Context is (Almost) Everything: Llama-3 on Structured Output and AMR Parsing

Maja Buljan, Stephan Oepen, Lilja Øvrelid
Language Technology Group, University of Oslo
{majabu,oe,liljao}@ifi.uio.no

Abstract

This paper evaluates the ability of an open-source LLM (Llama-3.1) to compute sentence-level semantics and encode it in formal language. We here compare two versions of the model on the task of generating a meaning representation graph for a given English sentence in the form of Abstract Meaning Representation. We explore the model's in-context learning capability, comparing zero-shot prompting to few-shot demonstrations of varying levels of specificity. We find that Llama-3.1 frequently makes errors when reproducing the syntactic structure of both seen and unseen structured output, and that it only achieves near-SotA parsing performance when shown highly specific demonstrations similar in structure to the target sentence graph. We include an in-depth analysis of the model output, considering performance through the lens of fine-grained semantic phenomena, different classes of graph properties, and graph complexity.

Keywords: meaning representation parsing, large language models, in-context learning

1. Introduction

Large language models have attracted great interest due to their ability to generate vast amounts of fluent natural language, and have permeated various aspects of NLP research, as well as becoming household names in daily life. But, of the many questions that still provoke scientific interest, we find two of particular importance: (1) LLMs' imitation of natural language is demonstrably successful, but how deep is these models' *understanding* of natural language? and (2) How many NLP challenges can be "solved" with LLMs?

In our work, we present a focused exploration of these two issues through the lens of meaning representation parsing (MRP). We evaluate an LLM's ability to grasp the semantics of a sentence, by producing structured output in the form of a meaning representation graph – specifically, the Abstract Meaning Representation (AMR) framework. Thus, we approach the high-level questions by asking: "Are LLMs good enough at computing meaning to serve as an out-of-the-box AMR parser?"

As we discuss in the background section, LLMs have been applied to numerous NLP problems, including tasks involving structured output (such as generating Python code), and machine translation (where the SotA is increasingly becoming to prompt an LLM). As sequence-to-meaning-representation parsing can be viewed as a translation problem, and as seq2seq parsers are an established approach in MRP, we are motivated to experiment with using LLMs as a meaning representation parser. By analysing and perfecting this use-case, MRP can eventually become one of the methods by which to quantify LLMs' grasp of language meaning.

The first challenge is to learn to what degree a general-purpose LLM can "translate" into formal languages, depending on (un)seen data, by testing for and contrasting memorisation vs. generalisation. If we can make an off-the-shelf LLM generate syntactically well-formed AMR graphs, then we can make use of a dataset such as GrAPES – an AMR parsing benchmark dataset, described in more detail in sections 2 and 3.1 – with its rich internal structure, to probe the LLM's ability to recognise subphenomena in sentence-level meaning. Are there specific differences in performance that correlate with certain semantic constructions? And finally, as LLMs are shown to learn from few-shot demonstrations, how does the choice of these examples, as well as prompt wording, affect the model's in-context learning abilities?

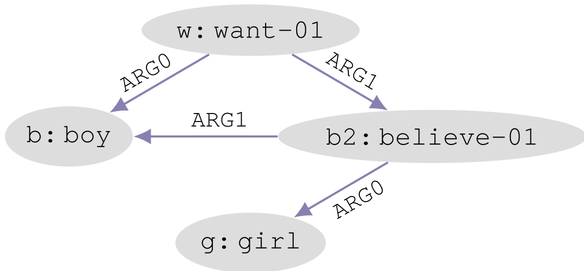


Figure 1: An AMR graph for the sentence "The boy wants the girl to believe him."

In this paper, we test the viability of using Llama-3.1, an open-source LLM, as an out-of-the-box seq2seq AMR parser, using output in PENMAN notation. We experiment with two model sizes – a small 8B-parameter model, and a medium-sized 70B, and find that, while Llama-3.1 does not

Proceedings of the Fifteenth Language Resources and Evaluation Conference (LREC 2026), pages 11679–11698
11–16 May 2026.

achieve performance close to SotA, it is still comparable to its larger GPT counterparts used in similar studies. We contribute to the issue of syntax errors in LLMs' output of formal language by introducing a robust post-processing procedure to correct these errors. We also examine how the choice of demonstrations in few-shot settings influences model output, and find that the process is brittle. In particular, we analyse the effect of including random demonstrations of AMR graphs, in contrast to highly specific demonstrations similar in sentence structure to the target sentence. This gives an idea of the model's upper bound on this task, devoid of fine-tuning. Finally, the paper includes a more fine-grained analysis of the strengths and weaknesses of Llama-3.1 as an AMR parser, analysing performance through the lens of semantic phenomena, graph complexity, and graph properties.

2. Background

Abstract Meaning Representation Meaning representation parsing (MRP) is the task of creating a general-purpose representation of sentence meaning in the form of a labelled directed graph. Of the many MR frameworks present in the field (see, e.g., Oepen et al., 2020, and Sadeddine et al., 2024, for extensive surveys), AMR (Banarescu et al., 2013) is widely used; being syntax-agnostic, it is suitable for cross-lingual applications (Damonte and Cohen, 2018) and has a steadily growing base of datasets in different languages (despite the fact that it is, by design, biased towards English; Banarescu et al., 2013). It has also found applications in downstream NLP tasks, such as text summarisation, paraphrasing, or question answering (Issa et al., 2018; Tohidi and Dadkhah, 2022; Mansouri, 2025).

An AMR datapoint is a directed graph comprising labelled nodes and edges, where nodes represent semantic units of the sentence – expressed using PropBank framesets (Kingsbury and Palmer, 2002) – and edges denote the relations between them.

Figure 1 demonstrates the properties of an AMR graph using an example from the AMR annotation guidelines[1]: "The boy wants the girl to believe him." In this graph, the four nodes represent instances, or semantic units of the sentence. Two of them are labelled with PropBank frames (`want-01`, `believe-01`), and the other two with common nouns (`boy`, `girl`). All the edges are labelled using PropBank frame arguments; `ARG0` labelling the (proto-)agent, and `ARG1` the (proto-)patient of a particular action.

In PENMAN notation (Matthiessen and Bateman, 1991; Goodman, 2020), this graph is serialised into a linear string as shown in Figure 2.

[1] github.com/amrisi/amr-guidelines

```
(w / want-01
:ARG0 (b / boy)
:ARG1 (b2 / believe-01
            :ARG0 (g / girl)
            :ARG1 b))
```

Figure 2: AMR graph for the sentence "The boy wants the girl to believe him.", in PENMAN notation.

The PENMAN notation was initially an adaptation for human writing and reading of graph annotation, and later adopted as a convenient way to encode graphs as character sequences for seq2seq parsing (described in the following subsection).

MR parsers in general, and AMR parsers in particular, are canonically divided into three categories according to their approach to graph generation. Transition-based models (Dozat and Manning, 2018; Lindemann et al., 2020), inspired by transition-based syntactic dependency parsers, generate graph elements following a sequence of parse transitions. Graph-based models (Cai and Lam, 2020), also called sequence-to-graph models, iteratively attend to a particular part of the input sequence and decode the graph structure globally. Sequence-to-sequence (seq2seq) models (Blloshmi et al., 2020) generate a linearised version of AMR from the input sentence string, through the intermediate step of generating a sequence of concepts. Thus, recent experiments (Bai et al., 2025) approach AMR parsing by using LLMs to "translate" an input sentence into formal language – AMR graphs in PENMAN notation.

LLMs and structured output For generating structured output, LLMs have most commonly been trained and evaluated on generating code (Chen et al., 2021; Xu et al., 2022). Concurrently, work has been done on pushing moderate-sized LLMs' translation capabilities closer to MT performance of SotA dedicated translation models (Xu et al., 2024). We see the task of generating AMR graphs, in a seq2seq setup, as a combination of these tasks.

On the task of LLM-based parsing, and particularly AMR parsing, Ettinger et al. (2023) set the groundwork with an in-depth qualitative study, manually analysing AMR graph output from the GPT family of models (Achiam et al., 2023). This study is the principal motivation for our work. Following their approach, we move towards a quantitative study, expanding the dataset used for testing, and complementing it with data introduced by the GrAPES benchmark (Groschwitz et al., 2023) for fine-grained analysis. Similarly, the concurrent work of Li and Fowlie (2025) expands on the work of Ettinger et al. (2023), and they conduct a set of quantitative experiments also using GPT models. Papakostas et al. (2025) experiment with a combination of symbolic and neural approaches, to evaluate LLMs on AMR parsing, as well as coreference resolution.

Elsewhere in the MRP space, we look to previous studies on in-depth meaning representation graph analysis (Szubert et al., 2020; Buljan et al., 2022, 2024), and follow their methodology in contrasting system performance according to graph complexity, and structural vs. linguistic factors of AMR graphs. In a similar vein, Ho (2025) evaluates the performance of fine-tuned AMRs and cites findings regarding the structural vs. semantic performance of various LLMs.

