Grammar-assisted neural semantic parsing

Maksims Galkins

Master of Science
Cognitive Science
School of Informatics
University of Edinburgh
2024

Abstract

Neural seq2seq semantic parsers are known to struggle with compositional generalisation - that is, parsing familiar elements recombined in novel ways, unseen during training. We propose a simple method of addressing the issue by providing the model with supertags produced by Combinatory Categorial Grammar parser, interleaved with natural language tokens. We train 3 models based on CodeT5 to evaluate the method: baseline, trained on natural language only, supertagged, trained with our method and, as a control condition, padded, where utterances are interleaved with meaningless tags. We show our method produces a significant improvement over other conditions **if** the embeddings for the supertags are pre-trained. We also conduct an error analysis, showing that most significant improvement happens in cases of verb ambiguity, which CCG supertagging is effective at resolving.

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

[illegible]

(*Maksims Galkins*)

Acknowledgements

Writing a Master's dissertation is not a trivial undertaking, so there are a lot of acknowledgements to give. Firstly I would like to thank Dr Mark Steedman for very precise and helpful guidance. Matt Grenander and Tianyi Li provided invaluable technical advice. Dr Milos Stanojevic (the creator of the CCG supertagger used in this work) took the time to advise me on setting it up correctly. I also thank Matthias Lindemann for advice on working with T5. My good friends and coursemates George Fairs and Dmitrii Tiron contributed by listening to me complain about something not working at least weekly, thus providing some emotional support, which they had to do while writing their own dissertations. I thank my parents for supporting me along the way and also paying for all the food, water and shelter I needed to conduct this work.

Table of Contents

Chapter 1

Introduction

One of the key subfields in Natural Language Processing is **semantic parsing**, defined broadly as mapping natural language (NL) utterances to semantic machine-readable representations. Such representations can be executable (e.g. database queries) or non-executable (e.g. lambda calculus, First Order Logic (FOL), graphs). An accurate and reliable system of domain-independent semantic parsing would be of tremendous value. The field of machine translation could make use of reliable semantic information to improve translations, new opportunities would emerge for large-scale knowledge base building, and some subfields of NLP, such as argument mining (Lawrence and Reed, 2019), would be solved.

Neural sequence-to-sequence models have been applied to the task of semantic parsing to good results, outperforming the previous generation of statistical, grammar-based semantic parsers. However, the relaxation of constraints introduced with the neural architectures brings several new issues. One of such issues is lack of consistent ability of the model to combine previously seen components in novel ways (**compositional generalisation**, CG). For example, successfully parsing “I swam in the sea” and “river” does not guarantee a correct parse for “I swam in the river”. Compositional generalisation is trivial for humans and is critical to our ability to learn language and to the expressive power of language, but remains out of reach for neural semantic parsers.

Current approaches to tackling issues with compositional generalisation employ complex specialised architectures (Lindemann et al., 2023) or ensembling (Shaw et al., 2021, Qiu et al., 2022). Taking inspiration from earlier grammar-based semantic parsers (Zettlemoyer and Collins, 2012, Kwiatkowksi et al., 2010) and recent exploration into providing neural models with Combinatory Categorial Grammar (CCG) (Steedman and Nivre, 2000) information (Nadejde et al., 2017, Arehalli and Linzen, 2024), we propose and evaluate a simple approach to improving compositional generalisation performance by fine-tuning a pre-trained LLM learner on a semantic parsing task where the NL utterances are interleaved with CCG tags.

We show (1) the model is unable to simultaneously learn useful embeddings for CCG tag tokens during fine-tuning on a semantic parsing task but (2) providing **pre-trained embeddings** for the CCG tags before fine-tuning results in the model noticably improving. We also provide an error analysis, highlightling in what cases our method works best.

Chapter 2

Background

This work combines developments from to branches of investigation, which we describe in turn below.

2.1 Combinatory Categorial Grammars

Combinatory Categorial Grammar (CCG) (Steedman and Nivre, 2000) is a grammar formalism used to model language. As part of parsing an utterance with CCG, words and utterance fragments get assigned a a category which describes them as functions, specifying the input argument and output. Word-level categories are referred to as supertags.

For example, in a sentence

```
Pavels befriended Eriks
```

the word "befriended" is assigned category `(S[dcl]\NP)/NP`, meaning this verb outputs a declarative sentence upon taking a a noun phrase to its right followed by a noun phrase to its left. The word-level assigned CCG categories are referred to as *supertags*.

Efficient and accurate CCG parsers (e.g. Stanojević and Steedman (2019)) make it possible to employ the information contained in supertags in a wide variety of downstream NLP tasks.

2.2 Semantic parsing

2.2.1 History

A broad definition of the task of semantic parsing is translating natural language utterances into meaning representations, which can be divided into executable representations (e.g. SQL queries) and non-executable representations, such as first order logic or graphs. Semantic parsing is different from other sequence-to-sequence tasks in a sense that the desired output is structured (Kamath and Das, 2018), yielding both unique challenges and opportunities.

Very early semantic parsing systems were rule-based. An early example is SAVVY, described in Johnson (1984), using a pattern recognition engine to map utterances to query for a database. An example of a grammar-based approach is EUFID (Templeton and Burger, 1983), employing a hand-crafted ”semantic grammar” and a multi-step process of converting an NL utterance to a parse tree with CYK algorithm, and eventually a database query. Being rule-based, such systems inherently provide good compositional generalisation abilities, but are limited in general usability.

