Compositional Generalization in Dependency Parsing

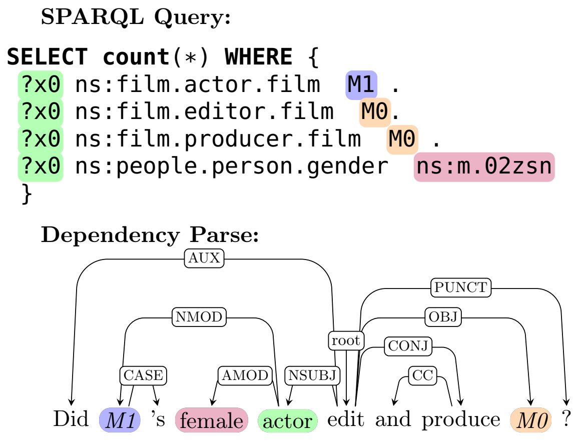
Emily Goodwin[*1,3] **Siva Reddy**[1,2,3,6] **Timothy J. O'Donnell**[1,3,5] **Dzmitry Bahdanau**[4, 5]
[1]Department of Linguistics [2]School of Computer Science McGill University, Canada
[3]Quebec Artificial Intelligence Institute (Mila) [4]ServiceNow Research
[5]Canada CIFAR AI Chair [6]Facebook CIFAR AI Chair

Abstract

Compositionality—the ability to combine familiar units like words into novel phrases and sentences—has been the focus of intense interest in artificial intelligence in recent years. To test compositional generalization in semantic parsing, Keysers et al. (2020) introduced Compositional Freebase Queries (CFQ). This dataset maximizes the similarity between the test and train distributions over primitive units, like words, while maximizing the *compound divergence*—the dissimilarity between test and train distributions over larger structures, like phrases. Dependency parsing, however, lacks a compositional generalization benchmark. In this work, we introduce a gold-standard set of dependency parses for CFQ, and use this to analyze the behavior of a state-of-the art dependency parser (Qi et al., 2020) on the CFQ dataset. We find that increasing compound divergence degrades dependency parsing performance, although not as dramatically as semantic parsing performance. Additionally, we find the performance of the dependency parser does not uniformly degrade relative to compound divergence, and the parser performs differently on different splits with the same compound divergence. We explore a number of hypotheses for what causes the non-uniform degradation in dependency parsing performance, and identify a number of syntactic structures that drive the dependency parser's lower performance on the most challenging splits.

1 Introduction

People understand novel combinations of familiar words in part due to the principle of *compositionality*: We expect the meaning of a phrase to be a predictable composition of the meanings of its parts. Unlike humans, many neural models fail to generalize compositionally; a growing interest in this area has led to novel architectures and datasets designed to test compositional generalization (see § 7).



SPARQL Query:

```
SELECT count(*) WHERE {
 ?x0 ns:film.actor.film  M1 .
 ?x0 ns:film.editor.film  M0.
 ?x0 ns:film.producer.film  M0 .
 ?x0 ns:people.person.gender  ns:m.02zsn
 }
```

Figure 1: An example question from the CFQ dataset, with the associated SPARQL query and dependency parse.

One recently-introduced semantic parsing dataset, Compositional Freebase Queries (CFQ), consists of English questions with corresponding database queries written in SPARQL. Figure 1 shows an example question and SPARQL query. To test compositional generalization, CFQ includes test and train sets with a highly similar distribution of primitive units (like words) and increasingly divergent distribution of larger *compound* units (like phrases). The most challenging of these splits, with the highest compound divergence, are dubbed *maximum compound divergence* (MCD) splits.

Although CFQ has proven to be a valuable resource, the difficulty of the splits appears to be influenced by factors other than compositional generalization. First, some evidence suggests that the complexity of the SPARQL output is in part responsible for CFQ performance (Furrer et al., 2020; Herzig et al., 2021). Furthermore, splits of the same compound divergence are not equally difficult. One

[*] Majority of work completed during internship at Element AI, now ServiceNow Research. Corresponding author, emily.goodwin@mail.mcgill.ca

possible explanation is a difference in the syntactic constructions of different splits; however, this has not yet been explored in CFQ. To address these issues, we created a dependency-parsing version of CFQ. Using our dataset, we evaluated a state-of-the-art dependency parser for compositional generalization, and used the dependency annotations to identify syntactic structures predictive of parsing failure on each MCD split.

We found that the dependency parser is more robust to increased compound divergence than the semantic parser, but performance still decreased with higher compound divergence. We also found the dependency parser, like semantic parsers, varied widely in performance on different splits of the same compound divergence. Finally, we found that a small number (less than seven) of syntactic constructions seem to drive the difficulty of the MCD splits. Our dataset is publically available on github.[1]

1.1 Motivation for Dependency Parsing

In this section, we discuss three problems of CFQ, and our motivation for studying compositional generalization in dependency parsing.

First, CFQ is hard: seq2seq models trained from scratch score at most 12% on MCD2 and MCD3 sets (Google Research, 2020). Because of its difficulty, CFQ may lack sensitivity to capture small but significant progress in neural modelling of compositionality. Second, recent work shows that CFQ's difficulty is in part due to the output representation being raw SPARQL: Models perform better when outputs are replaced with compressed versions of SPARQL, that are more aligned with the natural-language-like questions (Furrer et al., 2020; Herzig et al., 2021). In interpreting performance on CFQ, we might be conflating challenges of compositional generalization with challenges related to the output representation.