Research has shown that general-purpose LLMs, with no fine-tuning or gradient updates, demonstrate improved performance with the addition of task specification and few-shot demonstrations alone (Brown et al., 2020; Kong et al., 2024). However, other studies have delved into prompt programming that elicits improved results in zero-shot settings through careful wording that makes the model locate previously learned examples, rather than meta-learning with few-shot demonstrations (Reynolds and McDonell, 2021). While in-depth prompt engineering is beyond the scope of this paper, these studies nonetheless provide motivation to experiment with several zero-shot and few-shot settings, while also working under the assumption that this challenge is unlikely to present a significant portion of the model's seen data.

The model we use in this study – specifically, its size variants Llama-3.1-8B-Instruct and Llama-3.1-70B-Instruct – was published (Meta, 2024; Touvron et al., 2023) after the release of the datasets used in our experiments. However, given the proportion of freely available AMR data online compared to other NLP benchmarks, we proceed with the assumption that data contamination is minimal. To avoid contributing to the problem of LLM experimentation and indirect data contamination (Balloccu et al., 2024), we choose to use an open-source, locally run model for our experiments.

3. Experimental Setup

The aim of our experiments is to test the performance of Llama-3.1 as a semantic parser on two benchmark AMR datasets, described in 3.1. We vary the amount of guidance given in zero-shot and few-shot prompting scenarios, as detailed in 3.2, with the aim of searching for theoretical lower and upper bounds, and finding a balance between performance and practicality. A common issue seen with LLM output in this context is syntactically ill-formed AMR graphs, requiring automated syntax correction (3.3) before evaluation (3.4).

3.1. Data and Models

In our experiments, we use three datasets, containing English sentences manually annotated with AMR graphs: (1) the handcrafted GrAPES dataset[2] (1415 sentences[3]) created by Groschwitz et al. (2023); (2) the AMR 3.0 dataset[4] (Knight et al., 2020). specifically, the 1898-sentence test set, comprising web data such as news articles and discussion forum content; and (3) *The Little Prince* (LPP) dataset[5] (Bender et al., 2015; Oepen et al., 2019, 2020) (1562 sentences), comprising the English translation of Antoine de Saint-Exupéry's famous novel. The first two are comparable to the data used by Li and Fowlie (2025) in their work. We include the third dataset as a qualitative middle ground between the first two – a literary work (3), compared to web data (2), and manually crafted sentences for targeted challenges (1). The GrAPES dataset is of particular interest to our study, as it contains handcrafted sentences and their accompanying AMR graphs in PENMAN notation. The datapoints focus on challenges to structural generalisation, and are categorised by their particular semantic features, such as nested control, recursion and coreference, or PP attachment.

Output is acquired by prompting two versions of the Llama-3.1 model: the 8B- and 70B-parameter instruction-tuned text-only models. Both versions of the model are obtained through Hugging Face[6], and inference is run locally on an HPC cluster.

3.2. Prompting Strategies

As role-play is shown to improve the zero-shot reasoning of LLMs (Kong et al., 2024), we follow the wording of Ettinger et al. (2023), and begin each prompt with *"You are an expert linguistic annotator."*

In addition to zero-shot prompts with the annotator instruction, we experiment with variants of few-shot prompting. As LLMs are shown to be sensitive to the choice of examples in few-shot settings, we augment the prompt with one, and later five example demonstrations. The demonstrations are sentence-and-graph pairs, selected from the source dataset of the target sentence (we call these *generic* demonstrations). In the case of GrAPES data, we also experiment with demonstrations taken from the target sentence's category subset (*specific* demonstrations).

[2]https://github.com/jgroschwitz/GrAPES

[3]Note that, throughout the paper, we use the terms *sentence*, *datapoint*, and *graph* interchangeably for individual elements of the datasets. We note the exception when specifically referring to "surface sentence string" as different from "AMR graph".

[4]https://catalog.ldc.upenn.edu

[5]https://www.isi.edu

[6]https://huggingface.co/

For the sake of consistency, after being randomly selected, the demonstrations are “frozen” and reused for every instance of prompting, while respecting the principle of k-fold testing. The intention with the one-shot and five-shot generic demonstrations is to establish how much of an improvement is possible, in terms of both graph well-formedness and accuracy, when going from zero-shot prompting to showing a single example of a target sentence and its AMR graph – and, in turn, how much of a difference several demonstrations make compared to one.

The specific demonstrations, used only in experiments with GrAPES, are selected from the respective GrAPES category of the target sentence. As before, the random selection is frozen for all target sentences within the category, as a k-fold test setup. The aim of the specific demonstrations is to establish how much of an improvement in both syntactic (graph notation) and semantic (meaning representation) accuracy is visible when instructing the model on the example of a sentence that is, in semantic structure, highly similar to the target sentence, as opposed to a generic example of a target sentence and its AMR graph. In the case of five demonstrations, the setup is arguably pushing the theoretical upper limit of model performance, rather than testing for practical parsing purposes, as having five highly similar AMR graph examples for every target sentence we are tasking Llama to parse is highly unlikely in a practical scenario.

Additionally, every version of prompt wording includes a variant with the additional instruction to provide output in PENMAN notation. This is to test – particularly for the zero-shot scenario – whether such a prompt will instruct the model to adhere to a format it may have seen in its training data, or whether it would, arguably, introduce confusion by detracting from the focus of the prompt, which is the target sentence (Shi et al., 2023).

As discussed in Section 4.1, the best-performing prompt is *v2*: *“You are an expert linguistic annotator. Please generate an Abstract Meaning Representation graph [in PENMAN notation] for the following sentence: [target sentence]. Please begin your annotation with <amr>, and end it with </amr>.”*

We find that the last instruction is more successful at reducing chatter, in-graph comments, and multiple graph versions than other explicit wordings to avoid such output.

3.3. Post-processing

A considerable portion of the model’s output (ranging from 30–99%) is not well-formed and parseable, adhering to a superficial imitation of the PENMAN notation form, but containing syntax errors that cause the scorer to reject it as unacceptable input.

```
(w / woman :~(plur w)
:ARG (h / hate-01
  :ARG0 w
  :ARG1 (s / soldier)
  :polarity -
  :condition (a / after-01
    :time (t / time :mod all))))
:ARG (t2 / think-01
  :ARG0 (a / astronaut)
  :ARG1 (l / like-07
  :ARG0 (w2 / woman :~ (plur w2))
    :ARG1 s)))
```

Illegal characters

Duplicate variables

Mismatched parentheses

Figure 3: Raw system output (AMR graph in PENMAN notation) for the GrAPES sentence *“The women who the astronaut thought liked the soldier actually hated her after all.”*. The highlights show syntax errors that must be corrected by the superficial syntax-fixing post-processing step before the output can be automatically evaluated and scored.

As shown in Table 2, only 1-14% of raw system output is viable for parsing and scoring, across prompt variants and few-shot-setups – an issue also encountered and discussed by Li and Fowlie (2025). Performing scoring and evaluation on such a small subset of data would be next to pointless.

This means that the output requires superficial post-processing before evaluation. We develop an automated syntax correction script, comparable to that introduced in the parallel study of Li and Fowlie (2025), and expanding on some additional frequently seen issues in output syntax. Using regular expressions and heuristic rules, the syntax fixer corrects superficial issues such as mismatched parentheses, fragmented or concatenated node expressions, missing or duplicate variable names, and illegal characters that need removing or replacing; attempting to strike a balance between capturing the most commonly observed errors, and not requiring correction rules of great complexity that pertain to only a handful of observed errors.

Figure 3 shows an example of raw output for the GrAPES sentence *“The women who the astronaut thought liked the soldier actually hated her after all.”*, with syntax errors that will be corrected in the post-processing step.

These errors, such as mismatched parentheses or duplicate variable names, are not errors that damage the score during evaluation – but serious syntactic errors that prevent AMR scorers from running. Therefore, introducing this post-processing step does not “improve” evaluation results, apart from moving beyond the trivial solution of giving the system an F_1 score larger than 0.00 on only the aforementioned 1-14% of datapoints. Other issues visible in the example above, such as incorrect arguments, are not addressed by the post-processing script – these are examples of poor AMR annotation that are penalised in the evaluation process.

Our syntactic post-processing script is available online[7], and, in the interest of space, we further elaborate on the corrective steps in Appendix D.

Additionally, unless explicitly instructed otherwise, the model produces long strings of exposition, comments, and multiple versions of AMR graphs for the target sentence. The candidate graph is extracted from the commentary, and, in cases where the model outputs multiple graph versions, only the first graph is accepted.

The effectiveness of post-processing raw output before evaluation is discussed detail in Section 4.2.

3.4. Evaluation

We evaluate the output data using `mtool`[8], and retrieve Smatch (Cai and Knight, 2013) F_1 scores for the output graphs. Additionally, we use mtool's MRP metrics (Kuhlmann and Oepen, 2016) to glean additional information about the model's perfomance with regards to structural aspects of the graph (root nodes and edges), and node-local information (node labels and properties).

4. Results and Findings

The main results of our experiments are shown in Table 1. Each test run (prompting setup over the entire dataset) is repeated five times, and we report the average F_1 score. In the interest of space, we relegate the extended report of evaluation results and details to Appendix C, including the standard deviation between runs, and performance on well-formed graphs only.

Because of spatial limitations, we will also focus all in-depth analysis in Sections 4.2 and beyond only on the best-performing combination of prompt variation, datset, model size, and evaluation run. However, the unabridged in-depth analysis of all datasets is included in Appendix D.

