Compositional Generalization for Neural Semantic Parsing via Span-level Supervised Attention

Pengcheng Yin♠[*], **Hao Fang**♣, **Graham Neubig**♠, **Adam Pauls**♣,
Emmanouil Antonios Platanios♣, **Yu Su**♣, **Sam Thomson**♣, **Jacob Andreas**♣
♠Carnegie Mellon University ♣Microsoft Semantic Machines
{pcyin,gneubig}@cs.cmu.edu
{hao.fang,adam.pauls,anthony.platanios,
yusu2,samuel.thomson,jacob.andreas}@microsoft.com

Abstract

We describe a span-level supervised attention loss that improves compositional generalization in semantic parsers. Our approach builds on existing losses that encourage attention maps in neural sequence-to-sequence models to imitate the output of classical word alignment algorithms. Where past work has used word-level alignments, we focus on spans; borrowing ideas from phrase-based machine translation, we align subtrees in semantic parses to spans of input sentences, and encourage neural attention mechanisms to mimic these alignments. This method improves the performance of transformers, RNNs, and structured decoders on three benchmarks of compositional generalization.

1 Introduction

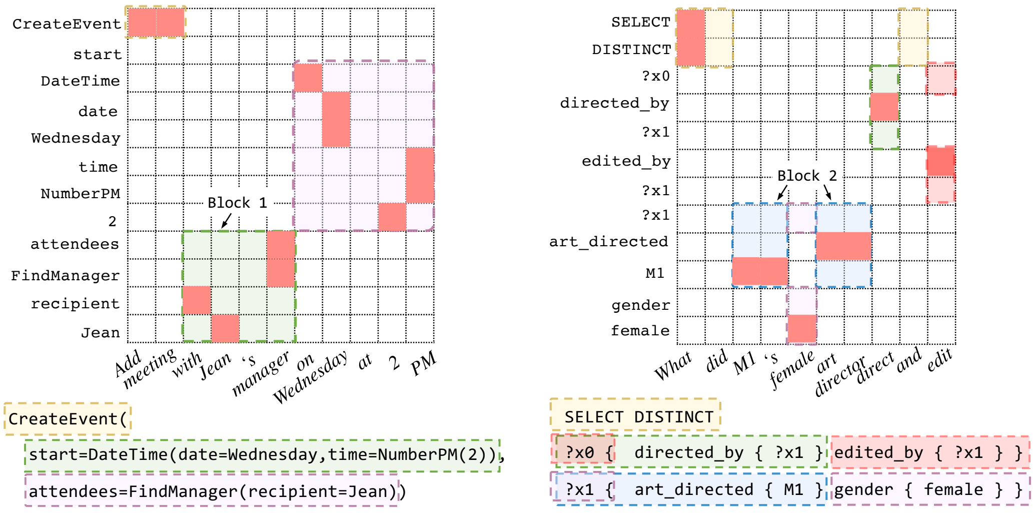
Semantic parsers translate natural language utterances (*e.g., Schedule a meeting with Jean*) into executable programs (*e.g.,* `CreateEvent(attendees=Jean)`), and play a crucial role in applications such as question answering systems and conversational agents (Liang, 2016; Gupta et al., 2018; Wen et al., 2017). As in many language understanding problems, a central challenge in semantic parsing is **compositional generalization** (Finegan-Dollak et al., 2018; Keysers et al., 2020). Consider a personal digital assistant for which developers have assembled separate collections of annotated utterances for user requests involving their calendars (*e.g., Schedule a meeting with Jean*) and their contact books (*e.g., Who is Jean's manager?*). An effective model should learn from this data how to additionally handle requests like *Schedule a meeting with Jean's manager*, composing skills from the calendar and contacts domains, with little or no supervision for such combinations.

Neural sequence-to-sequence models, which provide the foundation for state-of-the-art semantic parsers (Dong and Lapata, 2016; Yin and Neubig, 2017), tend to perform poorly at out-of-distribution generalization of this kind (Lake and Baroni, 2018; Furrer et al., 2020; Suhr et al., 2020). Methods have been proposed to bridge the generalization gap using meta-learning (Lake, 2019; Wang et al., 2020) or specialized model architectures (Russin et al., 2019; Li et al., 2019; Liu et al., 2020; Chen et al., 2020). These have registered impressive performance on small synthetic benchmark datasets, but it has proven difficult to effectively combine them with large-scale pre-training (Lewis et al., 2020; Raffel et al., 2020) and natural data (Furrer et al., 2020).

In contrast to this extensive literature on data transformations and model architectures, the design of **loss functions** to encourage compositional generalization has been under-explored. This paper investigates **attention supervision** losses that encourage attention matrices in neural sequence models to resemble the output of word alignment algorithms (Liu et al. (2016); Mi et al. (2016); Arthur et al. (2016); Lyu and Titov (2018), *inter alia*) as a source of inductive bias for compositional tasks. Previous work has found that aligning program tokens (*e.g.,* `FindManager` in Fig. 1) to natural language tokens (*manager*) improves model performance (Misra et al., 2018; Rabinovich et al., 2017; Goldman et al., 2018; Richardson et al., 2018; Herzig and Berant, 2020; Oren et al., 2020). However, the token-level alignments derived from off-the-shelf aligners are often noisy, and the correspondence between natural language and program tokens is not always a many-to-one map of the kind returned by standard alignment algorithms. On the other hand, programs also have explicit hierarchical structure, which could be useful to induce better attention regularizers (Wang et al., 2019). Here we investigate the use of **span-level alignments**, identifying sub-programs that should be predicted as a unit and aligning all tokens in a sub-program to a

[*] This work was mostly done during an internship at Microsoft Semantic Machines.

Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 2810–2823
June 6–11, 2021. ©2021 Association for Computational Linguistics



```
CreateEvent(
  start=DateTime(date=Wednesday,time=NumberPM(2)),
  attendees=FindManager(recipient=Jean))
```

```
SELECT DISTINCT
?x0 {  directed_by { ?x1 } edited_by { ?x1 } }
?x1 {  art_directed { M1 } gender { female } }
```

(a) Utterance-LISP Expression Alignments

(b) Utterance-SPARQL Alignments

Figure 1: Token and span level alignments (shown in $A^{\intercal}_{|\boldsymbol{u}|\times|\boldsymbol{z}|}$) between utterances and programs in LISP-style expressions (a) and SPARQL queries (b). Token alignments are marked in 🟥. Span-level alignments are marked using dashed bounding boxes (alignment to program sketch tokens are marked in ⬚). Programs in matrices are simplified for presentation. We use simplified SPARQL representation (Furrer et al., 2020) grouping relations (*e.g.,* `directed_by` and `edited_by`) by subjects (*e.g.,* `?x0`).

corresponding natural language span (Herzig and Berant, 2020).

We present a simple algorithm to derive span-level alignments from token-level alignments. Our approach is compatible with multiple models (RNNs, transformers, and structured tree-based decoders), pretrained or not. In experiments, span-based attention supervision consistently improves over token-level objectives, achieving strong results on three semantic parsing datasets featuring diverse formalisms and tests of generalization.

2 Span-level Supervised Attention

Neural Semantic Parsers A semantic parser maps a natural language (NL) utterance $\boldsymbol{u}$ to an executable program $\boldsymbol{z}$. In this paper, we consider neural parsers using token-based attentive decoders, in which $\boldsymbol{z}$ is predicted as a sequence of consecutive tokens $\{z_{j=1}^{|\boldsymbol{z}|}\}$ by attending to tokens in $\boldsymbol{u} = \{u_{i=1}^{|\boldsymbol{u}|}\}$. Examples include sequence-to-sequence models based on recurrent networks (Dong and Lapata, 2016; Jia and Liang, 2016) or transformers (Vaswani et al., 2017; Raffel et al., 2020), as well as structured parsing methods that predict a program following its syntactic structure (Dong and Lapata (2018), see §3 for more details).

