DOI: 10.14529/jsfi250308

One-Shot Prompting for Russian Dependency Parsing

Elena D. Shamaeva[1], ***Mikhail M. Tikhomirov***[1], ***Natalia V. Loukachevitch***[1]

This study investigates the application of Large Language Models (LLMs) for dependency parsing of Russian sentences. We evaluated several models (including Qwen, RuAdapt, YandexGPT, T-pro, T-lite, and Llama) in a one-shot mode across multiple Russian treebanks: SynTagRus, GSD, PUD, Poetry, and Taiga. Among the models tested, Llama70 achieved the highest scores in both UAS and LAS. Furthermore, we observed a general trend where larger models tended to perform better. Our analysis also revealed that parsing quality for Qwen4 and RuAdapt4 on the Taiga treebank was notably sensitive to prompt design. However, the results from all LLMs remained lower than those obtained from classical neural parsers. A key challenge encountered by many models was a difference between generated token sets and gold token sets, which was observed in a considerable portion of each treebank. Additionally, the T-pro and T-lite models produced a significant number of extra lines. The implementation for this study is publicly available at `https://github.com/Derinhelm/llm_parsing/tree/main`.

Keywords: LLM, parsing, dependency tree, one-shot, prompt-tuning, Russian language.

Introduction

In the era of Large Language Models (LLMs), syntax parsing remains an important task in Natural Language Processing (NLP), because syntax parsers allow for obtaining more interpretable results. These parsers are used as an auxiliary tool in tasks such as assessing text complexity [11], paraphrasing [10], named entity recognition [1, 9, 12, 18], and plagiarism detection [15]. Additionally, parsers are used for linguistic text analysis [3].

For syntax parsing, both classical neural networks and LLMs can be applied. While classical neural parsers have achieved a high level of performance[2], the application of LLMs to this task represents an emerging field of research. In this field, the widely adopted technique is prompt-based tuning of LLMs [5]. An important aspect of this prompt-based approach is the optimal design of prompts, particularly the selection of prompt examples. This research direction, however, remains notably underexplored for syntax parsing of the Russian language.

The article describes the experiment on applying LLMs in one-shot mode for syntax parsing of Russian sentences. The evaluation was conducted on five test samples from Russian corpora, which contain sentences with syntax annotation. This research evaluates models such as Qwen, RuAdapt, Llama, T-pro, T-lite, and YandexGPT. A significant feature of this study is the specification of the gold token set in the prompt.

The article is organized as follows. Section 1 discusses related works. Section 2 provides information about syntax parsing. Section 3 outlines the experimental setup. Section 4 describes the results. The Conclusion summarizes the study and suggests directions for future work.

[1]Lomonosov Moscow State University, Moscow, Russian Federation

[2]For the Russian language [14], parsers DeepPavlov and Stanza exceed 0.9 by the metric UAS (Unlabeled Attachment Score) on the PUD and SynTagRus treebanks, also, the parsers exceed 0.75 on the treebanks Taiga, Poetry and GSD.

1. Related Works

In the area of syntax parsing, an emerging field is the application of LLMs with an autoregressive decoder architecture. The main approaches for this purpose are both fine-tuning [5] and prompt-tuning [6] methods. Furthermore, these parsers are distinguished by the employed syntactic representation: dependency trees or constituency structures. These key distinctions are summarized in Tab. 1.

Table 1. Related works

Article	Research type	Russian language	Syntax structure	Prompting mode
[6]	Fine-tuning	yes	Dependency	—
[19]	Fine-tuning	no	Dependency	—
[2]	Both	no	Constituency	Zero-shot, few-shot
[5]	Prompt-tuning	no	Dependency	**One-shot**
[16]	Prompt-tuning	no	Constituency	Zero-shot, others*
[7]	Other	no	Constituency	—
This article	**Prompt-tuning**	**yes**	**Dependency**	One-shot

* Five-shot prompt-tuning, and zero-shot prompt-tuning in Chain-of-Thought mode.

The articles about prompt-tuning do not consider the Russian language. Moreover, among them, only article [5] is devoted to prompt-tuning for dependency trees[3], and therefore its prompt design is the basis for our work. However, since that article did not consider Russian sentences, the results of these studies are not directly comparable.

2. Syntax

2.1. Syntax Representation

The most popular ways to represent the syntax structure of a sentence are constituency structure and dependency structure (also called dependency tree). Figure 1 shows examples of the structures. For the Russian language most syntax datasets are represented by dependency trees. A dependency tree is a directed graph, the nodes of which correspond to sentence tokens (words, punctuation marks, etc.) and edges correspond to relations between tokens. Each token is connected to one main token, which is called parent token. The root token of the tree is connected to the auxiliary token ROOT.