Hand-crafted systems were outperformed by statistical semantic parsers (see Zelle and Mooney (1996) for an early example), even with training sets as small as 250 examples. CCG formalism was often employed in statistical semantic parsers, e.g. Zettlemoyer and Collins (2012) and Kwiatkowksi et al. (2010), where a Probabilistic CCG was learned from data.

Further improvements came with neural semantic parsing, framed as a sequence-to-sequence task, forgoing hand-crafted features and grammars. Most works employ LSTMs in encoder-decoder formation (Dong and Lapata, 2016) however later works use transformers.

Neural semantic parsers are inherently less constrained than previous approaches, displaying non-negligible amount of ungrammatical parses. Different ways of addressing this have been investigated; in Dong and Lapata (2016) the decoder is augmented to build tree structures hierarchically, level by level, employing additional connections to the parent node of the generated tree (”parent feeding”). Wang et al. (2019) and Yin and Neubig (2017) use rule based decoding, where instead of sequences the decoder outputs production rules.

Another major issue facing neural semantic parsers is lack of compositional generalisation (Lake and Baroni, 2017), which is described in detail in the next section.

2.2.2 Compositional Generalisation

Broadly, compositional generalisation refers to producing novel combinations out of known components (Lake and Baroni, 2017). This is something humans do effortlessly - knowing separately the utterance ”I swam in the sea” and the noun ”river”, it is trivial for us to understand the utterance ”I swam in the river”. Seq2seq neural models, however, struggle with this task (Lake and Baroni, 2017).

Several approaches to improving the situation have so far been attempted.

Ensembling. One intuitive idea is to ensemble a constrained grammar-based model with a neural model. Shaw et al. (2021) separately induce a QCFG (Quasi-synchronous CFG) for a grammar-powered neural parser and fine-tune T5. The two models are then put into an ensemble using a sort of fallback mechanism - when grammar-based model fails to produce an output, the ensemble returns the output of T5.

Data augmentation. Qiu et al. (2022) first train a QCFG-powered generative model on original training data to generate recombinations of atoms from the training data, then fine-tune T5 on a union of original and generated utterances. This ”distills” (Qiu et al., 2022) the knowledge of compositional structures from the generative model into T5, thus allowing it better compositionally generalise. This outperforms the ensemble of Shaw et al. (2021).

Trees. A range of approaches make the underlying tree structure of the logical form more explicit. Herzig and Berant (2021a) train a classifier to classify **spans** of NL utterances with categories, which are then built into a tree with CYK. Their approach, however, requires setting up ”domain constants” which are used as part of the possible categories during prediction, which reduces the flexibility of their model.

Multi-layer models. Lindemann et al. (2023) utilize the concept of *fertility* of a word in translation contexts (Brown et al., 1990) and produce a novel multi-level architecture with several predictions and a reordering step. The model suffers from high computational costs of the reordering layer for longer output sequences.

2.3 Related work outside semantic parsing

To our knowledge, no attempts in employing CCG supertags in neural semantic parsing have been made yet.

However, although not numerous, some work in combining neural models with CCG supertags has been conducted. Nadejde et al. (2017) supertag the **target sequence** in a machine translation task. This forces the model to ”think about” the syntactic structures explicitly, improving the translation quality. They experiment with both **interleaving** the supertags and having the supertag sequence as a secondary prediction task, showing that only interleaving produces improvements. They additionally experiment with providing CCG supertags in the input sequence, yielding a strong improvement too (e.g. 0.9 BLEU for English−>German).

Chapter 3

Methods

3.1 Model

We employ CodeT5 (Wang et al., 2021) as our starting model. CodeT5 builds up on T5 pre-trained model (Raffel et al., 2019) by fine-tuning it on code generation tasks. The selection of CodeT5 as a launch point is motivated by the intuition of relative similarity of semantic parsing to code generation, as both necessitate some level of semantic processing. We thus hope CodeT5 serves as a better foundation than an average model at the task of semantic parsing.

In particular, we run all the experiments on codeT5-small for reasons of limited computational and time resources available for the project. codeT5-small has 60,492,288 parameters and a vocabulary of 32100, such that the last 100 entries are mask tokens. codeT5-small tokenizer splits out-of-vocabulary (OOV) tokens into subword units, i.e. the number of words in a sentence does not always match the length of the output vector of the tokenizer.

3.2 Data

3.2.1 Dataset

COGS (Kim and Linzen, 2020) is a semantic parsing dataset specifically engineered for experiments on the ability of neural semantic parsing models to compositionally generalise. COGS contains a collection of pairings between natural language utterances (NL) and a logical formalism (LF) based on lambda calculus. More specifically, the lambda calculus variation used is `UDepLambda` (Reddy et al., 2017), a lambda calculus-

Partition	# examples	Notes
train	24154	
dev	3000	
test	3000	
gen	21000	
val_gen	250	Created by the authors

Table 3.1: Summarisation of COGS dataset (Kim and Linzen, 2020).

like language expressing Universal Dependencies v1 (Nivre et al., 2016).

For example, the train partition of COGS contains a pair

```
A rose was helped by a dog.

rose ( x_1 ) AND help . theme ( x_3 , x_1 ) AND
help . agent ( x_3 , x_6 ) AND dog ( x_6 )
```

In addition to the standard train/dev/test split, a forth subset is provided as **gen**, such that every sentence in gen requires compositional generalisation to be parsed correctly.