Supervised Attention Existing *token-level* supervised attention approaches assume access to an alignment matrix $A_{|\boldsymbol{u}|\times|\boldsymbol{z}|}$ with entries $a_{i,j}$, where $a_{i,j} = 1$ *iff* the i-th source (utterance) token u_i is aligned to the j-th target (program) token z_j. $A_{|\boldsymbol{u}|\times|\boldsymbol{z}|}$ can be inferred using latent variable models (Brown et al., 1993; Och and Ney, 2003; Dyer et al., 2013). During training, when the decoder predicts a target token z_j, supervised attention encourages the target-to-source attention distribution $p_{\text{att}}(u_i|z_j)$ to match the prior alignment distribution $p_{\text{prior}}(u_i|z_j) = \frac{a_{i,j}}{\sum_k a_{k,j}}$, which is normalized by the number of source tokens aligned to z_j. We use a squared error loss (Liu et al., 2016):

$$\mathcal{L}_{\text{sup\_att}} = \frac{1}{|\boldsymbol{z}|}\sum_{j=1}^{|\boldsymbol{z}|}\sum_{i=1}^{|\boldsymbol{u}|}\left(p_{\text{att}}(u_i|z_j) - p_{\text{prior}}(u_i|z_j)\right)^2. \quad (1)$$

Previous work has also used a cross entropy loss (Rabinovich et al., 2017; Oren et al., 2020).

Sub-program-to-Span Alignment We present a simple heuristic algorithm to extract span-level alignments between programs and utterances from existing token-level results (Algo. 1). Fig. 1 illustrates example span-level alignments for two types of programs (LISP and simplified SPARQL). Similarly to Dong and Lapata (2018), we assume each program can be decomposed into a top-level **sketch** and a set of **sub-programs**.[1] For the LISP expression in Fig. 1a, the sketch contains the top-level function call (`CreateEvent(`$\boxed{?}$`,`$\boxed{?}$`)`) and sub-programs are named arguments paired with values

[1] Unlike D&L, we allow sub-programs to include non-consecutive (and possibly overlapping) spans of program tokens, *e.g.,* `{?x0 {edited_by {?x1}}` in Fig. 1b. We also permit non-disjoint sub-programs.

Algorithm 1: Span Alignment Extraction

```
input  : Utterance $\boldsymbol{u}$, program $\boldsymbol{z}$, token-level
         alignment matrix $A_{|\boldsymbol{u}|\times|\boldsymbol{z}|}$
output : Span-level alignment matrix $A^{\text{span}}_{|\boldsymbol{u}|\times|\boldsymbol{z}|}$
1  Initialize set $A_S = \emptyset$ to store span-level alignments
2  foreach sub-program $\boldsymbol{z}^{\text{s}}$ do
3      $T_{\boldsymbol{z}^{\text{s}}} = \{u_i | \exists z_j \in \boldsymbol{z}^{\text{s}}, a_{i,j} = 1\}, U_{\boldsymbol{z}^{\text{s}}} = \emptyset$
4      ▷ Case 1 (Consecutive Alignment):
5      $m = \min_i\{u_i \in T_{\boldsymbol{z}^{\text{s}}}\}, n = \max_i\{u_i \in T_{\boldsymbol{z}^{\text{s}}}\}$
6      $U_{\boldsymbol{z}^{\text{s}}} = \{\boldsymbol{u}_{m:n}\}$
7      ▷ Case 2 (Nonconsecutive Alignment):
8      foreach consecutive span $\boldsymbol{u}_{m:n} \subset T_{\boldsymbol{z}^{\text{s}}}$ do
9          Add utterance span $\boldsymbol{u}_{m:n}$ to $U_{\boldsymbol{z}^{\text{s}}}$
10
11     foreach $\boldsymbol{z}_{p:q} \in \boldsymbol{z}^{\text{s}}$, $\boldsymbol{u}_{m:n} \in U_{\boldsymbol{z}^{\text{s}}}$ do
12         Add span alignment $\boldsymbol{z}_{p:q} \leftrightarrow \boldsymbol{u}_{m:n}$ to $A_S$
   ▷ Generate sketch-utterance span alignments:
13 foreach unaligned span $\boldsymbol{z}_{p:q} \in \boldsymbol{z}$ and $\boldsymbol{u}_{m:n} \in \boldsymbol{u}$ do
14     Add span alignment $\boldsymbol{z}_{p:q} \leftrightarrow \boldsymbol{u}_{m:n}$ to $A_S$
15 Generate $A^{\text{span}}_{|\boldsymbol{u}|\times|\boldsymbol{z}|}$, such that $a^{\text{span}}_{i,j} = 1$ iff
     $\exists \boldsymbol{z}_{p:q} \leftrightarrow \boldsymbol{u}_{m:n} \in A_S, i \in [m,n], j \in [p,q]$
16 return $A^{\text{span}}_{|\boldsymbol{u}|\times|\boldsymbol{z}|}$
```

(`attendees=FindManager...`). For the SPARQL expression in Fig. 1b, sketches include the query form (*e.g.,* `SELECT DISTINCT`) and sub-programs hold individual `subject-relation-object` assertions (*e.g.,* `?x0 edited_by ?x1`).[2]

In this paper, we use these program decompositions to guide span-level alignment. The underlying intuition is that *every* token in a sketch or sub-program will get aligned to the *same* set of utterance tokens. Algo. 1 extracts such set of utterance spans aligned to a sub-program $\boldsymbol{z}^{\text{s}}$ from the set $T_{\boldsymbol{z}^{\text{s}}}$ of NL tokens that are aligned to tokens in $\boldsymbol{z}^{\text{s}}$ (line 3). We present two variants of this approach, depending on the properties of the dataset (§3). In the first case (lines 5-6), similar to bilingual phrase extraction in machine translation (MT; Och, 2002), we create a single consecutive utterance span $\boldsymbol{u}_{m:n}$ via the outer bound of the aligned utterance tokens in $T_{\boldsymbol{z}^{\text{s}}}$ (*e.g.,* Block 1, Fig. 1a). In the second variant (lines 8-9), we find internally contiguous utterance spans (subsequences) in $T_{\boldsymbol{z}^{\text{s}}}$ and align them to $\boldsymbol{z}^{\text{s}}$. For instance, the sub-program (`?x1 art_directed M1`) in Block 2 of Fig. 1b aligns to two utterance spans: *M1 's* and *art director*. While this case does not have an exact analog in MT, it is reminiscent of the model of Chiang (2005) which extracts translation rules with discontinuous phrase segments, and could be useful in capturing long-range alignments of utterance subsequences to sub-programs as in Block 2 (Andreas et al., 2013). Span-level alignments for a sub-program are then generated by pairing its program spans $\boldsymbol{z}_{p:q}$ (spans with consecutive program tokens) with all its aligned utterance spans (lines 11-12). Finally, we generate alignments for sketch spans in $\boldsymbol{z}$ by pairing them with any utterance tokens that have not yet been aligned to a sub-program (lines 13-14).

Algo. 1 leverages the explicit hierarchical structures of programs to generate alignments between sub-programs and utterance spans. Such an idea of using structural information for alignment extraction has deep roots in statistical syntax-based MT, which leverages the syntactic structure of sentences to generate alignments between parse trees and NL constituents (Galley et al., 2004; Chiang, 2005; Liu et al., 2006). Our approach is also broadly related to lexicon induction models in semantic parsers based on probabilistic CCG grammars (Kwiatkowski et al., 2011) or other formalisms (Jones et al., 2012), which learn mapping rules between logical form templates and utterance tokens.

3 Experiments

We evaluate span-level supervised attention on three benchmarks of compositional generalization.

SMCALFLOW Compositional Skills (SMCALFLOW-CS) is a new dataset created in this study based on the task-oriented dialogue corpus SMCALFLOW (Semantic Machines et al., 2020), featuring real-world human-generated utterances about calendar management. Like the motivating story in §1, we create training data for skills $\mathbb{S}$ involving event creation (*e.g., Schedule a meeting with Adam*) and organization structure (*e.g., Who's on Adam's team?*), while evaluating on examples $\mathbb{C}$ featuring compositional skills (*e.g., Add meeting with Adam and his team*). Utterances are annotated with LISP-style programs (Fig. 1a). Since zero-shot compositional generalization is highly non-trivial due to novel language patterns (*e.g., Adam and his team*) and program structures (*e.g.,* usage of `List(·)` to specify multiple attendees) in compositional examples, we consider a few-shot learning scenario, where a handful of compositional examples are included in the training set. Readers are referred to Appendix A for details of dataset construction.