4.1. Prompt Variation

Table 1 presents evaluation results for output from the larger model (Llama-3.1-70B-Instruct) on the GrAPES, AMR, and LPP datasets. The numbers in the top row indicate the number of demonstrations used in zero- and few-shot setups (0, 1, 5),

		0	0+	1g	1g+	5g	5g+
GrAPES	v1	.380	**.410**	.502	.513	.451	.445
	v2	.228	.400	**.625**	**.624**	**.541**	**.541**
	v3	**.458**	.404	.547	.543	.500	.486
AMR	v1	.179	.198	.241	.237	.267	.256
	v2	.107	**.218**	**.313**	**.312**	**.347**	**.335**
	v3	**.219**	.181	.249	.247	.283	.277
LPP	v1	.312	.329	.376	.366	.419	.391
	v2	.283	**.354**	**.407**	**.404**	**.470**	**.442**
	v3	**.339**	.298	.386	.384	.449	.424

Table 1: F_1-scores for Llama-3.1-70B-Instruct on three datasets: GrAPESs (top), AMR 3.0 (middle), and LPP (bottom), across all variations of prompts and *k*-shot demonstrations. Note that this table does not include results for the experiments with specific demonstrations (*1s(+)*, *(s(+))*, as that evaluation setup is particular to the GrAPES dataset. These results are highlighted elsewhere in the text.

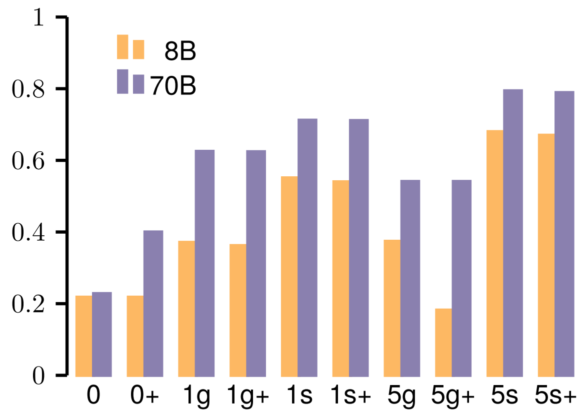


Figure 4: $F1$ scores on the GrAPES dataset for Llama-3.1 8B and 70B, showing the difference between 0-shot and k-shot prompting, with generic and specific demonstrations.

while the *plus* symbol indicates that the prompt includes an explicit instruction to produce output in PENMAN notation. The rows present results for different prompt versions (*v1–3*; the exact wording of all prompt variations is included in Appendix A).

Comparing both across datasets (Table 1) and model sizes (Figure 4), we can see that including the PENMAN instruction is only useful in the zero-shot setup, where the model is less likely to generate AMR graphs in PENMAN notation when given no additional instructions about the format. Otherwise, more often than not, including the explicit instruction harms performance. As previous research has shown (Shi et al., 2023), this is likely because including additional instructions at the end of the prompt detracts attention from the target sentence introduced closer to the start.

Overall, prompt version 2, without an explicit

[7]https://github.com/mabu42/syntaxfix
[8]https://github.com/cfmrp/mtool

	0	0+	1g	1g+	1s	1s+	5g	5g+	5s	5s+
correct	5.01	13.63	9.25	9.18	5.58	4.09	5.22	6.64	0.77	2.12
fixed	31.73	58.23	88.97	88.48	93.92	95.54	93.56	92.43	98.93	97.66
pass	36.74	71.87	98.23	97.66	99.50	99.64	98.79	99.08	99.71	99.78
broken	63.18	28.05	1.69	2.26	0.49	0.35	1.20	0.91	0.28	0.21
blank	0.07	0.07	0.07	0.07	0.00	0.00	0.00	0.00	0.00	0.00
fail	63.25	28.12	1.76	2.33	0.49	0.35	1.20	0.91	0.28	0.21

Table 2: Post-processing rate of well-structured output from Llama-3.1, for the GrAPES dataset. The *pass* rate is made up of datapoints with graphs that are correct in the raw output, and those that are fixed through heuristic post-processing, while the *fail* rate is made up of datapoints with incorrigible output, and output where no PENMAN notation is detected.

PENMAN instruction, is the best-performing version of the prompt, over all model sizes and datasets. Given that this wording generates output with the best semantic accuracy, we will be focusing on output generated from this prompt for the rest of the analysis in this section (4.2 and beyond).

Dataset Differences As seen in Figure 4, performance on GrAPES data is better than both AMR and LPP, arguably because the datasets are qualitatively different. The GrAPES data is manually crafted, specifically to target and challenge particular aspects of graph generation. Meanwhile, the AMR dataset is a representation of natural language seen in daily use, and is more likely to contain sentence fragments, or clauses with multiple challenges in one sentence.

Accuracy on the LPP dataset is broadly between GrAPES and AMR – and the qualitative difference argument is likely also applicable here. As pointed out earlier, the LPP data comprises literary fiction. On the one hand, this makes it comparable to AMR data in terms of the presence of sentence fragments, or multi-clause sentences. On the other hand, as it is a carefully written and edited piece of writing, it is closer to GrAPES in terms of curated sentences, devoid of errors, run-on clauses, and other features of online discourse.

As the training data for Llama contains elements similar to all of the above (e.g. data sourced from CommonCrawl, Wikpedia, or Project Gutenberg), this perfomance pattern between different datasets is interesting to observe.

Notably, AMR in particular also demonstrates considerable variations in sentence size. Roughly 10% of the AMR dataset is made up of small one-node graphs, usually generated from one-token metadata, while another 10% comprises unusually large graphs (28–130 nodes), created from long sentences and mischunked paragraphs. See subsection 4.5 below for a more detailed discussion of performance depending on graph size.

Compared to the results reported by Li and Fowlie (2025), Llama-3.1-70B (Table 1) performs comparably with GPT-3.5 in the zero-shot setup (14 vs. 22%), and worse with few-shot prompting (41–50 for GPT-3.5, and 60 for GPT-4o, vs. 25–35% for Llama-3.1). As the study does not report the average score on GrAPES (focusing instead on category sets), it is difficult to directly compare Llama to GPT on this task. However, on GrAPES we see Llama-3.1 score up to 62% in the one-shot setup with generic demonstrations, and up to 54% on generic 5-shot. As the GPT study reports highs of 64% Smatch on specific GrAPES categories, it is interesting to note that the smaller model (Llama-3.1 70B) can compete with the far larger one (GPT-4o, approx. 200B) on the hand-crafted data.

However, none of the models come near current SotA in AMR parsing, where LeakDistill (Vasylenko et al., 2023) scores 84.6% Smatch on AMR 3.0, showing that a general-purpose LLM with no finetuning cannot yet match a dedicated AMR parser.

4.2. Syntactic Evaluation

Before evaluation, it is necessary to automatically correct the system's raw output (using the correction script described in Section 3.3), to maximise the number of well-formed graphs that can be accepted by the scorer. For the sake of simplicity, in this subsection we present and discuss the post-processing statistics on the best-performing prompt variant discussed in Section 4.1 above.

Table 2 shows the percentage of acceptable and unparseable datapoints after post-processing raw output. We break down the statistics into four categories of candidate graphs: well-formed in the original, with no modifications needed (labelled *correct* in the table); with errors in the original, but corrected in post-processing (*fixed*); with errors in the original, unable to be corrected through iterative runs of the post-processing algorithm (*broken*); and no viable output found (*blank*). *Correct* and *fixed* make up the *pass* score, while *broken* and *blank* make up *fail*. Datapoints with incorrigible or missing output are replaced with `( d / dummy )` graphs, which yield a score of zero in the final evaluation.

GrAPES category	**0**	**0+**	**1g**	**1g+**	**1s**	**1s+**	**5g**	**5g+**	**5s**	**5s+**	avg npg	total #g
long lists	.194	.354	.572	.553	.625	.625	.322	.350	.751	.748	29.7	96
deep recursion rc contrastive coref	.293	.212	.573	.578	.828	.799	.499	.528	.919	.894	14.0	70
deep recursion pronouns	.166	.488	.823	.823	.918	.923	.825	.810	.932	.938	13.2	100
deep recursion 3s	.161	.443	.810	.827	.891	.899	.764	.760	.947	.944	13.0	82
deep recursion basic	.187	.406	.801	.812	.939	.932	.813	.802	.951	.948	13.0	100
winograd	.245	.293	.354	.359	.402	.392	.331	.330	.497	.480	11.0	150
deep recursion rc	.235	.426	.698	.694	.826	.832	.653	.644	.909	.893	10.8	60
nested control	.248	.538	.684	.675	.721	.724	.685	.655	.768	.773	9.7	50
deep recursion rc contrastive coref sanity check	.381	.345	.546	.494	.863	.848	.506	.494	.890	.848	8.0	5
keep from	.002	.283	.471	.475	.783	.782	.462	.436	.889	.887	7.7	50
read by	.394	.493	.599	.601	.742	.731	.644	.561	.845	.851	6.6	75
centre embedding	.373	.525	.666	.659	.832	.821	.681	.667	.874	.871	6.4	30
give up in	.033	.296	.436	.436	.492	.498	.389	.394	.699	.683	6.2	75
bought for	.359	.393	.572	.570	.589	.588	.463	.452	.690	.674	5.6	50
deep recursion rc sanity check	.501	.505	.726	.734	.826	.826	.766	.674	.850	.845	5.5	4
see with	.198	.534	.616	.609	.638	.628	.523	.515	.745	.745	5.2	75
pp attachment	.237	.555	.619	.619	.635	.630	.527	.536	.727	.740	5.1	30
adjectives	.171	.204	.375	.353	.581	.604	.392	.372	.710	.702	4.5	40
deep recursion basic sanity check	.364	.679	.718	.738	.783	.799	.742	.768	.769	.769	4.0	6
deep recursion 3s sanity check	.487	.689	.771	.775	.858	.831	.785	.776	.926	.921	4.0	8
nested control sanity check	.495	.676	.695	.694	.768	.759	.698	.680	.879	.844	3.6	13
deep recursion pronouns sanity check	.640	.702	.862	.862	.867	.831	.822	.788	.892	.904	3.5	16
long lists sanity check	.326	.662	.669	.695	.748	.763	.660	.695	.832	.837	3.3	111
berts mouth	.368	.406	.523	.526	.525	.533	.410	.400	.609	.600	3.0	95
centre embedding sanity check	.417	.633	.666	.680	.666	.666	.674	.682	.666	.663	2.4	13
g adjectives sanity check	.230	.328	.604	.551	.673	.658	.628	.538	.666	.651	2.0	11

