompt is finalized by joining all such sentences with a semicolon.

4 Decoding Strategies

To ensure well-formed output sequences that match the schemas (Section 3), a set of constrained decoding strategies is designed per task except for CON, which is already tackled as S2S modeling without constrained decoding (Vinyals et al., 2015b; Fernández-González and Gómez-Rodríguez, 2020). Formally, given an input $\mathbf{x}$ and any partial $\mathbf{y}_{<i}$, a constrained decoding algorithm defines a function NextY returning the set of all possible values for y_i that can immediately follow $\mathbf{y}_{<i}$ without violating the schema. For brevity, subwords handling is separately explained in Section 5.

4.1 Part-of-Speech Tagging

LS A finite set of POS tags, $\mathcal{D}$, are collected from the training set. Specifically, NextY returns $\mathcal{D}$ if $i \leq n$. Otherwise, it returns EOS.

LT Since tokens from the input sentence are interleaved with their POS tags in LT, NextY depends on the parity of i, as defined in Algorithm 1.

Algorithm 1: Constrained POS-LT

```
Function NextY(x, y_{<i}):
    if i > 2n then
        return {EOS}
    else
        if i is even then
            return {x_{i/2}}
        else
            return D
```

Algorithm 2: Prefix Matching

```
Function PrefixMatch(T, p):
    node ← T
    while node and p do
        node ← node.children[p_1]
        p ← p_{>1}
    return node
```

PT The PT generation can be divided into two phases: token and ‘‘is-a-tag’’ statement generations. A binary status u is used to indicate whether y_i is expected to be a token. To generate a token, an integer $k \leq n$ is used to track the index of the next token. To generate an ‘‘is-a-tag’’ statement, an offset o is used to track the beginning of an ‘‘is-a-tag’’ statement. Each description d_j of a POS tag t_j is extended to a suffix s_j = ‘‘is d_j ;’’.

Suffixes are stored in a trie tree $\mathcal{T}$ to facilitate prefix matching between a partially generated statement and all candidate suffixes, as shown in Algorithm 2. The full decoding is depicted in Algorithm 3.

4.2 Named Entity Recognition

LS Similar to POS-LS, the NextY for NER returns BIEOS tags if $i \leq n$ else EOS.

LT Opening tags (<>) in NER-LT are grouped into a vocabulary $\mathcal{O}$. The last generated output token y_{i-1} (assuming y_0 = BOS, a.k.a. beginning of a sentence) is looked up in $\mathcal{O}$ to decide what type of token will be generated next. To enforce label consistency between a pair of tags, a variable e is introduced to record the expected closing tag. Reusing the definition of k in Algorithm 3, decoding of NER-LT is described in Algorithm 4.

Algorithm 3: Constrained POS-PT

$u \leftarrow$ true, $k \leftarrow 0, o \leftarrow 0$
Function NextY($\mathbf{x}, \mathbf{y}_{<i}$)**:**
  if u **then**
    $u \leftarrow$ false, $k \leftarrow k+1, o \leftarrow i$
    return $\{x_k\}$
  else
    node $\leftarrow$ PrefixMatch($\mathcal{T}, \mathbf{y}_{>o}$)
    if *node.children is empty* **then**
      $u \leftarrow$ true
      return NextY($\mathbf{x}, \mathbf{y}_{<i}$)
    else
      return node.children

Algorithm 4: Constrained NER-LT

$k \leftarrow 0, e \leftarrow$ null
Function NextY($\mathbf{x}, \mathbf{y}_{<i}$)**:**
  $\mathcal{Y} \leftarrow \emptyset$
  if $y_{i-1} \in \mathcal{O}$ **then**
    $e \leftarrow$ the paired closing tag of y_{i-1}
  else if y_{i-1} *is* e **then**
    $e \leftarrow$ null
  else if $y_{i-1} \in \mathbf{x}$ **then**
    $k \leftarrow k+1$
  if e **then**
    $\mathcal{Y} \leftarrow \mathcal{Y} \cup \{e\}$
  else
    $\mathcal{Y} \leftarrow \mathcal{Y} \cup \mathcal{O}$
  if $k > n$ **then**
    $\mathcal{Y} \leftarrow \mathcal{Y} \cup \{EOS\}$
  else
    $\mathcal{Y} \leftarrow \mathcal{Y} \cup \{x_k\}$
  return $\mathcal{Y}$

PT For each entity type e_i, its description d_i is filled into the template ‘‘is d_i;’’ to create an ‘‘is-a’’ suffix s_i. Since the prompt is constructed using text while the number of entities is variable, it is not straightforward to tell whether a token belongs to an entity or an ‘‘is-a’’ suffix. Therefore, a noisy segmentation procedure is utilized to split a phrase into two parts: entity and ‘‘is-a’’ suffix. Each s_i is collected into a trie $\mathcal{S}$ to perform segmentation of a partially generated phrase $\mathbf{p}$ (Algorithm 5).

Once a segment is obtained, the decoder is constrained to generate the entity or the suffix. For the generation of an entity, string matching is used to find every occurrence o of its partial generation in $\mathbf{x}$ and add the following token x_{o+1} into the candidate set $\mathcal{Y}$. String matching could be noisy when an entity shares the same surface form with a non-entity phrase although no such cases are found in our datasets. Entities are generated sequentially and no nested entity is considered. To complete an ‘‘is-a’’ suffix, children of the prefix-matched node are added to the candidates (Algorithm 6).