Compositional Freebase Questions (CFQ) is a

[2] As we explain in Appendix B, such program decomposition could be easily generated using off-the-shelf syntax analyzers provided by the programming language.

$\lvert\mathbb{C}_{\text{train}}\rvert$	16		32		64		128	
Domain	$\mathbb{S}$	$\mathbb{C}$	$\mathbb{S}$	$\mathbb{C}$	$\mathbb{S}$	$\mathbb{C}$	$\mathbb{S}$	$\mathbb{C}$
BERT2SEQ	82.8 ±1.0	33.6 ±7.2	82.8 ±0.6	53.5 ±10.3	83.7 ±0.6	64.2 ±4.9	83.0 ±0.8	71.3 ±2.3
+TS (Token-level Sup.)	83.4 ±0.7	39.7 ±1.3	83.2 ±0.3	59.9 ±1.6	83.7 ±0.6	65.7 ±1.5	83.4 ±0.4	73.2 ±0.7
+SS (Span-level Sup.)	83.9 ±0.2	**46.8** ±1.2	83.5 ±0.7	**61.7** ±2.2	83.6 ±0.7	**66.9** ±1.0	83.5 ±0.9	**74.3** ±0.7
COARSE2FINE (DL18)	83.0 ±1.0	40.6 ±7.0	83.6 ±0.6	54.6 ±6.8	83.8 ±0.3	65.7 ±3.2	83.4 ±1.2	72.9 ±0.6
+TS (Token-level Sup.)	83.7 ±0.5	44.6 ±1.5	83.1 ±1.0	60.7 ±2.5	83.7 ±0.8	67.1 ±1.4	83.3 ±0.7	74.1 ±0.9
+SS (Span-level Sup.)	83.8 ±0.4	**47.4** ±2.1	83.7 ±1.0	**61.9** ±1.8	83.0 ±0.8	67.5 ±1.4	83.5 ±0.8	**75.0** ±1.2

Table 1: TEST. accuracies on the SMCALFLOW-CS Compositional Skills dataset w.r.t. the size of compositional examples included in the training set. We report both the results on the in-domain single-skill examples ($\mathbb{S}$) as well as the generalized multi-skill examples ($\mathbb{C}$). Results are averaged over five random random seeds. **Bold** results have p-values ≤ 0.05 when comparing to other systems in the same category under a permutation test.

challenging compositional generalization dataset of $130K$ synthetic utterances with SPARQL queries (Fig. 1b). Training and evaluation splits are constructed such that they have different distributions of compositional structures, while the distributions of atomic language (*e.g., director*) and program (*e.g.,* `film.director`) constructs remain similar (Keysers et al., 2020).

ATIS Text-to-SQL is a dataset of 3,809 SQL-annotated utterances about flight querying (*e.g., Flights from Seattle to Austin.*). We follow Oren et al. (2020) and use the **query split** (Finegan-Dollak et al., 2018), where training and evaluation programs do not overlap at template level.

Models We apply span-level supervised attention to strong neural models on each dataset. We evaluate two systems on SMCALFLOW-CS: BERT2SEQ, a sequence-to-sequence model with a BERT encoder and an LSTM decoder using copy mechanism, and COARSE2FINE (Dong and Lapata, 2018), which uses (a BERT encoder and) a structured decoder that factorizes the generation of a program into sketch and value predictions. On CFQ, we use T5-BASE (Raffel et al., 2020), and apply attention supervision on all the cross-attention heads in the last decoder layer. For ATIS, we take the best system from Oren et al. (2020) that is tuned for better generalization on this dataset, which is a sequence-to-sequence model with an ELMO encoder and coverage-based attention mechanism (See et al., 2017).

We extract word alignments using IBM Model 4 in `GIZA++` (Och and Ney, 2003), and canonicalize programs (*e.g.,* remove parentheses) to improve alignment quality. To extract span-level alignments, we use consecutive alignments (Case 1) in Algo. 1 for SMCALFLOW-CS and ATIS, as those datasets feature simple one-to-one mapping between sub-programs and utterance spans. For CFQ, we use nonconsecutive alignments (Case 2) to handle assertions aligned to disjoint NL spans (Fig. 1b). We apply Eq. (1) during model optimization using either the token and span level alignment matrix for token (+**TS**) and span (+**SS**) level supervised attention, respectively. See Appendix B for details.

3.1 Results

Tab. 1 lists the evaluation results on SMCALFLOW-CS with varying numbers of compositional examples in the training set ($\mathbb{C}_{\text{train}}$).[3] We report accuracies on both the in-domain single-skill examples ($\mathbb{S}$) as well as on the generalized compositional-skill examples ($\mathbb{C}$). Both methods improve compositional generalization for BERT2SEQ and COARSE2FINE, while span-level supervised attention is more effective. Intuitively, span-level alignments could better capture the correspondence between sub-structures in utterances and programs, helping the parser to correctly predict such sub-programs in compositionally novel contexts by focusing on the corresponding utterance span. Interestingly, in such a low-resource learning scenario with only a handful of training compositional samples, span-level supervised attention offers more gains in extreme low-resource settings ($\lvert\mathbb{C}_{\text{train}}\rvert = 16$), outperforming the base BERT2SEQ model by 13% absolute (33.6% v.s. 46.8% for BERT2SEQ).

Indeed, we found that more *alignment-like* attentions are associated with more accurate model predictions. For a BERT2SEQ model with span-level supervision trained on $\lvert\mathbb{C}_{\text{train}}\rvert = 64$, when predicting sub-programs for the `attendees` argument (*e.g.,* `attendees=FindManager(recipient=self)`) on compositional samples in $\mathbb{C}$, the model achieves 86% sub-program accuracy if it assigns a time-step

[3] We ran `GIZA++` and extracted span-level alignments for each training split separately.

Split	MCD$_1$			MCD$_2$			MCD$_3$			Average
	C	R	All	C	R	All	C	R	All	
T5-BASE	**55.8** ±4.8	77.4 ±4.7	**62.4** ±4.5	34.8 ±2.9	29.4 ±2.5	33.0 ±2.4	**21.6** ±8.6	34.4 ±2.8	23.0 ±1.7	39.5
+ TS	44.9 ±4.7	**86.4** ±2.4	57.7 ±3.4	32.4 ±3.1	32.7 ±1.4	32.5 ±2.1	14.3 ±1.5	36.6 ±1.7	22.0 ±0.7	37.4
+ SS	48.2 ±4.4	80.5 ±2.2	58.2 ±2.8	34.8 ±2.3	**36.4** ±2.8	**35.4** ±1.6	14.6 ±2.1	**40.1** ±3.5	23.8 ±1.0	39.1

Table 2: Mean Test Accuracies on CFQ MCD splits with 95% confidence interval, for **C**onjunctive, **R**ecursive, and **A**ll the samples. The last column lists averaged accuracies for the three splits. **Bold** results have p-values ≤ 0.01 when comparing to other systems in the same category.

Model	*Query Split*		*i.i.d. Split*	
	DEV.	TEST.	DEV.	TEST.
Oren et al. (2020)	28.9	34.4	78.4	74.5
+ Token-level Sup.	31.2 ±1.2	34.5 ±0.9	76.7 ±0.6	72.5 ±1.6
+ Span-level Sup.	31.1 ±0.6	35.0 ±2.0	78.4 ±0.8	74.0 ±0.5

Table 3: Accuracies and standard deviation on the ATIS text-to-SQL query (program template) and standard i.i.d. split splits. Results averaged over five random runs.

average of at least 90% of its attention weights over the aligned utterance spans (*e.g., with my manager*) identified by Algo. 1. Otherwise, the accuracy drops to 70% (more in Appendix C.1).