Sentences in gen set are marked with one of 21 types, signifying the type compositional generalisation required for them to be parsed correctly. The 21 types can be grouped into 5 categories.

Novel Combination of Familiar Primitives and Grammatical Roles (FPGR) refers to sentences where a word seen in the training set needs to processed in a different role, e.g. ”shark” is only ever a subject in the training set but an object in gen set.

Sentences in **Novel Combination Modified Phrases and Grammatical Roles (MPGR)** category require the model to parse a familiar modified word in a new role, e.g. ”a grey shark”.

Deeper Recursion (DEEP) category contains sentences with recursion deeper than seen in the training set, where recursion means additional ”that”-like phrases. For example, if ”A knew that B knew that X” is the longest sentence in the training set, sentences of the form ”A knew that B knew that C knew that X” and longer would be contained in the gen set.

Verb Argument Structure Alternation (VASA) category sentences modify the verbs seen in training set in novel ways. For example, the verb ”blessed” is only ever used in an active sense in the train set, but is employed as a passive verb in the gen set, and vice versa.

Dataset	Example NL sentence
COGS	`A rose was helped by a dog .`
Supertagged COGS	`A ccg_NP/N rose ccg_N was [...]`
Padded COGS	`A DU/D rose DU/D was [...]`

Table 3.2: Excerpts from COGS and two datasets derived from it. Each NL utterance is paired with its semantic form, which remains unchanged between datasets.

Final category is **Verb class (VCLASS)** contains sentences with previously unseen verb ”classes”, such that the `gen` set introduces a semantic role previously unseen semantic role for a previously seen verb. For example, the utterances `The hippo decomposed.` and `The hippo painted.` are syntactically similar, but imply a different level of agency from the hippo.

From gen partition we take the first 250 examples as our validation set, the `val_gen` partition. This is necessary as the validation set provided with COGS only contains in-distribution sentences, and thus can not be used for measuring compositional generalisation performance. `val_gen` is used for in-training evaluation to produce EM accuracy curves. Table 3.1 summarises the 5 partitions. A more detailed description of the 21 subtypes and 5 categories is provided in Kim and Linzen (2020).

3.2.2 Preprocessing

In addition to standard COGS, we produced two derivative datasets - a supertagged COGS and a padded COGS.

Supertagged COGS is a copy of COGS with every word in the sentences followed a CCG supertag, carrying information of that word’s syntactic properties. We used `ccg-tools` package by Miloš Stanojević[1] to produce the CCG supertags for each token.

Padded COGS replaces the CCG supertags with dummy tags of the form ”DU/D”, intended as a control condition to measure how explicitly delimiting words in a sentence with special tokens affects semantic parsing performance without the tokens carrying any meaning.

Since codeT5 tokenizer processes OOV tokens by breaking them down into subwords, there exists an overlap between simpler CCG supertags such as `N` or `NP` and subword tokens which are already in codeT5 vocabulary. To mitigate this overlap, all

[1]https://github.com/stanojevic/ccgtools

Model Name	Trained on	Pre-trained embeddings
codeT5-NL-only	COGS	n/a
codeT5-padded	Padded COGS	n/a
codeT5-ccg-naive	Supertagged COGS	No
codeT5-ccg-pretrained	Supertagged COGS	Yes

Table 3.3: List of all the models we train for this project, indicating their main differences.

supertags are preceded by `ccg_` prefix. Table 3.2 shows examples of one sentence excerpt per dataset, some sentences cropped to fit.

3.3 Training procedure

3.3.1 Tokenizer preparation

For both supertagged and padded datasets, we alter the tokenizer to process the new tokens (CCG supertags and dummy tags) as special tokens, i.e. not decompose them into subword units. This operation expands the model vocabulary and initialises the embeddings of the new tokens with random vectors.

We conduct experiments with both randomly initialised embeddings for CCG supertags **(naive condition)** and with pre-trained embeddings **(pre-trained condition)**.

To obtrain the pre-trained embeddings for the supertags, we fine-tune a separate instance of codeT5-small on a masked supertag prediction task. A model is given a sentence from Supertagged COGS training set with one of the supertags masked:

```
The ccg_NP/N captain ccg_N ate [MASK] . ccg_.
```

and is expected to return the missing supertag:

```
ccg_S[dcl]\NP
```

We fine-tuned the model for 5 epochs with learning rate of 5e-5, yielding a `0.995` accuracy on the validation set. We then extract the embeddings for CCG supertags **only** and transplant them into another instance of codeT5-small, which we then fine-tune on the semantic parsing task. The process is described in 3.4.

Step	Action
1	Set up a new instance of codeT5-small (codeT5-A)
2	Extend the vocabulary of codeT5-A with special symbols for CCG supertags
3	Fine-tune codeT5-A on masked supertag prediction task.
4	Extract **only** the embeddings for the supertags from codeT5-A into a matrix `M`
5	Set up a new instance of codeT5-small (codeT5-B)
6	Extend the vocabulary of codeT5-B with special symbols for CCG supertags
7	Replace the randomly initialised embeddings with the supertags from `M`
8	Train codeT5-B on the semantic parsing task.

Table 3.4: Steps involved in obtaining the pre-trained embeddings for CCG supertags and training the semantic parser with them. No unfair advantage is provided to the semantic parser model as, with the exception of the CCG supertag embeddings, it is a fresh codeT5 which has not seen any part of the dataset.