Algorithm 5: Segmentation

Function Segment($\mathcal{S}, \mathbf{p}$)**:**
  for $i \leftarrow 1$ **to** $|\mathbf{p}|$ **do**
    entity, suffix $= \mathbf{p}_{\leq i}, \mathbf{p}_{>i}$
    node $\leftarrow$ PrefixMatch($\mathcal{S}, \mathbf{p}_{>i}$)
    if *node* **then**
      return entity, suffix, node
  return null

Algorithm 6: Constrained NER-PT

Function NextY($\mathbf{x}, \mathbf{y}_{<i}$)**:**
  if *EOS in* $\mathbf{y}_{<i}$ **then return** $\{$EOS$\}$
  $\mathcal{Y} \leftarrow \emptyset$
  $\mathbf{p} \leftarrow$ last phrase in $\mathbf{y}_{<i}$ split by ‘‘;’’
  if Segment($\mathcal{S}, \mathbf{p}$) **then**
    entity, suffix, node
     $\leftarrow$ Segment($\mathcal{S}, \mathbf{p}$)
    if *node is terminal* **then**
      $\mathcal{Y} \leftarrow \mathcal{Y} \cup \{EOS\}$
    else
      $\mathcal{Y} \leftarrow \mathcal{Y} \cup$ node.children
      if *node is root* **then**
        foreach *occurrence* o **do**
          $\mathcal{Y} \leftarrow \mathcal{Y} \cup \{x_{o+1}\}$
  else
    $\mathcal{Y} \leftarrow \mathcal{Y} \cup \mathbf{x}$
  return $\mathcal{Y}$

4.3 Constituency Parsing

S2S on CON using LS and LT has been studied and their results are good without using constrained decoding (Vinyals et al., 2015b; Fernández-González and Gómez-Rodríguez, 2020)[2]; thus, we focus on only PT in our work.

PT The reverse linearization algorithm for CON-PT is non-trivial. To restore a constituency tree from PT, the prompt is split into sentences

[2]We experimented constrained decoding with LS on PTB and OntoNotes, and the improvement is marginal (+0.01).

Algorithm 7: Longest Prefix Matching

Function `LongestPrefixMatch(`$\mathcal{T}, \mathbf{x}$`)`**:**
- matches $\leftarrow \emptyset$
- **for** $i \leftarrow 1$ **to** $|\mathbf{x}| + 1$ **do**
  - node $\leftarrow \mathcal{T}.children[x_i]$
  - **if** *node is null* **then**
    - **continue**
  - $j \leftarrow i + 1$
  - $v \leftarrow$ node.value
  - **for** $k \leftarrow j$ **to** $|\mathbf{x}| + 1$ **do**
    - node $\leftarrow node.children[x_k]$
    - **if** *node is null* **then**
      - **break**
    - **if** *node.value* **then**
      - $v \leftarrow$ node.value
      - $j \leftarrow k + 1$
  - **if** v **then**
    - matches $\leftarrow$ matches $\cup \{(i, j, v)\}$
- **return** matches

Algorithm 8: Longest Prefix Splitting

Function `Split(`$\mathcal{T}, \mathbf{x}$`)`**:**
- (spans, o) $\leftarrow (\emptyset, 1)$
- **foreach** $(i, j, v) \in$ *LongestPrefixMatch(*$\mathcal{T}, \mathbf{x}$*)* **do**
  - **if** $i > o$ **then**
    - spans $\leftarrow$ spans $\cup \{(o, i, \text{null})\}$
  - spans $\leftarrow$ spans $\cup \{(i, j, v)\}$
- **if** $o < |\mathbf{x}| + 1$ **then**
  - spans $\leftarrow$ spans $\cup \{(o, |\mathbf{x}| + 1, \text{null})\}$
- **return** spans

creating new constituents, and sentences attaching new constituents to existing ones. Splitting is done by longest-prefix-matching (Algorithm 7) using a trie $\mathcal{T}$ built with the definite and indefinite article versions of the description of each constituent label e.g., ''*the noun phrase*'' and ''*a noun phrase*'' of `NP`.

Algorithm 8 describes the splitting procedure.

Once a prompt is split into two types of sentences, a constituency tree is then built accordingly. We use a variable `parent` to track the last constituent that gets attachments, and another variable `latest` to track the current new consistent that gets created. Due to the top-down nature of linearization, the target constituent that new constituents are attached to is always among the siblings of either `parent` or the ancestors of `parent`. The search of the target constituent is described in Algorithm 9.

Algorithm 9: Find Target

Function `FindTarget(`*parent*, *label*`)`**:**
- **while** *parent* **do**
  - **foreach** *sibling of parent* **do**
    - **if** *sibling.label is label and sibling has no children* **then**
      - **return** sibling
  - parent $\leftarrow$ parent.parent
- **return** null

Algorithm 10: Reverse `CON-PT`

Function `Reverse(`$\mathcal{T}, \mathbf{x}$`)`**:**
- root $\leftarrow$ parent $\leftarrow$ new TOP-tree
- latest $\leftarrow$ null
- **foreach** $(i, j, v) \in$ *Split(*$\mathcal{T}, \mathbf{x}$*)* **do**
  - **if** v **then**
    - **if** $\mathbf{x}_{i:j}$ *starts with ''the''* **then**
      - target $\leftarrow$ `FindTarget`(parent, v)
    - **else**
      - latest $\leftarrow$ new v-tree
      - add latest to parent.children
      - latest.parent $\leftarrow$ parent
  - **else**
    - **if** $\mathbf{x}_{i:j}$ *starts with ''has'' or ''which has''* **then**
      - parent $\leftarrow$ latest
    - add tokens in ''' into latest
- **return** root

Algorithm 10 shows the final reverse linearization.

4.4 Dependency Parsing

`LS` Arc-standard (Nivre, 2004) transitions are added to a candidate set and only transitions permitted by the current parsing state are allowed.

`LT` `DEP-LS` replaces all `SH` transitions with input tokens in left-to-right order. Therefore, an incremental offset is kept to generate the next token in place of each `SH` in `DEP-LT`.

