GPT makes a poor AMR parser

Anonymous ACL submission

Abstract

Much of Abstract Meaning Representation (AMR) parsing is currently concentrated on fine-tuning pre-trained language models. Large Language Models (LLMs) bring a new paradigm for NLP research, prompting. LLMs also show impressive 'reasoning' capabilities and a certain kind of interpretability with Chain-of-Thought (CoT) prompting. In this paper, we apply a variety of prompting strategies to induce GPT to do AMR parsing. We demonstrate that GPT models are insufficient as AMR parsers, but CoT prompting may shed light on how errors arise and show its potential in AMR parsing.

1 Introduction

Abstract Meaning Representation (AMR, Langkilde and Knight, 1998; Banarescu et al., 2013) is a graph-based semantic representation of a sentence, which represents the semantic relations within the sentence as a directed, acyclic graph (Fig. 1). State-of-the-art AMR parsers mainly adopt the paradigm of fine-tuning pre-trained language models (Vasylenko et al., 2023; Lee et al., 2022) to generate string representations of AMRs.

Even larger language models (LLMs), such as GPT-4 (OpenAI et al., 2024) have taken the world by storm. In addition to generating natural language texts, they can be used for downstream tasks by applying instructive prompts, without any fine-tuning (Zhao et al., 2023; Liu et al., 2023). This paper explores the possibility that GPT can work out of the box as an AMR parser. Despite trying a wide variety of prompting methods and models, we answer in the negative: GPT is not usable as an AMR parser.

The methods explored are: just asking the model to provide an AMR for a sentence (a.k.a. zero-shot), one-shot (in which one example is provided in the prompt), five-shot (5 examples; we tried sampling these from the training set randomly and by similarity), and two Chain-of-Thought prompts, in which it is demonstrated how to build the AMR step by step, with short explanations of each step (Wei et al., 2022b). The experiments took place on two 175B models (Instruct-GPT) of the GPT-3.5 family (Ouyang et al., 2022) and the GPT-4o model[2] (OpenAI et al., 2024).

We apply a range of evaluation techniques, from standard Smatch to fine-grained metrics to human evaluation. The best method-model combination we found – using GPT-4o with five-shot prompting and post-processing – achieves a Smatch F-score of only 60, compared to the SOTA at 85.4 (Lee et al., 2022). Moreover, without post-processing, even our best version yields strings that cannot be interpreted as single AMRs 15% of the time. The purest version of out-of-the-box methodology is the zero-shot setting; the results here are near 0. We found that CoT prompting demonstrates generalization and a sort of interpretability in AMR parsing, and specific CoT step design may benefit the parsing task.

2 Background & Related Work

AMR An AMR (Fig. 1) is composed of labelled nodes and edges, where nodes represent 'concepts' – roughly the words of the sentence – and the edges represent the relationships between the concepts. AMR guidelines include such details as the use of PropBank (Choi et al., 2010) verb senses (e.g. `receive-01`) and argument numbers (e.g. `ARG0`), complex subgraphs for named entities (e.g. the subgraph dominated by `person`), and negation, annotated with `(:polarity -)`. The numerical suffix such as `receive-01` represents the meaning of `receive`: `get something`, and `ARG0` is the `receiver`. AMRs can be encoded as text (Penman

[1] AMR graph visualisation powered by VULCAN (Groschwitz, 2023).

[2] https://openai.com/index/hello-gpt-4o/

The poor kid didn't receive the gift and the postcard that Dorothy Gale sent him on May 25th.

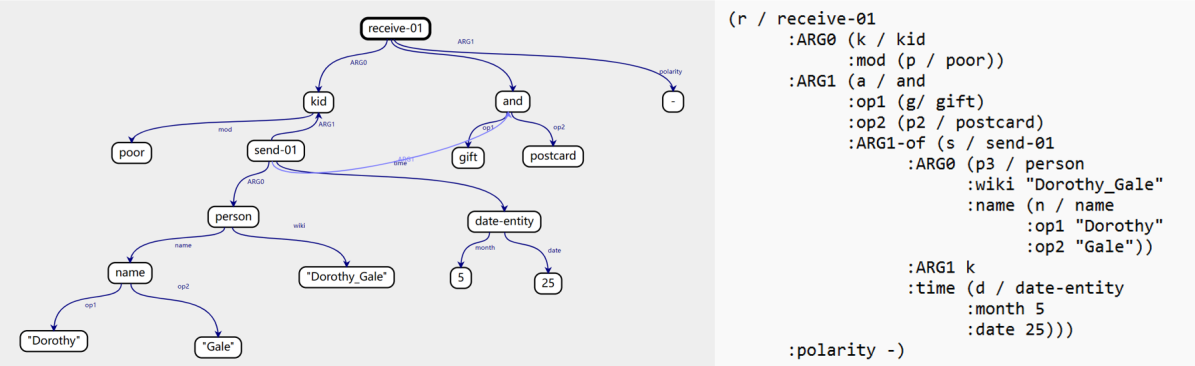


Figure 1: AMR graph example[1] (bottom left), corresponding Penman format (bottom left), and the sentence (top).

notation, which uses parentheses to express nested structures), so text-based models can be used to generate AMRs (van Noord and Bos, 2017).

AMR Parsing is the task of generating an AMR given a sentence. Existing AMR parsers mainly fall into three categories: transition-based models, sequence-to-graph models, and sequence-to-sequence (seq2seq) models. Transition-based models generate new nodes, edges, or subgraphs based on the words of the sentence (Peng et al., 2018; Naseem et al., 2019; Fernandez Astudillo et al., 2020; Zhou et al., 2021; Lindemann et al., 2020). Sequence-to-graph models derive the graph from existing nodes without transition processes, directly extending new nodes and edges (Zhang et al., 2019; Cai and Lam, 2020). Seq2seq models directly generate the text format of AMRs from raw sentences (van Noord and Bos, 2017; Blloshmi et al., 2020; Lee et al., 2022; Vasylenko et al., 2023; Bai et al., 2022). We use GPT as a seq2seq model.

Large Language Models (LLMs) such as GPT are pre-trained on vast quantities of raw text, and are trained to predict a continuation of given text (a *prompt*). Because this training data contains more than just natural language, a well-designed prompt can sometimes elicit text that can be interpreted as something other than language. For example, ChatGPT is capable of providing Python code given a plain-language description of what it should do (Poldrack et al., 2023). There are evidently some AMRs in GPT's training data – our zero-shot results are terrible, but some outputs are AMRs – and so AMR parsing may in principle be possible.

In-context learning/k-shot/few-shot prompting is a gradient-free "learning" strategy for language models that provides k task-related example question-answer pairs before asking the target question (Brown et al., 2020; Dong et al., 2023; Wei et al., 2022a). Few-shot prompting generally has better performance than zero-shot prompting (Zhao et al., 2021; Liu et al., 2022; Min et al., 2022), which only provides instructions. LLM performance is sensitive to the prompt, including the number of shots (Cao et al., 2020) and the choice of examples (Zhao et al., 2021).

Chain-of-Thought (CoT) Prompting Unlike regular few-shot prompting, CoT prompts include not only example question-answer pairs but also intermediate reasoning steps that can derive the final answer (Wei et al., 2022b). CoT prompting can significantly enhance the capabilities of LLMs in complex reasoning in several benchmarks (Lewkowycz et al., 2022; Saparov and He, 2023), and bring more interpretability with the generated reasoning process (Weng et al., 2023). Madaan and Yazdanbakhsh (2022) for instance claim that, through CoT prompting, LLMs can better understand the task by extracting commonsense knowledge from the questions, and generalize to unseen tasks by mimicking the expert's intermediate reasoning steps (Yang et al., 2022). However, final answers can be inconsistent with reasoning steps (Lyu et al., 2023).

Exactly what sort of interpretability CoT reasoning adds to LLMs is a matter of debate. Since CoT text is simply a part of the model's output, there is no particular reason to think that it's related to how the model actually produces its output. For instance, in our experiments, partial results in the CoT output are usually – but not always – part of the output AMR, and reasoning text is usually related to the partial results they are paired with, but

often doesn't precisely match them (§4.3).

3 Methods

3.1 Prompt Design

In our experiments, we compare several types of prompts, which we describe in this section, and give full examples in Appendix A. All of our prompts request an AMR given a sentence; some include examples. Asking GPT to role-play can improve model performance (Reynolds and McDonell, 2021; Kong et al., 2023), so all our prompts begin with *You are a computational linguist.* The simplest prompt template we use is in (1):

(1) *You are a computational linguist. Let's do some semantic parsing. Please give me an Abstract Meaning Representation (AMR) of* [sentence].

3.1.1 Zero-shot and Few-shot Prompt Design

The zero-shot setting uses the prompt in (1). One- and five-shot prompts use the zero-shot prompt but add examples, each introduced with *Q: Please give me an Abstract Meaning Representation (AMR) of* [sentence]. *A:* [AMR].

For k-shot, we use $k = 1$ as a minimal version and $k = 5$ as a maximal one, 5 being chosen because it provides a balance between using many examples and consistently fitting within the space constraints of GPT.

All one-shot prompts use the sentence in Fig. 1 as the example, built to demonstrate common AMR properties, such as reentrancy, different non-core roles, etc., and not be too long. Five-shot examples are sampled from the training set.

Two sampling strategies are applied, random sampling and sampling based on semantic similarity. Designing a prompt based on semantically similar examples to the query can improve LLMs' performance (Gao et al., 2021; Liu et al., 2022). We use Wang et al. (2020)'s model *sentence-transformers/all-MiniLM-L6-v2*[3] to calculate the cosine similarities between the target sentence and the training set, and the five most similar sentences and their AMRs are selected as the examples used in the prompt.

3.1.2 CoT Prompt Design

We use the same example sentence as the one-shot setting (Fig. 1) for all CoT prompts. We introduce two different styles of CoT prompting for deriving AMR, top-down and bottom-up. A CoT toy example of the sentence *Dorothy Gale danced* is shown in Fig. 2.

Top-down The top-down approach is to find the top node first, which is generally the main verb in the sentence, and then find its child nodes and their respective relationships according to the semantics of the sentence. We then recurse through the generated child nodes until LLM derives the final graph.

Bottom-up The bottom-up approach first extracts smaller subgraphs that make up the AMR, e.g. the subject and the object, the place and the time. Then it looks for the relationship between subgraphs and links them with nodes and edges until the graph is complete.

Prompt lengths The length of the CoT prompting is a trade-off. Longer reasoning steps for complex tasks boost LLMs' inference performance (Wei et al., 2022b; Jin et al., 2024), but leave less space for generating complete reasoning steps and results. Our top-down prompt has five steps and 800 tokens[4], and the bottom-up has four steps and 864 tokens. For comparability between the methods, CoT and one-shot have the same number of examples, and five-shot prompts are of similar length to the CoT prompts (∼800, tokens depending on the samples).