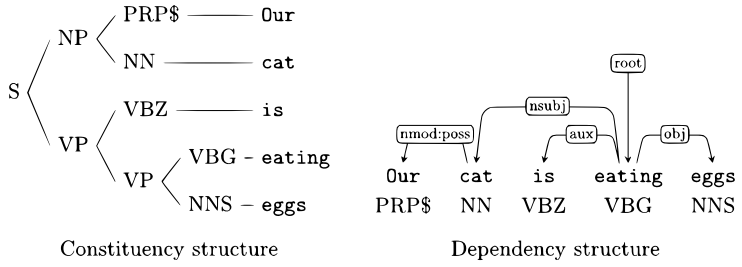

Figure 1. An example of constituency and dependency structures [8]

[3]An example of a prompt from [5] is provided in Appendix A.

The parsing by LLM is the task of generating a text sequence which describes the syntax structure of a sentence. So, a dependency tree should be represented in text format. There are two widely used formats: the CoNLL-U format and the bracket sequence format. The CoNLL-U format is also used in datasets of sentences with syntax markup (also called treebanks). In the CoNLL-U format, each line (except comment) corresponds to a token and consists of ten values, splitted by the tab character. The first value is the token identifier, the second is the token text, the seventh is the identifier of the parent token, the eighth is the relation tag. While the remaining values are morphological and other features of the token. Figure 2 shows an example of a CoNLL-U sentence.

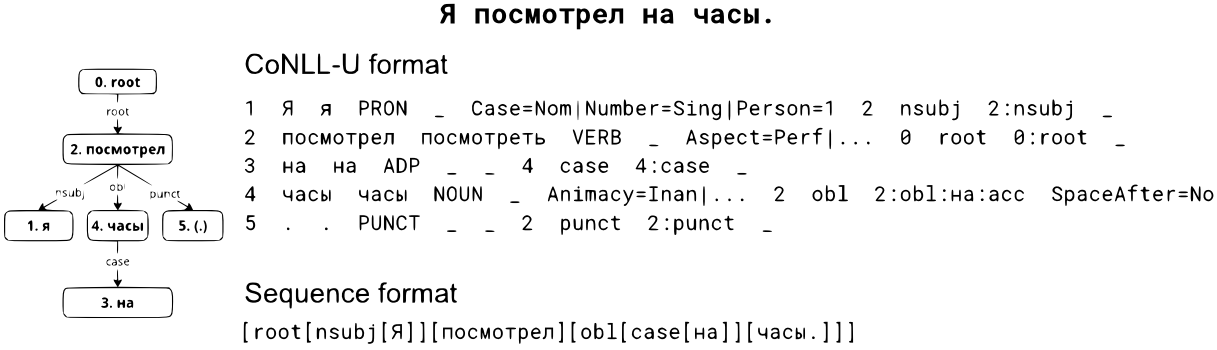

Figure 2. Examples of a sentence in the CoNLL-U and sequence formats

Datasets of dependency trees (treebanks) are stored primarily in the CoNLL-U format. So, the CoNLL-U format can be used by LLMs without fine-tuning [5], while fine-tuning is required for generating a dependency tree in the bracket sequence format [6]. Often, only four syntax columns of the CoNLL-U format are generated: token ID, form, parent ID, and relation type, while columns lexeme, part of speech and morphological features are replaced with the underscore symbol.

2.2. Evaluation of Syntax Parser

The standard way to evaluate parser results is to calculate metrics UAS (Unlabeled Attachment Score) and LAS (Labeled Attachment Score). The preliminary stage before evaluation is the aligning. At this stage, a correspondence is established between tokens from the dataset and tokens from the dependency tree, created by parser. After that, F1-score of UAS and LAS is calculated. Equations 1-6 show formulas for the calculation. G is a set of gold token, P is a set of dependency tree tokens, while $\mathbf{p(t)}$ is a function that maps a token to the parent token, $\mathbf{d(t)}$ is a function that maps a token to the relation between the token and the parent token.

$$UAS\_precision = \frac{\parallel gt | gt = pt, gt \in G, pt \in P, \mathbf{p}(gt) = \mathbf{p}(pt) \parallel}{\parallel pt | pt \in P \parallel}, \tag{1}$$

$$UAS\_recall = \frac{\parallel gt | gt = pt, gt \in G, pt \in P, \mathbf{p}(gt) = \mathbf{p}(pt) \parallel}{\parallel gt | gt \in G \parallel}, \tag{2}$$

$$LAS\_precision = \frac{\parallel gt | gt = pt, gt \in G, pt \in P, \mathbf{p}(gt) = \mathbf{p}(pt), \mathbf{d}(gt) = \mathbf{d}(pt) \parallel}{\parallel pt | pt \in P \parallel}, \tag{3}$$

$$LAS\_recall = \frac{\parallel gt|gt = pt, gt \in G, pt \in P, \mathbf{p}(gt) = \mathbf{p}(pt), \mathbf{d}(gt) = \mathbf{d}(pt) \parallel}{\parallel gt|gt \in G \parallel}, \quad (4)$$

$$UAS\_F1 = \frac{2 * UAS\_precision * UAS\_recall}{UAS\_precision + UAS\_recall}, \quad (5)$$

$$LAS\_F1 = \frac{2 * LAS\_precision * LAS\_recall}{LAS\_precision + LAS\_recall} \quad (6)$$