`PT` `DEP-PT` is more complicated than `CON-PT` because each sentence contains one more token. Its generation is therefore divided into 4 possible states: first token (`1st`), relation (`rel`), second token (`2ed`), and semicolon. An arc-standard

Algorithm 11: Recall Shift

Function `RecallShift(`*system*, i, x_j`)`**:**
  while *system*.s_i *is not* x_j **do**
    system.apply(`SH`)

transition system is executed synchronously with constrained decoding since `PT` is essentially a simplified transition sequence with all `SH` removed. Let $\mathbf{b}$ and $\mathbf{s}$ be the system buffer and stack, respectively. Let $\mathbf{c}$ be a set of candidate tokens that will be generated in $\mathbf{y}$, which initially contains all input tokens and an inserted token ''sentence'' that is only used to represent the root in ''the sentence has a root ...'' A token is removed from $\mathbf{c}$ once it gets popped out of $\mathbf{s}$. Since `DEP-PT` generates no `SH`, each input token x_j in $\mathbf{y}$ effectively introduces `SH`(s) till it is pushed onto $\mathbf{s}$ at index i ($i \in \{1, 2\}$), as formally described in Algorithm 11.

After the first token is generated, its offset o in $\mathbf{y}$ is recorded such that the following relation sequence $\mathbf{y}_{i>o}$ can be located. To decide the next token of $\mathbf{y}_{i>o}$, it is then prefix-matched with a trie $\mathcal{T}$ built with the set of ''has-'' and ''is-'' dependency relations. The children of the prefix-matched node are considered candidates if it has any. Otherwise, the dependency relation is marked as completed. Once a relation is generated, the second token will be generated in a similar way. Finally, upon the completion of a sentence, the transition it describes is applied to the system and $\mathbf{c}$ is updated accordingly. The full procedure is described in Algorithm 12. Since a transition system has been synchronously maintained with constrained decoding, no extra reverse linearization is needed.

5 Experiments

For all tasks, BART-Large (Lewis et al., 2020) is finetuned as our underlying S2S model. We also tried T5 (Raffel et al., 2020), although its performance was less satisfactory. Every model is trained three times using different random seeds and their average scores and standard deviations on the test sets are reported. Our models are experimented on the OntoNotes 5 (Weischedel et al., 2013) using the data split suggested by Pradhan et al. (2013). In addition, two other popular datasets are used for fair comparisons to previous works: the Wall Street Journal corpus from the Penn Treebank 3 (Marcus et al., 1993) for `POS`, `DEP`, and `CON`, as well as the English portion of the CoNLL'03 dataset (Tjong Kim Sang and De Meulder, 2003) for `NER`.

Algorithm 12: Constrained `DEP-PT`

(status, transition, t_1, t_2, o) $\leftarrow$ (1st, null, null, null, 0)
$\mathbf{c} \leftarrow \{$`sentence`$\} \cup \mathbf{y}$
Function `NextY(`$\mathbf{x}, \mathbf{y}_{<i}$`)`**:**
  if *EOS in* $\mathbf{y}_{<i}$ **then**
    return {`EOS`}
  $\mathcal{Y} \leftarrow \emptyset$
  $\mathbf{p} \leftarrow$ last phrase in $\mathbf{y}_{<i}$ split by ''**;**''
  if *status is semicolon* **then**
    $\mathcal{Y} \leftarrow \mathcal{Y} \cup \{;\}$
    status $\leftarrow$ 1st
  else if *status is 1st* **then**
    $\mathcal{Y} \leftarrow \mathcal{Y} \cup \mathbf{c}$
    $o \leftarrow i$
    status $\leftarrow$ rel
  else if *status is rel* **then**
    $t_1 \leftarrow y_o$
    node $\leftarrow$ `PrefixMatch(`$\mathcal{T}, \mathbf{y}_{>o}$`)`
    if *node.children* **then**
      $\mathcal{Y} \leftarrow \mathcal{Y} \cup \{$node.children$\}$
    else
      $relation \leftarrow$ the relation in $\mathbf{y}_{>o}$
      if $\mathbf{y}_{>o}$ *starts with ''is''* **then**
        $transition \leftarrow$ LA-relation
      else
        $transition \leftarrow$ RA-relation
      status $\leftarrow$ 2ed
  else if *status is 2ed* **then**
    $\mathcal{Y} \leftarrow \mathcal{Y} \cup \mathbf{c}$
    status $\leftarrow$ semicolon
  else if *status is semicolon* **then**
    $t_2 \leftarrow y_{i-1}$
    $\mathcal{Y} \leftarrow \mathcal{Y} \cup \{;\}$
    `RecallShift(`system, 1, t_2`)`
    `RecallShift(`system, 2, t_1`)`
    if *transition starts with LA* **then**
      remove s_1 from $\mathbf{c}$
    else
      remove s_2 from $\mathbf{c}$
    system.`apply(`transition`)`
    if *system is terminal* **then**
      $\mathcal{Y} \leftarrow \mathcal{Y} \cup \{$`EOS`$\}$
    status $\leftarrow$ 1st
  return $\mathcal{Y}$

Model	PTB	OntoNotes
Bohnet et al. (2018)	97.96	–
He and Choi (2021b)	–	98.32 ± 0.02
LS	97.51 ± 0.11	98.21 ± 0.02
LT	**97.70** ± 0.02	**98.40** ± 0.01
PT	97.64 ± 0.01	98.37 ± 0.02

Table 2: Results for POS.

Each token is independently tokenized using the subword tokenizer of BART and merged into an input sequence. The boundary information for each token is recorded to ensure full tokens are generated in LT and PT without broken pieces. To fit in the positional embeddings of BART, sentences longer than 1,024 subwords are discarded, which include 1 sentence from the Penn Treebank 3 training set, and 24 sentences from the OntoNotes 5 training set. Development sets and test sets are not affected.

5.1 Part-of-Speech Tagging

Token level accuracy is used as the metric for POS. LT outperforms LS although LT is twice as long as LS, suggesting that textual tokens positively impact the learning of the decoder (Table 2). PT performs almost the same with LT, perhaps due to the fact that POS is not a task requiring a powerful decoder.

5.2 Named Entity Recognition

For CoNLL'03, the provided splits without merging the development and training sets are used. For OntoNotes 5, the same splits as Chiu and Nichols (2016), Li et al. (2017); Ghaddar and Langlais (2018); He and Choi (2020, 2021b) are used. Labeled span-level F_1 score is used for evaluation.