Moreover, supervised attention may be a sufficient *substitute* for structured model architectures in some cases. Despite the unstructured BERT2SEQ model's generally inferior performance without supervised attention, it matches the accuracies of COARSE2FINE when both models are trained with span-level supervision.[4] We also remark that span-based supervision maintains or improves performance on in-domain single-skill examples ($\mathbb{S}$). For instance, the accuracy for BERT2SEQ increases from 82.8% to 83.9% when $|\mathbb{C}_{\text{train}}| = 16$.

Next, on CFQ (Tab. 2), we report break-down results based on the syntactic types of questions: **R**ecursive questions with chained multi-hop relations (*e.g.,* $\boldsymbol{u}_r$:*Was M1 influenced by a German writer?*), and **C**onjunctive ones with only conjunctions of entities and relations and without chained relations (*e.g.,* $\boldsymbol{u}_c$:*Was M1 directed and edited by M2 and M3?*). While supervised attention is effective on recursive questions, it struggles on conjunctive ones. This may be because the model learns to attend to discontinuous utterance spans (*e.g.,* "*M1 directed*" and "*M2 and M3*" in $\boldsymbol{u}_c$) when predicting a relation (*e.g.,* `directed_by`) in a conjunction, which could be more sensitive to alignment errors. Additionally, utterance spans aligned to a sub-program in conjunctive questions are usually longer and more complex (*e.g.,* having multiple conjunctive entity mentions like *Did M1 write M2, M3, M4, and M5?*), which might require more fine-grained supervision than uniformly treating every aligned utterance tokens equally as in Eq. (1). More analysis is in Appendix C.2.

Finally, we present the results on the ATIS query splits in Tab. 3, where span-level supervision is comparable with token-level one, further improving upon an already-strong model that targets for compositional generalization (ELMO with coverage based attention). Interestingly, token-level supervised attention is slightly worse than the baseline model on the standard *i.i.d.* splits, while span-level supervision does not offer further improvements. Empirically we observe that the utterance-SQL alignments in ATIS are much noisier than other two datasets due to redundant structures in SQL queries (*e.g.,*`Join` statements with intermediary tables), whose aligned NL constituents are often not well defined (See Appendix B for more details).

4 Conclusion

This paper demonstrated the effectiveness of span-level supervised attention as a simple and flexible tool for improving neural sequence models in a diverse set of architectures and tests of generalization. Future work might explore applications to other prediction tasks and joint learning of alignments with sequence model parameters.

Acknowledgements

We thank the Semantic Machines team and anonymous reviewers for their valuable feedbacks. Pengcheng Yin was supported in part by an IBM Ph.D. fellowship.

[4] We found that the sketch and sub-program decoders in COARSE2FINE do not achieve their best DEV. accuracy at the same iteration during training, which could hurt performance in our few-short learning setting.

References

Jacob Andreas, Andreas Vlachos, and Stephen Clark. 2013. Semantic parsing as machine translation. In *Proceedings of ACL*.

Philip Arthur, Graham Neubig, and Satoshi Nakamura. 2016. Incorporating discrete translation lexicons into neural machine translation. In *Proceedings of EMNLP*.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2).

Xinyun Chen, Chen Liang, Adams Wei Yu, D. Song, and Denny Zhou. 2020. Compositional generalization via neural-symbolic stack machines. In *Proceedings of NeurIPS*.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of ACL*.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of ACL*.

Li Dong and Mirella Lapata. 2018. Coarse-to-fine decoding for neural semantic parsing. In *Proceedings of ACL*.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of IBM model 2. In *Proceedings of NAACL*.

Catherine Finegan-Dollak, Jonathan K. Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. 2018. Improving text-to-SQL evaluation methodology. In *Proceedings of ACL*.

Daniel Furrer, Marc van Zee, Nathan Scales, and Nathanael Scharli. 2020. Compositional generalization in semantic parsing: Pre-training vs. specialized architectures. *ArXiv*, abs/2007.08970.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proceedings of HLT-NAACL*.

Omer Goldman, Veronica Latcinnik, Ehud Nave, Amir Globerson, and Jonathan Berant. 2018. Weakly supervised semantic parsing with abstract examples. In *Proceedings of ACL*.

Sonal Gupta, Rushin Shah, Mrinal Mohit, Anuj Kumar, and Mike Lewis. 2018. Semantic parsing for task oriented dialog using hierarchical representations. In *Proceedings of EMNLP*.

Jonathan Herzig and Jonathan Berant. 2020. Span-based semantic parsing for compositional generalization. In *Proceedings of EMNLP*.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of ACL*.

B. Jones, Mark Johnson, and S. Goldwater. 2012. Semantic parsing with Bayesian tree transducers. In *Proceedings of ACL*.

Daniel Keysers, Nathanael Schärli, Nathan Scales, H. Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, D. Tsarkov, Xiao Wang, Marc van Zee, and O. Bousquet. 2020. Measuring compositional generalization: A comprehensive method on realistic data. In *Proceedings of ICLR*.

T. Kwiatkowski, Luke Zettlemoyer, S. Goldwater, and Mark Steedman. 2011. Lexical generalization in CCG grammar induction for semantic parsing. In *Proceedings of EMNLP*.

Brenden Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *Proceedings of ICML*.

Brenden M Lake. 2019. Compositional generalization through meta sequence-to-sequence learning. In *Proceedings of NeurIPS*.

M. Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, A. Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of ACL*.

Yuanpeng Li, Liang Zhao, JianYu Wang, and Joel Hestness. 2019. Compositional generalization for primitive substitutions. In *Proceedings of EMNLP/IJCNLP*.

Percy Liang. 2016. Learning executable semantic parsers for natural language understanding. *Commun. ACM*.

Lemao Liu, Masao Utiyama, Andrew Finch, and Eiichiro Sumita. 2016. Neural machine translation with supervised attention. In *Proceedings of COLING*.

Qian Liu, Shengnan An, Jianguang Lou, B. Chen, Zeqi Lin, Yan Gao, Bin Zhou, Nanning Zheng, and Dongmei Zhang. 2020. Compositional generalization by learning analytical expressions. In *Proceedings of NeurIPS*.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proceedings of ACL*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of EMNLP*.

Chunchuan Lyu and Ivan Titov. 2018. AMR parsing as graph prediction with latent alignment. In *Proceedings of ACL*.

Haitao Mi, Zhiguo Wang, and Abe Ittycheriah. 2016. Supervised attentions for neural machine translation. In *Proceedings of EMNLP*.

Dipendra Kumar Misra, Ming-Wei Chang, X. He, and Wen tau Yih. 2018. Policy shaping and generalized update equations for semantic parsing from denotations. In *Proceedings of EMNLP*.

Franz Josef Och. 2002. *Statistical machine translation: From single word models to alignment templates*. Ph.D. thesis, RWTH Aachen University, Germany.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1).

Inbar Oren, Jonathan Herzig, Nitish Gupta, Matt Gardner, and Jonathan Berant. 2020. Improving compositional generalization in semantic parsing. In *Proceedings of EMNLP-Findings*.

Maxim Rabinovich, Mitchell Stern, and Dan Klein. 2017. Abstract syntax networks for code generation and semantic parsing. In *Proceedings of ACL*.

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).

Kyle Richardson, Jonathan Berant, and Jonas Kuhn. 2018. Polyglot semantic parsing in APIs. In *Proceedings of NAACL-HLT*.

Jake Russin, Jason Jo, R. O'Reilly, and Yoshua Bengio. 2019. Compositional generalization in a deep seq2seq model by separating syntax and semantics. *ArXiv*, abs/1904.09708.

Abigail See, Peter Liu, and Christopher Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of ACL*.

Semantic Machines, Jacob Andreas, John Bufe, David Burkett, Charles Chen, Josh Clausman, Jean Crawford, Kate Crim, Jordan DeLoach, Leah Dorner, Jason Eisner, Hao Fang, Alan Guo, David Hall, Kristin Hayes, Kellie Hill, Diana Ho, Wendy Iwaszuk, Smriti Jha, Dan Klein, Jayant Krishnamurthy, Theo Lanman, Percy Liang, Christopher H Lin, Ilya Lintsbakh, Andy McGovern, Aleksandr Nisnevich, Adam Pauls, Dmitrij Petters, Brent Read, Dan Roth, Subhro Roy, Jesse Rusak, Beth Short, Div Slomin, Ben Snyder, Stephon Striplin, Yu Su, Zachary Tellman, Sam Thomson, Andrei Vorobev, Izabela Witoszko, Jason Wolfe, Abby Wray, Yuchen Zhang, and Alexander Zotov. 2020. Task-oriented dialogue as dataflow synthesis. *Transactions of the Association for Computational Linguistics*, 8.