3. Experimental Setup

3.1. Prompts

In our article, as in [5], the prompt contains one sentence with the gold dependency tree in the CoNLL-U format. In both articles, the length of gold sentence is from 4 to 7. The sentence is used more as an example of the format than as a source of linguistic information. In [5] test sentences are randomly selected from the train set of a treebank, while in our work each prompt contains one gold dependency tree. Another difference from [5] lies in the format of token set representation. While in the original article, tokens from test sentences are simply separated by spaces, in our article the token set is represented as a python list. This approach is based on the assumption that LLMs work well with program code. In future, comparison between the token set representations will be planned.

Figure 3 shows an example of our prompt. In addition to prompts from [5], the article prompts also include gold token set and some restrictions. In our study, we experiment with 10 prompts. In our article, as in [5], one-shot prompting is considered. Each test prompt is passed to a tested LLM once. Figure 4 shows the gold sentences for the article prompts.

```
Пример: Предложение <Рядом проходит автомобильная дорога .> в формате
CONLL:
1 Рядом _ _ _ _ 2 advmod _ _
2 проходит _ _ _ _ 0 root _ _
3 автомобильная _ _ _ _ 4 amod _ _
4 дорога _ _ _ _ 2 nsubj _ _
5 . _ _ _ _ 2 punct _ _
Задание: Верни в формате CONLL предложение <С 2012 года центр
занимается также вопросом об освещении изменения климата .>:
Результат должен состоять из 12 строк в формате CONLL. Во втором
столбце должны быть токены ['С', '2012', 'года', 'центр', 'занимается',
'также', 'вопросом', 'об', 'освещении', 'изменения', 'климата', '.'].
Нельзя нарушать порядок токенов. Нельзя добавлять токены. Нельзя
удалять токены.
```

Figure 3. The prompt example of the article

1. Рядом проходит автомобильная дорога.
2. Соглашение рассчитано на два года.
3. Доцент Саратовского государственного университета.
4. Девушка ждала его 4 года.
5. Началу работ препятствовал недостаток финансирования.
6. Сила трения незначительная.
7. В России встречаются 2 вида.
8. Николай Резанов родился в Ленинграде.
9. Жена: Ольга Александровна Михайлова.
10. Женат, имеет трех сыновей.

Figure 4. Gold sentences

3.2. Models

In the study, we considered ten LLMs, working with the Russian language. The models are selected from open-sources projects Qwen[4], RuAdapt[5] [17], T-Tech[6], Yandex[7], LLaMA[8]. RuAdapt models are Russian adapted versions of Qwen models. Additionally, LLMs from T-Tech are based on Qwen models.

The study considers LLMs with different amount of parameters. Table 2 shows the values.

Table 2. Statistics on the amount of LLM parameters

Model	Parameters (billions)
Qwen/Qwen3-32B	32.8
Qwen/Qwen3-8B	8.19
Qwen/Qwen3-4B	4.02
RefalMachine/RuadaptQwen3-32B-Instruct	32.7
RefalMachine/RuadaptQwen3-8B-Hybrid	8.14
RefalMachine/RuadaptQwen3-4B-Instruct	4.01
t-tech/T-pro-it-2.0	32.8
t-tech/T-lite-it-1.0	7.61
yandex/YandexGPT-5-Lite-8B-instruct	8.04
unsloth/Llama-3.3-70B-Instruct	70.6

As a baseline, classical neural parsers DeepPavlov[9], Stanza [13], Natasha[10] were chosen. Russian parsers DeepPavlov and Stanza have demonstrated the best UAS and LAS results, while Natasha parser has shown the worst results [14].

[4] https://huggingface.co/Qwen
[5] https://huggingface.co/collections/RefalMachine/ruadaptqwen3-682e12092a5d2b3a3efbba2e
[6] https://huggingface.co/t-tech
[7] https://huggingface.co/yandex
[8] https://huggingface.co/meta-llama
[9] https://docs.deeppavlov.ai/en/master/features/models/syntax_parser.html
[10] https://natasha.github.io/

3.3. Test Data

The test sentences are taken from Russian treebanks in the Universal Dependencies project. The treebanks comprise documents from different genres [4]. E-communication texts (blogs and social media) are used to create the Taiga treebank[11]. The Poetry treebank[12] contains samples of Russian poetry from the 19th to early 21st centuries. The SynTagRus[13] treebank also includes texts from a variety of genres such as contemporary fiction, popular science, newspaper and journal articles written in a period from 1960-s to 2016, as well as online news texts. Sentences in the PUD[14] treebank are taken from the news and Wikipedia (where the Wikipedia texts were translated into Russian), while the GSD[15] treebank consists of sentences extracted from the Russian Wikipedia.