We acknowledge that the performance of NER systems can be largely improved by rich embeddings (Wang et al., 2021), document context features (Yu et al., 2020), dependency tree features (Xu et al., 2021), and other external resources. While our focus is the potential of S2S, we mainly consider two strong baselines that also use BART as the only external resource: the generative BART-Pointer framework (Yan et al., 2021) and the recent template-based BART NER (Cui et al., 2021).

As shown in Table 3, LS performs the worst on both datasets, possibly attributed to the fact

Model	CoNLL'03	OntoNotes 5
Clark et al. (2018)	92.60	–
Peters et al. (2018)	92.22	–
Akbik et al. (2019)	93.18	–
Straková et al. (2019)	93.07	–
Yamada et al. (2020)	92.40	–
Yu et al. (2020)†	92.50	89.83
Yan et al. (2021)‡$^{\mathcal{S}}$	93.24	90.38
Cui et al. (2021)$^{\mathcal{S}}$	92.55	–
He and Choi (2021b)	–	89.04 ± 0.14
Wang et al. (2021)	94.6	–
Zhu and Li (2022)	–	91.74
Ye et al. (2022)	–	91.9
LS	70.29 ± 0.70	84.61 ± 1.18
LT	92.75 ± 0.03	89.60 ± 0.06
PT	**93.18** ± 0.04	**90.33** ± 0.04

Table 3: Results for NER. $\mathcal{S}$ denotes S2S.

that the autoregressive decoder overfits the high-order left-to-right dependencies of BIEOS tags. LT performs close to the BERT-Large biaffine model (Yu et al., 2020). PT performs comparably well with the Pointer Networks approach (Yu et al., 2020) and it outperforms the template prompting (Cui et al., 2021) by a large margin, suggesting S2S has the potential to learn structures without using external modules.

5.3 Constituency Parsing

All POS tags are removed and not used in training or evaluation. Terminals belonging to the same non-terminal are flattened into one constituent before training and unflattened in post-processing. The standard constituent-level F-score produced by the EVALB[3] is used as the evaluation metric.

Table 4 shows the results on OntoNotes 5 and PTB 3. Incorporating textual tokens into the output sequence is important on OntoNotes 5, leading to a +0.9 F-score, while it is not the case on PTB 3. It is possibly due to the fact that OntoNotes is more diverse in domains, requiring a higher utilization of pre-trained S2S for domain transfer. PT performs the best, and it has a competitive performance to recent works, despite the fact that it uses no extra decoders.

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

Model	PTB 3	OntoNotes 5
Fernández-González and Gómez-Rodríguez (2020)$^{\mathcal{S}}$	91.6	–
Mrini et al. (2020)	96.38	–
He and Choi (2021b)	–	94.43 ± 0.03
Yang and Tu (2022)$^{\mathcal{S}}$	96.01	–
LS	95.23 ± 0.08	93.40 ± 0.31
LT	95.24 ± 0.04	94.32 ± 0.11
PT	**95.34** ± 0.06	**94.55** ± 0.03

Table 4: Results for CON. $\mathcal{S}$ denotes S2S.

Model	UAS	LAS
Wiseman and Rush (2016)$^{\mathcal{S}}$	91.17	87.41
Zhang et al. (2017)$^{\mathcal{S}}$	93.71	91.60
Li et al. (2018)$^{\mathcal{S}}$	94.11	92.08
Mrini et al. (2020)	97.42	96.26
LS	92.83 ± 0.43	90.50 ± 0.53
LT	95.79 ± 0.07	93.17 ± 0.16
PT	**95.91** ± 0.06	**94.31** ± 0.09

(a) PTB results for DEP.

Model	UAS	LAS
He and Choi (2021b)	95.92 ± 0.02	94.24 ± 0.03
LS	86.54 ± 0.12	83.84 ± 0.13
LT	94.15 ± 0.14	91.27 ± 0.19
PT	**94.51** ± 0.22	**92.81** ± 0.21

(b) OntoNotes results for DEP.

Table 5: Results for DEP. $\mathcal{S}$ denotes S2S.

5.4 Dependency Parsing

The constituency trees from PTB and OntoNotes are converted into the Stanford dependencies v3.3.0 (De Marneffe and Manning, 2008) for DEP experiments. Forty and 1 non-projective trees are removed from the training and development sets of PTB 3, respectively. For OntoNotes 5, these numbers are 262 and 28. Test sets are not affected.

As shown in Table 5, textual tokens are crucial in learning arc-standard transitions using S2S, leading to +2.6 and +7.4 LAS improvements, respectively. Although our PT method underperforms recent state-of-the-art methods, it has the strongest performance among all S2S approaches. Interestingly, our S2S model manages to learn a transition system without explicitly modeling the stack, the buffer, the partial parse, or pointers.

We believe that the performance of DEP with S2S can be further improved with a larger and more recent pretrained S2S model and dynamic oracle (Goldberg and Nivre, 2012).

Model	PTB	OntoNotes
LS	97.51 ± 0.11	98.21 ± 0.02
w/o CD	97.51 ± 0.11	98.21 ± 0.02
LT	97.70 ± 0.02	98.40 ± 0.01
w/o CD	97.67 ± 0.02	98.39 ± 0.01
PT	97.64 ± 0.01	98.37 ± 0.02
w/o CD	97.55 ± 0.02	98.29 ± 0.05

(a) Accuracy of ablation tests for POS.

Model	CoNLL 03	OntoNotes 5
LS	70.29 ± 0.70	84.61 ± 1.18
w/o CD	66.33 ± 0.73	84.57 ± 1.16
LT	92.75 ± 0.03	89.60 ± 0.06
w/o CD	92.72 ± 0.02	89.50 ± 0.07
PT	93.18 ± 0.04	90.33 ± 0.04
w/o CD	93.12 ± 0.06	90.23 ± 0.05

(b) F1 of ablation tests for NER.