3.2 Dataset & Models

Our experiment took place on the English AMR 2.0 (Knight et al., 2017) and AMR 3.0 (Knight et al., 2021) test set with example selections for few-shot prompting on the training set of AMR 2.0 and AMR 3.0 respectively. AMR 2.0 has 1,371 (test) and 36,521 (train) AMRs and AMR 3.0 has 1,898 and 55,635. Used within the terms of the license, LDC User Agreement for Non-Members.

The AMR 2.0 test set is used to test different prompting strategies on the English "text-davinci-003" (davinci) model and the "gpt-3.5-turbo-instruct" (turbo-instruct) model; both models belong to the GPT-3.5 family and have 175B parameters with 4,097 token length limitations (Ouyang et al., 2022). These length limitations are shared

[3] https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2?library=sentence-transformers

[4] Reported token lengths are calculated based on the OpenAI official tokenizer for GPT-3.5 and GPT-4, https://platform.openai.com/tokenizer. GPT-4o uses a new tokenizer which has not officially been released at the moment.

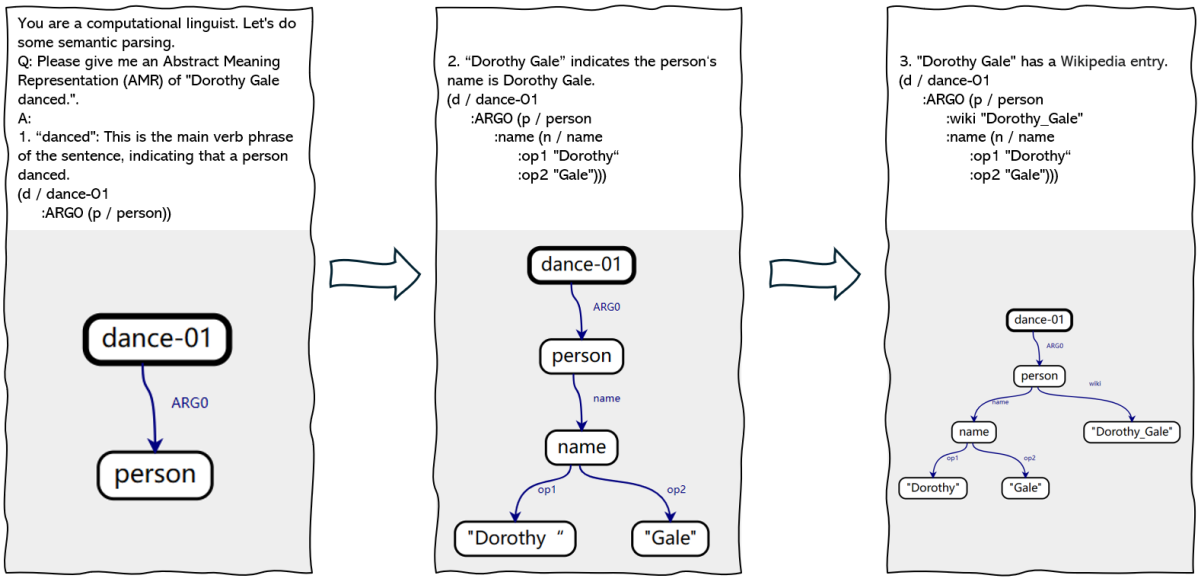


Figure 2: A toy example of CoT reasoning for *Dorothy Gale danced.* Each box is a reasoning step, where the top part is the CoT reasoning text and the bottom part is a visualization[1] of the corresponding subgraph.

between the input and prediction. The best prompting strategy overall (5-shot with similarity example sampling) and the best CoT prompting strategy (top-down CoT) on AMR 2.0 with GPT-3.5 were determined. These two methods were then applied to the GPT-4o model with 128,000 token limitations on the AMR 3.0 test set. Our experiments use the official OpenAI API and the models'[5] temperatures are set to 0 for reproducibility.

4 Results

Our experiments include three different LLMs (turbo-instruct and davinci for the AMR 2.0 dataset and GPT-4o for the AMR 3.0 dataset). The full results are reported in Appendix B; in this section, we report only the best results of each method. All GPT-3.5 results are therefore for the davinci model. The best five-shot method used similarity sampling, and for CoT, the top-down method.

We begin our analysis with a look at the parsability of output strings into AMRs (§4.1). No method produces more than 85% parsable outputs, but a post-processing script significantly improves parsability. Next, we evaluate with standard Smatch scores (§4.2). We find that the 5-shot method with GPT-4o is the best, followed by one-shot, Chain-of-Thought (CoT), and zero-shot. We then apply the fine-grained analysis provided by Damonte et al. (2017) and GrAPES (Groschwitz et al., 2023), finding that the more complex and AMR-specific the subtask, the harder it is for GPT (§4.2.1). Finally, we sample from the CoT outputs and do an error analysis, finding that the steps the model outputs generally make some sense, but are riddled with errors, and the reasoning text does not always match the subgraph generated (§4.3).

4.1 Parseability and post-processing

Some of the LLM output texts are not parsable as AMRs, and some produce multiple AMRs. While the ability of the model to produce a single, legal, AMR is the main question of interest, also of interest is whether a pipeline that includes minor post-processing works as an AMR parser. For this reason, we apply a post-processing script[6] that fixes mismatched parentheses, splits multiply-labeled nodes into separate nodes, and combines multiple AMRs into one with a root node labeled `multi-sentence`. The proportion of legal and illegal AMRs for each method, before and after post-processing, is in Fig. 3.

4.2 Smatch

The standard evaluation metric for AMR is Smatch (Cai and Knight, 2013). For Smatch evaluation, unparsable graphs are replaced by a dummy graph `(d / dummy)`. We evaluate with and without post-

[5]The turbo-instruct and GPT-4o model point to "gpt-3.5-turbo-0613" and "gpt-4o-2024-05-13" respectively at the time of our experiments. Used within term of use: https://openai.com/policies/eu-terms-of-use/

[6]All code will be made available upon publication.

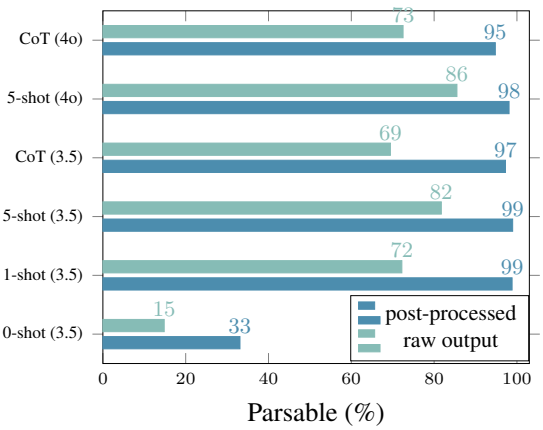


Figure 3: Parsability before/after post-processing

Dataset AMR 3.0	**Smatch**
5-shot (GPT-4o)	60
CoT (GPT-4o)	55
LeakDistill (SOTA) (Vasylenko et al., 2023)	**84.6**
Dataset AMR 2.0	
Zero-shot (GPT-3.5)	14
1-shot (GPT-3.5)	41
5-shot (GPT-3.5)	50
CoT (GPT-3.5)	36
LeakDistill (SOTA) (Vasylenko et al., 2023)	**86.1**
Dataset LDC2013E117	
JAMR (Flanigan et al., 2014)	58

Table 1: Smatch for our methods (all post-processed), LeakDistill (SOTA), and JAMR (an early baseline).

processing. Fig. 4 is the Smatch of the best version of each method. (Full results in Appendix B.1)

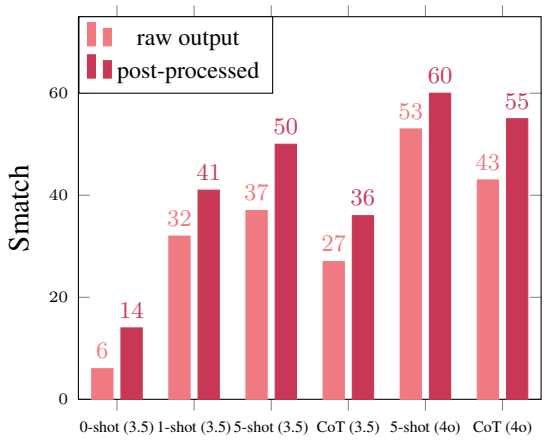


Figure 4: Best Smatch of each method

In each evaluation, the ranking of each method conforms to the following hierarchy: 5-shot > 1-shot > CoT > zero-shot.

Out best results are for the 5-shot method with GPT-4o (60), but nothing approaches the SOTA AMR parser (Vasylenko et al. (2023) with 86.1 on AMR 2.0 and with 84.6 on AMR 3.0). GPTs are loosely comparable to an early AMR baseline parser, JAMR (Flanigan et al., 2014), with a 58 Smatch on the LDC2013E117 dataset (Table 1).

4.2.1 Fine-grained results

In addition to Smatch, we evaluated the GPT-4o outputs with the Granular AMR Parsing Evaluation Suite, or GrAPES (Groschwitz, 2023), a fine-grained evaluation with 36 categories divided into 9 sets. 23 of the categories are extracted from the AMR 3.0 test set; evaluating with these metrics grants insights into the strengths and weaknesses of GPT as an AMR parser. For comparison, we include a high-performing fine-tuned BART model, AMRBart (Bai et al., 2022) (Smatch 84). The full GrAPES results on the AMR 3.0 test set categories are in Appendix B.2. We also ran Damonte et al. (2017)'s fine-grained Smatch is run on all outputs (see Appendix B.1) and highlighted some relevant results.

Because the overall quality of GPT as a parser is so poor, AMRBart outperforms GPT in nearly all categories. However, there is substantial, and, we argue, principled, cross-categorical variation. Overall, GPT is much better at simpler and less AMR-specific tasks.

Seen vs Unseen Unlike with purpose-trained AMR parsers, there is very little difference in performance on subcategories of things seen and unseen in the AMR 3.0 training set: while AMRBart performs on average 38% worse on unseen items, our best GPT model is only 10% worse, and for rare and unseen node labels the performance is identical for GPT, but 69 vs 45 for AMRBart (Table 2). This is as expected, as GPT is not trained specifically on the AMR training set.

Simple vs complex, AMR-specific subtasks GPT is not too bad at simple tasks like node labeling. Fine-grained Smatch includes the metric *Concepts*, the F-score over the multiset of node labels. GPT here scores 67, which is not high (AMRBart scores 90), but is quite a bit higher than its overall Smatch F-score of 60. Similarly, GrAPES shows GPT trailing AMRBart in rare node labels only 61 to 68, and outperforming it in unseen node

labels 61 to 45 in Table 2. It is also reasonable at finding and categorizing named entities (Fine-grained Smatch NER score 73). GPT even outperforms AMRBart on the GrAPES category *Hard unseen wiki links*, which are wiki links for named entities that are not templatic. Evidently, these unpredictable URLs occur in GPT's training data, and it is able to make use of them in some cases.