4. Results and Analysis

4.1. Metrics UAS and LAS

To evaluate parsing quality, we considered only correct CoNLL-U lines. Some lines, such as those containing extra underscore symbols, were also fixed and used in the evaluation. Moreover, for UAS and LAS calculation, only the last created line was considered among lines with identical identifiers. These duplicates arise from the reasoning processes of certain LLMs.

LLM results are significantly lower than the results of classical neural parsers. It was also found that the Russian language adaptation of LLMs do not lead to a significant improvement in quality. However, as the number of parameters increases, the UAS and LAS values increase too. The Llama70 model demonstrates the best result, while results of Qwen4 and RuAdapt4 models are the worst.

Table 3 and Table 4 show mean values of UAS and LAS metric[16]. In each treebank and parser the best prompt was chosen. Values greater than or equal to 0.5 for UAS and 0.4 for LAS are shown in bold in the tables. In each treebank the best value is underlined.

4.2. Prompt Analysis

In most experiments, the best results are achieved on the prompts with sentences 1 and 7[17], while the worst results are obtained on the prompts with sentences 9 and 10. The relationships between prompts and metrics differ depending on datasets and LLMs. Figure 5 shows boxplot diagrams for the Taiga treebank and LLMs Qwen4 and Qwen32. For Qwen32 the results are similar, while for Qwen4 there are some prompts with better results. More experiments with different gold dependency trees in prompts are planned.

4.3. Sentences with Mismatched Token Set

In each treebank and for each LLM there are sentences, for which the LLM generates a token set, different from the gold token set. Figure 6 shows an example of the sentence. Table 5 shows

[11] https://universaldependencies.org/treebanks/ru_taiga/index.html
[12] https://universaldependencies.org/treebanks/ru_poetry/index.html
[13] https://universaldependencies.org/treebanks/ru_syntagrus/index.html
[14] https://universaldependencies.org/treebanks/ru_pud/index.html
[15] https://universaldependencies.org/treebanks/ru_gsd/index.html
[16] For relation types with several parts ('nummod:gov') only first parts ('nummod') are considered.
[17] The sentences are shown in the Fig. 4.

Table 3. Maximum of UAS

	gsd	pud	taiga	poetry	syntagrus
Stanza (b)	0.85	0.93	0.79	0.82	0.94
DeepPavlov (b)	0.88	0.94	0.78	0.85	0.91
Natasha (b)	0.79	0.88	0.70	0.64	0.83
Llama70	**0.55**	**0.55**	**0.55**	**0.56**	**0.54**
T-pro (32)	**0.53**	**0.54**	**0.56**	**0.56**	**0.53**
Qwen32	**0.51**	**0.51**	**0.51**	**0.52**	**0.50**
RuAdapt32	0.49	**0.52**	0.47	0.49	0.49
Qwen8	0.44	0.46	0.45	0.48	0.44
RuAdapt8	0.38	0.40	0.42	0.44	0.39
YandexGPT (8)	0.43	0.42	0.43	0.46	0.42
T-lite (7)	0.39	0.39	0.39	0.43	0.38
Qwen4	0.41	0.43	0.44	0.47	0.43
RuAdapt4	0.33	0.33	0.34	0.38	0.34

Table 4. Maximum of LAS

	gsd	pud	taiga	poetry	syntagrus
Stanza (b)	0.73	0.76	0.79	0.87	0.91
DeepPavlov (b)	0.71	0.78	0.79	0.86	0.88
Natasha (b)	0.64	0.58	0.75	0.84	0.79
Llama70	**0.47**	**0.48**	**0.45**	**0.47**	**0.45**
T-pro (32)	**0.42**	**0.40**	**0.44**	**0.45**	**0.41**
Qwen32	**0.43**	**0.43**	**0.43**	**0.44**	**0.41**
RuAdapt32	0.36	0.40	0.37	**0.40**	0.37
Qwen8	0.32	0.33	0.33	0.36	0.33
RuAdapt8	0.27	0.28	0.30	0.33	0.28
YandexGPT (8)	0.30	0.31	0.30	0.35	0.30
T-lite (7)	0.25	0.25	0.25	0.30	0.25
Qwen4	0.24	0.24	0.26	0.31	0.26
RuAdapt4	0.18	0.18	0.20	0.24	0.19

minimal and maximal proportions of the sentences (for different prompts). For all treebanks and LLMs the values are different, so the proportion depends on the prompt.