Alane Suhr, Ming-Wei Chang, Peter Shaw, and Kenton Lee. 2020. Exploring unexplored generalization challenges for cross-database semantic parsing. In *Proceedings of ACL*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of NIPS*.

Bailin Wang, Mirella Lapata, and Ivan Titov. 2020. Meta-learning for domain generalization in semantic parsing. *arXiv:2010.11988*.

Bailin Wang, Ivan Titov, and Mirella Lapata. 2019. Learning semantic parsers from denotations with latent structured alignments and abstract programs. In *EMNLP/IJCNLP*.

Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gašić, Lina M. Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *Proceedings of EACL*.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. In *Proceedings of ACL*.

Compositional Generalization for Neural Semantic Parsing via Span-level Supervised Attention
Supplementary Materials

A SMCalFlow Compositional Skills Dataset

SMCalFlow (Semantic Machines et al., 2020) is a large-scale semantic parsing dataset for task-oriented dialogue, featuring multi-turn utterances between a user and a dialogue agent that updates the user's schedule using LISP-style programs (see Fig. 1a for an example). In line with the motivating story in §1 about learning compositional skills for task-oriented semantic parsers, we created a new dataset based on SMCalFlow to evaluate a semantic parser's ability to generalize to utterances that require compositional skills when trained on examples of simpler ones.

Specifically, we extract all single-turn, context-free[5] examples from SMCalFlow in the domains of EventCreation (*e.g., Add meeting with Adam* ↦ `CreateEvent(attendees=Adam)`) and OrgChart (*e.g., Who are in Adam's team?* ↦ `FindTeam(recipient=Adam)`), and divide the examples into a training set $\mathbb{S}$ consisting of samples from single domains, and an compositional evaluation set $\mathbb{C}$ with examples covering both of the two skills (*e.g., Set up meeting with Adam and his team* ↦ `CreateEvent(attendees=List(Adam, FindTeam(recipient=Adam)))`). We generate validation and testing sets by evenly dividing the compositional samples in $\mathbb{C}$, while including the same amount ($\frac{|\mathbb{C}|}{2}$) of single-skill examples from $\mathbb{S}$. Tab. 4 presents more examples in SMCalFlow-CS.

Zero-shot generalization in this setting is highly non-trivial due to novel language patterns (*e.g., Adam and his team*) and program structures (*e.g.,* usage of `List(·)` to concatenate entities) in the compositional evaluation set. We therefore consider a few-shot learning scenario, where we include a few compositional examples $\{16, 32, 64, 128\}$ into the training sets (denoted as $\mathbb{C}_{\text{train}}$). To ensure the representativeness of those handful of compositional examples used for training, we generate $\mathbb{C}_{\text{train}}$ using rejection sampling. Specifically, we randomly splitting $\mathbb{C}$ into $\mathbb{C}_{\text{train}}$ and $\mathbb{C}_{\text{dev+test}}$, and repeat this process until examples in $\mathbb{C}_{\text{train}}$ cover a pre-defined list of NL patterns (*e.g., "with Amy and her team", "with Tom's reports", "with my manager", etc*). The sizes of training (without compositional samples)/development/test splits are 25,404/1,325/1,325, respectively.

$\mathbb{S}_{\text{EventCreation}}$ (24,763 Examples)	*Schedule dinner with Adam tomorrow.* *Please add dinner with Adam Wallen next Wednesday night at 6:00 PM.* *Put a reminder on my calendar half an hour before my dinner.*
$\mathbb{S}_{\text{OrgChart}}$ (641 Examples)	*Who's on Abby's team now?* *Who are the reports of Dan Schoffel?*
Compositional Skills ($\mathbb{C}$) (1,453 Examples)	*Add a meeting with my manager after lunch.* *Add Amanda and her boss to project meeting.* *Right after I'm done with breakfast, put a meeting with Sally's team.*

Table 4: Examples from SMCalFlow-CS

B Model Configuration and Alignment Generation

SMCalFlow-CS All models use the `BERT-base-uncased` model as encoder. Both Bert2Seq and Coarse2Fine use two-layer LSTM networks as decoder, following the formulation in Luong et al. (2015), with a hidden size of 256. For Coarse2Fine, we use a slightly different definition of sketch-subprogram decomposition as in §2, where a sketch includes named arguments as well (*e.g.,* `CreateEvent(attendees=Jim)` is decomposed to a sketch `CreateEvent(attendees=`$\boxed{?}$`))` and sub-program (*e.g.,* $\boxed{?}$`=Jim`)). The sketch and sub-program decoders in Coarse2Fine share the same LSTM, as we find this will improve its performance in our few-shot learning setting. During training, we use an

[5] Context-freeness could be determined by checking if a program has function calls that refer to previous dialogue context.

Adam optimizer using a batch size of 64 for 30 epochs, with separate learning rates for BERT (3×10^{-5}) and the rest of the model parameters (0.001). We add supervised attention loss Eq. (1) to the model's loss function with a tuning weight of $\lambda \in \{2.0, 4.0\}$ for BERT2SQL and $\lambda \in \{1, 0, 2.0\}$ for COARSE2FINE. For each training split with $|\mathbb{C}_{\text{train}}|$ compositional training examples, we perform grid search and chose the λ that achieves the best DEV. accuracy on compositional samples $\mathbb{C}$. We use beam search (beam size of 5) for decoding.

CFQ We use T5-BASE, with a constant tuning weight of $\lambda = 0.1$ for the supervised attention loss. We train the model using an Adafactor optimizer with a batch-size of 128 examples and a learning rate of 0.001 for 15 epochs ($\sim 110K$ iterations). We warmup the learning rate using the first 1,100 iterations. Target program sequences with a length longer than 300 after `sentencepiece` subtokenization are clipped. For efficiency, we use greedy search for decoding.

ATIS TEXT-TO-SQL We use the original implementation and hyper-parameters provided by Oren et al. (2020), and apply supervised attention loss with a tuning weight validated from $\{0.05, 0.1, 1.0, 2.0\}$.

Alignment Extraction We run `GIZA++` to get token-level alignments. As noted in Oren et al. (2020), raw alignments between program and utterance tokens generated by off-the-shelf word aligners are often noisy, we therefore applied the following heuristics to improve alignment quality: On SMCALFLOW-CS, we canonicalize programs by removing parentheses. We use the source-to-target direction alignments generated by `GIZA++`, as we find alignments in this direction have better coverage and higher quality than the results from the other direction. On CFQ, we use the union of the alignments for both directions, and removed alignments to intermediary variables (*e.g.,* `?x0`, `?x1`), as their alignments are often noisy. For ATIS, we follow Oren et al. (2020) and canonicalize programs by removing punctuations. We use the source-to-target direction alignments from `GIZA++`. To extract sub-programs from SQL queries in ATIS for span-level alignment extraction, we define sub-programs in SQL as tables (*e.g.,* `Flight.ID`) and comparison statements (*e.g.,* `City.City_Name = "city_name0"`) in the `SELECT` and `WHERE` clauses, respectively. We use this restricted strategy to extract sub-programs only from `SELECT` and `WHERE` clauses because we find words alignments to other constructs in SQL queries (*e.g.,* statements that specify tables to be joined) are often noisy. For this reason, we do not generate span-level alignments for program sketch tokens, as they as under-specified.

Finally, for all datasets, we remove alignments between non-content program tokens (*e.g.,* the '`=`' sign) and stop words in utterances.