However, the more complex and AMR-specific the subtask, the worse GPT gets. Scores on PropBank tasks (sense disambiguation and argument numbers) average to 30 (vs 63 for AMRBart). Overall, for tasks we classified as AMR-specific (PropBank tasks, multinode word meanings (e.g. *teacher* is annotated `(person :ARG0-of teach-01)`), imperatives, ellipsis, and special entities), GPT performed 49% worse than its average GrAPES score, while AMRBart performed only 4% worse.

One GrAPES metric we categorize as AMR-specific deserves special mention: Special Entities, which are things like dates, URLs, and names. GPT does much better here than on other AMR-specific subtasks at 64. We suspect GPT succeeds here because this task is very templatic.

We also found an effect of sentence length, with longer sentences much more likely to yield unparsable outputs. For instance, GPT-4o five-shot, before post-processing, has an error rate of 6% for sentences up to 20 words but 33% for sentences longer than 40 (see Appendix D).

Category	**5-shot 4o**	**CoT 4o**	**AMR Bart**
Seen vs Unseen			
Rare node labels	61	57	69
Unseen node labels	**61**	56	45
Hard unseen wiki links	**33**	5	9
Seen	71	59	93
Unseen	57	48	58
Seen − Unseen as % of Seen	10%	18%	38%
AMR-specific			
PropBank	30	25	63
Multinode word meanings	14	4	84
Imperatives	4	0	66
Ellipsis	12	15	55
Special Entities	64	55	77
Average AMR-spec.	25	20	69
Average all categories	48	42	72
AMR-spec − all as % of all	49%	53%	4%

Table 2: Selection of fine-grained categories from GrAPES. *Italicised* categories are averages across multiple categories. Scores are (averages of) recall.

Negation (Damonte et al., 2017)'s fine-grained Smatch reveals that the one-shot and CoT methods out-perform the five-shot on exactly one thing: negation, which is F-score on the subgraph `(:polarity -)`. On AMR 2.0, the best five-shot score is 9, while CoT top-down and bottom-up are 14 and 19 respectively, and even one-shot has an 11. On AMR 3.0 with GPT-4o, CoT also slightly outperforms five-shot (34 vs 32). This is likely because the hand-built example used in one-shot and CoT includes negation, and the CoT example even includes a step for the negation. Informal sampling of CoT outputs revealed a lot of output reasoning steps about negation, even occasional over-generalizations to creating a `(:polarity +)` edge, which doesn't exist in AMR. In one case (*Do they **not** realize that you **don't** get piles of change from the bank*), because there are two negations, the CoT text in Step 3 includes the sentence *The subordinate clause is also negated, so there is a polarity change at the top level*, and the AMR at that step actually (incorrectly) changes `(:polarity -)` to `(:polarity +)`. (See Appendix C.1 for details.)

4.3 Error analysis: Chain of Thought

Chain-of-Thought prompting can be taken to lend some interpretability to LLMs. To that end, from the AMR 2.0 test set, we randomly sampled 6 sentences and did an error analysis of the top-down and bottom-up CoT outputs of GPT-3.5. No output was without mistakes; the 96 mistakes noted fall into 10 broad categories (Table 3). To get an idea of what this can look like, consider the bottom-up parse of example (2) in Fig. 5. The gold and predicted AMRs are visualized in Fig. 6. Steps in bottom-up parses were largely unnumbered, so for ease of exposition, steps are annotated (e.g. `[a]`).

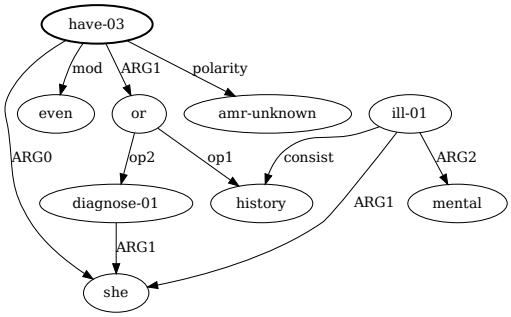
(2) *Does she even have a history of mental illness or a diagnosis?*

Overall, we can see that the subgraphs generated are not unreasonable, usually make it into the final AMR, and relate at least somewhat to the CoT text. This was true of every item sampled. Step `[a]` is in fact good, with *she* correctly identified and a correct AMR generated for it. All samples started with the correct `ARG0` in AMR.

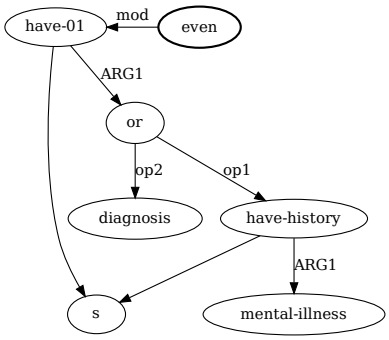
Step `[b]`, however, is very bad. Not only does *even* have nothing to do with this being a question, but no interrogative marker is added (a Missing Component error). Thus Step `[b]` has two CoT errors: it doesn't make sense, and the text does not match the subgraph. Step `[c]` is good, only making a PropBank sense error (`have-01` *vs* `-03`). Note as

```
[a] 1. This sentence has the following
    components: "Does she": indicates that "she"
    is the subject of the sentence.
    (s / she)
[b] "even": indicates that the sentence is
    asking a question.
    (e / even)
[c] "have": is the main verb phrase of the
    sentence, indicating that "she" has
    something.
    (h / have-01
          :ARG0 s)
[d] "a history of mental illness": indicates
    that "mental illness" is the object of the
    verb "have".
    (h2 / have-history
          :ARG0 s
          :ARG1 (m / mental-illness))
[e] "or a diagnosis": indicates that "diagnosis"
    is an alternative to "mental illness".
    (o / or
          :op1 h2
          :op2 (d / diagnosis))
[f] 2. The AMR is:
    (e / even
          :mod (h / have-01
                :ARG0 s
                :ARG1 (o / or
                      :op1 (h2 / have-history
                         :ARG0 s
                         :ARG1
                         (m / mental-illness))
                      :op2 (d / diagnosis))))
```

Figure 5: CoT bottom-up output for (2)



(a) Gold AMR



(b) Predicted AMR

Figure 6: AMRs for example (2)

well that the ARG0 is correctly made node s. In Penman notation, only one instance of a node can be labeled as in (s / she); any other mentions of it should just be the node name, here s. No outputs we sampled had double-labeled nodes.

In step [d] we see two common errors. First, notice the node labeled mental-illness, which should be a complex subgraph with ill-01 and mental. We found this kind of "simplification" error 12 times. Node h2, labeled have-history, is another simplification error, and it is also a CoT error, because the CoT text refers to the verb *have*, but instead of using the existing node h, it adds a new node h2. Adding a new version of an existing node happened 4 times in the sample, and never was an existing node label updated to reflect the new step's analysis.

In step [e], *diagnosis* should be annotated as a PropBank predicate diagnose-01. "Predicate" errors like this occurred 6 times in the sample. Perhaps node h2 should be thought of as a history node; if so, the subgraph dominated by or is correct; however, the CoT text here doesn't match: it should mean op1 is node m, not node h2, as it refers to *mental illness*.

In step [f], the subgraphs are put together. Note first that the ARG0 (s / she) is missing. Such "Missing earlier component" errors occur 3 times in the sample. Conversely, the h2 daughter of (o / or) has correctly substituted in the full subgraph dominated by h2.

The even node has incorrectly been made the root. Its edge to h is close: the correct graph has a mod edge from even to have-03.

5 Discussion

GPT performs poorly LLM performance is correlated with the amount of task-relevant data during pre-training (Kandpal et al., 2023). Only a tiny portion of pre-training data can be AMR-related, yielding near-zero results in the zero-shot setting.

The zero-shot setting without post-processing yields a Smatch F-score of only 6 (14 with post-processing). The purest version of GPT as an out-of-the-box AMR parser is therefore right out. This is in contrast to, for instance, Python programming, where Poldrack et al. (2023) found that natural-

Category	BU	TD	Sum
CoT	10	9	19
PropBank sense	4	3	7
Simplification	8	4	12
Predicate	3	3	6
Missing component	8	7	15
Missing earlier component	3	0	3
Named Entity	4	3	7
Duplication	2	2	4
Whole AMR in step 1	0	2	2
Other	2	18	23
Total	46	50	96
Steps	31	21	52

Table 3: CoT Error classification over 6 sentences (GPT-3.5; BU = bottom up, TD = top down).

language prompts for Python code were usable on the first try in 38% of cases. Parsing into Python code is arguably just as difficult a task as AMR parsing, so we might expect similar outcomes. A major difference is likely the training data. While the training data for GPT is not publicly available, we suspect it contains a large amount of Python code. AMR corpora are proprietary, but even if the whole AMR 3.0 dataset slipped into the training data, it only has 59,255 sentences. We predict that any other task that has little GPT training data and is complex and not *a priori* predictable would have similar performance issues.

Fine-grained analysis reveals a large discrepancy between subtasks that are fairly simple and easy to predict, such as basic node labeling, and subtasks that are complex or AMR-specific, such as imperatives. Because language phenomena have a Zipfian distribution, it is impossible to create a single, short enough example that contains every phenomenon that can – or even is likely to – arise. Notice as well that even though GPT-4o in the CoT setting was given an explicit example of negation, its fine-grained Smatch Negation score is only 34, so a single example may not be enough.

Interpretability of CoT AMR parsing Although CoT uses a similar setup to one-shot, it is not an extension of the 1-shot method. The output graphs are not necessarily the same, and even when they are, there is no way to know whether the chain of thought is in any way related to how the model built the graph in the one-shot case.

An advantage of CoT is that it to a certain extent reflects how GPT derives an AMR in the CoT case, since the subgraphs in the chain of reasoning are usually in fact subgraphs of the final output. We can often easily find errors through the CoT reasoning process, which can make it easier to correct errors by hand.

Prompt tuning – updating the prompt or its vector representation in response to a learning signal – may improve the CoT outcome. While there is no way to automatically evaluate the validity of the relationship between a reasoning step and the subgraph it describes, the quality of the subgraph itself may work as a proxy for the quality of the reasoning step. The difference between the subgraph and gold AMR could then be used as the loss signal for prompt tuning. This method opens the possibility to subsequently use LLM as a generative model for data augmentation, especially for complex sentences.

The potential of CoT prompting for AMR parsing An interesting part of the CoT results is that: it did outperform other approaches on Negation, likely due to a specific step design, and it shows generalization and inference abilities (see Double Negation example in the Appendix (3)). Although Negation is less complex than tasks like Semantic Role Labeling in AMR, the required knowledge for these complex tasks is open source and can be elicited with proper prompting if an LLM is pretrained. This suggests that difficult tasks can also be solved using explicit CoT reasoning.

Additionally, the general problem of language phenomena being in a Zipfian distribution might be somewhat mitigated by the demonstrated generalization ability of the CoT method.

6 Conclusion

We compared the capabilities of GPT models on AMR parsing under various prompting strategies. We found that GPT-3.5 and GPT-4o make poor AMR parsers, with a maximum Smatch of 60.