3.4 Fine-tuning

We fine-tune codeT5-small separately on the training partition of each of our 3 datasets (24155 NL-LF pairs), using a batch size of 12 for training. Supertagged COGS was fine-tuned for twice, once in naive supertags condition and once with pre-trained embeddings transplanted. For the sake of brevity, we refer to the fine-tuned models with a `codeT5-*` convention, e.g. a model fine-tuned on the Supertagged COGS with pre-trained embeddings we refer to as `codeT5-ccg-pretrained`. Our 4 models are summarised in Table 3.3.

For most experiments we employed a learning rate of 5e-6, however we experimented with a higher learning rate of 1e-5 for `codeT5-ccg-pretrained`.

All training is conducted on the University of Edinburgh teaching cluster, using HuggingFace transformers library (Wolf et al., 2020). We employ AdamW (Loshchilov and Hutter, 2017) optimizer with a linear learning rate scheduler (without warmup).

3.5 Evaluation

Following other work on COGS (Lindemann et al. 2023) and generally compositional generalisation (Oren et al. 2020, Herzig and Berant 2021b) we employ exact match (EM) accuracy as our evaluation metric.

Chapter 4

Results

4.1 Naive supertags

Naively employed supertags fail to outperform padded condition, but outperform unaltered natural language on validation set. Figure 4.1 provides validation loss curves and EM accuracy on validation set over training.

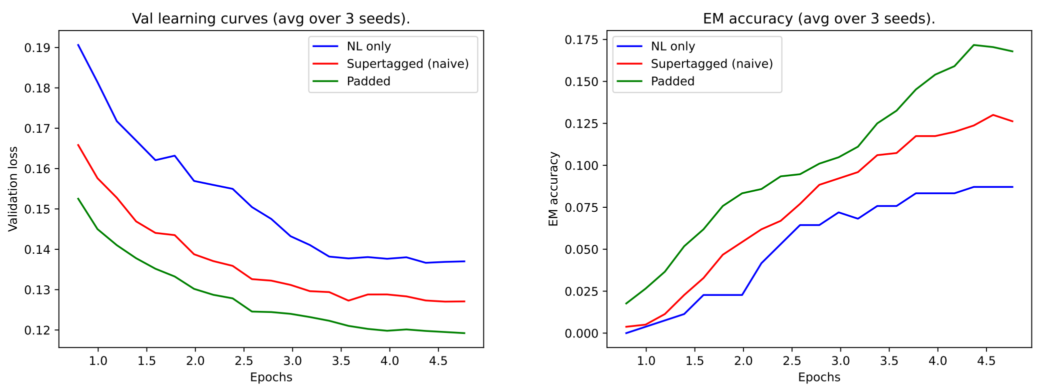

Figure 4.1: Validation loss and EM accuracy loss curves for our 3 semantic parsers. Blue denotes the parser training on natural language only source utterances, green denotes parser training on NL source utterances interleaved with padding tags and red denotes the model training on CCG supertagged NL source utterances. Training for 5 epochs with lr of 5e-6.

While `codeT5-padded` performing best may seem unintuitive at first, note that OOV tokens are broken up into subword units in codeT5. The padding tokens (`DU/D`) negate that effect by demarcating words in a simple way, giving the model a kind of demarcation of logical entities. `codeT5-NL-only` does not enjoy this additional benefit.

`codeT5-ccg-pretrained` likely underperforms since the model fails to learn useful embeddings for the supertags, while supertags themselves due to their diversity do not accomplish the word demarcation role as well as the `DU/D` tags.

4.2 Pre-trained supertags

To even the playing field, we rid the model of the need to simultaneously learn embeddings for the supertags and learn semantic parsing. We provide the model pre-trained embeddings for each CCG tag.

Pre-trained supertag condition (`codeT5-ccg-pretrained`) outperforms all other conditions by a large margin, as presented in Figure 4.2.

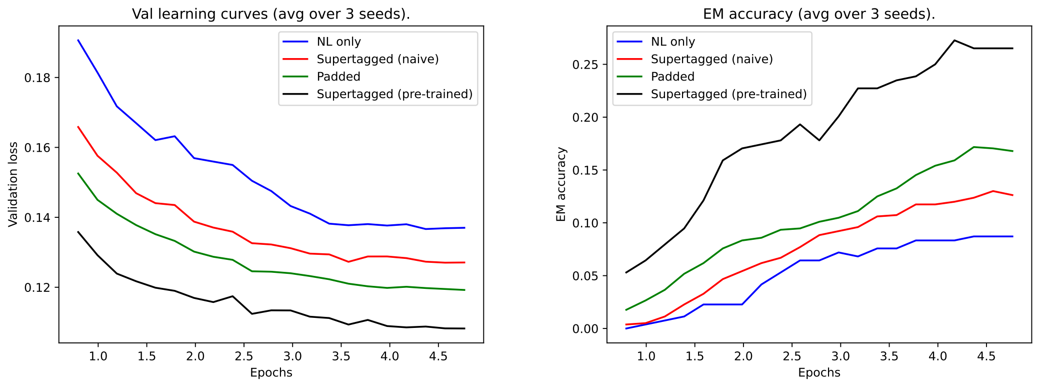

Figure 4.2: Figure 4.1 augmented with the metrics for the fourth model - semantic parser training on CCG supertagged NL utterances **with** pre-trained embeddings provided for supertag tokens (**black**).