4.4. Amount of Wrong Lines

A significant problem with using LLM is the generation of extra lines. Figure 7 shows an example of a sentence with extra lines. The statistics for extra line amount is shown in Fig. 8 and Fig. 9. The green color corresponds to sentences without extra lines. The yellow color corresponds to sentences, in which the amount of extra values is less than or equal to the number of correct CoNLL-U lines. The orange color shows the sentences, in which the ratio of extra lines to the correct lines is between 1 to 2, while the red color indicates sentences, in which the ratio of extra lines to the correct lines is more than 2. The black color indicates sentences without right lines.

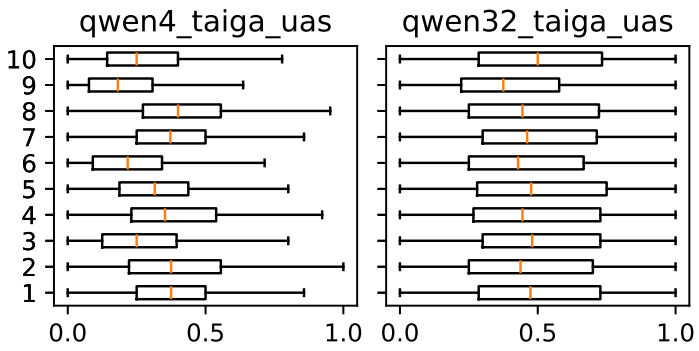

Figure 5. Examples of different relationships between prompts and UAS

Original dependency tree

```
1 Сперва          _ _ _ _ 5 advmod _ _
2 главный         _ _ _ _ 3 amod   _ _
3 балетмейстер    _ _ _ _ 5 nsubj  _ _
4 не              _ _ _ _ 5 advmod _ _
5 понял           _ _ _ _ 0 root   _ _
6 .               _ _ _ _ 5 punct  _ _
```

Generated dependency tree

```
1 Сперва          _ _ _ _ 2 advmod _ _
2 главный         _ _ _ _ 4 amod   _ _
3 балетмейстер    _ _ _ _ 4 nsubj  _ _
4 понял           _ _ _ _ 2 verb   _ _
5 .               _ _ _ _ 4 punct  _ _
```

Figure 6. An example of a mismatch between gold and generated token sets

Table 5. The proportion of sentences with different token sets

	gsd	pud	taiga	poetry	syntagrus
Llama70	6%–52%	5%–48%	2%–26%	2%–19%	4%–35%
Qwen32	24%–54%	29%–54%	12%–29%	11%–21%	21%–41%
RuAdapt32	23%–53%	13%–42%	8%–28%	7%–19%	12%–35%
T-pro	34%–63%	38%–61%	19%–36%	20%–30%	29%–47%
Qwen8	4%–17%	3%–12%	2%–9%	1%–10%	2%–12%
RuAdapt8	39%–76%	37%–73%	24%–61%	25%–57%	33%–67%
YandexGPT	18%–77%	18%–66%	8%–45%	11%–43%	15%–55%
T-lite	24%–54%	25%–45%	20%–30%	16%–24%	22%–36%
Qwen4	3%–43%	4%–44%	4%–28%	2%–22%	4%–36%
RuAdapt4	37%–94%	37%–90%	27%–84%	26%–78%	32%–84%

T-Pro and T-Lite models generate the most amount of sentences, in which the number of incorrect lines exceeds the number of correct ones. Also, Qwen8 and RuAdapt8 models generate many sentences with extra lines. RuAdapt4, RuAdapt8 and Qwen 8 models create the most amount of sentences with no correct lines.

```
Вот предложение в формате CONLL:

```
1 Пришел _ _ _ _ 2 nsubj _ _
2 и _ _ _ _ 0 root _ _
3 он _ _ _ _ 2 advmod _ _
4 , _ _ _ _ 2 punct _ _
5 тот _ _ _ _ 6 det _ _
6 незабвенный _ _ _ _ 5 amod _ _
7 день _ _ _ _ 2 obj _ _
8 , _ _ _ _ 7 punct _ _
```

Объяснение:
- "Пришел" является подлежащим (nsubj) и связано с глаголом "пришел".
- "и" является корневым словом (root) и соединяет два предложения.
- "он" является дополнением (advmod) к глаголу "пришел".
- Запятая после "он" является пунктуацией (punct).
- "тот" является определением (det) для прилагательного "незабвенный".
- "незабвенный" является прилагательным (amod), модифицирующим
существительное "день".
- "день" является объектом (obj) глагола "пришел".
- Запятая после "день" является пунктуацией (punct).
```