Model	PTB	OntoNotes
LS	90.50 ± 0.53	83.84 ± 0.13
w/o CD	90.45 ± 0.47	83.78 ± 0.13
LT	93.17 ± 0.16	91.27 ± 0.19
w/o CD	93.12 ± 0.14	91.05 ± 0.20
PT	94.31 ± 0.09	92.81 ± 0.21
w/o CD	81.50 ± 0.27	81.76 ± 0.36

(c) LAS of ablation tests for DEP.

Table 6: Ablation test results.

6 Analysis

6.1 Ablation Study

We perform an ablation study to show the performance gain of our proposed constrained decoding algorithms on different tasks. Constrained decoding algorithms (CD) are compared against free generation (w/o CD) where a model freely generates an output sequence that is later post-processed into task-specific structures using string-matching rules. Invalid outputs are patched to the greatest extent, e.g., POS label sequences are padded or truncated. As shown in Table 6, ablation of constrained decoding seldom impacts the performance of LS on all tasks, suggesting that the decoder of seq2seq can acclimatize to the newly added label

tokens. Interestingly, the less performant NER-LS model degrades the most, promoting the necessity of constrained decoding for weaker seq2seq models. The performance of LT on all tasks is marginally degraded when constrained decoding is ablated, indicating the decoder begins to generate structurally invalid outputs when textual tokens are freely generated. This type of problem seems to be exacerbated when more tokens are freely generated in the PT schemas, especially for the DEP-PT.

Unlike POS and NER, DEP is more prone to hallucinated textual tokens as early errors in the transition sequence get accumulated in the arc-standard system which shifts all later predictions off the track. It is not yet a critical problem as LS generates no textual tokens while a textual token in LT still serves as a valid shift action even if it is hallucinated. However, a hallucinated textual token in PT is catastrophic as it could be part of any arc-standard transitions. As no explicit shift transition is designed, a hallucinated token could lead to multiple instances of missing shifts in Algorithm 12.

6.2 Case Study

To facilitate understanding and comparison of different models, a concrete example of input (**I**), gold annotation (**G**), and actual model prediction per each schema is provided below for each task. Wrong predictions and corresponding ground truth are highlighted in red and teal, respectively.

POS In the following example, only PT correctly detects the past tense (VBD) of ‘‘put’’.

I: The word I put in boldface is extremely interesting.
G: DT NN PRP VBD IN NN VBZ RB JJ .
LS: DT NN PRP VBP IN NN VBZ RB RB JJ
LT: The/DT word/NN I/PRP put/VBP in/IN boldface/NN is/VBZ extr./RB interesting/JJ./.
PT: ‘‘The’’ is a determiner; ‘‘word’’ is a singular noun; ‘‘I’’ is a personal pronoun; ‘‘put’’ is a past tense verb; ‘‘in’’ is a preposition or subordinating conjunction; ‘‘boldface’’ is a singular noun; ‘‘is’’ is a 3rd person singular present verb; ‘‘extremely’’ is an adverb; ‘‘interesting’’ is an adjective; ‘‘.’’ is a period.

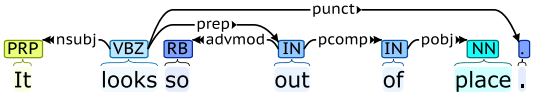
NER In the following example, LS and LT could not correctly recognize ‘‘HIStory’’ as an art work, possibly due to its leading uppercase letters.

I: Large image of the Michael Jackson HIStory statue.
G: Large image of the Michael Jackson HIStory statue.
PERSON(PER) WOA
LS: O O O O B-PER E-PER S-ORG O O O
LT: Large image of the <PERSON>Michael Jackson </PERSON> HIStory statue.
PT: ‘‘Michael Jackson’’ is a person; ‘‘HIStory’’ is an art work.

CON As highlighted with strikeout text below, LS and LT failed to parse ‘‘how much’’ as a wh-noun phrase and a wh-adverb phrase, respectively.

I: It’s crazy how much he eats.
G: (S (NP It)) (VP ’s (ADJP crazy) (SBAR (WHNP (WHADJP how much)) (S (NP he) (VP eats)))) .)
LS: N-S N-NP N-NP SH RE RE N-VP SH N-ADJP SH RE N-SBAR ~~N-WHNP~~ N-WHADVP SH SH RE ~~RE~~ N-S N-NP SH RE N-VP SH RE RE RE RE SH RE
LT: (S (NP (NP It)) (VP ’s (ADJP crazy) (SBAR (WHNP ~~(WHADJP~~ how much~~)~~) (S (NP he) (VP eats)))) .)
PT: a sentence has a simple clause, which has a noun phrase and a verb phrase and ‘‘.’’; the noun phrase has a noun phrase ‘‘It’’, the verb phrase has ‘‘’s’’ and an adjective phrase ‘‘crazy’’ and a subordinating clause, which has a wh-noun phrase and a simple clause; the wh-noun phrase has a wh-adjective phrase ‘‘how much’’, the simple clause has a noun phrase ‘‘he’’ and a verb phrase ‘‘eats’’.

DEP In the following example, LS incorrectly attached ‘‘so out of’’ to ‘‘place’’, and LT wrongly attached ‘‘so’’ to ‘‘looks’’.

I: It looks so out of place.
G:


LS: SH SH LA-nsubj SH SH SH SH LA-advmod LA-advmod LA-advmod RA-acomp SH RA-punct RA-root
LT: It looks LA-nsubj so out of place RA-pobj RA-pcomp RA-prep RA-ccomp . RA-punct RA-root
PT: ‘‘It’’ is a nominal subject of ‘‘looks’’; ‘‘so’’ is an adverbial modifier of ‘‘out’’; ‘‘of’’ has an object of a preposition ‘‘place’’; ‘‘out’’ has a prepositional complement ‘‘of’’; ‘‘looks’’ has a prepositional modifier ‘‘out’’; ‘‘looks’’ has a punctuation ‘‘.’’; ‘‘sentence’’ has a root ‘‘looks’’.