Table 3: Breakdown of GrAPES F_1-scores (Llama-70B, prompt v2), by grapes category. The categories are sorted, in descending order, by average node count per category graph. The rightmost column indicates the total number of datapoints per category, for reference.

In the zero-shot case, the rate of acceptable and correctable output improves with the introduction of the *"use PENMAN notation"* instruction; considerably for GrAPES data, and only moderately for unedited correct AMR datapoints. The use of even a single generic demonstration dramatically increases the rate of correctable output, and with it the *pass* rate as well, demonstrating the model's in-context learning capability and showing that few-shot prompting surpasses carefully worded textual instructions. However, including any number of demonstrations also progressively decreases the rate of correctly formed output compared to the zero-shot setup. The errors introduced in these cases largely fall into the categories of illegal characters and superfluous closing parentheses. The latter might be explained away by the system overfitting to the closing parentheses of example graphs, but no clear explanation exists for the illegal character artifacts in the former.

4.3. In-context Learning

In Figure 4, we focus on the performance of both models on GrAPES, and particularly on the effect of demonstration choice in k-shot prompting. Using specific demonstrations markedly improves performance for all model sizes and all prompt versions. In the 1-shot setup, improvement ranges from 8 to 19%, and from 21 to 49% in 5-shot. It is important to point out that using a larger number of generic demonstrations reduces performance compared to a single generic demonstration, as in the case of 36% for *1g* vs. 18% for *5g*, indicating that the model may be overfitting to example data.

Specific demonstrations, particularly in the 5-shot setup, may give an indication of the upper bounds of model performance, but are not practically applicable in a use-case setting, as having this amount of correctly parsed and highly specific training data defeats the purpose of using Llama as a parser. However, this setup demonstrates the models' prompt sensitivity, and suggests an upper bound for in-context learning in the case of AMR parsing, or, more generally, generating structured output in underrepresented formal language.

4.4. GrAPES Categories

The fine-grained GrAPES categories allow for a more detailed results analysis. In this section we will look at differences in model performance considering these categories of semantic challenges.

Table 3 shows the GrAPES results, broken down by predefined categories. The categories are sorted in descending order by average graph size (the average number of nodes per datapoint, in each category), and, for reference, we include the number of datapoints in each category (i.e. category size) in the rightmost column. We discuss the effect of graph size on model output in Section 4.5.

Overall, the best-performing categories are the recursion-with-pronouns challenges, scoring up to 86% in the generic-demonstrations setup, and up to 93% with specific demonstrations. The highest-scoring category is *deep recursion basic*, in which the specific-demonstrations setup scores 93% in the 1-shot, and 95% in the 5-shot case respectively.

An example short sentence in this category is: *"I said that the boys won."* A mid-length sentence is: *"The boys believed that the soldier heard that the women mentioned that the lawyer mentioned that the kids arrived."*, with other sentences getting longer still. An illustrated example for this category can be found in Appendix 9 (Figure 7).

This challenge tests the system on consistency in capturing argument structure in subordinate clauses (Groschwitz et al., 2023). It seems to be

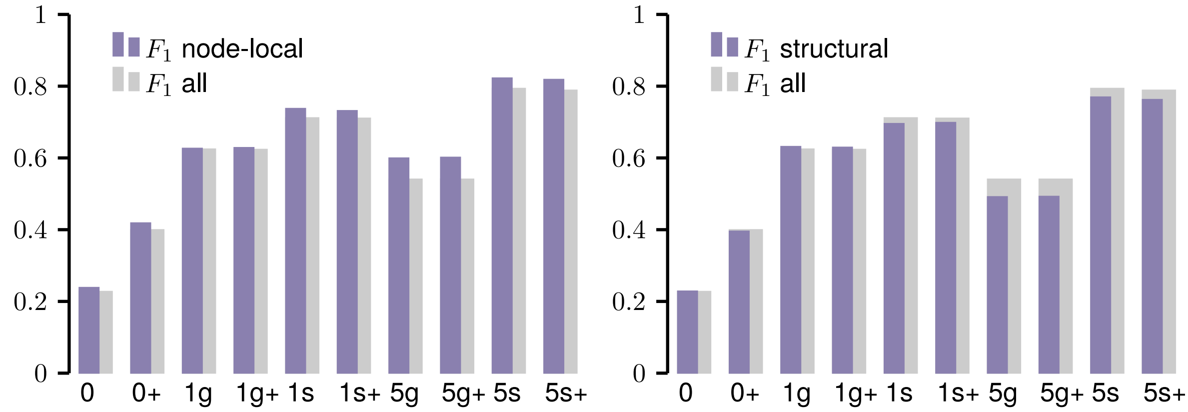


Figure 5: F_1-scores for node-local (left) and structural (right) graph properties on the GrAPES dataset (Llama-70B, prompt v2). The property-specific scores are shown in the foreground, with the overall F_1 score included in the background in light grey, for reference.

fairly straightforward for the system to learn the repetitive recursive pattern from five samples of the same structure, and replicate it.

A sentence from *pronouns*, which scores best for generic demonstrations, is: *"You thought that the girls believed that we said that we sneezed"*, which is semantically close to the examples above. Presumably, correctly labelling pronouns is an easy task for the system to learn, and any example of coreference in the demonstrations is enough guidance to generate accurate graphs for recursion.

On the other hand, among the worst-performing categories in the generic-demonstrations setup are phrasal verbs (*bought for*, 57; *keep from*, 47; *give up in*, 43%), *Bert's mouth* (a word disambiguation task first introduced by Karidi et al. (2021); 52%), *adjectives* (35–39%), and, overwhelmingly, in the case of both generic and specific demonstrations, the *Winograd* schema (Levesque et al., 2012) (33–35%, and 39–47%, respectively). Visual examples of datapoints from these categories may be found in Appendix 9 (Figures 8 and 9).

An example sentence from *adjectives* is *"A strange beautiful round plate."* – notably, these sentences lack predicates, unlike any random selection of samples drawn from the datasets, which may lead to errors. The prepositional verbs are another form of *PP attachment* challenge, testing for accurate edge attachment in the case of ambiguities between the NP and VP. The correct attachment varies by context, so the task aims to test whether the system consistently prefers one over the other, or is sensitive to the context of the particular sentence (Groschwitz et al., 2023).

Also visible from the heatmap is the largely consistent pattern where using one generic demonstration improves performance, but using five demonstrations causes the model to perform worse than when given only one. The same is, predictably, not true in the case of specific demonstrations.

This drop when going from one to many demonstrations is most visible in the case of *long lists* (25-point drop), *PP attachment* (9-point drop), phrasal verbs (up to 12 points), and *Bert's mouth* (12 points). Some deep recursion subtasks also suffer this drop in performance, ranging from 4 to 8 points.

4.5. Fine-grained Analysis

Graph features Previous studies on parser performance (Buljan et al., 2022) have analysed output quality by graph factors. We follow this methodology to examine Llama's performance on node decoration (correctly generating node labels and node properties), and graph structure (top node assignment, edge direction and edge labels) of AMR graphs. We use the aforementioned `mtool` scorer to extract F_1-scores per set of features.

Our findings are presented in Figure 5. Overall, node decoration is slightly easier for the model to generate correctly, while graph structure is more difficult. The phenomenon of more generic demonstrations being damaging for performance compared to only one is not present in node decoration, but has a much more dramatic effect when it comes to generating correct graph structure. Presumably, generating node labels from the surface string is an easier task than generating graph structure, so the risk of overfitting or learning conflicting information is higher in the case of graph structure.

Curiously, performance on node decoration is overall higher than the average F_1-score, and for graph structure it is lower, even in the case of specific demonstrations.

Graph complexity Again following the methodology of Buljan et al. (2022), we look at model performance depending on graph complexity, i.e. node count for gold set datapoints. Figure 6 illustrates the variation according to graph-size decile bins.