Two CoT prompting methods for AMR parsing (bottom-up and top-down) were introduced. CoT prompting worsens performance over both one-shot and five-shot methods, but arguably adds some interpretability.

A detailed analysis revealed GPT's limited ability to produce AMRs in alignment with structural AMR guidelines. These guidelines specify precise ways to annotate certain things, such as named entities, that can't be predicted *a priori*. GPT struggles to express many linguistic phenomena within the AMR paradigm.

Limitations

Example selection A better selection of the examples in the five-shot may help LLM get better performance. Since we only need to illustrate phenomena for one sentence at a time. However, the problem of identifying the phenomena to demonstrate, and finding the AMRs that exemplify them, is in itself a kind of parsing. A proof-of-concept experiment could use the gold AMR and measure graph similarity, but this would not be usable as an AMR parsing method.

Multi-step reasoning system In Addition to the CoT method we provided in the paper, another potential reasoning technique is using a multi-step reasoning system, by breaking the different functions (steps) of a single CoT into multiple prompts on multiple LLMs (Wu et al., 2022a,b). This provides additional space for tailoring prompts for specific subtasks (such as providing extra examples), and each prompt could be applied to a certain LLM that is more suitable for a specific subtask. However, one of our attempts, showed that this approach performs poorly on the AMR parsing task (Details can be found in Appendix E and result shown in Table 10). The first step of the two-step version is finding the core roles (the ARGs), followed by a separate step in which the model is given the output of step one and a list of non-core roles, instructions, and an example. Other designs would be worth trying in a future paper

CoT Prompt Length To easily adjust different experiment settings and test different prompts and model responses. The prompt design, initialization, and tuning happened on the OpenAI official playground[7]. However, the playground has a 2048 token constraint at the time of our experiments, so our prompt token lengths do not take advantage of the maximum limit for the API, at 4,096. We also kept our prompt lengths the same for GPT-4o as for GPT-3.5, but GPT-4o has a much longer token constraint of 128,000, so future experiments do have room to expand to longer prompts.

CoT Prompt Design The prompting design of this paper is based on expert knowledge of AMR parsing. Not everything needed to generate an AMR is explicitly given in the prompt; for instance, most edge labels are never mentioned in the prompt. As mentioned in Smatch section, the LLMs are designed for general purposes, even if the pre-trained LLM has some AMR or semantic knowledge, it is still unknown whether it should be assumed that the LLM is already familiar with certain terms or knowledge.

This affects the design process of the prompt. In our experimental design, we assumed that LLM could understand and handle those problems. In fact, LLM does perform well to a certain extent. But if using a CoT prompt that is more simple and informative, in other words, crafting CoT more like a "rookie" rather than an expert, the results still need to be tested.

But this involves another issue, the limit on the number of tokens. A more detailed CoT prompt may have less space for generation, which can be solved by using a larger model.

LLM models Due to limitations of data and model availability, our initial experiments, that explored the largest space of possibilities, were all run on the AMR 2.0 test set, using GPT-3.5 models. We only had the resources to try out on GPT-4o the methods that worked best on GPT-3.5. The results across GPT versions are also therefore not completely comparable, as there are some differences between the APR 2.0 and 3.0 test sets, including about 500 additional sentences. Our best GPT-3.5 methods all used the `text-davinci-003` model, which has now been deprecated.

This experiment only focused on decoder-only architecture LLMs (GPT-3.5 family models and GPT-4o model). As for the performance of other LLM structures, such as encoder-only models and decoder-encoder models, which still need testing.

Data contamination The mainstream way to obtain pre-training data for LLM is to crawl the Internet and it's hard to know whether a benchmark is involved in pretraining, especially for closed LLM (Jacovi et al., 2023). The contamination leads to overestimating LLM performance on the benchmark (Sainz et al., 2023). However, Duan et al. (2024) shows that there is no effective way to examine if a certain data is a member of pertaining data. The AMR 2.0 and 3.0 are closed-source datasets and to some extent might avoid data contamination problems.

To examine this, a new dataset is sampled from the AMR 2.0 training set, which has the same amount of data as the test set. We apply it with

[7] https://platform.openai.com/playground/

the same setup as 5-shot, and the result of this new dataset is similar to the test set (Table 11 in Appendix F.). Therefore, it suggests two possibilities: either the training set and test set are contaminated with the pretraining data or neither not.

English only We only experimented with English AMR using GPT, which is trained mostly on English.

Ethics Statement

We believe this work does not involve any ethical issues.

References

Xuefeng Bai, Yulong Chen, and Yue Zhang. 2022. Graph pre-training for AMR parsing and generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6001–6015, Dublin, Ireland. Association for Computational Linguistics.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for sembanking. In *Proceedings of the 7th linguistic annotation workshop and interoperability with discourse*, pages 178–186.

Rexhina Blloshmi, Rocco Tripodi, and Roberto Navigli. 2020. XL-AMR: Enabling cross-lingual AMR parsing with transfer learning techniques. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2487–2500, Online. Association for Computational Linguistics.

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

Deng Cai and Wai Lam. 2020. AMR parsing via graph-sequence iterative inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1290–1301, Online. Association for Computational Linguistics.

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 748–752, Sofia, Bulgaria. Association for Computational Linguistics.

Tianshi Cao, Marc T Law, and Sanja Fidler. 2020. A theoretical analysis of the number of shots in few-shot learning. In *International Conference on Learning Representations*.

Jinho D. Choi, Claire Bonial, and Martha Palmer. 2010. Propbank instance annotation guidelines using a dedicated editor, jubilee. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta. European Language Resources Association (ELRA).

Marco Damonte, Shay B. Cohen, and Giorgio Satta. 2017. An incremental parser for Abstract Meaning Representation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 536–546, Valencia, Spain. Association for Computational Linguistics.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, Lei Li, and Zhifang Sui. 2023. A survey on in-context learning.

Michael Duan, Anshuman Suri, Niloofar Mireshghallah, Sewon Min, Weijia Shi, Luke Zettlemoyer, Yulia Tsvetkov, Yejin Choi, David Evans, and Hannaneh Hajishirzi. 2024. Do membership inference attacks work on large language models?

Ramón Fernandez Astudillo, Miguel Ballesteros, Tahira Naseem, Austin Blodgett, and Radu Florian. 2020. Transition-based parsing with stack-transformers. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1001–1007, Online. Association for Computational Linguistics.

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. 2014. A discriminative graph-based parser for the Abstract Meaning Representation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1426–1436, Baltimore, Maryland. Association for Computational Linguistics.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. 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 3816–3830, Online. Association for Computational Linguistics.

Jonas Groschwitz. 2023. Introducing VULCAN: A visualization tool for understanding our models and

data by example. In *Proceedings of the 6th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 199–211, Singapore. Association for Computational Linguistics.

Jonas Groschwitz, Shay Cohen, Lucia Donatelli, and Meaghan Fowlie. 2023. AMR parsing is far from solved: GrAPES, the granular AMR parsing evaluation suite. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10728–10752, Singapore. Association for Computational Linguistics.

Alon Jacovi, Avi Caciularu, Omer Goldman, and Yoav Goldberg. 2023. Stop uploading test data in plain text: Practical strategies for mitigating data contamination by evaluation benchmarks. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5075–5084, Singapore. Association for Computational Linguistics.

Mingyu Jin, Qinkai Yu, Dong Shu, Haiyan Zhao, Wenyue Hua, Yanda Meng, Yongfeng Zhang, and Mengnan Du. 2024. The impact of reasoning step length on large language models.

Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. 2023. Large language models struggle to learn long-tail knowledge. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Kevin Knight, Bianca Badarau, Laura Baranescu, Claire Bonial, Madalina Bardocz, Kira Griffitt, Ulf Hermjakob, Daniel Marcu, Martha Palmer, Tim O'Gorman, and Nathan Schneider. 2017. Abstract Meaning Representation (AMR) Annotation Release 2.0.

Kevin Knight, Bianca Badarau, Laura Baranescu, Claire Bonial, Madalina Bardocz, Kira Griffitt, Ulf Hermjakob, Daniel Marcu, Martha Palmer, Tim O'Gorman, et al. 2021. Abstract Meaning Representation (AMR) annotation release 3.0. https://catalog.ldc.upenn.edu/LDC2020T02.

Aobo Kong, Shiwan Zhao, Hao Chen, Qicheng Li, Yong Qin, Ruiqi Sun, and Xin Zhou. 2023. Better zero-shot reasoning with role-play prompting.

Irene Langkilde and Kevin Knight. 1998. Generation that exploits corpus-based statistical knowledge. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1*, pages 704–710, Montreal, Quebec, Canada. Association for Computational Linguistics.

Young-Suk Lee, Ramón Astudillo, Hoang Thanh Lam, Tahira Naseem, Radu Florian, and Salim Roukos. 2022. Maximum Bayes Smatch ensemble distillation for AMR parsing. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5379–5392, Seattle, United States. Association for Computational Linguistics.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. 2022. Solving quantitative reasoning problems with language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 3843–3857. Curran Associates, Inc.

Matthias Lindemann, Jonas Groschwitz, and Alexander Koller. 2020. Fast semantic parsing with well-typedness guarantees. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3929–3951, Online. Association for Computational Linguistics.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2022. What makes good in-context examples for GPT-3? In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 100–114, Dublin, Ireland and Online. Association for Computational Linguistics.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Comput. Surv.*, 55(9).

Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. 2023. Faithful Chain-of-Thought reasoning. In *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 305–329, Nusa Dua, Bali. Association for Computational Linguistics.

Aman Madaan and Amir Yazdanbakhsh. 2022. Text and patterns: For effective Chain of Thought, it takes two to tango.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11048–11064, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Tahira Naseem, Abhishek Shah, Hui Wan, Radu Florian, Salim Roukos, and Miguel Ballesteros. 2019. Rewarding Smatch: Transition-based AMR parsing with reinforcement learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4586–4592, Florence, Italy. Association for Computational Linguistics.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2024. Gpt-4 technical report.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc.

Xiaochang Peng, Daniel Gildea, and Giorgio Satta. 2018. AMR parsing with cache transition systems. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1).

Russell A Poldrack, Thomas Lu, and Gašper Beguš. 2023. Ai-assisted coding: Experiments with gpt-4.

Laria Reynolds and Kyle McDonell. 2021. Prompt programming for large language models: Beyond the few-shot paradigm. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI EA '21, New York, NY, USA. Association for Computing Machinery.

Oscar Sainz, Jon Campos, Iker García-Ferrero, Julen Etxaniz, Oier Lopez de Lacalle, and Eneko Agirre. 2023. NLP evaluation in trouble: On the need to measure LLM data contamination for each benchmark. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 10776–10787, Singapore. Association for Computational Linguistics.

Abulhair Saparov and He He. 2023. Language models are greedy reasoners: A systematic formal analysis of Chain-of-Thought. In *The Eleventh International Conference on Learning Representations*.