Third, different splits of the same compound divergence vary widely in difficulty: seven of the nine semantic parsers currently listed on the leaderboard perform at least twice as well on MCD1 as MCD 2, despite the splits having the same compound divergence (Google Research, 2020). Performance on CFQ is thus heavily influenced by some factor about the splits other than compound divergence.

Finally, CFQ lacks a description of the specific syntactic generalizations tested by each split. Related benchmarks, like COGS (Kim and Linzen, 2020) and CLOSURE (Bahdanau et al., 2020), test a clearly-defined set of generalizations (for example, training a noun in subject position and testing in object position). CFQ splits, by contrast, optimize a gross metric over the distribution of all syntactic compounds in the dataset. This complicates in-depth analyses of CFQ results: For a particular split, it is unclear what syntactic constructions are tested in out-of-distribution contexts. Meanwhile, for a particular test sentence, it is unclear which of its syntactic structures caused the model to fail.

To address the issues with the CFQ semantic parsing benchmark, we studied compositional generalization in syntactic parsing. While syntactic parsing is simpler than mapping to a complete meaning representation, a language-to-SPARQL semantic parser must understand the question's syntax. For example, to generate the triple `?x0 ns:film.editor.film M0` in the SPARQL query shown in Figure 1, a semantic parser must first identify that "actor" is the subject of "edit".

We chose dependency trees as the target syntactic formalism due to the maturity of the universal dependencies annotation standard, the popularity of dependency trees among the NLP practitioners, and the availability of popular high-performance software such as Stanza (Qi et al., 2020). Importantly, dependency parsing does not require auto regressive models; instead, graph dependency parsers independently predict edge labels. This different way of employing deep learning for parsing has the additional advantage of allowing us to separate the challenge of compositional generalization from challenges related to auto regressive models' teacher forcing training. Finally, having gold dependency annotations for CFQ questions enables detailed analysis of the relation between the model errors and syntactic discrepancies that are featured by the MCD splits.

2 Compound Divergence in CFQ

CFQ is designed to test compositional generalization by combining familiar units in novel ways. To ensure the primitive units are familiar to the learner, CFQ test and train sets are sampled in a way that ensures a low divergence in the frequency distribution of *atoms*. Here, atoms refers to individual predicates or entities, (like "produced" or "Christopher Nolan"), and the rules used to generate questions.

To ensure the compounds in test are novel, train

[1]https://github.com/emilygoodwin/CFQ-dependencies

and test sets were sampled in a way that ensures higher divergence between the frequency distribution of compounds, weighted to prevent double-counting of any nested compounds which co-occur frequently.

Keysers et al. (2020) released dataset splits with compound divergence on a scale between 0 (a random split) and .7 (Maximum Compound Divergence, or MCD, splits).

3 Corpus Construction: Dependency Parses for CFQ

To train a dependency parser and analyze syntactic structures in the CFQ dataset, we created a corpus of gold dependency parses. Because the questions in CFQ are synthetically generated, we were able to write a full-coverage context-free grammar for the CFQ language (see Appendix C). Using this grammar, and the chart parser available in Python's natural language toolkit, we generated a constituency parse for each question. Finally, we designed an algorithm to map to the dependency parse.

To map from constituency to dependency parses, we wrote a dependency-mapping rule for each production rule in the CFG (Collins, 2003). Each dependency rule describes the dependency relation between the elements in the constituent; for example, if the production rule is VP $\longrightarrow$ V NP, the dependency-mapping rule connects the head of the right-hand node (the head of NP) as a dependent of the left-hand node (the V), with the arc label OBJ. We follow version two of the Universal Dependencies Corpus annotation standards (Nivre et al., 2020),[2] but simplify the categorization of nominal subjects for active and passive verbs into one category (NSUBJ), and do not include part of speech tags in the dataset.

Our algorithm then recursively walks the constituency tree from bottom to top, mapping non-head children of each node to their syntactic heads and passing the head of each constituent up the tree. A number of sentences in the CFQ dataset exhibit dependency structures which cannot be directly read off the constituency parse in this manner: Such *right-node-raising* constructions involve a word without a syntactic head in the immediate constituent. For example, in "Was *Tonny* written by and executive produced by Mark Marabella?" the first instance of "by" is a dependent of "Mark Marabella", but its immediate constituent is "directed by". To handle right-node raising cases, our dependency-mapping algorithm identifies prepositions with no head in the immediate constituent, and passes them up the tree until they can be attached to their appropriate syntactic head.

Finally, we performed a form of anonymization on the questions, replacing entities with single-word proper names. This reflects the anonymization strategy used in Keysers et al. (2020), and prevents the dependency parser from failing because of named entities with particularly complex internal syntax (for example, "Did a Swedish film producer edit *Giliap* and *Who Saw Him Die?*")

The experiments in this paper are based on the original CFQ splits. However, these validation sets are constructed from the same distribution as the test sets; some information about the test distribution is therefore available during train. To ensure that the model only had access to the training distribution during the training phase, we followed the suggestion of Keysers et al. (2020) and discarded the MCD validation sets,[3] randomly sampling 20% of the training data to use instead (see § 5.1 of that paper for more details). The resulting splits have $11,968$ test sentences and $76,595$ train sentences.

4 Compound Divergence Effect on Dependency Parsing

4.1 Training Stanza

To evaluate the effect of compound divergence on dependency parsing, we used Stanza (Qi et al., 2020), a state-of-the-art dependency parser, on the gold label dependency parses described in §3. We trained Stanza five times on each of 22 splits from the CFQ release: one random split (which has a compound divergence of 0), 18 splits with increasing compound divergence (ranging from .1 to .6) and three MCD splits (divergence of .7).