6.3 Design Choices

In the interest of experimentally comparing the schema variants, we would like each design we

Model	PTB 3	OntoNotes 5
POS-PT	**97.64** ± 0.01	**98.37** ± 0.02
dec.LEX	97.63 ± 0.02	98.35 ± 0.03
DEP-PT	**94.31** ± 0.09	**92.81** ± 0.21
dec.LEX	93.89 ± 0.18	91.19 ± 0.86

Table 7: Study of lexicality on POS and DEP.

Model	CoNLL 03	OntoNotes 5
NER-PT	**93.18** ± 0.04	**90.33** ± 0.04
inc.VRB	92.47 ± 0.03	89.63 ± 0.23
Model	PTB 3	OntoNotes 5
CON-PT	**95.34** ± 0.06	**94.55** ± 0.03
inc.VRB	95.19 ± 0.06	94.02 ± 0.49

Table 8: Study of verbosity on NER and CON.

consider to be equivalent in some systematic way. To this end, we fix other aspects and variate two dimensions of the prompt design, lexicality, and verbosity, to isolate the impact of individual variables.

Lexicality We call the portion of textual tokens in a sequence its lexicality. Thus, LS and PT have zero and full lexicality, respectively, while LT falls in the middle. To tease apart the impact of lexicality, we substitute the lexical phrases with corresponding tag abbreviations in PT on POS and DEP, e.g., ‘‘friend’’ is a *noun* → ‘‘friend’’ is a NN, ‘‘friend’’ is a *nominal subject* of ‘‘bought’’ → ‘‘friend’’ is a nsubj of ‘‘bought’’. Tags are added to the BART vocabulary and learned from scratch as LS and LT. As shown in Table 7, decreasing the lexicality of PT marginally degrades the performance of S2S on POS. On DEP, the performance drop is rather significant. Similar trends are observed comparing LT and LS in Section 5, confirming that lexicons play an important role in prompt design.

Verbosity Our PT schemas on NER and CON are designed to be as concise as human narrative, and as easy for S2S to generate. Another design choice would be as verbose as some LS and LT schemas. To explore this dimension, we increase the verbosity of NER-PT and CON-PT by adding ‘‘isn’t an entity’’ for all non-entity tokens and substituting each ‘‘which’’ to its actual referred phrase, respectively. The results are presented in Table 8. Though increased verbosity would eliminate any ambiguity, unfortunately, it hurts performance. Emphasizing a token ‘‘isn’t an entity’’ might encounter the over-confidence issue as the boundary annotation might be ambiguous in gold NER data (Zhu and Li, 2022). CON-PT deviates from human language style when reference is forbidden, which eventually makes it lengthy and hard to learn.

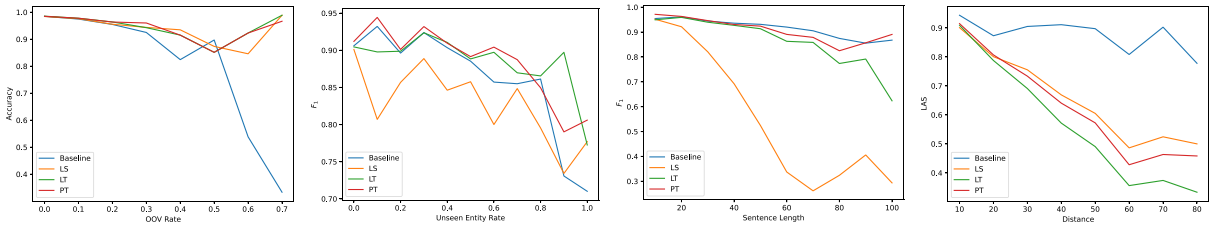
6.4 Stratified Analysis

Section 5 shows that our S2S approach performs comparably to most ad-hoc models. To reveal its pros and cons, we further partition the test data using task-specific factors and run tests on them. The stratified performance on OntoNotes 5 is compared to the strong BERT baseline (He and Choi, 2021b), which is representative of non-S2S models implementing many state-of-the-art decoders.

For POS, we consider the rate of Out-Of-Vocabulary tokens (OOV, tokens unseen in the training set) in a sentence as the most significant factor. As illustrated in Figure 1a, the OOV rate degrades the baseline performance rapidly, especially when over half tokens in a sentence are OOV. However, all S2S approaches show strong resistance to OOV, suggesting that our S2S models unleash greater potential through transfer learning.

For NER, entities unseen during training often confuse a model. This negative impact can be observed on the baseline and LT in Figure 1b. However, the other two schemas generating textual tokens, LT and PT, are less severely impacted by unseen entities. It further supports the intuition behind our approach and agrees with the finding by Shin et al. (2021): With the output sequence being closer to natural language, the S2S model has less difficulty generating it even with unseen entities.

Since the number of binary parses for a sentence of $n + 1$ tokens is the nth *Catalan Number* (Church and Patil, 1982), the length is a crucial factor for CON. As shown in Figure 1c, all models, especially LS, perform worse when the sentence gets longer. Interestingly, by simply recalling all the lexicons, LT easily regains the ability to parse long sentences. Using an even more natural representation, PT outperforms them with a performance on par with the strong baseline. It again



(a) OOV for POS (b) Unseen entities for NER (c) Length for CON (d) Distance for DEP

Figure 1: Factors impacting each task: the rate of OOV tokens for POS, the rate of unseen entities for NER, the sentence length for CON, and the head-dependent distance for DEP.

supports our intuition that natural language is beneficial for pretrained S2S.

For DEP, the distance between each dependent and its head is used to factorize the overall performance. As shown in Figure 1d, the gap between S2S models and the baseline increases with head-dependent distance. The degeneration of relatively longer arc-standard transition sequences could be attributed to the static oracle used in finetuning.

Comparing the three schemas across all subgroups, LT uses the most special tokens but performs the worst, while PT uses zero special tokens and outperforms the rest two. It suggests that special tokens could harm the performance of the pretrained S2S model as they introduce a mismatch between pretraining and finetuning. With zero special tokens, PT is most similar to natural language, and it also introduces no extra parameters in finetuning, leading to better performance.