Rik van Noord and Johan Bos. 2017. The meaning factory at SemEval-2017 task 9: Producing AMRs with neural semantic parsing. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 929–933, Vancouver, Canada. Association for Computational Linguistics.

Pavlo Vasylenko, Pere Lluís Huguet Cabot, Abelardo Carlos Martínez Lorenzo, and Roberto Navigli. 2023. Incorporating graph information in transformer-based AMR parsing. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1995–2011, Toronto, Canada. Association for Computational Linguistics.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *Advances in Neural Information Processing Systems*, volume 33, pages 5776–5788. Curran Associates, Inc.

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. 2022a. Emergent abilities of large language models. *Transactions on Machine Learning Research*. Survey Certification.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022b. Chain-of-Thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc.

Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Shengping Liu, Bin Sun, Kang Liu, and Jun Zhao. 2023. Large language models are better reasoners with self-verification. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 2550–2575, Singapore. Association for Computational Linguistics.

Tongshuang Wu, Ellen Jiang, Aaron Donsbach, Jeff Gray, Alejandra Molina, Michael Terry, and Carrie J Cai. 2022a. Promptchainer: Chaining large language model prompts through visual programming. In *Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems*, CHI EA '22, New York, NY, USA. Association for Computing Machinery.

Tongshuang Wu, Michael Terry, and Carrie Jun Cai. 2022b. Ai chains: Transparent and controllable human-ai interaction by chaining large language model prompts. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, CHI '22, New York, NY, USA. Association for Computing Machinery.

Mengjiao (Sherry) Yang, Dale Schuurmans, Pieter Abbeel, and Ofir Nachum. 2022. Chain of Thought imitation with procedure cloning. In *Advances in Neural Information Processing Systems*, volume 35, pages 36366–36381. Curran Associates, Inc.

Sheng Zhang, Xutai Ma, Kevin Duh, and Benjamin Van Durme. 2019. Broad-coverage semantic parsing as transduction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3786–3798, Hong Kong, China. Association for Computational Linguistics.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023. A survey of large language models.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 12697–12706. PMLR.

Jiawei Zhou, Tahira Naseem, Ramón Fernandez Astudillo, and Radu Florian. 2021. AMR parsing with action-pointer transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5585–5598, Online. Association for Computational Linguistics.

A Appendix: Prompt Design Detail

A.1 Zero-shot Prompt

```
You are a computational linguist. Let's do some
semantic parsing.
Q: Please give me an Abstract Meaning
Representation (AMR) of '[Target Sentence]'.
A:
```

A.2 Few-shot Prompt

A.2.1 1-shot Prompt

```
You are a computational linguist. Let's do some
semantic parsing.
Q: Please give me an Abstract Meaning
Representation (AMR) of "The poor kid didn't
receive the gift and the postcard that Dorothy
Gale sent him on May 25th.".
A:
(r / receive-01
      :ARG0 (k / kid
            :mod (p / poor))
      :ARG1 (a / and
            :op1 (g/ gift)
            :op2 (p2 / postcard)
            :ARG1-of (s / send-01
                  :ARG0 (p3 / person
                        :wiki "Dorothy_Gale"
                        :name (n / name
```

```
                              :op1 "Dorothy"
                              :op2 "Gale"))
                  :ARG1 k
                  :time (d / date-entity
                        :month 5
                        :date 25)))
      :polarity -)
Q: Please give me an Abstract Meaning
Representation (AMR) of '[Target Sentence]'.
A:
```

A.2.2 Example 5-shot Prompt

```
You are a computational linguist. Let's do some
semantic parsing.
Q: Please give me an Abstract Meaning
Representation (AMR) of 'However, when it comes
to the soul and the physical being, in the
Church's eyes, the soul trumps the former.'.
A:
(c2 / contrast-01
      :ARG2 (t / trump-01
            :ARG0 (s2 / soul)
            :ARG1 b
            :ARG1-of (v / view-02
                  :ARG0 (o / organization
                        :wiki "Catholic_Church"
                        :name (n / name
                            :op1 "Church")))
            :condition (c3 / come-12
                  :ARG1 (a / and
                        :op1 (s / soul)
                        :op2 (b / be-02
                              :mod
                              (p / physical))))))

Q: Please give me an Abstract Meaning
Representation (AMR) of 'Thanks for the article
and the background - I read both.'.
A:
(m / multi-sentence
      :snt1 (t / thank-01
            :ARG2 (a / and
                  :op1 (a2 / article)
                  :op2 (b / background)))
      :snt2 (r / read-01
            :ARG0 (i / i)
            :ARG1 (b2 / both)))

Q: Please give me an Abstract Meaning
Representation (AMR) of 'International; weapons;
proliferation; dissent; politics.'.
A:
(a / and
      :op1 (i / international)
      :op2 (w / weapon)
      :op3 (p / proliferation)
      :op4 (d / dissent-01)
      :op5 (p2 / politics))

Q: Please give me an Abstract Meaning
Representation (AMR) of 'Speakers addressed
opium production and trafficking in Afghanistan.'.
A:
(a / address-01
      :ARG0 (p / person
            :ARG0-of (s / speak-01))
      :ARG1 (a2 / and
            :op1 (p2 / produce-01
                  :ARG1 (o / opium))
            :op2 (t / traffic-01)
            :location (c / country
                    :wiki "Afghanistan"
                    :name (n / name :op1
                            "Afghanistan"))))

Q: Please give me an Abstract Meaning
Representation (AMR) of 'I'm sorry for the
swearing, i'm not one to usually swear, but the
CIA et al are a joke.'.
A:
(a2 / and
      :op1 (s / sorry-01
            :ARG1 (i / i)
            :ARG2 (s2 / swear-02
                  :ARG0 i))
      :op2 (s3 / swear-02 :polarity -
            :ARG0 i
            :mod (u / usual)
            :concession-of (j / joke
                  :domain (a / and
                        :op1 (g /
                        government-organization
                        :wiki "Central
                        _Intelligence
                        _Agency"
                            :name (n / name
                            :op1 "CIA"))
                            :op2 (o / other)))))

Q: Please give me an Abstract Meaning
Representation (AMR) of 'well, where did it go
in the first place?'.
A:
(g / go-01
      :ARG1 (i / it)
      :ARG4 (a / amr-unknown)
      :prep-in (p / place
            :ord (o / ordinal-entity :value 1)))

Q: Please give me an Abstract Meaning
Representation (AMR) of '[Target Sentence]'.
A:
```

A.3 Top-down CoT Prompt

```
You are a computational linguist. Let's do some
semantic parsing.
Q: Please give me an Abstract Meaning
Representation (AMR) of "The poor kid didn't
receive the gift and the postcard that Dorothy
Gale sent him on May 25th.".
A:
1. "didn't receive": This is the main verb
phrase of the sentence, indicating that "the
poor kid" did not receive something. "the gift
and the postcard" refers to the direct object
of the verb "receive", specifying what the poor
kid didn't receive.
(r / receive-01
      :ARG0 (k / kid)
      :ARG1 (a / and)
      :polarity -)
2. "the poor kid" indicates that "kid" is poor.
(r / receive-01
      :ARG0 (k / kid
            :mod (p / poor))
      :ARG1 (a / and)
      :polarity -)
3. "the gift and the postcard" are the two
```

```
objects that the poor kid didn't receive.
(r / receive-01
      :ARG0 (k / kid
            :mod (p / poor))
      :ARG1 (a / and
            :op1 (g/ gift)
            :op2 (p2 / postcard))
      :polarity -)

4. "that Dorothy Gale sent him on May 25th":
This relative clause modifies "the gift and the
postcard" which are sent by a person ("Dorothy
Gale") to the kid. The kid is the recipient of
"the gift and the postcard". Hence, there is a
reentrancy at node k. "sent him on May 25th":
The action "sent" happened on May 25th, which
is a date-entity. The ":month" is 5 and the
":date" is 25.
(r / receive-01
      :ARG0 (k / kid
            :mod (p / poor))
      :ARG1 (a / and
            :op1 (g/ gift)
            :op2 (p2 / postcard)
            :ARG1-of (s / send-01
                  :ARG0 (p3 / person)
                  :ARG1 k
                  :time (d / date-entity
                        :month 5
                        :date 25)))
      :polarity -)
5.  "Dorothy Gale sent" indicates the person
who sent "the gift and the postcard" is named
"Dorothy Gale".
(r / receive-01
      :ARG0 (k / kid
            :mod (p / poor))
      :ARG1 (a / and
            :op1 (g/ gift)
            :op2 (p2 / postcard)
            :ARG1-of (s / send-01
                  :ARG0 (p3 / person
                        :wiki "Dorothy_Gale"
                        :name (n / name
                              :op1 "Dorothy"
                              :op2 "Gale"))
                  :ARG1 k
                  :time (d / date-entity
                        :month 5
                        :date 25)))
      :polarity -)

The AMR is:
(r / receive-01
      :ARG0 (k / kid
            :mod (p / poor))
      :ARG1 (a / and
            :op1 (g/ gift)
            :op2 (p2 / postcard)
            :ARG1-of (s / send-01
                  :ARG0 (p3 / person
                        :wiki "Dorothy_Gale"
                        :name (n / name
                              :op1 "Dorothy"
                              :op2 "Gale"))
                  :ARG1 k
                  :time (d / date-entity
                        :month 5
                        :date 25)))
      :polarity -)
```

```
Q: Please give me an Abstract Meaning
Representation (AMR) of '[Target Sentence]'.
A:
```