To evaluate performance on each test set, we used the CoNLL18 shared task dependency parsing challenge evaluation script (CoNLL Shared Task, 2018), which gives a Labeled Attachment Score (LAS) and Content-word Labeled Attachment Score (CLAS), reflecting how many of the total dependency arcs in the test set were correctly labeled, and how many of the arcs connecting content words were correctly labeled, respectively.

In addition, we calculated the percentage of

[2] www.universaldependencies.org

[3] We also re-sampled a validation set from the random split training set, so that MCD and random splits are trained on the same amount of data.

test questions for which every content word arc was correctly labeled, which we call *Whole Sentence Content-word Labeled Attachment Score* (WSCLAS). This all-or-nothing evaluation scheme for each sentence more closely resembles the exact-match accuracy of semantic parser evaluation.[4]

4.2 Dependency Parsing Results

We plot Stanza's performance as a function of the split compound divergence in Figure 2. Increasing compound divergence had a negative effect on performance: Stanza's accuracy on the random split (zero compound divergence) was near perfect, with an average CLAS of 99.98% and WSCLAS of 99.89%. Meanwhile, accuracy on the three MCD splits (divergence of .7) dropped to an average CLAS of 92.85% and WSCLAS of 74.92%.

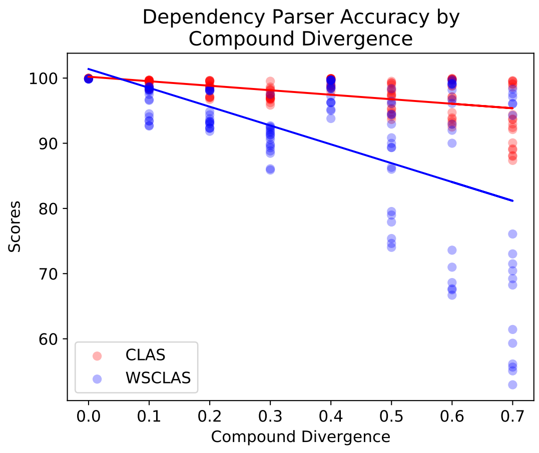


Figure 2: The effect of compound divergence on Content-word Labeled Attachment Score (CLAS) and Whole-Sentence Content Labeled Attachment Score (WSCLAS).

A linear regression predicting CLAS found a slope of −6.91, and predicting WSCLAS found a slope of −28.89; in other words, for each .1 increase in compound divergence the linear model predicts a 2.889% lower WSCALS, and .691% lower CLAS. These linear models are also shown in Figure 2.

We note, however, two exceptions to the generally negative relationship between compound divergence and accuracy, which indicate that other characteristics of the test set have a large effect on accuracy. First, all splits with a target compound divergence of .4 performed stronger than those at divergence .3 and .2. Secondly, we observed considerable variation in performance on different splits that have the same compound divergence, particularly the MCD splits.

Split	Dependency Parser WSCLAS mean (sd)	CLAS mean (sd)	LAS mean (sd)	Semantic Parser Exact Match mean (95% conf interval)
MCD1	96.57 ±1.31	99.38 ±0.29	99.64 ±0.16	37.4 ±2.2
MCD2	71.42 ±2.59	91.53 ±1.00	93.28 ±0.88	8.1±1.6
MCD3	56.76 ±2.81	87.66 ±0.93	90.87 ±1.01	11.3 ±0.3
Random	99.89 ±0.01	99.98±0.00	99.99 ±0.00	98.0 ±0.3

Table 1: Stanza's performance on MCD splits in terms of Whole Sentence Content-word Labeled Attachment Score (WSCLAS), Content-word Labeled Attachment Score (CLAS), and Labeled Attachment Score (LAS). Means and standard deviations are calculated over five randomly-seeded runs. The semantic parsing scores are reproduced from (Keysers et al., 2020); the mean exact-match of 5 experiments with 95% confidence intervals is reported for each MCD split in their github repository.[5]

Stanza's performance on the three maximum-compound-divergence splits and one random split is shown in Table 1. While all three MCD splits were harder than the random split, performance varied from 96.57% WSCLAS (MCD1) to 56.76% WSCLAS (MCD3). Thus, while compound divergence is a factor in performance, idiosyncrasies in the individual splits also have large effects on performance.

Finally, we note that while Stanza was more robust to compound divergence than the semantic parser, it also ranked the splits differently in difficulty. Table 1 reproduces mean accuracies from Keysers et al. (2020)'s strongest-performing semantic parser, a universal transformer (Dehghani et al., 2019). The universal transformer's exact-match is lower than Stanza's WSCLAS on every MCD split. Additionally, while Stanza performed worst on MCD3, the universal transformer and most other semantic parsers in the CFQ leaderboard performed worst on MCD2 (Google Research, 2020). In the next sections, we explore what causes the variation in performance on different MCD splits.

5 Construction Complexity and the MCD Splits

The compound divergence metric treats all compounds of any number of words identically; therefore, the differences between the MCD splits may be driven by differing distributions of compounds of different complexities. In this section, we show

[4]The code to calculate WSCLAS is also available at https://github.com/emilygoodwin/CFQ-dependencies

that this is not the case. We first describe how we characterize syntactic constructions using the dependency annotations.