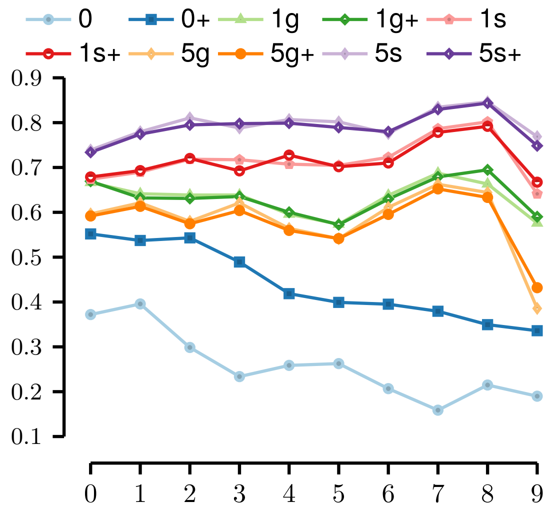


Figure 6: F_1-score on the GrAPES dataset, by graph complexity decile bin.

Previous work on AMR data found that, predictably, smaller graphs are generally easier to generate correctly than larger ones, with the exception of very small graphs that equate to sentence fragments or metadata. However, we see conflicting data on the GrAPES dataset – the expected trends are seen in the case of zero-shot prompts, but with the introduction of demonstrations, regardless of type and number, graphs on the higher end seem to be easier to generate for this particular dataset.

This is interesting to compare with the findings in Section 4.4 and Table 3. The largest graphs fall into the *long lists* category, which is one of the worse-scored challenges in the GrAPES dataset, but the next few highest decile bins (with the exception of *Winograd*) are made up of graphs in the *deep recursion* subcategories, which Llama overall performs well in. For comparison, performance through the lens of graph size for all three datasets is included in Appendix D.

5. Conclusion

We evaluated a small and medium-sized Llama model on generating AMR graphs, as a stepping stone towards quantifying LLMs' grasp of sentence meaning. Considering the prevalence of syntactic errors in generated formal language, we introduced a robust syntax correction script to maximise usable output. In our experiments, we evaluated the effects of prompt variation, with particular focus on in-context learning with demonstrations of varying count and specificity. Finally, we conducted an analysis of the data considering different perspectives – categories of semantic challenges, contrasting properties of AMR graphs, and graph complexity.

Our findings align with observations from previous studies: LLMs perform poorly on generating AMR graphs, and the syntax of raw output is imperfect (Ettinger et al., 2023; Li and Fowlie, 2025). These technical limitations must be overcome in order to apply MRP as a tool for more in-depth analysis of LLMs' ability to capture sentence-level semantics. As an AMR parser, general-purpose LLMs' performance is still well below SotA. However, on the GrAPES dataset, the medium-sized open-source Llama-3.1-70B performed competitively to its larger counterparts from other studies, which is notable from an accessibility standpoint. Still, without fine-tuning, the model has demonstrable limitations to performance as an AMR parser.

The effect of demonstrations on performance is noteworthy, and there is more room for exploration in this space. Even though the use-case of the specific demonstrations described here is more informative from a theoretical than a practical standpoint, it still indicates that demonstrations selected to be more informative may positively affect model accuracy. For example, the model may benefit from a small set of examples individually demonstrating semantic phenomena, such as those categorised in the GrAPES dataset, which is a practical middle ground between completely random demonstrations, and unrealistically specific ones.

6. Acknowledgements

The work described here was funded by the University of Oslo, and carried out on resources provided by Sigma2 — the Norwegian national research infrastructure provider for high-performance computing and large-scale data storage.

We thank the reviewers for their insightful comments and suggestions for improvement.

We would also like to thank Étienne Simon for his assistance with LaTeX wizardry; the first author takes full responsibility for any formatting decisions that are not up to the highest standards of the craft.

7. Bibliographical References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Xuefeng Bai, Jialong Wu, Yulong Chen, Zhongqing Wang, Kehai Chen, Min Zhang, and Yue Zhang. 2025. Constituency parsing using llms. *IEEE Transactions on Audio, Speech and Language Processing*.

Simone Balloccu, Patrícia Schmidtová, Mateusz Lango, and Ondrej Dusek. 2024. Leak, cheat, repeat: Data contamination and evaluation malpractices in closed-source LLMs. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 67–93, St. Julian's, Malta. 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.

Emily M Bender, Dan Flickinger, Stephan Oepen, Woodley Packard, and Ann Copestake. 2015. Layers of interpretation: On grammar and compositionality. In *Proceedings of the 11th international conference on Computational Semantics*, pages 239–249.

Rexhina Blloshmi, Rocco Tripodi, Roberto Navigli, et al. 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. The 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, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Maja Buljan, Joakim Nivre, Stephan Oepen, and Lilja Øvrelid. 2022. A tale of four parsers: methodological reflections on diagnostic evaluation and in-depth error analysis for meaning representation parsing. *Language Resources and Evaluation*, 56(4):1075–1102.

Maja Buljan, Stephan Oepen, and Lilja Øvrelid. 2024. Argument sharing in meaning representation parsing. In *Proceedings of the Fifth International Workshop on Designing Meaning Representations@ LREC-COLING 2024*, pages 77–87.

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.

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.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Marco Damonte and Shay B Cohen. 2018. Cross-lingual abstract meaning representation parsing. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1146–1155.

Timothy Dozat and Christopher D. Manning. 2018. Simpler but more accurate semantic dependency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 484–490, Melbourne, Australia. Association for Computational Linguistics.

A Ettinger, J Hwang, V Pyatkin, et al. 2023. You are an expert linguistic annotator": limits of LLMs as analyzers of abstract meaning representation. findings of the association for computational linguistics.

Michael Wayne Goodman. 2020. Penman: An open-source library and tool for amr graphs. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 312–319.

Jonas Groschwitz, Shay Cohen, Lucia Donatelli, Meaghan Fowlie, et al. 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. Association for Computational Linguistics.

Shu Han Ho. 2025. Evaluation of finetuned llms in amr parsing. *arXiv preprint arXiv:2508.05028*.

Fuad Issa, Marco Damonte, Shay Cohen, Xiaohui Yan, and Yi Chang. 2018. Abstract meaning representation for paraphrase detection. In *16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 442–452. Association for Computational Linguistics.

Taelin Karidi, Yichu Zhou, Nathan Schneider, Omri Abend, and Vivek Srikumar. 2021. Putting words in bert's mouth: Navigating contextualized vector spaces with pseudowords. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10300–10313.

Paul R Kingsbury and Martha Palmer. 2002. From treebank to propbank. In *LREC*, pages 1989–1993.

Aobo Kong, Shiwan Zhao, Hao Chen, Qicheng Li, Yong Qin, Ruiqi Sun, Xin Zhou, Enzhi Wang, and Xiaohang Dong. 2024. Better zero-shot reasoning with role-play prompting. In *NAACL-HLT*.

Marco Kuhlmann and Stephan Oepen. 2016. Towards a catalogue of linguistic graph banks. *Computational Linguistics*, 42(4):819–827.

Hector J Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning*, pages 552–561.

Yanming Li and Meaghan Fowlie. 2025. GPT makes a poor AMR parser. *Journal for Language Technology and Computational Linguistics*, 38(2):43–76.

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.

Behrooz Mansouri. 2025. Survey of abstract meaning representation: Then, now, future. *arXiv preprint arXiv:2505.03229*.

Christian MIM Matthiessen and John A Bateman. 1991. Text generation and systemic-functional linguistics: experiences from english and japanese. *(No Title)*.

Meta. 2024. Introducing meta llama 3: The most capable openly available llm to date. https://ai.meta.com/blog/meta-llama-3/. Accessed: 2025-10-23.

Stephan Oepen, Omri Abend, Lasha Abzianidze, Johan Bos, Jan Hajic, Daniel Hershcovich, Bin Li, Tim O'Gorman, Nianwen Xue, and Daniel Zeman. 2020. MRP 2020: The second shared task on cross-framework and cross-lingual meaning representation parsing. In *Proceedings of the CoNLL 2020 Shared Task: Cross-Framework Meaning Representation Parsing*, pages 1–22, Online. Association for Computational Linguistics.

Stephan Oepen, Omri Abend, Jan Hajic, Daniel Hershcovich, Marco Kuhlmann, Tim O'Gorman, Nianwen Xue, Jayeol Chun, Milan Straka, and Zdenka Uresova. 2019. Mrp 2019: Cross-framework meaning representation parsing. In *Proceedings of the Shared Task on Cross-Framework Meaning Representation Parsing at the 2019 Conference on Natural Language Learning*, pages 1–27.

Christos Papakostas, Christos Troussas, Akrivi Krouska, and Cleo Sgouropoulou. 2025. A hybrid neuro-symbolic pipeline for coreference resolution and amr-based semantic parsing. *Information*, 16(7):529.

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*, pages 1–7.

Zacchary Sadeddine, Juri Opitz, and Fabian Suchanek. 2024. A survey of meaning representations-from theory to practical utility. In *2024 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 2877–2892. Association for Computational Linguistics.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H Chi, Nathanael Schärli, and Denny Zhou. 2023. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning*, pages 31210–31227. PMLR.