4.3 Longer training

Having established a strong effect from providing a semantic parser with CCG supertags (with pre-trained embeddings), we ran an experiment with a higher learning rate (1e-5) for longer (10 epochs) to see whether the effect scales.

The results are curious - `codeT5-NL-only` eventually catches up and outperforms `codeT5-padded` and gets closer to `codeT5-ccg-pretrained`n.

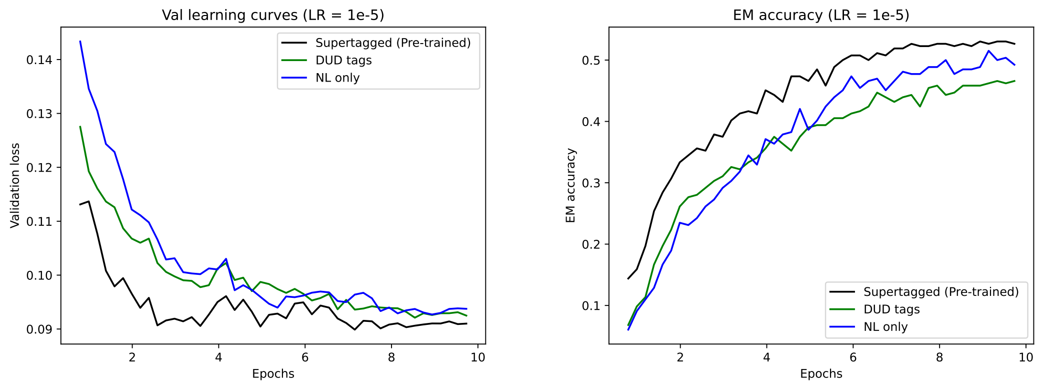

Figure 4.3: Validation loss and EM accuracy loss curves for 3 semantic parsers. Blue denotes the parser training on natural language only source utterances, green denotes parser training on NL source utterances interleaved with padding tags and **black** denotes the model training on CCG supertagged NL source utterances **with pre-trained embeddings** for the supertag tokens. Training for 10 epochs with lr of 1e-5.

Figure 4.3 provides the validation loss curves and EM accuracy curves for the three models. Despite the improved performance of `codeT5-NL-only`, the provision of CCG supertags still clearly outperforms all other conditions.

Model	Accuracy (EM)					
	Total	FPGR	MPGR	DEEP	VASA	VCLASS
codeT5-NL-only	0.579	0.892	0.0	0.002	0.228	0.918
codeT5-padded	0.534	0.871	0.0	>0.001	0.220	0.681
codeT5-CCG-pretrained	**0.609**	**0.896**	0.0	**0.007**	**0.306**	**0.954**

Table 4.1: Comparison of EM accuracy between the 3 models on gen set of COGS, presented as total (unweighted) accuracy and by CG category.

Table 4.1 presents the `gen` set results for all four models trained, reported in total and for each of the five generalisation categories. VASA and VCLASS categories see the greatest improvement, which we discuss in more detail in Section 4.4, along with a theory for why `codeT5-NL-only` sees an improvement of performance.

4.4 Error analysis

We provide examples of frequent errors the models make. We look at two cases : (1) where `codeT5-ccg-pretrained` is the **only model** producing a correct parse (except naive supertags, which we do not consider here) and (2) where **all other conditions** produce a correct parse *except* `codeT5-ccg-pretrained`. The error analysis is performed on errors made by models train as described in Section 4.3.

We provide examples of errors from each of our categories except MPGR and DEEP. We do not provide any examples from the class `DEEP` as they are lengthy and would not provide new error kinds, as the errors happening within the `DEEP` subcategory are also occurring in other classes. For `MPGR` there is nothing to report at all, as all models fail to produce a correct parse.

4.4.1 Case 1

Original utterance:	The student thought that a cockroach planned to sketch .
Supertagged utterance:	`The ccg_NP/N student ccg_N thought ccg_(S[dcl]\NP)/S[em] that ccg_S[em]/S[dcl] a ccg_NP/N cockroach ccg_N planned ccg_(S[dcl]\NP)/(S[to]\NP) to ccg_(S[to]\NP)/(S[b]\NP) sketch ccg_S[b]\NP . ccg_.`
CG class	FPGR
CCG-assisted output	* student (x _ 1) ; think . agent (x _ 2 , x _ 1) AND think . ccomp (x _ 2 , x _ 6) AND cockroach (x _ 5) AND plan . agent (x _ 6 , x _ 5) AND plan . xcomp (x _ 6 , x _ 8) AND sketch . agent (x _ 8 , x _ 5)
Baseline output	* student (x _ 1) ; think . agent (x _ 2 , x _ 1) AND think . ccomp (x _ 2 , x _ 6) AND cockroach (x _ 5) AND plan . agent (x _ 6 , x _ 5) AND plan . xcomp (x _ 6 , x _ 8) AND sketch . agent (x _ 8 , x _ 10)

Table 4.2: Error example 1. Non-CCG models frequently make variable alignment errors, in other words the ”mix up” indices. CCG-assisted model correctly identifies the cockroach as the entity potentially about to sketch, whereas non-CCG models fail to do so.