A.4 Bottom-up CoT Prompt

```
You are a computational linguist. Let's do some
semantic parsing.
Q: Please give me an Abstract Meaning
Representation (AMR) of "The poor kid didn't
receive the gift and the postcard that Dorothy
Gale sent him on May 25th.".
A:
1.  This sentence has the following components:
"The poor kid": indicates that "kid" is poor.
(k / kid
      :mod (p / poor))
"the gift and the postcard": "the gift" and
"the postcard" are in an "and" relationship.
(a / and
      :op1 (g/ gift)
      :op2 (p2 / postcard))
"Dorothy Gale": "Dorothy Gale" is the "name"
of a person.
(p3 / person
      :wiki "Dorothy_Gale"
      :name (n / name
            :op1 "Dorothy"
            :op2 "Gale"))
"him": refers to the "The poor kid".
k
"on May 25th": is a date-entity. The ":month"
is 5 and the ":date" is 25.
(date-entity
      :month 5
      :day 25)
2. "didn't receive": This is the main verb
phrase of the sentence, indicating that "the
poor kid" did not receive something. "the gift
and the postcard" is the direct object of the
verb "receive" specifying what the poor kid
didn't receive.
(r / receive-01
      :ARG0 (k / kid
            :mod (p / poor))
      :ARG1 (a / and
            :op1 (g/ gift)
            :op2 (p2 / postcard))
      :polarity -)
3. "that Dorothy Gale sent him on May 25th":
The action "sent" is performed by a person
named "Dorothy Gale" to the kid, and it
happened on May 25th. The kid is the recipient
of "the gift and the postcard". Hence, there is
a reentrancy at node k.
(s / send-01
      :ARG0 (p3 / person
            :wiki "Dorothy_Gale"
            :name (n / name
                :op1 "Dorothy"
                :op2 "Gale"))
      :ARG1 k
      :time (d / date-entity
            :month 5
            :date 25))
4. "the gift and the postcard that Dorothy Gale
sent him on May 25th": "the gift and the
postcard" is the object of "sent". This is a
relative clause, so we make "sent" an "ARG1-of"
belonging to "and".
```

```
(r / receive-01
      :ARG0 (k / kid
            :mod (p / poor))
      :ARG1 (a / and
            :op1 (g/ gift)
            :op2 (p2 / postcard)
            :ARG1-of (s / send-01
                  :ARG0 (p3 / person
                        :wiki "Dorothy_Gale"
                        :name (n / name
                            :op1 "Dorothy"
                            :op2 "Gale"))
                  :ARG1 k
                  :time (d / date-entity
                        :month 5
                        :date 25)))
      :polarity -)
The AMR is:
(r / receive-01
      :ARG0 (k / kid
            :mod (p / poor))
      :ARG1 (a / and
            :op1 (g/ gift)
            :op2 (p2 / postcard)
            :ARG1-of (s / send-01
                  :ARG0 (p3 / person
                        :wiki "Dorothy_Gale"
                        :name (n / name
                              :op1 "Dorothy"
                              :op2 "Gale"))
                  :ARG1 k
                  :time (d / date-entity
                        :month 5
                        :date 25)))
      :polarity -)
Q: Please give me an Abstract Meaning
Representation (AMR) of '[Target Sentence]'.
A:
```

B Appendix: Full Results

B.1 Fine-grained Smatch Results

Smatch sub-metrics	Definition
Unlabeled (Unlab.)	Smatch score after pruning the edge labels.
NoWSD	Smatch score which ignores Propbank senses.
Concepts (Con.)	F-score on the concept identification task.
Named Entity Recognition (NER.)	F-score on the named entity recognition.
Negations (Neg.)	F-score on the negation detection.
Wikification (Wiki.)	F-score on the wikification.
Semantic Role Labeling (SRL.)	Smatch score computed on :ARG-i roles only.
Reentrancy (Reen.)	Smatch score on reentrant edges only.

Table 4: Fine-grained Smatch definition (Damonte et al., 2017)[8]

		Model	Smatch	Unlab.	NoWSD	Con.	NER.	Neg.	Wiki.	Reen.	SRL.
Baseline	Zero-shot	turbo-instruct	3	4	3	4	3	0	4	2	4
		davinci	6	7	6	7	4	1	4	2	8
	1-shot	turbo-instruct	28	34	28	33	27	3	19	17	29
		davinci	32	39	33	36	34	9	40	21	33
	5-shot random	turbo-instruct	34	41	35	**43**	35	3	38	19	**35**
		davinci	**37**	**44**	**38**	**43**	40	8	**48**	21	**35**
	5-shot similarity	turbo-instruct	33	39	34	39	41	7	39	20	32
		davinci	**37**	43	**38**	42	**51**	10	**48**	**23**	**35**
CoT approach	CoT top-down	turbo-instruct	14	17	14	15	10	5	7	6	14
		davinci	27	32	27	30	12	13	14	16	30
	CoT bottom-up	turbo-instruct	12	15	12	13	15	7	11	4	12
		davinci	24	29	25	28	19	**14**	16	14	24

Table 5: Fine-grained Smatch result of baseline and CoT approach (AMR 2.0, raw output)

		Model	Smatch	Unlab.	NoWSD	Con.	NER.	Neg.	Wiki.	Reen.	SRL.
Baseline	Zero-shot	turbo-instruct	5	7	6	7	5	0	1	3	7
		davinci	14	18	14	16	10	1	8	8	18
	1-shot	turbo-instruct	39	48	40	47	39	6	28	27	40
		davinci	41	50	42	47	42	11	51	29	42
	5-shot random	turbo-instruct	42	50	43	52	42	4	47	25	43
		davinci	44	52	45	51	48	9	57	26	42
	5-shot similarity	turbo-instruct	44	52	46	53	55	10	54	30	43
		davinci	**50**	**58**	**51**	**57**	**69**	13	**66**	**34**	**47**
CoT approach	CoT top-down	turbo-instruct	34	44	34	37	23	11	17	22	36
		davinci	36	45	37	41	17	14	22	25	40
	CoT bottom-up	turbo-instruct	30	37	30	34	30	13	24	15	31
		davinci	33	41	34	39	26	**19**	24	22	34

Table 6: Fine-grained Smatch result of baseline and CoT approach (AMR 2.0, post-processed)

Method	Evaluation object	Smatch	Unlab.	NoWSD	Con.	NER.	Neg.	Wiki.	Reen.	SRL.
5-shot similarity	well-formed AMR only	61	68	63	69	75	33	71	42	57
	raw output	53	59	54	59	66	28	62	35	49
	post-processed	60	67	61	67	73	32	69	41	56
CoT top-down	well-formed AMR only	58	66	59	65	72	37	29	38	55
	raw output	43	48	44	47	55	27	19	26	39
	post-processed	55	63	56	62	70	34	25	36	52

Table 7: Fine-grained Smatch result on the GPT-4o model (AMR 3.0)

B.2 GrAPES results

[8] https://github.com/mdtux89/amr-evaluation

Set ID	Dataset	Metric	5-shot	CoT	AMRBart	#
1	**Pragmatic reentrancies**					
	Pragmatic coreference (testset)	Edge recall	08 [03, 22]	06 [02, 18]	39 [25, 55]	36
		Prerequisites	19 [10, 35]	22 [12, 38]	61 [45, 75]	36
2	**Unambiguous reentrancies**					
	Syntactic (gap) reentrancies	Edge recall	15 [07, 28]	27 [16, 42]	49 [34, 64]	41
		Prerequisites	54 [39, 68]	39 [26, 54]	68 [53, 80]	41
	Unambiguous coreference	Edge recall	39 [24, 56]	23 [11, 40]	65 [47, 79]	31
		Prerequisites	61 [44, 76]	52 [35, 68]	77 [60, 89]	31
4	**Rare and unseen words**					
	Rare node labels	Label recall	61 [57, 64]	57 [53, 61]	69 [66, 73]	676
	Unseen node labels	Label recall	61 [52, 69]	56 [47, 65]	45 [37, 54]	117
	Rare predicate senses (excl. -01)	Label recall	21 [13, 34]	18 [10, 30]	45 [32, 58]	56
		Prerequisites	82 [70, 90]	73 [60, 83]	91 [81, 96]	56
	Rare edge labels (ARG2+)	Edge recall	15 [07, 29]	12 [05, 26]	35 [22, 50]	40
		Prerequisites	35 [22, 50]	35 [22, 50]	72 [57, 84]	40
5	**Special entities**					
	Seen names	Recall	69 [67, 71]	71 [69, 73]	94 [93, 95]	1788
	Unseen names	Recall	70 [67, 73]	67 [64, 70]	76 [73, 79]	910
	Seen dates	Recall	68 [62, 73]	66 [59, 71]	94 [90, 96]	233
	Unseen dates	Recall	51 [45, 58]	56 [49, 63]	86 [81, 90]	204
	Other seen entities	Recall	88 [83, 91]	79 [73, 84]	97 [94, 99]	237
	Other unseen entities	Recall	88 [81, 93]	70 [61, 78]	78 [69, 85]	109
6	**Entity classification and linking**					
	Types of seen named entities	Recall	59 [57, 62]	61 [58, 63]	92 [90, 93]	1628
		Prerequisites	67 [64, 69]	69 [67, 71]	94 [93, 95]	1628
	Types of unseen named entities	Recall	39 [35, 43]	36 [32, 40]	51 [47, 55]	659
		Prerequisites	60 [56, 64]	57 [53, 61]	70 [66, 73]	659
	Seen and/or easy wiki links	Recall	73 [71, 75]	19 [17, 21]	87 [85, 88]	2064
	Hard unseen wiki links	Recall	33 [28, 39]	05 [03, 08]	09 [06, 13]	277
7	**Lexical disambiguation**					
	Frequent predicate senses (incl. -01)	Label recall	46 [43, 48]	39 [36, 41]	86 [84, 88]	1654
		Prerequisites	78 [76, 80]	73 [70, 75]	94 [93, 95]	1654
	Passives	Edge recall	47 [37, 58]	28 [19, 38]	76 [66, 84]	83
		Prerequisites	57 [46, 67]	39 [29, 49]	80 [70, 87]	83
	Unaccusatives	Edge recall	21 [12, 34]	27 [17, 41]	71 [57, 82]	48
		Prerequisites	52 [38, 66]	48 [34, 62]	79 [66, 88]	48
9	**Non-trivial word-to-node relations**					
	Ellipsis	Recall	12 [05, 27]	15 [07, 31]	55 [38, 70]	33
		Prerequisites	58 [41, 73]	45 [30, 62]	94 [80, 98]	33
	Multinode word meanings	Recall	14 [07, 26]	04 [01, 13]	84 [71, 92]	50
	Imperatives	Recall	04 [01, 11]	00 [00, 05]	66 [55, 75]	76
		Prerequisite	66 [55, 75]	59 [48, 70]	89 [81, 95]	76

Table 8: Results on all GrAPES categories extracted from the AMR 3.0 test set. Grey numbers in square brackets are 95%-Wilson confidence intervals. Our 5-shot and CoT results are for GPT-4o with similarity sampling and top-down methods respectively. AMRBart (Bai et al., 2022) is a competitive fine-tuned BART model.

C Appendix: CoT Samples

C.1 Polarity Change Analysis

Figure 7 is the reasoning steps sampled from a top-down CoT generated by turbo-instruct model. The target sentence is in example (3).

(3) *Do they not realise that you don't get piles of change from the bank..*

In the earlier step, GPT added a `(:polarity -)`, because there is a *"not"* in the text. But in the later step, GPT stated that there was another negation, so it needed to switch the polarity. The explanation seems sort of reasonable. This polarity switching is not something AMR does, and was not in the example CoT; GPT 'learned' this by itself. However, here GPT is wrong anyway: the second *not* negates *get*, so there should instead be a second `(:polarity -)` in the AMR.