Ida Szubert, Marco Damonte, Shay B Cohen, and Mark Steedman. 2020. The role of reentrancies in abstract meaning representation parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2198–2207.

Nasim Tohidi and Chitra Dadkhah. 2022. A short review of abstract meaning representation applications. *Modeling and Simulation in Electrical and Electronics Engineering*, 2(3):1–9.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

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.

Frank F Xu, Uri Alon, Graham Neubig, and Vincent Josua Hellendoorn. 2022. A systematic evaluation of large language models of code. In *Proceedings of the 6th ACM SIGPLAN international symposium on machine programming*, pages 1–10.

Haoran Xu, Amr Sharaf, Yunmo Chen, Weiting Tan, Lingfeng Shen, Benjamin Van Durme, Kenton Murray, and Young Jin Kim. 2024. Contrastive preference optimization: pushing the boundaries of llm performance in machine translation. In *Proceedings of the 41st International Conference on Machine Learning*, pages 55204–55224.

8. Language Resource References

Kevin Knight, Bianca Badarau, Laura Baranescu, Claire Bonial, Madalina Bardocz, Kira Griffitt, Ulf Hermjakob, Daniel Marcu, Martha Palmer, Tim O'Gorman, and Nathan Schneider. 2020. Abstract meaning representation (amr) annotation release 3.0. *Philadelphia: Linguistic Data Consortium*, (LDC2020T02).

9. Appendices

In these appendices, we have included additional information, such as prompt variations, and complete result tables, from experiments with worse-performing prompts and model setups. This material is not crucial to understanding the work, nor is it necessary for a complete and informative report of the experiments and findings. However, it comprises the complete output of the work, and therefore we include this data as additional points of interest.

Appendix A. Prompt Variations

We experiment with three prompt variations (denoted in the paper as *v1* through *v3*). Each of the prompts has an optional explicit instruction to use PENMAN notation in the output (indicated in the paper with a +). Finally, each of the prompts can be expanded with example sentences, so we compare a zero-shot prompt with one-shot and five-shot (denoted in the paper as *0*, *1*, *5*).

Below we will show all versions of the zero-shot prompts, and follow with a compact print of prompts *v2* and *v3*.

v1 **0**

> You are an expert linguistic annotator. Please provide an AMR parse for the following utterance:
> `[target sentence]`

v1 **0+**

> You are an expert linguistic annotator. Please provide an AMR parse in PENMAN notation for the following utterance:
> `[target sentence]`

v1 **1[+]**

> You are an expert linguistic annotator. I will give you an example annotation in the form of Abstract Meaning Representation for the sentence:
> *The boy wants the girl to believe him.*
> ```
> (w / want-01
>   :ARG0 (b / boy)
>   :ARG1 (b2 / believe-01
>      :ARG0 (g / girl)
>      :ARG1 b))
> ```
> Please provide an AMR parse [in PENMAN notation] for the following utterance:
> `[target sentence]`

v1 **5[+]**

> You are an expert linguistic annotator.
> I will give you some example annotations in the form of Abstract Meaning Representation.
> `[example sentence [1-5]]`
> `[example graph [1-5]]`
> Please provide an AMR parse [in PENMAN notation] for the following utterance:
> `[target sentence]`

In a similar vein, prompts *v2* and *v3* state:

v2

> You are an expert linguistic annotator.
> [I will give you an example annotation ...]
> Please generate an Abstract Meaning Representation graph [in PENMAN notation] for the following sentence:
> `[target sentence]`
> Please begin your annotation with `<amr>`, and end it with `</amr>`.

v3

> You are an expert linguistic annotator.
> [I will give you an example annotation ...]
> Please generate an Abstract Meaning Representation graph [in PENMAN notation] for the following sentence:
> `[target sentence]`
> Don't include any comments, explanations, alternative versions, or any other output except the AMR graph for the target sentence.

As discussed in the main body of the paper, and shown in Appendix D, prompt *v2* is the best-performing phrasing. Even though the focus of this work was not prompt engineering, we experimented with this small pool of variations to check and account for possible brittleness within the model.

Our conclusion was that unrestricted output, as in *v1*, produces comments, interjections, and multiple versions that make the output graph difficult to parse, so limiting the output to clearly denote the graph, and include nothing else within – as in *v2* – improves performance. However, overly wordy instructions, as in *v3*, detract attention from the examples and target sentence introduced closer to the start of the propt, as mentioned in Section 4.1, and shown by previous work (Shi et al., 2023).

Appendix B. Graph Examples

Although qualitative analysis is not the focus of this paper, here we have included a selection of system output graphs as points of interest.

As discussed in Section 4.4, there are several prominent categories in the GrAPES dataset that demonstrate great improvement between different prompt setups (with regards to demonstrations), or are challenging overall, across prompt variations.

Figure 7 presents output graphs for the sentence "The astronaut knew that we won.", from the *deep recursion* category of GrAPES. This is one of the best-performing challenge categories overall, showing significant improvement between the quality of output based on a zero-shot prompt, versus using even a single demonstration. In Figure 7, the difference between the gold graph and the two versions of system output is visible, highlighting false negatives in red, and false positives in blue. Both the structure and graph features are visibly improved with the addition of a single demonstration in the prompt.

Figure 8 presents output graphs for the phrase "A fantastic new dark plate", from the *adjectives* subset of GrAPES. This is one of the most poorly performing challenge categories, but improvement is visible between using a single generic demonstration, and multiple subset-specific demonstrations. In Figure 8, it is again visible that the graph structure improves with more (specific) demonstrations, though many labels are still incorrect.

Finally, Figure 9 presents output graphs for the sentence "The lawyer asked the witness a question, but he was reluctant to repeat it.", from the *winograd* subset of GrAPES. This is the worst-performing challenge category of all, with little improvement between using a single generic demonstration, and multiple subset-specific demonstrations. In Figure 9, it is visible that many structural features and labels are missing in the output produced from a one-shot prompt (top). However, in the case of five subset-specific demonstrations (bottom), there are less missing structural features, but more false positives in the form of arguments and node-labels, likely as a result of overfitting to the demonstrations.

Appendix C. Full Evaluation Results

Below we present the full evaluation results.

In the case of the GrAPES dataset, we include full results using the 8B (Table 4) and 70B (Table 5) versions of the model. Note that the other two datasets (AMR, Table 6; and LPP, Table 7) are evaluated only on output from the 70B model: given the difference between results for the two model versions, we did not wish to waste computing resources on experiments that we know with certainty will perform badly.

Additionally, the GrAPES dataset is scored as the average of five experimental runs. Tables 4 and 5 include the standard deviation between these. Given the negligible variations, evaluations for the other two datasets – AMR (Table 6) and LPP (Table 7) – are evaluated on a single run of the 70B model, again out of consideration for reasonable use of computing resources.

Finally, note that the AMR and LPP datasets (Tables 6 and 7) do not include results for experiments with specific demonstrations (*1s(+)*, *5s(+)*), as this experimental setup is particular to the GrAPES dataset. See Section 4.3 for details.

Appendix D. Full Analysis Results

Below we include full results of the various analyses described in Section 4.2 and onwards.

Syntactic Post-processing

As discussed in Section 3.3, raw system output is syntactically correct and viable for scoring only for 1-14% of datapoints. We therefore introduce a post-processing step, similar to the approach of Li and Fowlie (2025), to superficially correct the syntax before evaluation.

The steps include correcting mismatched parentheses, fragmented or concatenated node expressions, missing or duplicate variable names, and illegal characters that need removing or replacing.

The syntax-fixing script is a simple heuristic algorithm, using regular expressions and list comprehension to capture the most commonly observed syntactic errors.

Table 8 shows the complete statistics of the syntactic viability of raw system output, across all three datasets. (The experimental setup remains the same as described in the main body of the paper: Llama-70B, prompt version *v2*.)

Graph Features

An extension of Figure 5 in the main body of the paper, Table 9 shows the quality of system output by node-local and structural properties of AMR graphs, across all three datasets. (The experimental setup remains the same as described in the main body of the paper: Llama-70B, prompt version *v2*.)

Graph Complexity

An extension of Figure 6 in the main body of the paper, Table 10 shows the quality of system output by graph size, expressed in node count, across all three datasets. (The experimental setup remains the same as described in the main body of the paper: Llama-70B, prompt version *v2*.)

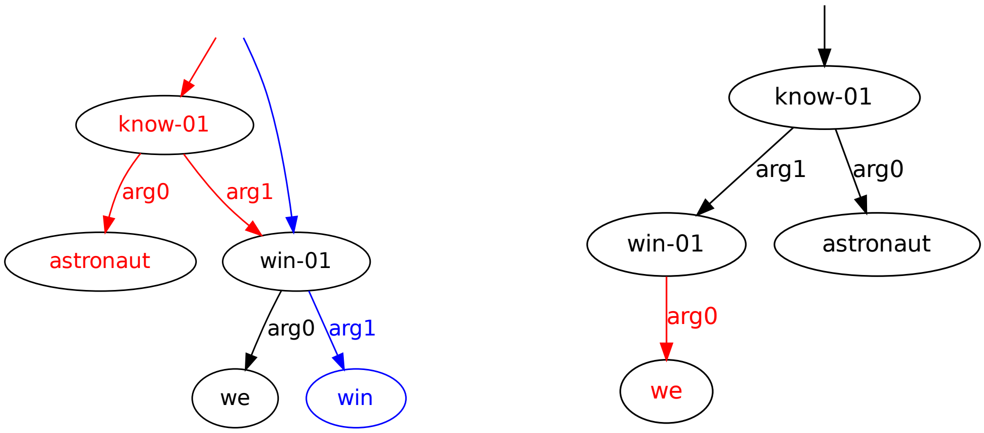


Figure 7: AMR graphs for the sentence “The astronaut knew that we won.”, from the *deep recursion* subset of GrAPES. The graphs show the intersection of the gold and system output graphs, with false negatives highlighted in red, and false positives in blue. Left: system output in the **0+** setup (no demonstrations, explicit PENMAN instruction; F_1=0.428). Right: system output in the **1g+** setup (one generic demonstration, explicit PENMAN instruction; F_1=0.857).