Most frequent error the models that do not employ CCG tags make is variable alignment errors, i.e. mixing up indices. We provide a full example in Table 4.2 (note in this subsection, ”CCG-assisted output” always exactly matches the gold standard parse); it occurs in about 33% of cases where `codeT5-ccg-pretrained` is the only model making a correct prediction. Estimating the exact rate at which it **causes** the error is difficult as it often co-occurs with other types, but it is a frequent error non-CCG models make.

A more interesting error is `theme/agent` error non-CCG models make, illustrated in Table 4.3. Here we see that the CCG supertags aid the model in distinguishing between semantic roles - whether the verb refers to its subject as an agent or not. In 4.5 illustrates this - the verb ”examined” here is intransitive, rather than passive (meaning the shark did the examining instead of being a patient), which the supertagger correctly picks up. This allows our model to assign the shark the role of an agent, whereas the non-CCG models fail to correctly deduce whether the verb is employed passively or as an intransitive.

Original utterance: Supertagged utterance:	A shark examined . `A ccg_NP/N shark ccg_N examined ccg_S[dcl]\NP . ccg_.`
CG class	FPGR
CCG-assisted output Baseline output	shark (x _ 1) AND examine . agent (x _ 2 , x _ 1) shark (x _ 1) AND examine . theme (x _ 2 , x _ 1)

Table 4.3: Error example 2. The model utilising CCG tags exhibits an ability to distinguish between ”examined” as an intransitive verb or a passive verb. ”examined” here is intransitive, and the parser correctly identifies the shark as the agent. CCG-assisted output exactly matches gold standard.

Similarly, the presence of CCG supertags allows the model to infer relations with higher granularity, as seen in Table 4.4. We observe that `codeT5-ccg-pretrained` was able to explicitly highlight Olivia as a recipient of the action ”ship” and the box (i.e. the thing being shipped) as the theme, whereas non-CCG models consider Olivia a theme of the action and do make the box explicitly related to the act of shipping. This is because the word ”ship” is supertagged as *ditransitive*, meaning it expects one subject and **two** objects, one of which the model can infer as the recipient.

An interesting and unexpected error non-CCG models make is spelling and tense errors, e.g. using `baked . agent` instead of the correct `bake . agent` , or `cob ( x _ 1 )` instead of `cobra ( x _ 1 )` . We suspect both `codeT5-NL-only` and `codeT5-padded` are able to learn to do this correctly eventually and produce correct parses, but it is of interest that providing CCG supertags aids faster learning in this seemingly unrelated way. These errors likely account of the narrowing of the gap between `codeT5-ccg-pretrained` and other models with higher learning rate and longer training.

Original utterance:	The boy respected that Samuel shipped Olivia the box in the room .
Supertagged utterance:	`The ccg_NP/N boy ccg_N respected ccg_(S[dcl]\NP)/S[em] that ccg_S[em]/S[dcl] Samuel ccg_N shipped ccg_((S[dcl]\NP)/NP)/NP Olivia ccg_N the ccg_NP/N box ccg_N in ccg_((S\NP)\(S\NP))/NP the ccg_NP/N room ccg_N . ccg_.`
CG class	VASA
CCG-assisted output	boy (x _ 1) ; * box (x _ 8) ; * room (x _ 11) ; respect . agent (x _ 2 , x _ 1) AND respect . ccomp (x _ 2 , x _ 5) AND ship . agent (x _ 5 , Samuel) AND ship . recipient (x _ 5 , Olivia) AND ship . theme (x _ 5 , x _ 8) AND box . nmod . in (x _ 8 , x _ 11)
Baseline output	* boy (x _ 1) ; * box (x _ 8) ; * room (x _ 11) ; respect . agent (x _ 2 , x _ 1) AND respect . ccomp (x _ 2 , x _ 5) AND ship . agent (x _ 5 , Samuel) AND ship . theme (x _ 5 , Olivia) AND box . nmod . in (x _ 5 , x _ 8)

Table 4.4: Error example 3. Non-CCG models are unable to process the details of a relation to the same granularity as the model employing CCG supertags. The verb ”ship” is supertagged as ditransitive, allowing the model to infer that it expects two objects, one of which (Olivia) is the recepient. CCG-assisted output exactly matches gold standard.

4.4.2 Case 2

On the flip side, there are errors that **only** `codeT5-ccg-pretrained` makes, and these are of three types. The first type, again, is variable alignment errors (same type as example in Table 4.2). These are, however, much rarer in this case, making only 11.5% of total errors. The other two types are presented below. In this section, the NL only parses serve a role of the gold standard.

In cases of error of the supertagger, the error propagates to our model as well. Example in Table 4.5 shows a sentence where two distinct individuals (Ava and Paula) are interpreted by the supertagger to be one entity (in the same sense as in phrase ”dog bone”), and the verb ”sent” is supertagged as transitive (where it should be ditransitive).

Original utterance: Supertagged utterance:	Sophia sent Ava Paula . `Sophia ccg_N sent ccg_(S[dcl]\NP)/NP Ava ccg_N/N Paula ccg_N . ccg_.`
CG class	FPGR
CCG-assisted output Baseline output	send . agent (x _ 1 , Sophia) AND send . theme (x _1 , Ava) AND send . recipient (x _ 1 , Paula) send . agent (x _ 1 , Sophia) AND send . recipient (x _ 1 , Ava) AND send . theme (x _ 1 , Paula)

Table 4.5: Error example 3. Errors in the supertagger output propagate to the semantic parser output. Here two distinct entities were supertagged to be one (”Ava Paula” being one entity in the same way a ”dog bone” is one entity), leading to an incorrect output.