5.1 Syntactic Constructions

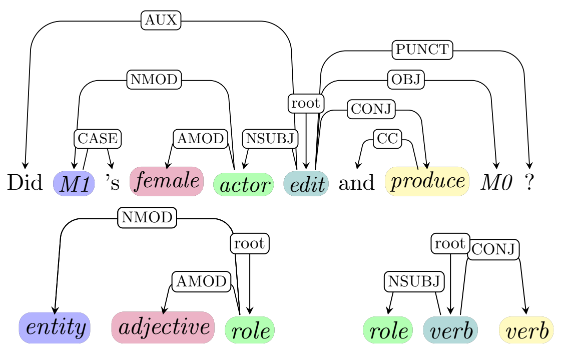


Figure 3: A dependency parse and two of its subtrees

We explored differences in the distributions of syntactic constructions by looking at a restricted set of the subtrees of each dependency parse, which we will now describe.

With respect to any *target node* in the corpus, we consider a syntactic construction to be any subtree that consists of that target node together with a *constituent-contiguous* subset of the target node's immediate children. Here, constituent-contiguous means the subsets of child nodes which are heads of phrases that are adjacent to one another or to the target node in the string. We include only the immediate children in the subtree (excluding their descendants). We also replace words with their category label in CFQ: in addition to traditional parts of speech like *verb* and *adjective*, the category labels include nominal categories *role* (which occurs in possessive constructions like "mother" in "Alice's mother"), *entity* for proper nouns, and *noun* for common nouns.

For the analyses in this and the following section, we extract every syntactic construction for every dependency parse in our corpus, and compare their *complexity*. We define complexity to be the number of arcs in the subtree, discounting the dummy ROOT arc. Two of the subtrees for sentence "Did *M1*'s female actor edit and produce *M0*?" are shown in Figure 3 (these subtrees have a complexity of two). Table 2 shows the number of unique constructions in each test and train set.

5.2 Analysis of MCD Splits

One possible source of the differences between MCD splits may be that they differ in their distributions of subtrees at differing complexities. In this section, we present two analyses showing that this is not the case.

	Total Sentences		Unique Constructions	
	Train	Test	Train	Test
MCD1	76,595	11,968	2,093	2,048
MCD2	76,595	11,967	2,006	1,884
MCD3	76,595	11,968	2,300	1,823
Random	76,596	11,967	4,082	3,251

Table 2: Number of sentences and unique constructions for each test and train set in our experiments (see § 5.1 for an explanation of constructions).

In our first analysis, we analyzed the distance between test and train distribution for each split. To do this we calculated the Jensen-Shannon (JS) distance between the test and train histograms of syntactic constructions at differing complexities.[6]

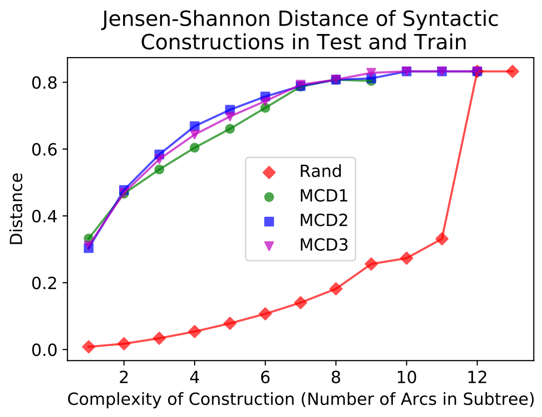


Figure 4: Divergence between test and train of the MCD and random splits. Roughly, divergence increases with subtree complexity, although much more rapidly for MCD splits than random. Additionally, there is little difference between the different MCD splits.

The JS distances for constructions of each complexity are plotted in Figure 4. As can be seen in the figure, the distances between test and train are similar for all MCD splits at all subtree complexities. Even the MCD1 distances pattern with the other MCD splits, despite the parser performance on MCD1 being more similar to the random split.

[6]The JS distance for histograms p and q is defined as

$$\sqrt{\frac{D(\,p\|m\,)+D(\,q\|m\,)}{2}} \tag{1}$$

where m is the pointwise mean of p and q, and D is the Kullback-Leibler divergence.

Thus, differences between the test and train distributions at different complexities cannot explain the MCD splits' differential performance.

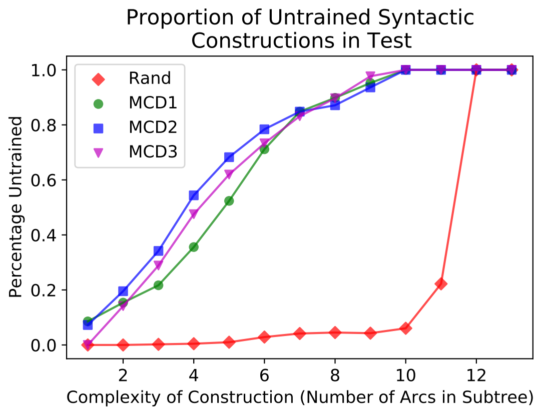


Figure 5: The proportion of syntactic constructions in test which are untrained, for all splits and all subtree complexities. Subtree complexity is measured in number of arcs.

In our second analysis, we examined whether the MCD splits differ in the proportion of untrained subtrees at different complexities. The proportions are plotted in Figure 5. The MCD splits pattern together, with far more untrained constructions at each complexity than the random split.

We thus conclude it is unlikely that gross distributional properties of the MCD splits explain the differences in parser performance. In the next section, we show that parser mistakes for all splits seem to be driven by a very small number of hard-to-parse subtrees. Thus, performance differences between splits likely depend on idiosyncratic interactions between the specific data splits and models.