D Appendix: Parsability Analysis

Here we provide graphs illustrating the distribution of unparsable raw results by sentence length. The first two figures are the sentence length distribution over the test sets.

```
1. This sentence has the following components:
"they": refers to a group of people.
(p / person
      :quant 2)
"not realise": This is the main verb phrase of
the sentence, indicating that the group of
people do not realize something.
(r / realize-01
      :ARG0 (p / person
            :quant 2)
      :polarity -)
"that": introduces a subordinate clause.
2. "you don't get piles of change": This is the
subordinate clause, indicating that "you" do not
receive a large amount of change.
(r2 / receive-01
      :ARG0 you
      :ARG1 (c / change
            :quant (p2 / pile)))
"from the bank": specifies where the change is
not received from.
(f / from
      :ARG0 c
      :ARG1 (b / bank))
3. "Do they not realise that you don't get
piles of change from the bank.": The main verb
phrase is "realise", and the subordinate clause
is the object of "realise". The subordinate
clause is also negated, so there is a polarity
change at the top level.
(r / realize-01
      :ARG0 (p / person
            :quant 2)
      :ARG1 (r2 / receive-01
            :ARG0 you
            :ARG1 (c / change
                  :quant (p2 / pile))
            :ARG2 (f / from
                  :ARG0 c
                  :ARG1 (b / bank)))
      :polarity +)
The AMR is:
(r / realize-01
      :ARG0 (p / person
            :quant 2)
      :ARG1 (r2 / receive-01
            :ARG0 you
            :ARG1 (c / change
                  :quant (p2 / pile))
            :ARG2 (f / from
                  :ARG0 c
                  :ARG1 (b / bank)))
      :polarity +)

Golden:
(r / realize-01 :polarity - :mode interrogative
      :ARG0 (t / they)
      :ARG1 (g / get-01 :polarity -
            :ARG0 (y / you)
            :ARG1 (p / pile
                  :mod (c / change))
            :ARG2 (b / bank)))
```

Figure 7: CoT sampling: polarity change in ex. (3)

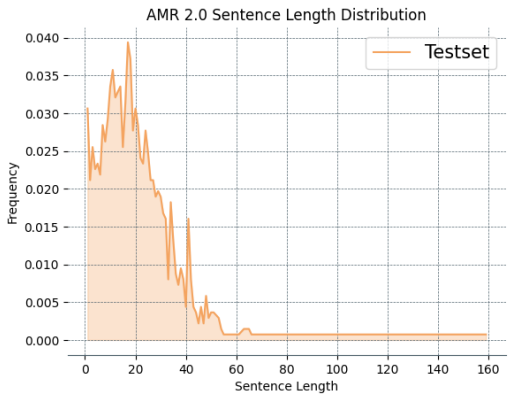


Figure 8:
Distribution of Testset Sentence Length (AMR 2.0)

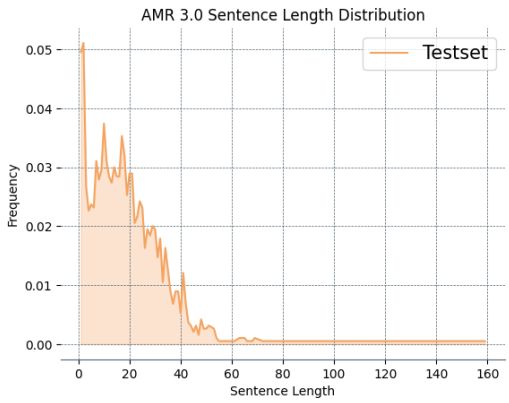


Figure 9:
Distribution of Testset Sentence Length (AMR 3.0)

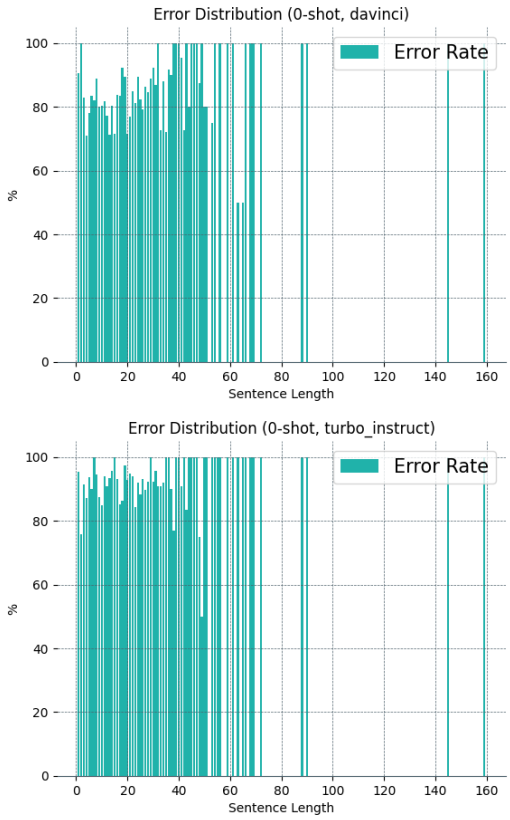


Figure 10: Error Distribution of Zero-shot (AMR 2.0)

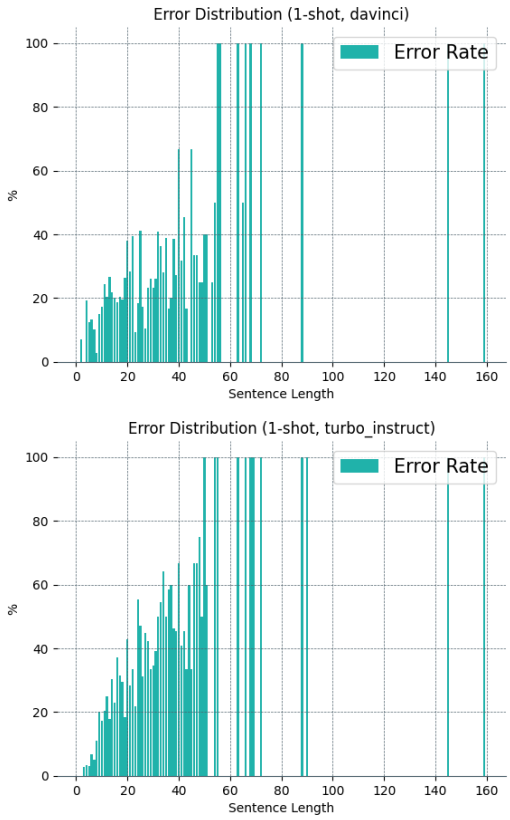


Figure 11: Error Distribution of 1-shot (AMR 2.0)

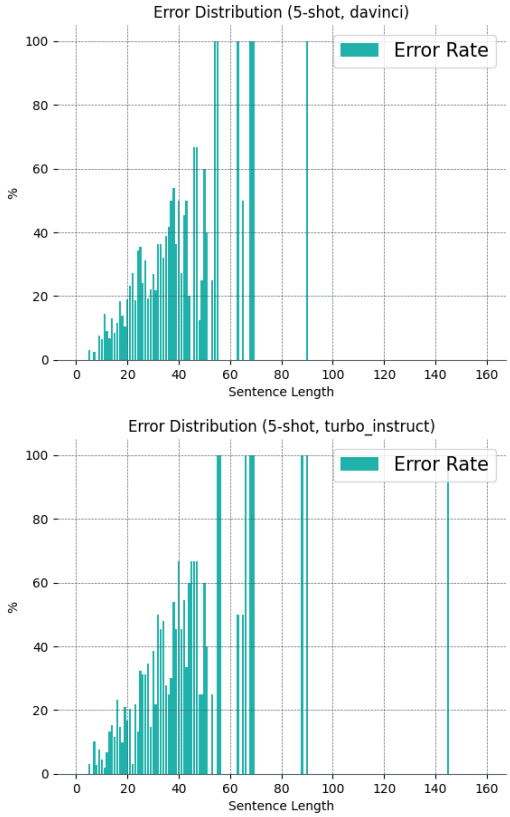


Figure 12: Error Distribution of 5-shot random sampling (AMR 2.0)

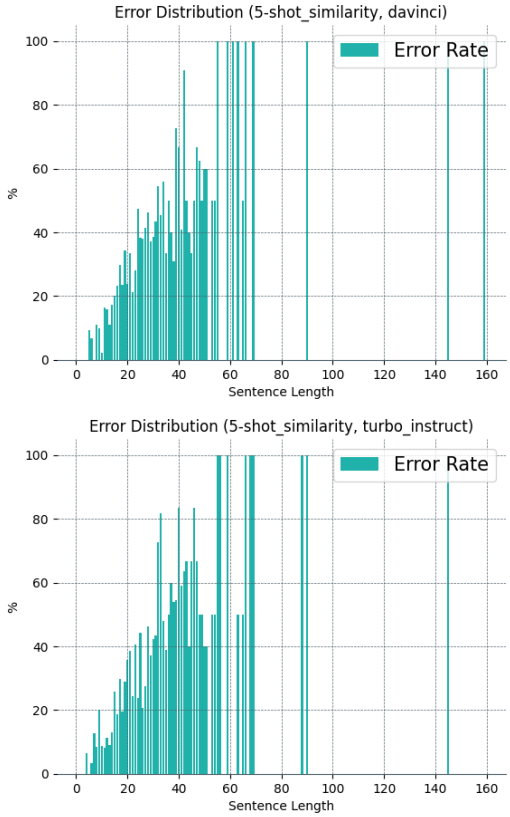


Figure 13: Error Distribution of 5-shot similarity sampling (AMR 2.0)

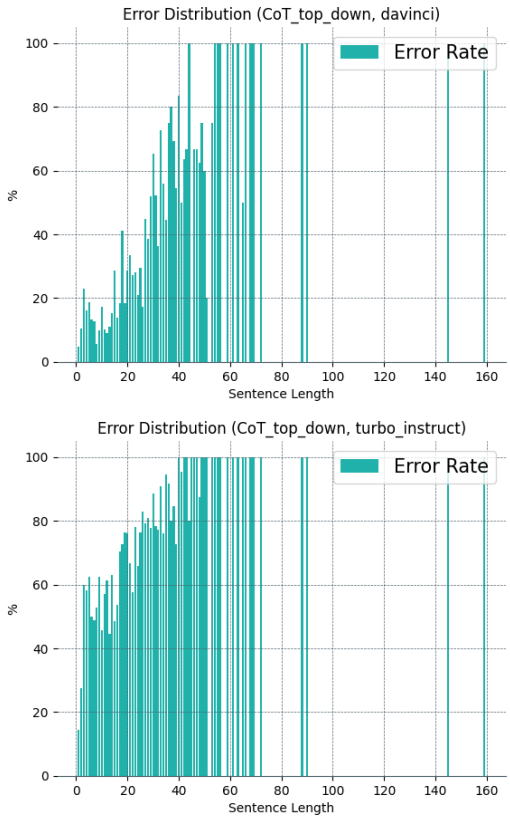


Figure 14: Error Distribution of CoT top-down (AMR 2.0)