Figure 7. An example of a sentence with extra lines

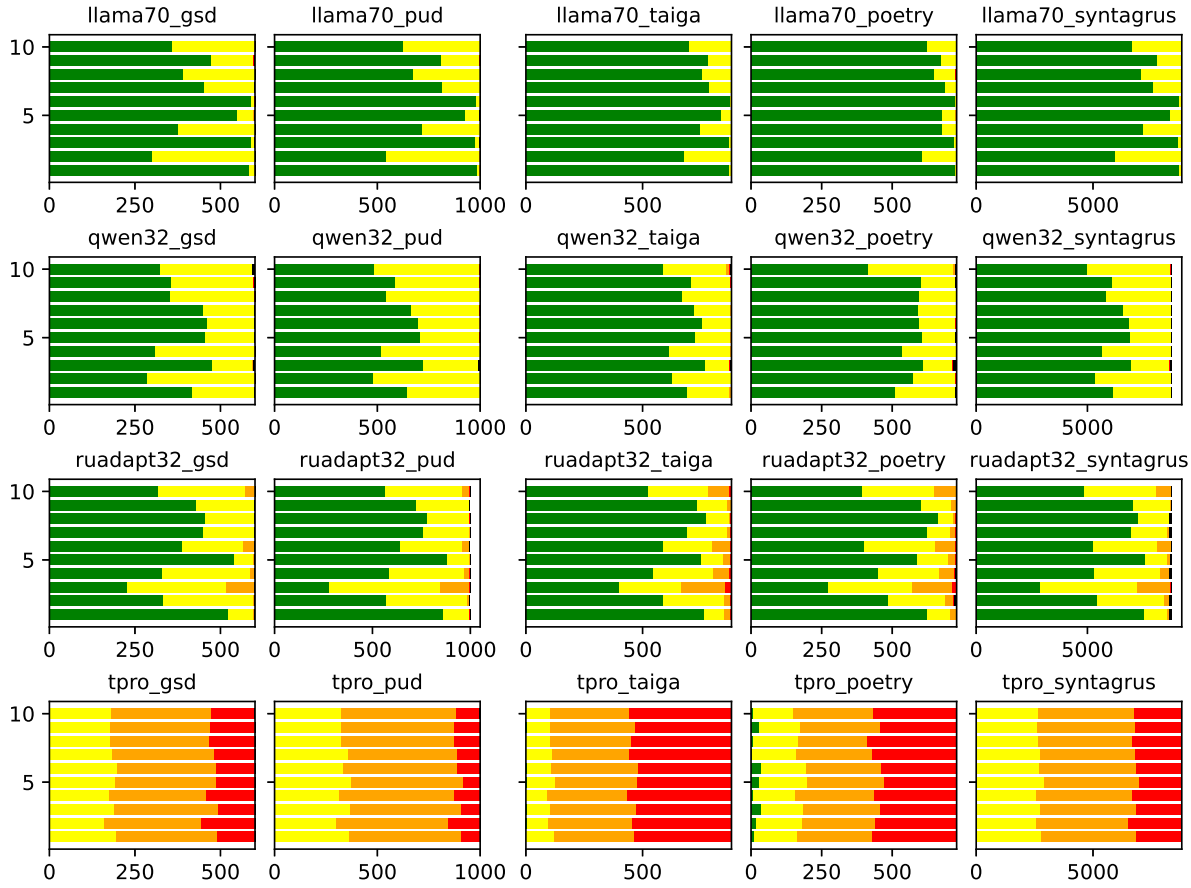

Figure 8. Statistics for extra lines for LLMs with 32–70 billion parameters

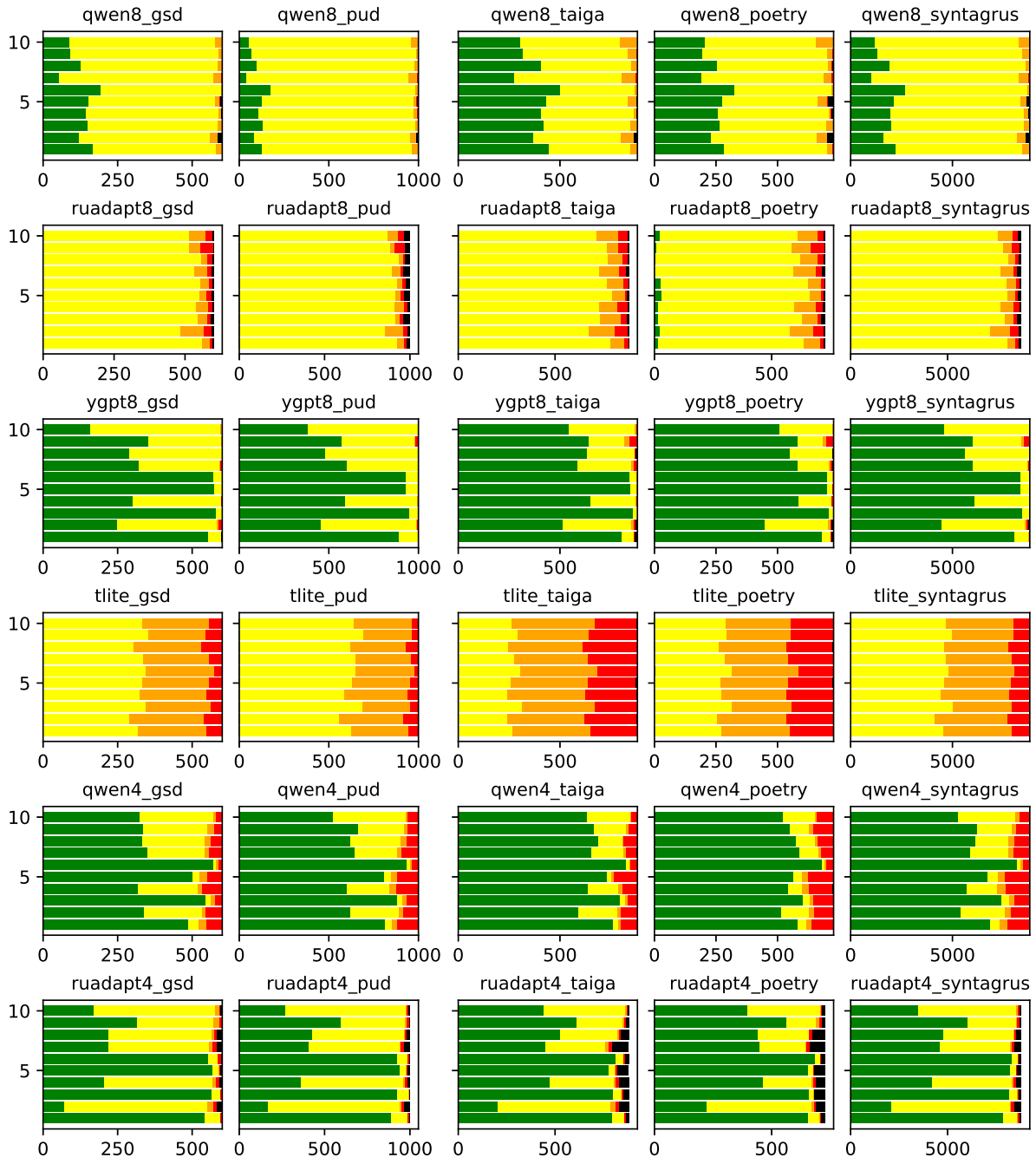

Figure 9. Statistics for extra lines for LLMs with 4–8 billion parameters

Conclusion

The study explores the applicability of Large Language Models in one-shot mode for dependency parsing of Russian sentences. An evaluation of ten LLMs was conducted across five Russian treebanks from the UD project.

The results demonstrate a connection between model parameters and quality, with the largest in our research model, Llama-70B achieving the highest scores.

Also, for some LLMs and treebanks the sentence used in the prompt is observed to influence syntax parser quality.

One of the identified problems is the generation of extra lines, which was particularly severe in the T-pro and T-lite models. For many sentences, these models produced more extra lines than correct ones. RuAdapt4, RuAdapt8 and Qwen models did not generate any correct CoNLL-U lines for a considerable number of sentences. Moreover, a significant difference was detected between the generated token sets and the gold token sets in a considerable fraction of the treebanks examined.

LLM results remain lower than that of classical neural syntax parser. It is partially affected by extra and incorrect generated lines.