6 Syntactic Analyses of Dependency Parser Outputs

6.1 Identifying Difficult Subtrees

To identify syntactic constructions that are predictive of dependency parsing error, we fit a logistic model predicting Stanza's performance on each test question from the question's syntactic constructions. Because we trained five randomly-initialized versions of Stanza, the model was fit with five instances of each question. To encourage sparse subtree feature weights, we used L1 regularization. We used 90% of the test set to train the logistic model, and the remaining 10% to test it and select a regularization coefficient of .01.

To analyze the subtrees most predictive of parsing failure, we extracted from the model all subtrees with a coefficient less than or equal to -1. Finally, to quantify the effect these trees have on test performance, we removed all the sentences containing the trees for each split, and calculated Stanza's accuracy on the remaining test sentences.

6.2 Subtrees Predictive of Parsing Error

Table 3 shows the number of subtrees found to be predictive of parsing error, together with the accuracy when those trees are removed from test. Removing five subtrees from MCD2's test set improves the accuracy to 92.46% (an increase of 21.05%), and removing seven trees from MCD3's test set improves the accuracy to 93.09% (an increase of 36.33%). We thus conclude that the performance degradation of Stanza on higher compound divergence splits is driven by a relatively small number of syntactic constructions.

Table 4 shows the subtrees most predictive of a dependency parsing error, with their test and train frequency. To quantify the effect of each subtree on the test accuracy, we also report the *Test set* Δ: $\text{WSCLAS}(T') - \text{WSCLAS}(T)$ where T is the original test set and T' is all test sentences which do not include the construction. A positive Δ means that removing the subtree from the test set improved performance, while a negative Δ indicates that removing the subtree from the test set degraded performance.

Subtrees that are predictive of error for a particular split are often missing from train, together with others that share a similar syntactic structure. For instance, there are a set of trees that form questions with common nouns as subject and predicate, and a copula verb "was" appearing to the left of the subject (e.g. "Was an art director of Palm County a person?").[7] The fourth, fifth and sixth subtrees in Table 4 are subtrees which form these questions; all three are missing from train for both MCD2 and MCD3, and all are predictive of parser error for these splits. In contrast, the MCD1 training set includes one of the subtrees (fourth in Table 4) and leaves the other two untrained; none are predictive of parser error (with Δ of 0.0, -0.06 and 0.02, the performance on these trees is close to average for MCD1). The model performs better on the untrained trees in MCD1, perhaps because of the similar trees in train; with no evidence of this

[7]Note that CFQ has two part-of-speech categories which are common nouns: a *role*, like the word "mother" in the phrase "Henry's mother", and a category labeled *noun*, like "person" in the phrase "a person".

Split	Num Trees removed	Sentences in reduced test	WSCLAS, StDv	CLAS, StDv	LAS, StDv	Num Sentences removed
mcd1	3	11,139	98.41 ±0.71	99.75 ±0.15	99.84±0.1	828
mcd2	5	8,440	92.46 ±3.82	98.59±0.47	98.89±0.37	3527
mcd3	7	5,692	93.09±2.77	98.79±0.50	98.91±0.36	6275
rand	0	11,967	99.89±0.01	99.98 ±0.00	99.99 ±0.00	0

Table 3: Re-calculated accuracy on the test sets, when removing all sentences whose subtrees are most predictive of Stanza's failure.

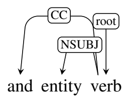
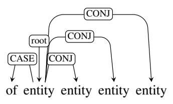
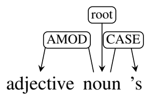
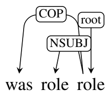
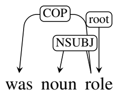
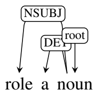
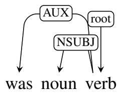
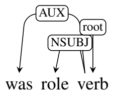
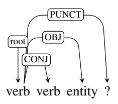
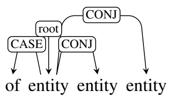
Tree	MCD1 Train #	Test #	Δ%	MCD2 Train #	Test #	Δ%	MCD3 Train #	Test #	Δ%
				Predictive for MCD1					
	2342	192	0.26	2099	144	-0.15	1546	118	0.01
	0	484	1.29	0	405	-0.15	0	382	1.78
	2319	137	0.23	3033	3	-0.0	3255	50	-0.1
				Predictive for MCD2 and MCD3					
	367	792	-0.0	0	940	5.73	0	1437	6.98
	0	530	-0.06	0	535	2.25	0	593	0.54
	0	541	0.02	0	547	2.38	0	604	2.01
	872	60	-0.02	0	512	2.94	0	547	2.61
	1484	231	-0.03	0	993	2.53	0	1093	3.06
				Predictive for MCD3					
	877	1513	0.1	451	1435	-2.53	355	1407	1.47
	0	1153	1.33	509	905	-0.97	0	931	3.95

Table 4: Syntactic constructions most predictive of dependency parsing failure for each split. "Predictive" means the subtree is associated with a coefficient ≤ -1 by the logistic model. # stands for the number of occurrences. $\Delta\%$ is defined as the change in mean WSCLAS after all instances of the construction have been removed from test.

kind of structure in MCD2 and MCD3, the model struggles.[8]

Another group of subtrees with similar syntactic structure is the second and last subtrees in Table 4. These coordinate three and four entities in an "of"-type prepositional phrase, which occurs in phrases like "the mother of Alice, Bob, Carl and Dave". Both trees are absent from MCD1 train, and both have a large effect on performance for MCD1 (Δ of 1.29 and 1.33). In MCD2, only the tree with four coordinated entities is absent from train, and it is not difficult for the model (Δ of -0.15, indicating that removing it from test reduces performance); the model is likely able to parse four coordinated entities based on the training examples with three coordinated entities.