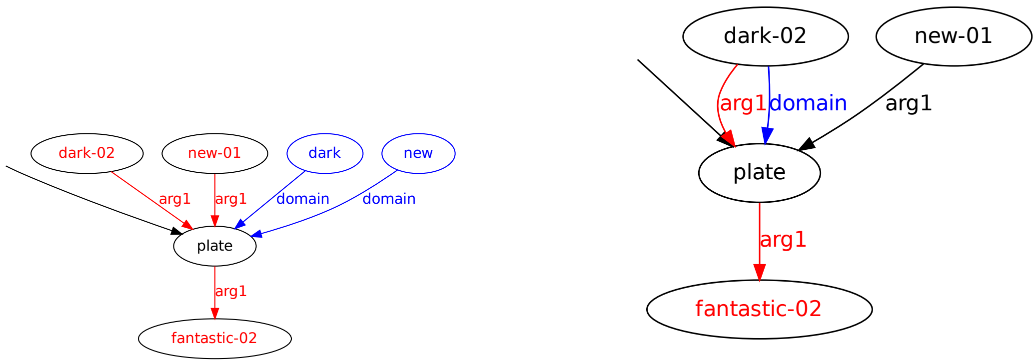
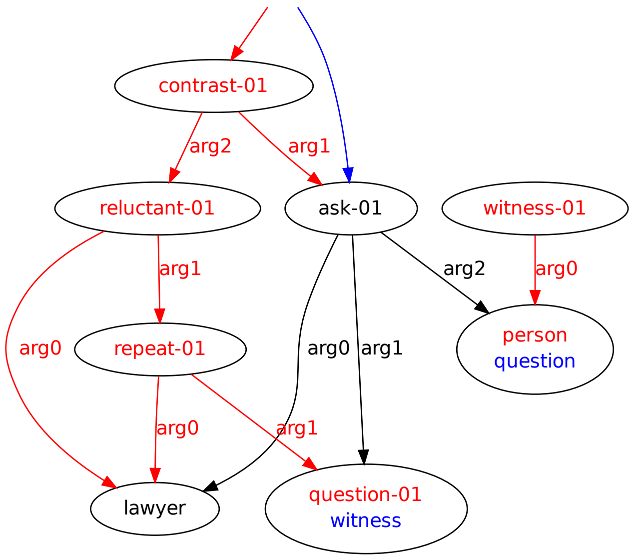


Figure 8: AMR graphs for the phrase “A fantastic new dark plate”, from the *adjectives* subset of GrAPES. The graphs show the intersection of the gold and system output graphs, with false negatives highlighted in red, and false positives in blue. Left: system output in the **1g** setup (one generic demonstration, no PENMAN instruction; F_1=0.285). Right: system output in the **5s** setup (five subset-specific demonstrations, no PENMAN instruction; F_1=0.714).



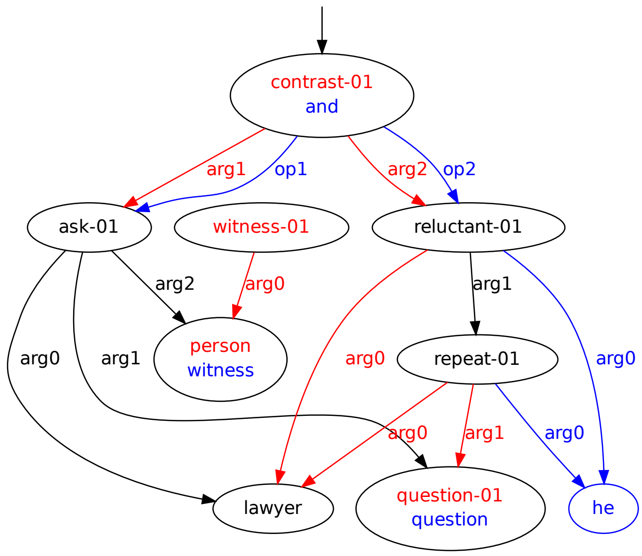


Figure 9: AMR graphs for the sentence “The lawyer asked the witness a question, but he was reluctant to repeat it.”, from the *winograd* subset of GrAPES. The graphs show the intersection of the gold and system output graphs, with false negatives highlighted in red, and false positives in blue. Top: system output in the **1g** setup (one generic demonstration, no PENMAN instruction; F_1=0.370). Bottom: system output in the **5s** setup (five subset-specific demonstrations, no PENMAN instruction; F_1=0.444).

[8B]	**0**	**0+**	**1g**	**1g+**	**1s**	**1s+**	**5g**	**5g+**	**5s**	**5s+**	
v1 F_1	.050	.023	.366	.154	.484	.337	.197	.181	.483	.507	
stddev	.002	.003	.005	.006	.003	.010	.005	.008	.005	.007	.0054
F_1, non-zero	.133	.118	.386	.356	.506	.514	.340	.364	.633	.648	
v2 F_1	.218	.218	.371	.362	.551	.540	.374	.182	.680	.670	
stddev	.268	.268	.003	.003	.009	.013	.007	.004	.002	.003	.0058
F_1, non-zero	.467	.460	.380	.375	.560	.562	.390	.446	.710	.732	
v3 F_1	.005	.009	.380	.354	.510	.487	.388	.258	.610	.588	
stddev	.001	.001	.003	.007	.005	.009	.006	.002	.004	.008	.0046
F_1, non-zero	.129	.127	.389	.388	.527	.534	.398	.435	.612	.633	

Table 4: Llama-8B results on the GrAPES dataset, average of five runs.

[70B]	**0**	**0+**	**1g**	**1g+**	**1s**	**1s+**	**5g**	**5g+**	**5s**	**5s+**	
v1 F_1	.380	.410	.502	.513	.630	.587	.451	.445	.695	.684	
stddev	.053	.007	.004	.005	.004	.008	.002	.003	.004	.002	.0091
F_1, non-zero	.460	.475	.545	.557	.644	.606	.460	.460	.703	.690	
v2 F_1	.228	.400	.625	.624	.712	.711	.541	.541	.794	.789	
stddev	.006	.012	.005	.004	.006	.006	.003	.004	.002	.002	.0048
F_1, non-zero	.529	.535	.586	.628	.681	.683	.516	.526	.777	.778	
v3 F_1	.458	.404	.547	.543	.657	.653	.500	.486	.711	.706	
stddev	.007	.003	.006	.005	.004	.003	.002	.003	.002	.000	.0036
F_1, non-zero	.497	.477	.575	.571	.663	.659	.505	.493	.712	.707	

Table 5: Llama-70B results on the GrAPES dataset, average of five runs.

[70B]	**0**	**0+**	**1g**	**1g+**	**5g**	**5g+**
v1 F_1	.179	.198	.241	.237	.267	.256
F_1, non-zero	.227	.227	.254	.255	.278	.270
v2 F_1	.107	.218	.313	.312	.347	.335
F_1, non-zero	.317	.298	.321	.322	.358	.342
v3 F_1	.219	.181	.249	.247	.283	.277
F_1, non-zero	.246	.235	.262	.263	.292	.287

Table 6: Llama-70B results on the AMR dataset (single run).

[70B]	**0**	**0+**	**1g**	**1g+**	**5g**	**5g+**
v1 F_1	.312	.329	.376	.366	.419	.391
F_1, non-zero	.362	.371	.396	.389	.435	.409
v2 F_1	.283	.354	.407	.404	.470	.442
F_1, non-zero	.403	.393	.419	.416	.477	.452
v3 F_1	.339	.298	.386	.384	.449	.424
F_1, non-zero	.380	.376	.400	.400	.457	.434

Table 7: Llama-70B results on the LPP dataset (single run).