Future work will involve experiments with different prompt instructions, gold token set representations, few-shot learning modes and multi-stage prompting. We will also examine the effect of gold dependency relations in an example from a prompt on the parsing results for different dependency types. Moreover, difficult cases, such as complex sentences and sentences with large dependency trees, will be considered. Another direction is a systematic investigation of the problem of generating an incorrect number of tokens.

Acknowledgements

The research is carried out using the equipment of the shared research facilities of HPC computing resources at Lomonosov Moscow State University.

The research was supported by the Russian Science Foundation, project No. 25-11-00191, https://rscf.ru/project/25-11-00191/.

References

1. Alonso, M.A., Gómez-Rodríguez, C., Vilares, J.: On the use of parsing for named entity recognition. Applied Sciences 11(3) (2021). https://doi.org/10.3390/app11031090

2. Bai, X., Wu, J., Chen, Y., *et al.*: Constituency parsing using LLMs. IEEE Transactions on Audio, Speech and Language Processing 33, 3762–3775 (2025). https://doi.org/10.1109/TASLPRO.2025.3600867

3. Corbetta, C., Passarotti, M., Moretti, G.: The Rise and Fall of Dependency Parsing in Dante Alighieri's Divine Comedy. In: Sprugnoli, R., Passarotti, M. (eds.) Proceedings of the Third Workshop on Language Technologies for Historical and Ancient Languages (LT4HALA) @ LREC-COLING-2024. pp. 50–56. ELRA and ICCL (2024), https://aclanthology.org/2024.lt4hala-1.7/