7 Related work

A growing body of work uses CFQ to investigate better models for compositional generalization in semantic parsing (Herzig and Berant, 2021; Guo et al., 2020; Furrer et al., 2020). Tsarkov et al. (2020) also recently released an expanded version of CFQ called *-CFQ, which remains challenging for transformers even when they are trained on much more data. Our methodology can be easily be applied to *-CFQ at the cost of a straight-forward extension of the grammar.

Other datasets focused on compositional generalization include SCAN (Lake and Baroni, 2018), a dataset of English commands and navigation sequences; gSCAN (Ruis et al., 2020), a successor to SCAN with grounded navigation sequences; and COGS (Kim and Linzen, 2020), where English sentences are paired with semantic representations based on lambda calculus and the UDepLambda framework (Reddy et al., 2017). In contrast to CFQ, these datasets challenge models by targeting specific, linguistically-motivated generalizations. For example, COGS includes tests of novel verb argument structures (like training on a verb in active voice and testing in passive voice), and novel grammatical roles for primitives (like training with a noun in object position and testing in subject position); similarly, SCAN includes splits which test novel combinations of specific predicates (training a predicate "jump" or "turn left" in isolation, and testing it composed with additional predicates from train). Finally, the CLOSURE benchmark for visual question answering tests systematic generalization of familiar words by constructing novel referring expressions; for example, "a cube that is the same size as the brown cube" (Bahdanau et al., 2020).

8 Conclusion

In this paper, we presented a dependency parsing version of the Compositional Freebase Queries (CFQ) dataset. We showed that a state-of-the-art dependency parser's performance degrades with increased compound divergence, but varies on different splits of the same compound divergence. Finally, we showed the majority of the parser failures on each split can be characterized by a small (seven or fewer) number of specific syntactic structures.

To our knowledge, this is the first explicit test of compositional generalization in dependency parsing. We hope that the gold-standard dependency parses that we have developed will be a useful resource in future work on compositional generalization. Existing work on syntactic (and in particular dependency) parsing can provide researchers in compositional generalization with ideas and inspiration which can then be empirically validated using our corpus.

Finally, our work represents a step forward in understanding the syntactic structures which drive lower performance on MCD test sets. Predicting parser performance from the syntactic constructions contained in the question provides a new method for understanding the syntactic structures that can cause parser failure; in future work, similar methods can also be used to better understand failures of semantic parsers on the CFQ dataset.

Ethical Considerations

This article contributes to compositional generalization research, a foundational concern for neural natural natural language processing models. Breakthroughs in this research might eventually lead to smaller, and more efficient models, as well as better performance on low-resource languages. The ethical and societal consequences of these improvements will depend on downstream applications.

The resource released in this work is a new set of annotations for CFQ, an existing dataset. The origi-

[8]Not shown in Table 4 is the tree with a left-edge copula and simple nouns in both predicate and subject position. This structure was also absent from train in the MCD2 and MCD3 splits, but present in MCD1 train. It was not found to be strongly predictive of errors by the logistic model, likely because it was infrequent in test (occurring 188 times in MCD2 and 208 in MCD3).

nal CFQ dataset was artificially generated, so there was no process of data collection and therefore no ethics review process. The dataset was annotated by the author, so there was no ethics review of the annotation process or demographic information of this population to report.

Acknowledgements

We thank Christopher Manning, the Montreal Computational and Quantitative Linguistics lab at McGill University, and the Human and Machine Interaction Through Language group at ServiceNow Research for helpful feedback. We are grateful to ServiceNow Research for providing extensive compute and other support. We also gratefully acknowledge the support of the MITACS accelerate internship, the Natural Sciences and Engineering Research Council of Canada, the Fonds de Recherche du Québec, Société et Culture, and the Canada CIFAR AI Chairs Program.

References

Dzmitry Bahdanau, Harm de Vries, Timothy J. O'Donnell, Shikhar Murty, Philippe Beaudoin, Yoshua Bengio, and Aaron Courville. 2020. CLOSURE: Assessing Systematic Generalization of CLEVR Models. ArXiv:1912.05783. Version 2.

Michael Collins. 2003. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4):589–637.

CoNLL Shared Task. 2018. Evaluation. http://universaldependencies.org/conll18/evaluation.html.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. 2019. Universal transformers. In *International Conference on Learning Representations*.

Daniel Furrer, Marc van Zee, Nathan Scales, and Nathanael Schärli. 2020. Compositional Generalization in Semantic Parsing: Pre-training vs. Specialized Architectures. *arXiv:2007.08970*.

Google Research. 2020. Compositional freebase questions leaderboard. https://github.com/google-research/google-research/tree/master/cfq. Accessed March 12, 2022.

Yinuo Guo, Zeqi Lin, Jian-Guang Lou, and Dongmei Zhang. 2020. Hierarchical poset decoding for compositional generalization in language.

Jonathan Herzig and Jonathan Berant. 2021. Span-based semantic parsing for compositional generalization. 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 908–921, Online. Association for Computational Linguistics.