Original utterance: Supertagged utterance:	The professor confessed that a hippo ate . `The ccg_NP/N professor ccg_N confessed ccg_(S[dcl]\NP)/S[em] that ccg_S[em]/S[dcl] a ccg_NP/N hippo ccg_N ate ccg_S[dcl]\NP . ccg_.`
CG class	VCLASS
CCG-assisted output Baseline output	* professor (x _ 1) ; confessor (x _ 2) ; confessor . agent (x _ 2 , x _ 1) AND confessor . ccomp (x _ 2 , x _ 6) AND hippo (x _ 5) AND eat . agent (x _ 6 , x _ 5) * professor (x _ 1) ; confess . agent (x _ 2 , x _ 1) AND confess . ccomp (x _ 2 , x _ 6) AND hippo (x _ 5) AND eat . agent (x _ 6 , x _ 5)

Table 4.6: Error example 4. Semantic parser employing CCG supertags in rare cases may hallucinate new entities, which we think is a sequence-to-sequence noise artifact appearing due to the task of processing a supertagged sequence being more complex.

The second type of error (which is not very frequent but interesting) is ”hallucination” of new entities by the LLM learner, as displayed in Table 4.6. `codeT5-ccg-pretrained` introduces a new entity, ”the confessor”, which does the actions that the professor is supposed to do. The parsing of a supertagged sentence is inherently a harder problem (due, at least, to increased length), thus the LLM learner sometimes might get confused.

Chapter 5

Discussion

5.1 Limitations

The most prominent limitation of this work is that, due to time and computational constraints, we did not set out to measure our effect on any SOTA-level models. We did not perform a comprehensive parameter search to achieve best possible accuracy for any of the conditions, and we did not run any experiments on large models (e.g. codeT5-large). This also prevents us from comparing our results with published work on COGS (e.g. Lindemann et al., 2023), as published works use at least `base` size models.

COGS is also artificially constructed, meaning it does not contain some phenomena observed in natural language. In fact, Supertagged COGS only contains **27** unique CCG supertags, implying a low variety in syntactic structures. Compositional generalisation performance may be drastically different on a dataset of organic utterances, although a dataset that is both organic and allows for CG testing (to our knowledge) does not yet exist. Organically gathered sentences could also give an unfair advantage to LLMs as those sentences may have been obvserved during pre-training, which is part of the reason for making COGS artificially constructed (Kim and Linzen, 2020).

An extension of COGS, SLOG (Li et al., 2023) adds 17 new types of structural generalisation, where a model has to process unfamiliar syntactic structures (e.g. deeper recursion). We were not able to conduct any tests on SLOG as the available CCG parsers struggle to parse some of the sentences in SLOG, running out of memory. More efficient parsers running on beam search instead of A* do exist (e.g. Stanojević and Steedman, 2019) but are older and currently unmaintained, thus are hard to get operational due to out-of-date and unavailable software dependencies.

5.2 Further work

Most urgent work to carry out is test the effect of CCG supertags on bigger models with optimal hyperparameters. LLMs have exhibited acquiring new abilities with scale (Wei et al., 2022) so the impact of providing CCG supertags is not wholly predictable.

SLOG is also a promising direction, as we suspect CCG supertags will be even more helpful for models in a task which emphasises structural generalisation more, however producing CCG supertags for SLOG is non-trivial and might require augmenting the codebase for existent parsers and/or reducing the scope of the search and settling for a lower-quality parse.

Chapter 6

Conclusions

We set out to test the hypothesis that providing a neural semantic parser with CCG supertags would benefit the performance, in particular in cases where compositional generalisation is required for a successful parse.

We trained 3 models by fine-tuning codeT5-small on 3 datasets, COGS (Kim and Linzen, 2020) and two datasets derived from it - Supertagged COGS, which interleaved words with their supertags, and Padded COGS, which interleaved words with meaningless tags ("DU/D").

Our results show a significant improvement of the model utilising the supertags over the baseline model (and the model trained on padded utterances), most explicitly on utterances in VASA and VCLASS subcategories. These require the model to interpret a verb in a previously unseen way, e.g. as a passive verb where it was previously only seen as active, or in a new semantic role (e.g. intransitive vs passive). Strong effect shown on this subset of the problem is **consistent** with the hypothesis, and the effect way in which supertags improve the parses can be seen in our error analysis.

While supertagging is not flawless, we demonstrate that it improves the model performance overall, meaning the rare instances of a flawed supertag influencing the parse negatively is a good trade-off for reducing the occurrence of compositional generalisation related errors.

Bibliography

Suhas Arehalli and Tal Linzen. Neural networks as cognitive models of the processing of syntactic constraints. *Open Mind*, 8:558–614, 05 2024. doi: 10.1162/opmi_a_00137.

Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85, 1990. URL https://aclanthology.org/J90-2002.

Li Dong and Mirella Lapata. Language to logical form with neural attention. pages 33–43, 01 2016. doi: 10.18653/v1/P16-1004.

Jonathan Herzig and Jonathan Berant. Span-based semantic parsing for compositional generalization. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *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, August 2021a. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.74. URL https://aclanthology.org/2021.acl-long.74.

Jonathan Herzig and Jonathan Berant. Span-based semantic parsing for compositional generalization. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *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, August 2021b. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.74. URL https://aclanthology.org/2021.acl-long.74.