7 Conclusion

We aim to unleash the true potential of S2S models for sequence tagging and structure parsing. To this end, we develop S2S methods that rival state-of-the-art approaches more complicated than ours, without substantial task-specific architecture modifications. Our experiments with three novel prompting schemas on four core NLP tasks demonstrated the effectiveness of natural language in S2S outputs. Our systematic analysis revealed the pros and cons of S2S models, appealing for more exploration of structure prediction with S2S.

Our proposed S2S approach reduces the need for many heavily engineered task-specific architectures. It can be readily extended to multi-task and few-shot learning. We have a vision of S2S playing an integral role in more language understanding and generation systems. The limitation of our approach is its relatively slow decoding speed due to serial generation. This issue can be mitigated with non-autoregressive generation and model compression techniques in the future.

Acknowledgments

We would like to thank Emily Pitler, Cindy Robinson, Ani Nenkova, and the anonymous TACL reviewers for their insightful and thoughtful feedback on the early drafts of this paper.

References

Alan Akbik, Tanja Bergmann, and Roland Vollgraf. 2019. Pooled contextualized embeddings for named entity recognition. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 724–728, Minneapolis, Minnesota. Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-1078

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual String Embeddings for Sequence Labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, COLING'18, pages 1638–1649.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7–9, 2015, Conference Track Proceedings*.

Xuefeng Bai, Yulong Chen, and Yue Zhang. 2022. Graph pre-training for AMR parsing

and generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6001–6015, Dublin, Ireland. Association for Computational Linguistics.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1415–1425, Baltimore, Maryland. Association for Computational Linguistics. https://doi.org/10.3115/v1/P14-1133

Michele Bevilacqua, Rexhina Blloshmi, and Roberto Navigli. 2021. One spring to rule them both: Symmetric AMR semantic parsing and generation without a complex pipeline. In *Proceedings of AAAI*. https://doi.org/10.1609/aaai.v35i14.17489

Bernd Bohnet, Ryan McDonald, Gonçalo Simões, Daniel Andor, Emily Pitler, and Joshua Maynez. 2018. Morphosyntactic tagging with a Meta-BiLSTM model over context sensitive token encodings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, ACL'18, pages 2642–2652. https://doi.org/10.18653/v1/P18-1246

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Jiawei Chen, Qing Liu, Hongyu Lin, Xianpei Han, and Le Sun. 2022. Few-shot named entity recognition with self-describing networks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5711–5722, Dublin, Ireland. Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.acl-long.392

Lingzhen Chen and Alessandro Moschitti. 2018. Learning to progressively recognize new named entities with sequence to sequence models. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2181–2191, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Jason Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics*, 4:357–370. https://doi.org/10.1162/tacl_a_00104

Chinmay Choudhary and Colm O'riordan. 2021. End-to-end mBERT based seq2seq enhanced dependency parser with linguistic typology knowledge. In *Proceedings of the 17th International Conference on Parsing Technologies and the IWPT 2021 Shared Task on Parsing into Enhanced Universal Dependencies (IWPT 2021)*, pages 225–232, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.iwpt-1.24

Kenneth Church and Ramesh Patil. 1982. Coping with syntactic ambiguity or how to put the block in the box on the table. *American Journal of Computational Linguistics*, 8(3–4):139–149.

Kevin Clark, Minh-Thang Luong, Christopher D. Manning, and Quoc Le. 2018. Semi-supervised sequence modeling with cross-view training. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1914–1925, Brussels, Belgium. Association for Computational Linguistics. https://doi.org/10.18653/v1/D18-1217

Leyang Cui, Yu Wu, Jian Liu, Sen Yang, and Yue Zhang. 2021. Template-based named entity recognition using BART. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1835–1845, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.findings-acl.161

Marie-Catherine De Marneffe and Christopher D. Manning. 2008. The Stanford typed dependencies representation. In *COLING 2008: Proceedings of the Workshop on Cross-framework and Cross-domain Parser Evaluation*, pages 1–8. https://doi.org/10.3115/1608858.1608859

Daniel Deutsch, Shyam Upadhyay, and Dan Roth. 2019. A general-purpose algorithm for constrained sequential inference. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 482–492, Hong Kong, China. Association for Computational Linguistics. https://doi.org/10.18653/v1/K19-1045

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Daniel Fernández-González and Carlos Gómez-Rodríguez. 2020. Enriched in-order linearization for faster sequence-to-sequence constituent parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4092–4099, Online. Association for Computational Linguistics.

Abbas Ghaddar and Phillippe Langlais. 2018. Robust lexical features for improved neural network named-entity recognition. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1896–1907, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Yoav Goldberg and Joakim Nivre. 2012. A dynamic oracle for arc-eager dependency parsing. In *Proceedings of COLING 2012*, pages 959–976, Mumbai, India. The COLING 2012 Organizing Committee.

Tatsunori B. Hashimoto, Kelvin Guu, Yonatan Oren, and Percy S. Liang. 2018. A retrieve-and-edit framework for predicting structured outputs. *Advances in Neural Information Processing Systems*, 31.

Han He and Jinho Choi. 2020. Establishing strong baselines for the new decade: Sequence tagging, syntactic and semantic parsing with bert. In *The Thirty-Third International Flairs Conference*.

Han He and Jinho D. Choi. 2021a. Levi graph AMR parser using heterogeneous attention. In *Proceedings of the 17th International Conference on Parsing Technologies and the IWPT 2021 Shared Task on Parsing into Enhanced Universal Dependencies (IWPT 2021)*, pages 50–57, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.iwpt-1.5

Han He and Jinho D. Choi. 2021b. The stem cell hypothesis: Dilemma behind multi-task learning with transformer encoders. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5555–5577, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.emnlp-main.451

Chris Hokamp and Qun Liu. 2017. Lexically constrained decoding for sequence generation using grid beam search. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1535–1546, Vancouver, Canada. Association for Computational Linguistics. https://doi.org/10.18653/v1/P17-1141

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12–22, Berlin, Germany. Association for Computational Linguistics. https://doi.org/10.18653/v1/P16-1002

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, pages 282–289.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016*

Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 260–270, San Diego, California. Association for Computational Linguistics. https://doi.org/10.18653/v1/N16-1030

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. 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*, pages 7871–7880, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.703

Peng-Hsuan Li, Ruo-Ping Dong, Yu-Siang Wang, Ju-Chieh Chou, and Wei-Yun Ma. 2017. Leveraging linguistic structures for named entity recognition with bidirectional recursive neural networks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2664–2669, Copenhagen, Denmark. Association for Computational Linguistics.