C Additional Results

C.1 Full Results on SMCALFLOW-CS

Quality of attention distribution w.r.t sub-program prediction accuracy In §3, we briefly described the positive correlation between the "quality" of the attention distribution $p_{\text{att}}(u_i|z_j)$ (how concentrated $p_{\text{att}}(u_i|z_j)$ is) over an utterance span (*e.g., with my manager*) and the prediction accuracy of its target sub-program (*e.g.,* `attendees=FindManager(·)`). Here we present more results. Specifically, we identify compositional examples in the Dev. set for which a model predicts sub-programs $\boldsymbol{z}^{\text{s}}$ for the `attendees`, `start`, and `location` arguments in a `CreateEvent` function call (refer to Fig. 1a for the first two arguments, `location` is used to specify event location). We compute the sum of the attention weights over the "oracle" utterance span identified by Algo. 1, and averaged over the decoder's time step when predicting $\boldsymbol{z}^{\text{s}}$. We measure the sub-program prediction accuracy w.r.t. the attention weights, as illustrated in Fig. 2. We observe that models trained with span-level supervised attention shows a stronger correlation between the sub-program accuracy and the degree the attention focuses on utterance tokens within the oracle span.

Results using a Previous Version of SMCALFLOW For completeness, we also report results on another version of our SMCALFLOW-CS benchmark based on a previous version of the SMCALFLOW dataset. Tab. 5 list the results. The main differences between this version of

$\lvert\mathbb{C}_{\text{train}}\rvert$	16		32		64		128	
Domain	$\mathbb{S}$	$\mathbb{C}$	$\mathbb{S}$	$\mathbb{C}$	$\mathbb{S}$	$\mathbb{C}$	$\mathbb{S}$	$\mathbb{C}$
BERT2SEQ	82.8 ±1.0	37.7 ±1.0	82.8 ±0.8	57.4 ±7.1	82.4 ±0.2	71.1 ±2.7	81.8 ±0.9	75.8 ±2.0
+TS (Token-level Sup.)	82.9 ±0.5	47.1 ±4.0	82.5 ±0.7	65.1 ±1.8	83.1 ±0.4	72.1 ±0.9	82.3 ±0.6	77.5 ±1.5
+SS (Span-level Sup.)	83.3 ±0.7	**54.9** ±3.4	**83.4** ±0.6	**67.5** ±2.0	82.8 ±0.6	**76.0** ±1.3	82.6 ±0.3	**78.7** ±0.9
COARSE2FINE (DL18)	82.5 ±0.8	44.7 ±4.9	83.0 ±1.0	60.0 ±4.2	82.5 ±0.4	72.4 ±1.4	83.0 ±0.9	75.0 ±0.9
+TS (Token-level Sup.)	83.0 ±0.3	51.0 ±4.6	82.9 ±0.9	64.2 ±1.8	82.6 ±0.6	74.0 ±0.5	82.8 ±0.4	78.1 ±0.9
+SS (Span-level Sup.)	83.1 ±0.4	**54.2** ±3.0	83.1 ±0.5	**66.6** ±1.6	**83.5** ±0.9	74.8 ±1.1	82.9 ±0.4	78.2 ±0.5

Table 5: TEST. accuracies on the SMCALFLOW-CS Compositional Skills dataset w.r.t. the size of compositional examples included in the training set. We report both the results on the in-domain single-skill examples ($\mathbb{S}$) as well as the generalized multi-skill examples ($\mathbb{C}$). Results averaged over five random random seeds. **Bold** results have p-values ≤ 0.01 when comparing to other systems in the same category using paired permutation test.

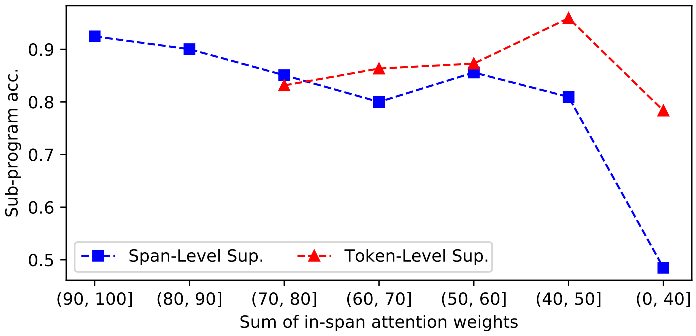


Figure 2: Sub-program prediction accuracy w.r.t. the sum of attention weights over oracle utterance spans. Models are trained on $\lvert\mathbb{C}_{\text{train}}\rvert = 32$. Results averaged over three runs.

SMCALFLOW-CS and the one used in Tab. 1 are (1) ordering of named arguments in LISP expressions (*e.g.*, `CreateEvent(attendees=`$\boxed{?}$`, start=`$\boxed{?}$`, subject=`$\boxed{?}$`)` *v.s.* `CreateEvent(subject=`$\boxed{?}$`, attendees=`$\boxed{?}$`, start=`$\boxed{?}$`)`), and (2) some "cosmetic" changes to simply the domain-specific LISP programs. Interestingly, compared with the results in Tab. 1, we find that both the token and span-level supervised attention methods are sensitive to such changes in the representation of programs. While we didn't observe significant changes of quality in the underlying word alignments produced by `GIZA++`, we leave investigating these results as interesting future work.

C.2 Complementary Span-level Supervised Attention Loss

In §2 we present span-level supervised attention, which minimizes the mean-squared error loss between the decoder's attention distribution $p(u_i|z_j)$ and the prior alignment distribution derived from the span-level alignment matrix (§2). Models trained with such a loss function learns a uniform attention distribution over tokens in an utterance span.

An alternative loss function is to relax the uniform attention constraint, and let the model to decide how to allocate the attention mass over tokens inside a predefined utterance span. Specifically, we consider a masked version of the main-squared error loss in Eq. (1), where we only apply the loss on utterance tokens u_i that are *not* aligned to z_j according to the alignment matrix (*i.e.*, $a_{i,j} = 0$):

$$\mathcal{L}_{\text{sup\_att}} = \frac{1}{|\boldsymbol{z}|}\sum_{j=1}^{|\boldsymbol{z}|}\sum_{i=1}^{|\boldsymbol{u}|}\left(\|p_{\text{att}}(u_i|z_j) - p_{\text{prior}}(u_i|z_j)\|_{a_{i,j}=0}\right)^2. \tag{2}$$

where $\|\cdot\|_{a_{i,j}=0} \triangleq p_{\text{att}}(u_i|z_j) - p_{\text{prior}}(u_i|z_j)$ iff $a_{i,j} = 0$. Intuitively, Eq. (2) forces zero attention to tokens outside an aligned utterance span, while leaving the model with the freedom to attend to any tokens inside the span. We term this loss function the *complementary span-level supervised attention* loss.

We first compare complementary and standard span-level supervised attention on SMCALFLOW-CS. Results are listed in Tab. 6. We didn't report on all the training splits since in our pilot study we observe

Model	$\mathbb{S}$	$\mathbb{C}$
Bert2Seq	82.6 ($\Delta = 1.9$)	55.2 ($\Delta = 6.2$)
+ Complementary	82.1 ($\Delta = 2.3$)	63.0 ($\Delta = 8.3$)
+ Span Attn. Sup.	**83.3** ($\Delta = 0.3$)	**67.3** ($\Delta = 2.6$)

Table 6: Complementary vs standard span-level supervised attention on SMCalFlow-CS with 32 compositional training examples averaged over three runs. Δ indicates the difference between the best and worse results.