Tim Johnson. Natural language computing: The commercial applications. *The Knowledge Engineering Review*, 1(3):11–23, 1984. doi: 10.1017/S0269888900000588.

Aishwarya Kamath and Rajarshi Das. A survey on semantic parsing. *CoRR*, abs/1812.00978, 2018. URL http://arxiv.org/abs/1812.00978.

Najoung Kim and Tal Linzen. COGS: A compositional generalization challenge based on semantic interpretation. *CoRR*, abs/2010.05465, 2020. URL https://arxiv.org/abs/2010.05465.

Tom Kwiatkowksi, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. Inducing probabilistic CCG grammars from logical form with higher-order unification. In Hang Li and Lluís Màrquez, editors, *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1223–1233, Cambridge, MA, October 2010. Association for Computational Linguistics. URL https://aclanthology.org/D10-1119.

Brenden M. Lake and Marco Baroni. Still not systematic after all these years: On the compositional skills of sequence-to-sequence recurrent networks. *CoRR*, abs/1711.00350, 2017. URL http://arxiv.org/abs/1711.00350.

John Lawrence and Chris Reed. Argument mining: A survey. *Computational Linguistics*, 45(4):765–818, December 2019. doi: 10.1162/coli_a_00364. URL https://aclanthology.org/J19-4006.

Bingzhi Li, Lucia Donatelli, Alexander Koller, Tal Linzen, Yuekun Yao, and Najoung Kim. SLOG: A structural generalization benchmark for semantic parsing. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3213–3232, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.194. URL https://aclanthology.org/2023.emnlp-main.194.

Matthias Lindemann, Alexander Koller, and Ivan Titov. Compositional generalization without trees using multiset tagging and latent permutations. 2023. URL https://arxiv.org/abs/2305.16954.

Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101, 2017. URL http://arxiv.org/abs/1711.05101.

Maria Nadejde, Siva Reddy, Rico Sennrich, Tomasz Dwojak, Marcin Junczys-Dowmunt, Philipp Koehn, and Alexandra Birch. Syntax-aware neural machine translation using CCG. *CoRR*, abs/1702.01147, 2017. URL http://arxiv.org/abs/1702.01147.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. Universal Dependencies v1: A multilingual treebank collection. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Helene Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1659–1666, Portorož, Slovenia, May 2016. European Language Resources Association (ELRA). URL https://aclanthology.org/L16-1262.

Inbar Oren, Jonathan Herzig, Nitish Gupta, Matt Gardner, and Jonathan Berant. Improving compositional generalization in semantic parsing. In Trevor Cohn, Yulan He, and Yang Liu, editors, *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2482–2495, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.225. URL https://aclanthology.org/2020.findings-emnlp.225.

Linlu Qiu, Peter Shaw, Panupong Pasupat, Pawel Nowak, Tal Linzen, Fei Sha, and Kristina Toutanova. Improving compositional generalization with latent structure and data augmentation. In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz, editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4341–4362, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.323. URL https://aclanthology.org/2022.naacl-main.323.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, abs/1910.10683, 2019. URL http://arxiv.org/abs/1910.10683.

Siva Reddy, Oscar Täckström, Slav Petrov, Mark Steedman, and Mirella Lapata. Universal semantic parsing. *arXiv preprint arXiv:1702.03196*, 2017. URL https://arxiv.org/pdf/1702.03196.pdf.

Peter Shaw, Ming-Wei Chang, Panupong Pasupat, and Kristina Toutanova. Compositional generalization and natural language variation: Can a semantic parsing approach handle both? In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *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 922–938, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.75. URL https://aclanthology.org/2021.acl-long.75.

Miloš Stanojević and Mark Steedman. CCG parsing algorithm with incremental tree rotation. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 228–239, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1020. URL https://aclanthology.org/N19-1020.

Mark Steedman and Joakim Nivre. *The Syntactic Process*. 01 2000.

Marjorie Templeton and John Burger. Problems in natural-language interface to DBMS with examples from EUFID. In *First Conference on Applied Natural Language Processing*, pages 3–16, Santa Monica, California, USA, February 1983. Association for Computational Linguistics. doi: 10.3115/974194.974197. URL https://aclanthology.org/A83-1002.

Bailin Wang, Ivan Titov, and Mirella Lapata. Learning semantic parsers from denotations with latent structured alignments and abstract programs. *CoRR*, abs/1909.04165, 2019. URL http://arxiv.org/abs/1909.04165.

Yue Wang, Weishi Wang, Shafiq R. Joty, and Steven C. H. Hoi. Codet5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation. *CoRR*, abs/2109.00859, 2021. URL https://arxiv.org/abs/2109.00859.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models, 2022. URL https://arxiv.org/abs/2206.07682.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. Transformers: State-of-the-art natural language processing. In Qun Liu and David Schlangen, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-demos.6. URL https://aclanthology.org/2020.emnlp-demos.6.

Pengcheng Yin and Graham Neubig. A syntactic neural model for general-purpose code generation. *CoRR*, abs/1704.01696, 2017. URL http://arxiv.org/abs/1704.01696.

John M. Zelle and Raymond J. Mooney. Learning to parse database queries using inductive logic programming. In *AAAI/IAAI, Vol. 2*, 1996. URL https://api.semanticscholar.org/CorpusID:263135.

Luke S. Zettlemoyer and Michael Collins. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. *CoRR*, abs/1207.1420, 2012. URL http://arxiv.org/abs/1207.1420.