Jonathan Herzig, Peter Shaw, Ming-Wei Chang, Kelvin Guu, Panupong Pasupat, and Yuan Zhang. 2021. Unlocking Compositional Generalization in Pre-trained Models Using Intermediate Representations. *arXiv preprint arXiv:2104.07478*.

Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, Dmitry Tsarkov, Xiao Wang, Marc van Zee, and Olivier Bousquet. 2020. Measuring compositional generalization: A comprehensive method on realistic data. In *International Conference on Learning Representations*.

Najoung Kim and Tal Linzen. 2020. COGS: A compositional generalization challenge based on semantic interpretation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9087–9105, Online. Association for Computational Linguistics.

Brenden Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. volume 80 of *Proceedings of Machine Learning Research*, pages 2873–2882, Stockholmsmässan, Stockholm Sweden. PMLR.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Jan Hajič, Christopher D. Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, and Daniel Zeman. 2020. Universal Dependencies v2: An evergrowing multilingual treebank collection. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 4034–4043, Marseille, France. European Language Resources Association.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Siva Reddy, Oscar Täckström, Slav Petrov, Mark Steedman, and Mirella Lapata. 2017. Universal semantic parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 89–101, Copenhagen, Denmark. Association for Computational Linguistics.

Laura Ruis, Jacob Andreas, Marco Baroni, Diane Bouchacourt, and Brenden M. Lake. 2020. A benchmark for systematic generalization in grounded language understanding.

Dmitry Tsarkov, Tibor Tihon, Nathan Scales, Nikola Momchev, Danila Sinopalnikov, and Nathanael

Schärli. 2020. *-cfq: Analyzing the scalability of machine learning on a compositional task.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30.

A Correlation of Semantic Parsing and Dependency Parsing Errors

Because syntactic parsing is a necessary sub-task for semantic parsing, we also explored the possibility that dependency parsing errors might be predictive of semantic parsing errors. We extracted the predictions from Keysers et al. (2020)'s transformer model (which is based on Vaswani et al. (2017)'s model), and compared them to those of Stanza on the same test set. For each test sentence, we calculated the number of times the parsers correctly parsed the sentence (out of five experiments each).

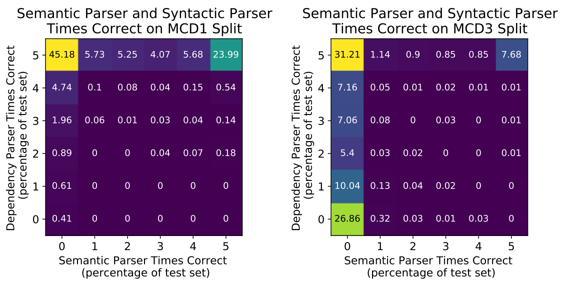


Figure 6: Percentage of the test set which is correctly parsed by five semantic parsing experiments and five dependency parsing experiments, for MCD1 and MCD3.

The results for MCD1 and MCD3 are shown in Figure 6: for example, the top-right hand corner of the MCD1 matrix means that 23.99% of the test set was correctly parsed in all semantic and dependency parsing experiments, while the top right-hand corner in the MCD3 matrix indicates that only 7.68% of the sentences were correctly parsed by both models in all experiments. The semantic parser fails for all five experiments on the majority of sentences. We do note some trends in error patterns between the models: for example, no sentences are correctly parsed by all semantic parsers without also being correctly parsed by the dependency parser at least a few times. However, overall it does not appear that dependency parsing performance is strongly related to semantic parsing performance.

B Proportion of Few-shot Constructions in MCD Splits

We examined whether the MCD splits differ in the proportion of test syntactic constructions which are *few-shot*, meaning they appear in train fewer than four times. This analyses is similar to the ones described in § 5.2.

The proportions are plotted in Figure 7. The MCD splits pattern together, with far more few-shot constructions at each complexity than the random split.

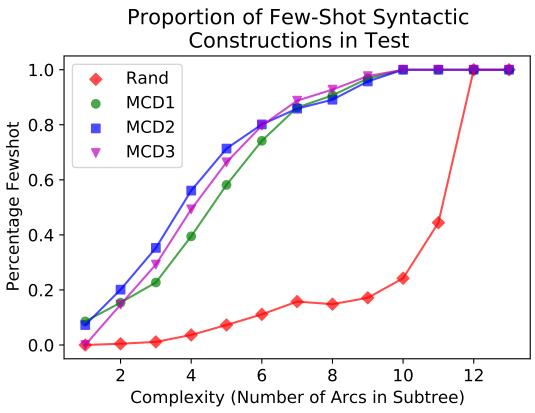


Figure 7: The proportion of syntactic constructions in test which appear in train fewer than four times, for all subtree complexities. Complexity is measured in number of arcs.

C CFG

Below are the rules in our Context Free Grammar. Using these rules we parsed CFQ into constituency trees, and then mapped to dependency trees as described in § 3.