Split	MCD$_1$			MCD$_2$			MCD$_3$		
	C	R	All	C	R	All	C	R	All
T5-base	55.6	75.6	61.7	**35.3**	28.1	32.8	**17.7**	33.4	23.2
+ TS	43.5	**88.1**	57.2	33.1	32.6	33.0	14.3	36.2	21.9
+ SS	46.1	80.1	56.6	34.8	**35.1**	**34.9**	15.0	**39.2**	**23.4**
+ Complementary TS	**58.3**	79.3	**64.7**	**35.4**	32.1	34.3	17.5	33.2	22.9

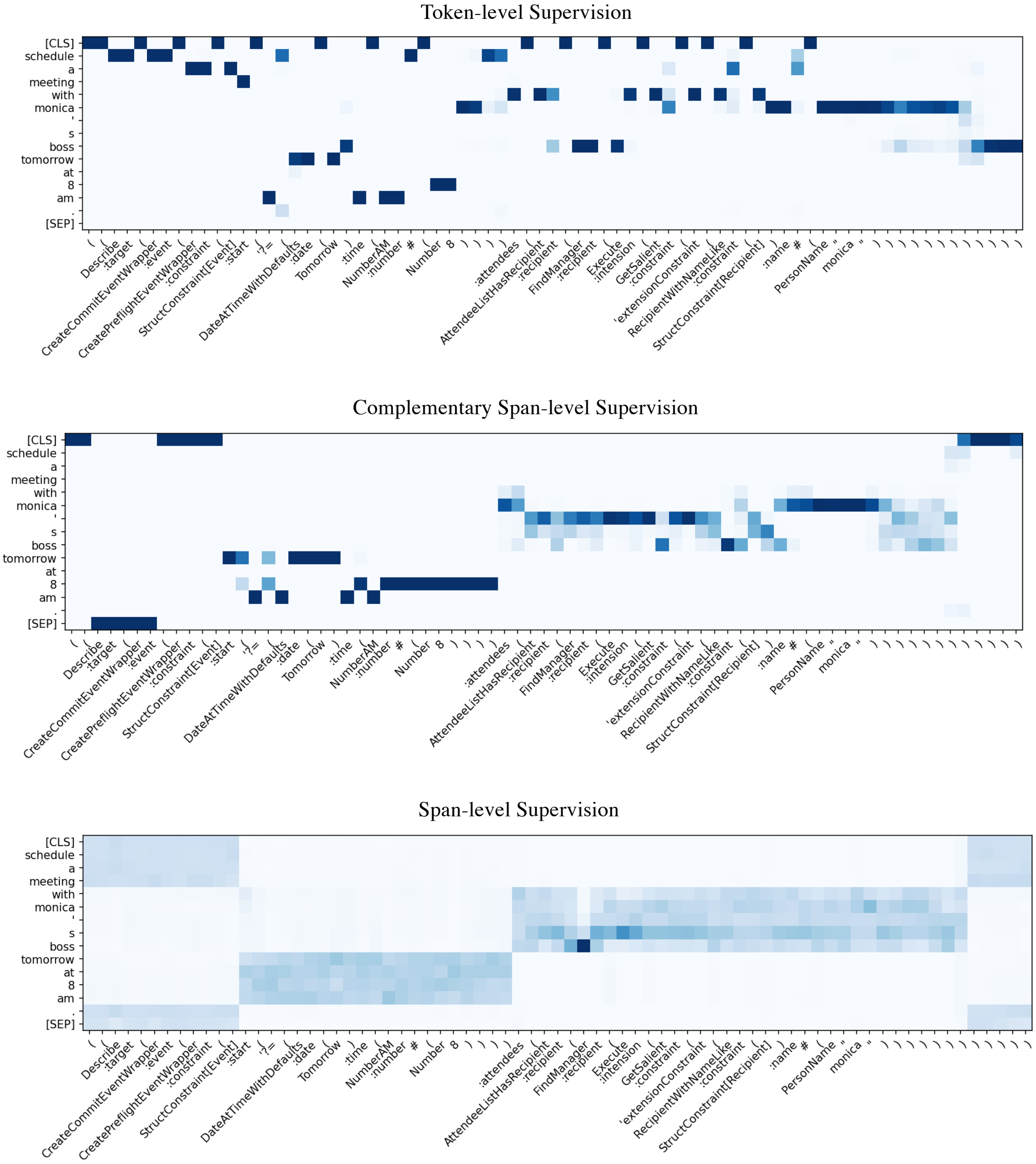
Table 7: Test Accuracies on CFQ MCD splits, for **C**onjunctive, **R**ecursive, and **A**ll the samples, averaged over three restarts.

that complementary span-level attention supervision does not perform well on SMCalFlow due to its high variance. We hypothesize that is because complementary attention allows the model to freely attend to any utterance tokens within a predefined span boundary, as long as the attention weights for the tokens within the span sum up to 1. Therefore, it is possible that the attention distribution becomes sparse and degenerates to the scenario with token-level supervision, as illustrated by the example in Fig. 3.

Next, we evaluate complementary supervised attention on CFQ, with results listed in Tab. 7. Interestingly, we observe the standard span-level objective is more effective on recursive (**R**) splits, and the complementary objective on conjunctive (**C**) splits. First, we find models trained with the complementary objective are better at handling questions with long conjunctive entity lists (*e.g., Who directed, produced, wrote, and edited M1, M2, and M3?*). This is probably due to that the model has the freedom to attend to specific utterance tokens (*e.g.,* an entity mention *M1*) in an utterance span that are most relevant to predicting a target token (*e.g.,* the entity variable `M1` in z), while enforcing uniform distribution as in the standard span-level supervision will cause the model to "lose focus".

Fig. 4 shows such an example, the model trained with complementary supervision selectively attends the relevant entity mentions when generating the three object variables (`M1 M2 M3`) for the relation `film.film.directed_by`, while the model with vanilla span-level supervision, using a more flattened attention distribution, failed to predict the complete list of objects (only `M1` is predicted). However, we note that is not always the case, as we observe that when the number of conjunctive entities grows larger, models trained with the complementary objective could still correctly predict the entire variables in an entity list without attending to their individual mentions separately in the utterance (Fig. 5). We leave investigating this as interesting future work.

Next, we attempt to understand the relative advantage of the standard span-level supervised attention *v.s.* the complementary objective. We sampled 50 failure cases for the model with the complementary objective on MCD$_3$. Interestingly, we find that more than half of the errors are due to the model got confused about the syntactic role of entities mentions in complex questions with chained relations. Fig. 6 gives such an example, where the the model incorrectly identifies `M1` as the subject of the relation `people.person.employment_history...` when interpreting the utterance span *a employee of M1*, a pattern that we usually observed for models without using supervised attention. One possible explanation is that models trained with complementary objective use a sparser attention distribution, which might not consider the full utterance span when making predictions, while a model trained with the standard span-level objective learns to parse an utterance span using information from all its tokens.