Zuchao Li, Jiaxun Cai, Shexia He, and Hai Zhao. 2018. Seq2seq dependency parsing. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3203–3214, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Jiangming Liu and Yue Zhang. 2017. In-order transition-based constituent parsing. *Transactions of the Association for Computational Linguistics*, 5:413–424. https://doi.org/10.1162/tacl_a_00070

Chunpeng Ma, Lemao Liu, Akihiro Tamura, Tiejun Zhao, and Eiichiro Sumita. 2017. Deterministic attention for sequence-to-sequence constituent parsing. In *Thirty-First AAAI Conference on Artificial Intelligence*. https://doi.org/10.1609/aaai.v31i1.10967

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330. https://doi.org/10.21236/ADA273556

Khalil Mrini, Franck Dernoncourt, Quan Hung Tran, Trung Bui, Walter Chang, and Ndapa Nakashole. 2020. Rethinking self-attention: Towards interpretability in neural parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 731–742, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.findings-emnlp.65

Joakim Nivre. 2004. Incrementality in deterministic dependency parsing. In *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*, pages 50–57, Barcelona, Spain. Association for Computational Linguistics. https://doi.org/10.3115/1613148.1613156

Giovanni Paolini, Ben Athiwaratkun, Jason Krone, Jie Ma, Alessandro Achille, RISHITA ANUBHAI, Cicero Nogueira dos Santos, Bing Xiang, and Stefano Soatto. 2021. Structured prediction as translation between augmented natural languages. In *International Conference on Learning Representations*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics. https://doi.org/10.18653/v1/N18-1202

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards Robust Linguistic Analysis using OntoNotes. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 143–152, Sofia, Bulgaria. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael

Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Kenji Sagae and Alon Lavie. 2005. A classifier-based parser with linear run-time complexity. In *Proceedings of the Ninth International Workshop on Parsing Technology*, pages 125–132, Vancouver, British Columbia. Association for Computational Linguistics. https://doi.org/10.3115/1654494.1654507

Richard Shin, Christopher Lin, Sam Thomson, Charles Chen, Subhro Roy, Emmanouil Antonios Platanios, Adam Pauls, Dan Klein, Jason Eisner, and Benjamin Van Durme. 2021. Constrained language models yield few-shot semantic parsers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7699–7715, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.emnlp-main.608

Jana Straková, Milan Straka, and Jan Hajic. 2019. Neural architectures for nested NER through linearization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5326–5331, Florence, Italy. Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1527

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems*, 27.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147. https://doi.org/10.3115/1119176.1119195

Bo-Hsiang Tseng, Jianpeng Cheng, Yimai Fang, and David Vandyke. 2020. A generative model for joint natural language understanding and generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1795–1807, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.163

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015a. Pointer networks. In *Advances in Neural Information Processing Systems*, volume 28.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015b. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Xinyu Wang, Yong Jiang, Nguyen Bach, Tao Wang, Zhongqiang Huang, Fei Huang, and Kewei Tu. 2021. Automated concatenation of embeddings for structured prediction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2643–2660, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.acl-long.206

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, and Mohammed El-Bachouti, Robert Belvin, and Ann Houston. 2013. Ontonotes release 5.0 ldc2013t19. *Linguistic Data Consortium, Philadelphia, PA*.

Sam Wiseman and Alexander M. Rush. 2016. Sequence-to-sequence learning as beam-search optimization. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1296–1306, Austin, Texas. Association for Computational Linguistics. https://doi.org/10.18653/v1/D16-1137

Lu Xu, Zhanming Jie, Wei Lu, and Lidong Bing. 2021. Better feature integration for named entity recognition. In *Proceedings of the 2021 Conference of the North American Chapter of the*

Association for Computational Linguistics: Human Language Technologies, pages 3457–3469, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.naacl-main.271

Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. 2020. LUKE: Deep contextualized entity representations with entity-aware self-attention. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6442–6454, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.emnlp-main.523

Hang Yan, Tao Gui, Junqi Dai, Qipeng Guo, Zheng Zhang, and Xipeng Qiu. 2021. A unified generative framework for various NER subtasks. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5808–5822, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.acl-long.451

Songlin Yang and Kewei Tu. 2022. Bottom-up constituency parsing and nested named entity recognition with pointer networks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2403–2416, Dublin, Ireland. Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.acl-long.171

Deming Ye, Yankai Lin, Peng Li, and Maosong Sun. 2022. Packed levitated marker for entity and relation extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4904–4917, Dublin, Ireland. Association for Computational Linguistics.

Juntao Yu, Bernd Bohnet, and Massimo Poesio. 2020. Named entity recognition as dependency parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6470–6476, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.577

Zhirui Zhang, Shujie Liu, Mu Li, Ming Zhou, and Enhong Chen. 2017. Stack-based multi-layer attention for transition-based dependency parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1677–1682, Copenhagen, Denmark. Association for Computational Linguistics. https://doi.org/10.18653/v1/D17-1175

Enwei Zhu and Jinpeng Li. 2022. Boundary smoothing for named entity recognition. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7096–7108, Dublin, Ireland. Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.acl-long.490

Huiming Zhu, Chunhui He, Yang Fang, and Weidong Xiao. 2020. Fine grained named entity recognition via seq2seq framework. *IEEE Access*, 8:53953–53961. https://doi.org/10.1109/ACCESS.2020.2980431

Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. 2013. Fast and accurate shift-reduce constituent parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 434–443, Sofia, Bulgaria. Association for Computational Linguistics.