		0	0+	1g	1g+	1s	1s+	5g	5g+	5s	5s+
GrAPES	*correct*	5.01	13.63	9.25	9.18	5.58	4.09	5.22	6.64	0.77	2.12
	fixed	31.73	58.23	88.97	88.48	93.92	95.54	93.56	92.43	98.93	97.66
	pass	36.74	71.87	98.23	97.66	99.50	99.64	98.79	99.08	99.71	99.78
	broken	63.18	28.05	1.69	2.26	0.49	0.35	1.20	0.91	0.28	0.21
	blank	0.07	0.07	0.07	0.07	0.00	0.00	0.00	0.00	0.00	0.00
	fail	63.25	28.12	1.76	2.33	0.49	0.35	1.20	0.91	0.28	0.21
AMR	*correct*	8.53	25.81	20.70	21.12	-	-	10.43	14.43	-	-
	fixed	20.60	41.35	73.91	73.23	-	-	85.56	82.71	-	-
	pass	29.13	67.17	94.62	94.36	-	-	95.99	97.15	-	-
	broken	70.70	32.45	4.47	4.21	-	-	2.95	2.47	-	-
	blank	0.15	0.36	0.89	1.42	-	-	1.05	0.36	-	-
	fail	70.86	32.82	5.37	5.63	-	-	4.00	2.84	-	-
LPP	*correct*	20.99	31.17	18.75	17.34	-	-	7.10	12.61	-	-
	fixed	45.07	62.67	80.85	81.49	-	-	92.76	87.32	-	-
	pass	66.06	93.85	99.61	98.84	-	-	99.87	99.93	-	-
	broken	33.93	6.14	0.38	1.15	-	-	0.12	0.06	-	-
	blank	0.00	0.00	0.00	0.00	-	-	0.00	0.00	-	-
	fail	33.93	6.14	0.38	1.15	-	-	0.12	0.06	-	-

Table 8: Pre- and post-processing rates of well-structured output from Llama-3.1. Top: GrAPES dataset; middle: AMR 3.0 test set; bottom: LPP dataset. The *pass* rate is made up of datapoints with graphs that are correct in the raw output, and those that are fixed through heuristic post-processing, while the *fail* rate is made up of datapoints with incorrigible output, and output where no PENMAN notation is detected. Note that the AMR and LPP dataset rows do not include results for specific demonstrations (*1s(+)*, *5s(+)*), as that evaluation setup is particular to the GrAPES dataset.

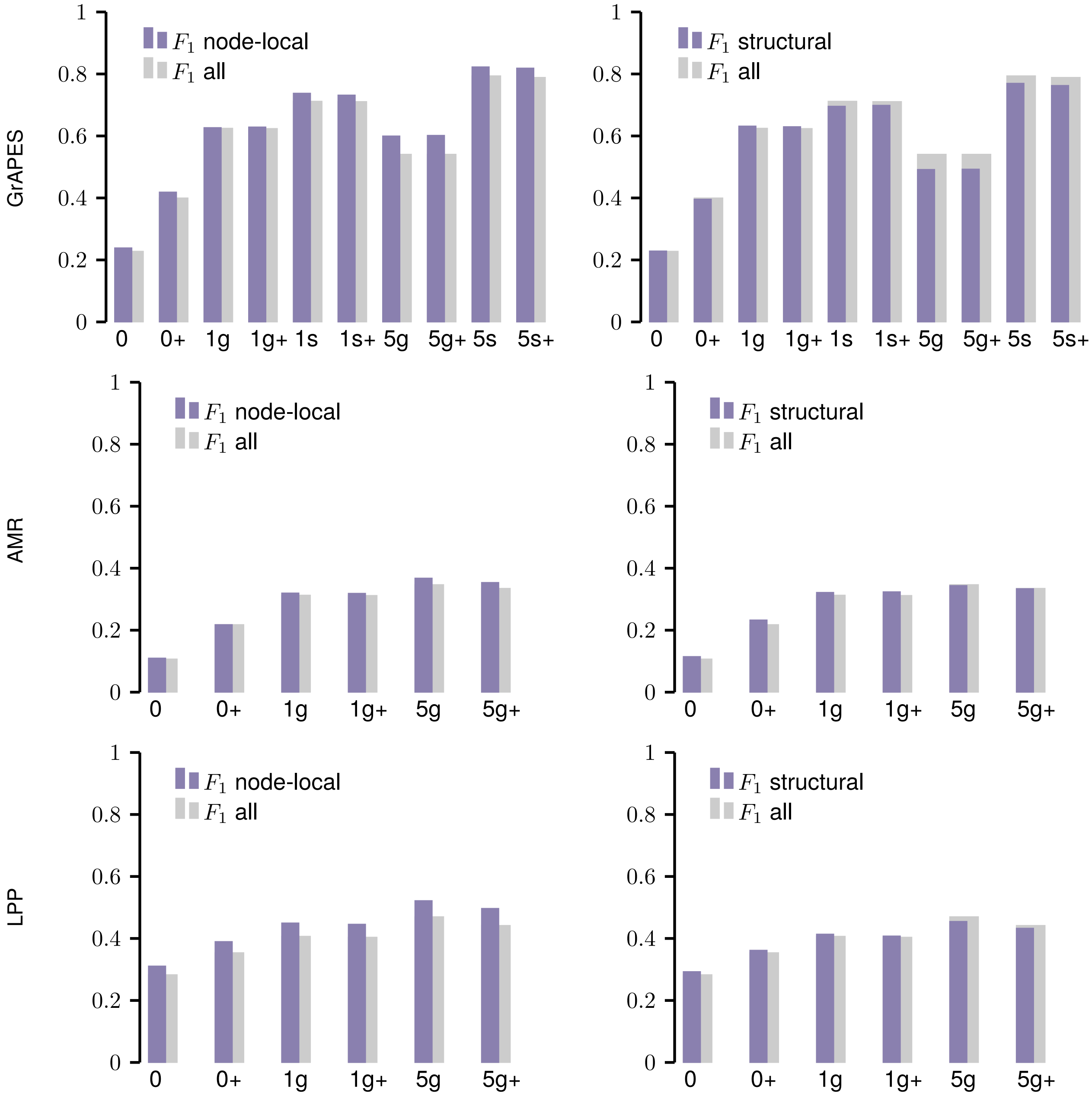


Table 9: F_1-scores for node-local (left) and structural (right) graph properties on the GrAPES, AMR, and LPP datasets (Llama-70B, prompt v2). The property-specific scores are shown in the foreground, with the overall F_1 score included in the background in light grey, for reference. Note that the AMR and LPP datasets have no results for the experiments with specific demonstrations (*1s(+)*, *(s(+)*), as that evaluation setup is particular to the GrAPES dataset.

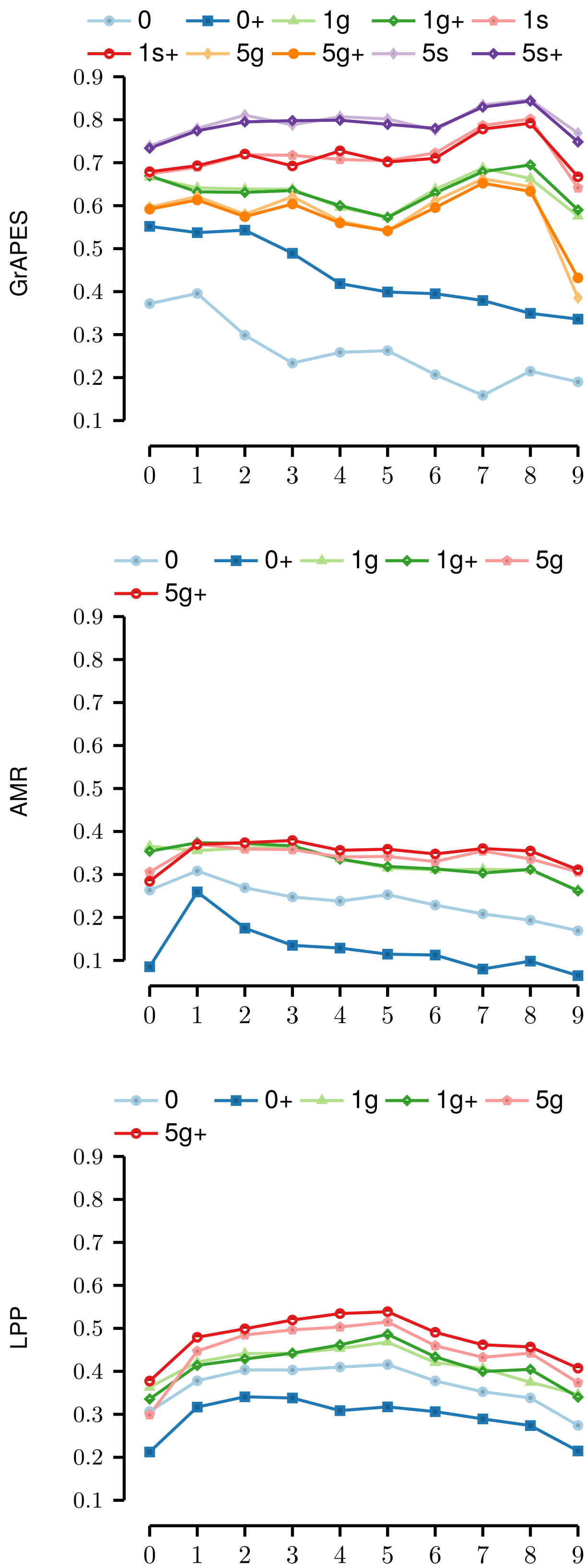


Table 10: F_1-score on the GrAPES, AMR, and LPP datasets (Llama-70B, prompt v2), by decile bins of graph complexity (expressed as node count). Note that the AMR and LPP datasets have no results for the experiments with specific demonstrations (*1s(+)*, *(s(+))*), as that evaluation setup is particular to the GrAPES dataset.