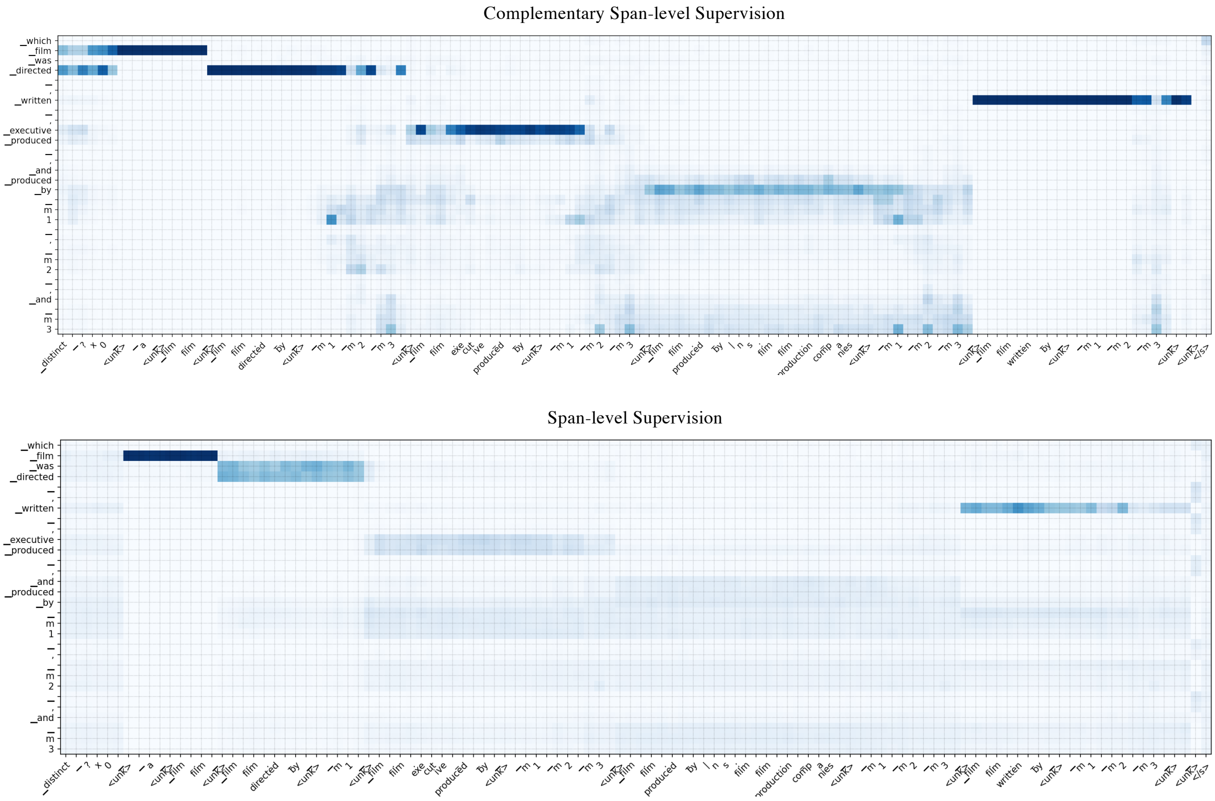

Token-level Supervision

Complementary Span-level Supervision

Span-level Supervision

$\boldsymbol{u}$: *Schedule a meeting with Monica's boss tomorrow at 8 am.*

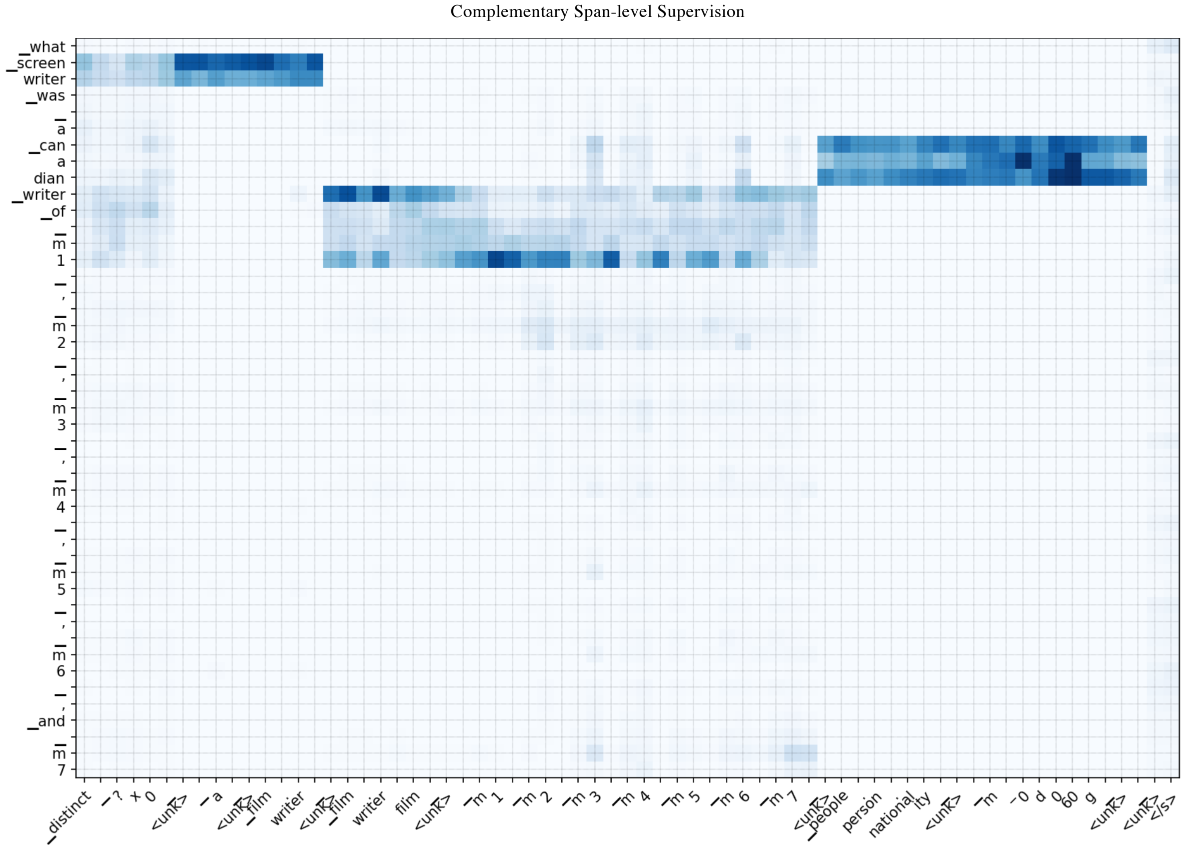
Figure 3: Example attention visualizations for BERT2SEQ trained with token-level, complementary, and standard span-level supervised attention on SMCALFLOW-CS.


Complementary Span-level Supervision

Span-level Supervision

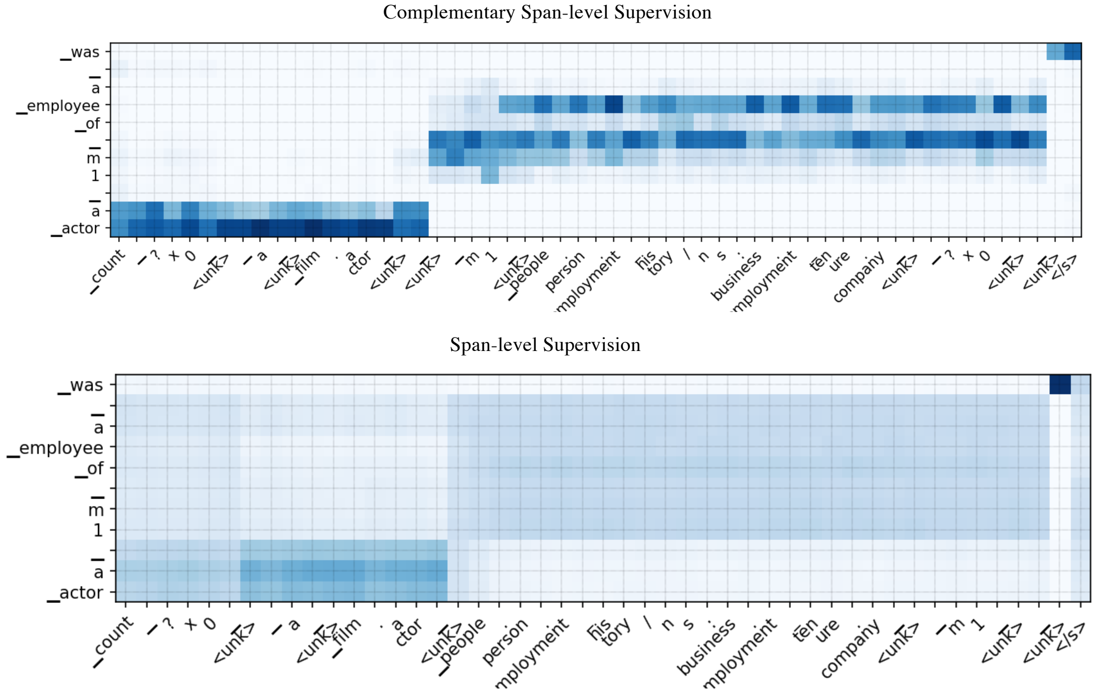
$\boldsymbol{u}$: *Which film was directed, written, executive produced, and produced by M1, M2, and M3?*

Figure 4: Example attention visualizations for T5-BASE trained with complementary and standard span-level supervised attention for a CFQ question. Attention distributions are taken from the last decoder layer and maxed over all the attention heads.



u: *What screen writer was a canadian writer of M1, M2, M3, M4, M5, M6, and M7?*

Figure 5: Example attention visualization for T5-BASE with complementary span-level supervision. The model correctly generates the 7 object variables without attending to their individual mentions in $\boldsymbol{u}$.



u: *Was a(n) employee of M1 an actor?*

Figure 6: Example attention visualization for a question in the recursive evaluation split of CFQ.