S ⟶ NPQ VP Qmark
S ⟶ NPQ was Nominal Qmark
S ⟶ NPQ did NPV Qmark
S ⟶ was Nominal Vobl Qmark
S ⟶ NPQ Vobl Qmark
S ⟶ was Nominal Adj Qmark
S ⟶ was Nominal Nominal Qmark
S ⟶ did Nominal VP Qmark

NPV ⟶ Nominal V
NPV ⟶ Nominal VPrep

VP ⟶ V Nominal
VP ⟶ was Vobl
VPrep ⟶ was VPrep
VPrep ⟶ V by

Vobl $\longrightarrow$ VPrep Nominal
NPQ $\longrightarrow$ WhW Nominal
NPQ $\longrightarrow$ WhW role caseO

commonNoun $\longrightarrow$ commonNoun RC

RC $\longrightarrow$ Vobl
RC $\longrightarrow$ R VP
RC $\longrightarrow$ R NPV
RC $\longrightarrow$ whose role VP

VP $\longrightarrow$ VP andVP
VP $\longrightarrow$ VPx andVP
VPx $\longrightarrow$ VP punctVP
VPx $\longrightarrow$ VPx punctVP
andVP $\longrightarrow$ conj VP
andVP $\longrightarrow$ punct conj VP
punctVP $\longrightarrow$ punct VP

Vobl $\longrightarrow$ Vobl andVobl
Vobl $\longrightarrow$ Voblx andVobl
Voblx $\longrightarrow$ Vobl punctVobl
Voblx $\longrightarrow$ Voblx punctVobl
andVobl $\longrightarrow$ conj Vobl
andVobl $\longrightarrow$ punct conj Vobl
punctVobl $\longrightarrow$ punct Vobl

VPrep $\longrightarrow$ VPrep andVPrep
VPrep $\longrightarrow$ VPrepX andVPrep
VPrepX $\longrightarrow$ VPrep punctVPrep
VPrepX $\longrightarrow$ VPrepX punctVPrep
andVPrep $\longrightarrow$ conj VPrep
andVPrep $\longrightarrow$ punct conj VPrep
punctVPrep $\longrightarrow$ punct VPrep

V $\longrightarrow$ V andV
V $\longrightarrow$ Vx andV
Vx $\longrightarrow$ V punctV
Vx $\longrightarrow$ Vx punctV
andV $\longrightarrow$ conj V
andV $\longrightarrow$ punct conj V
punctV $\longrightarrow$ punct V

Vx $\longrightarrow$ Vx punctVPrep
Vx $\longrightarrow$ V punctVPrep
V $\longrightarrow$ Vx andVPrep
V $\longrightarrow$ V andVPrep
VPrep $\longrightarrow$ VPrep andV
VPrep $\longrightarrow$ VPrepX andV
VPrepX $\longrightarrow$ VPrep punctV
VPrepX $\longrightarrow$ VPrepX punctV

NPV $\longrightarrow$ NPV andNPV
NPV $\longrightarrow$ NPVx andNPV
NPVx $\longrightarrow$ NPV punctNPV
NPVx $\longrightarrow$ NPVx punctNPV
andNPV $\longrightarrow$ conj NPV
andNPV $\longrightarrow$ punct conj NPV
punctNPV $\longrightarrow$ punct NPV

V $\longrightarrow$ F V

Nominal $\longrightarrow$ Name
Nominal $\longrightarrow$ DP
Nominal $\longrightarrow$ commonNoun

DP $\longrightarrow$ caseS role
caseS $\longrightarrow$ DP pS
caseS $\longrightarrow$ Name pS
DP $\longrightarrow$ det role caseO
caseO $\longrightarrow$ of DP
caseO $\longrightarrow$ of Name

DP $\longrightarrow$ det commonNoun

Name $\longrightarrow$ Name andName
Name $\longrightarrow$ Namex andName
Namex $\longrightarrow$ Name punctName
Namex $\longrightarrow$ Namex punctName
andName $\longrightarrow$ conj Name
andName $\longrightarrow$ punct conj Name
punctName $\longrightarrow$ punct Name

commonNoun $\longrightarrow$ commonNoun andCommonNoun
commonNoun $\longrightarrow$ commonNounx andCommonNoun
commonNounx $\longrightarrow$ commonNoun punctCommonNoun
commonNounx $\longrightarrow$ commonNounx punctCommonNoun
andCommonNoun $\longrightarrow$ conj commonNoun
andCommonNoun $\longrightarrow$ punct conj commonNoun
punctCommonNoun $\longrightarrow$ punct commonNoun

role $\longrightarrow$ role androle
role $\longrightarrow$ rolex androle
rolex $\longrightarrow$ role punctrole
rolex $\longrightarrow$ rolex punctrole
androle $\longrightarrow$ conj role
androle $\longrightarrow$ punct conj role

punctrole $\longrightarrow$ punct role

commonNoun $\longrightarrow$ F commonNoun
role $\longrightarrow$ F role
role $\longrightarrow$ Cnt of nat
commonNoun $\longrightarrow$ P commonNoun

commonNoun $\longrightarrow$ Adj commonNoun
role $\longrightarrow$ Adj role

punct $\longrightarrow$,
Cnt $\longrightarrow$ country
nat $\longrightarrow$ nationality
P $\longrightarrow$ production
F $\longrightarrow$ film | art | executive | costume
V $\longrightarrow$ VP_SIMPLE | direct | produce ...
Name $\longrightarrow$ entity | Alice | Bob ...
commonNoun $\longrightarrow$ NP_SIMPLE | character | person ...
role $\longrightarrow$ ROLE_SIMPLE | character | person ...
NPQ $\longrightarrow$ who | what
WhW $\longrightarrow$ What | Which | what | which
did $\longrightarrow$ did | Did
conj $\longrightarrow$ and
pS $\longrightarrow$ ‘s
of $\longrightarrow$ of
det $\longrightarrow$ a | an
by $\longrightarrow$ by
Adj $\longrightarrow$ ADJECTIVE_SIMPLE | female | American ...
was $\longrightarrow$ was | were
R $\longrightarrow$ that
whose $\longrightarrow$ whose
Qmark $\longrightarrow$?