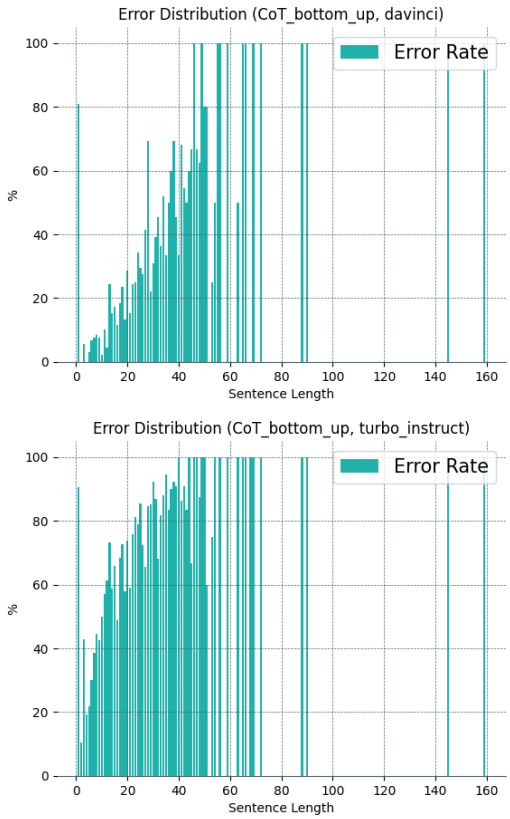


Figure 15: Error Distribution of CoT bottom-up (AMR 2.0)

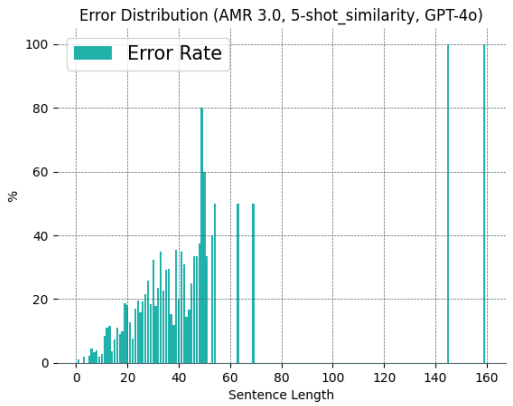


Figure 16: Error Distribution of 5-shot similarity sampling (AMR 3.0)

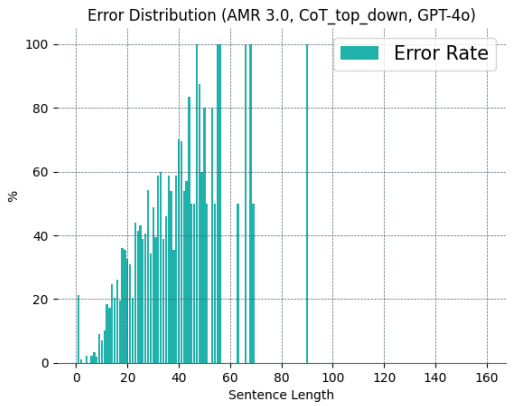


Figure 17: Error Distribution of CoT top-down (AMR 3.0)

E Appendix: Two-step Version

E.1 Two-step Version: Prompt Design Detail

Since there are hundreds of non-core roles of AMR and prompts have token limitations, it's impossible to provide sufficient numbers of examples in one single sentence or in one CoT prompt.

To address this problem, we break the whole AMR generation process into two parts, firstly generate core roles (e.g. the ARGs) and then add non-core roles in a separate step.

Step 1 is to generate core roles with 5-shot prompting, which the prompts only provides AMR with core roles. There are two different AMR styles of Step 1, one is simply pruning all non-core roles in the examples, which might break an AMR graph into several sub-graphs (notated as "multi-graph"). Another is only pruning the non-core roles leave nodes, which might be some remnants of non-core roles in the AMR graph, but the AMR graph will remain as a single graph (notated as "one-graph"). A "one-graph" style example is shown in E.1.1.

Step 2 is a CoT prompt with an example and a guideline that includes several non-core roles, an example is shown in E.1.2. The results are shown in Table 9 and Table 10.

E.1.1 Step 1: Example 5-shot Prompt for generating core roles

```
You are a computational linguist. Let's do some
semantic parsing.
Q: Please give me an Abstract Meaning
Representation (AMR) of 'However, when it comes
to the soul and the physical being, in the
Church's eyes, the soul trumps the former.'.
A:
(c2 / contrast-01
      :ARG2 (t / trump-01
            :ARG0 (s2 / soul)
            :ARG1 b
            :ARG1-of (v / view-02
                  :ARG0 (o / organization))
            :condition (c3 / come-12
                  :ARG1 (a / and))))

Q: Please give me an Abstract Meaning
Representation (AMR) of 'Thanks for the article
and the background - I read both.'.
A:
(m / multi-sentence
      :snt1 (t / thank-01
            :ARG2 (a / and))
      :snt2 (r / read-01
            :ARG0 (i / i)
            :ARG1 (b2 / both)))

Q: Please give me an Abstract Meaning
Representation (AMR) of 'International; weapons;
proliferation; dissent; politics.'.
A:
(a / and)

Q: Please give me an Abstract Meaning
Representation (AMR) of 'Speakers addressed
opium production and trafficking in Afghanistan.'.
A:
(a / address-01
      :ARG0 (p / person
            :ARG0-of (s / speak-01))
      :ARG1 (a2 / and
            :op1 (p2 / produce-01
                  :ARG1 (o / opium))))

Q: Please give me an Abstract Meaning
Representation (AMR) of 'I'm sorry for the
swearing, i'm not one to usually swear, but the
CIA et al are a joke.'.
A:
(a2 / and
      :op1 (s / sorry-01
            :ARG1 (i / i)
            :ARG2 (s2 / swear-02
                  :ARG0 i))
      :op2 (s3 / swear-02 :polarity -
            :ARG0 i))

Q: Please give me an Abstract Meaning
Representation (AMR) of 'well, where did it go
in the first place?'.
A:
(g / go-01
      :ARG1 (i / it)
      :ARG4 (a / amr-unknown))

Q: Please give me an Abstract Meaning
Representation (AMR) of '[Target Sentence]'.
A:
```

E.1.2 Step 2: CoT with guildline

```
You are a computational linguist. Let's do some
semantic parsing.
non-core roles list:
:accompanier, :age, :beneficiary, :concession,
:condition, :consist-of,
:degree, :destination, :direction, :domain,
:duration, :example,
:extent, :frequency, :instrument, :li,
:location, :manner,
:medium, :mod, :mode, :name, :ord, :part,
:path, :polarity, :polite, :poss, :purpose,
:quant,
:range, :scale, :source, :subevent, :time,
:topic,
:unit, :value, :wiki
date-entity:
:calendar, :century, :day, :dayperiod, :decade,
:era, :month, :quarter, :season, :timezone,
:weekday, :year, :year2
conjunctions:
:op1, :op2, :op3, :op4, ...
other components:
:country

Q: Here is the sentence and its half-completed
AMR graph.
Sentence:
'The poor kid didn't receive the gift and the
postcard that Dorothy Gale sent him on May 25th.'
```

```
Half-completed AMR graph:
(r / receive-01
      :ARG0 (k / kid)
      :ARG1 (a / and
            :ARG1-of (s / send-01
                  :ARG0 (p3 / person)
                  :ARG1 k)))

Please give me an Abstract Meaning
Representation (AMR) of the sentence according
to its half-completed AMR graph, and
aforementioned non-core roles list.
A:
"(r / receive-01)" requires a ":polarity -"
because the receiving "didn't" happen.
(r / receive-01
      :polarity -)

For ":ARG0 (k / kid)", because the "kid" is
"poor", it is a modifier. That is the ":mod" in
the non-core roles list, so the ":mod" is
"(p / poor)".
:ARG0 (k / kid
      :mod (p / poor))

"the gift and the postcard" is a conjunction,
so ":ARG1 (a / and)" has "op1" and "op2" from
the list; the ":op1" is "(g/ gift)" and the
":op2" is "(p2 / postcard)"
:ARG1 (a / and
      :op1 (g/ gift)
      :op2 (p2 / postcard))

For ":ARG0-of (s / send-01)", "sent him on May
25th", indicates that the sending happened on
May 25th, which is a time. We use ":time" in
the non-core roles list, and it is a
"date-entity"; the ":month" is 5 and the ":day"
is 25.
:ARG1-of (s / send-01
      :ARG0 (p3 / person)
      :ARG1 k
      :time (d / date-entity
            :month 5
            :date 25))

For ":ARG0 (p3 / person)",  "Dorothy Gale", is
a name, which is ":name" in the non-core roles
list, and also ":wiki" in the non-core roles
list, so ":wiki" is "Dorothy_Gale".
For the ":name", the two parts of this name are
introduced with "op1" and "ops2". ":op1" is
"Dorothy" and the ":op2" is "Gale".
:ARG0 (p3 / person
      :wiki "Dorothy Gale"
      :name (n / name
            :op1 "Dorothy"
            :op2 "Gale"))

The AMR is:
(r / receive-01
      :ARG0 (k / kid
            :mod (p / poor))
      :ARG1 (a / and
            :op1 (g/ gift)
            :op2 (p2 / postcard)
            :ARG1-of (s / send-01
                  :ARG0 (p3 / person
                        :wiki "Dorothy_Gale"
                        :name (n / name
                              :op1 "Dorothy"
                              :op2 "Gale"))
                  :ARG1 k
                  :time (d / date-entity
                        :month 5
                        :date 25)))
      :polarity -)

Q: Here is the sentence and its half-completed
AMR graph.
Sentence:
'[Target_Sentence]'

Half-completed AMR graph:
[Target_Graph]

Please give me an Abstract Meaning
Representation (AMR) of the sentence according
to its half-completed AMR graph, and
aforementioned non-core roles list.
A:
```

E.2 Two-step Version Result

Step 1 style	Model name	Smatch	Unlab.	NoWSD	Con.	NER.	Neg.	Wiki.	Reen.	SRL.
one_graph	turbo-instruct	12	14	12	14	16	3	13	7	13
	davinci	**24**	**29**	**25**	**28**	**30**	**8**	**25**	**17**	**26**
multi_graphs	turbo-instruct	5	5	5	5	4	3	1	2	5
	davinci	11	13	11	13	10	4	5	5	12

Table 9: Fine-grained Smatch result of two-step version (AMR 2.0, raw output)

	Method	Model	Smatch	Unlab.	NoWSD	Con.	NER.	Neg.	Wiki.	Reen.	SRL.
Baseline	zero-shot	davinci	6	7	6	7	4	1	4	2	8
	1-shot	davinci	32	39	33	36	34	9	40	21	33
	5-shot (similarity)	davinci	**37**	**43**	**38**	**42**	**51**	10	**48**	**23**	**35**
CoT	CoT top-down	davinci	27	32	27	30	12	**13**	14	16	30
Two-step approach	one_graph	davinci	24	29	25	28	30	8	25	17	26

Table 10: Fine-grained Smatch result comparison among baseline, CoT, and two-step version (AMR 2.0, raw output)

F Appendix: Data Contamination

Dataset	Model	Smatch	Unlab.	NoWSD	Con.	NER.	Neg.	Wiki.	Reen.	SRL.
training	turbo-instruct	36	43	37	45	39	6	39	19	37
	davinci	34	41	35	41	38	6	44	19	35
test	turbo-instruct	34	41	35	43	35	3	38	19	35
	davinci	37	44	38	43	40	8	48	21	35

Table 11: Fine-grained Smatch result of data contamination experiment (AMR 2.0, raw output)