4. Droganova, K., Lyashevskaya, O., Zeman, D.: Data Conversion and Consistency of Monolingual Corpora: Russian UD Treebanks. In: Proceedings of the 17th International Workshop on Treebanks and Linguistic Theories (TLT 2018). pp. 53–66. Linköping University Electronic Press, Linkping, Sweden (2018), https://ep.liu.se/ecp/155/007/ecp18155007.pdf

5. Ezquerro, A., Gómez-Rodríguez, C., Vilares, D.: Better benchmarking LLMs for zero-shot dependency parsing. In: Johansson, R., Stymne, S. (eds.) Proceedings of the Joint 25th Nordic Conference on Computational Linguistics and 11th Baltic Conference on Human Language Technologies (NoDaLiDa/Baltic-HLT 2025). pp. 121–135. University of Tartu Library (2025), https://aclanthology.org/2025.nodalida-1.13/

6. Hromei, C.D., Croce, D., Basili, R.: U-DepPLLaMA: Universal Dependency Parsing via Auto-regressive Large Language Models. IJCoL. Italian Journal of Computational Linguistics 10(1) (2024). https://doi.org/10.4000/125nm

7. Kim, T.: Revisiting the practical effectiveness of constituency parse extraction from pre-trained language models. In: Calzolari, N., Huang, C.R., Kim, H., *et al.* (eds.) Proceedings of the 29th International Conference on Computational Linguistics. pp. 5398–5408. International Committee on Computational Linguistics (2022), https://aclanthology.org/2022.coling-1.479/

8. Le-Hong, P., Cambria, E.: Integrating graph embedding and neural models for improving transition-based dependency parsing. Neural Computing and Applications 36, 2999–3016 (2024). https://doi.org/10.1007/s00521-023-09223-3

9. Lin, L., Ziyang, C., Shuxing, L., *et al.*: Event extraction in complex sentences based on dependency parsing and longformer. In: Nianyin, Z., Pachori, R.B. (eds.) Proceedings of 2024 International Conference on Machine Learning and Intelligent Computing. Proceedings of Machine Learning Research, vol. 245, pp. 1–7. PMLR (2024), https://proceedings.mlr.press/v245/lin24a.html

10. Liu, T., Sun, Y., Wu, J., *et al.*: Unsupervised paraphrasing under syntax knowledge. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 37, pp. 13273–13281 (2023). https://doi.org/10.1609/aaai.v37i11.26558

11. Morozov, D., Lagutina, K., Drozhashchikh, G., *et al.*: Exploring the feature space for cross-domain assessing the complexity of russian-language texts. In: 2024 Ivannikov Ispras Open Conference (ISPRAS). pp. 1–8 (2024). https://doi.org/10.1109/ISPRAS64596.2024.10899137

12. Nikolaev, I.E.: Knowledge and skills extraction from the job requirements texts. Ontology of Designing 13(2), 282–293 (2023). https://doi.org/10.18287/2223-9537-2023-13-2-282-293

13. Qi, P., Zhang, Y., Zhang, Y., *et al.*: Stanza: A Python Natural Language Processing Toolkit for Many Human Languages. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations. pp. 101–108 (2020)

14. Shamaeva, E.: Russian parser comparison. In: International Journal of Open Information Technologies Proceedings (2025)

15. Taufiq, U., Pulungan, R., Suyanto, Y.: Named entity recognition and dependency parsing for better concept extraction in summary obfuscation detection. Expert Systems with Applications 217, 119579 (2023). https://doi.org/10.1016/j.eswa.2023.119579

16. Tian, Y., Xia, F., Song, Y.: Large language models are no longer shallow parsers. Commun. ACM 58(4), 7131–7142 (2024). https://doi.org/10.18653/v1/2024.acl-long.384

17. Tikhomirov, M., Chernyshev, D.: Facilitating large language model russian adaptation with learned embedding propagation. Journal of Language and Education 10(4), 130–145 (2024). https://doi.org/10.17323/jle.2024.22224

18. Vasiliev, S., Korobkin, D., Fomenkov, S.: Extracting the Component Composition Data of Inventions from Russian Patents using Dependency Tree Analysis. In: 2023 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM). pp. 1030–1034 (2023). https://doi.org/10.1109/ICIEAM57311.2023.10139170

19. Zhou, H., Chersoni, E., Hsu, Y.Y.: Branching out: Exploration of chinese dependency parsing with fine-tuned large language models. In: Conference on Recent Advances in Natural Language Processing (RANLP 2025), Varna, September 8-10, 2025. pp. 1437–1445. Association for Computational Linguistics (2025). https://doi.org/10.26615/978-954-452-098-4-166

Appendix A. Prompt Examples

Figure 10 demonstrates an example of prompts from [5]. The tokens from gold and test sentences are splitted by spaces.

```
In dependency parsing the CoNLL format for the sentence <The
trial begins again Nov 28 .> is:
1 The _ _ _ _ 2 det _ _
2 trial _ _ _ _ 3 nsubj _ _
3 begins _ _ _ _ 0 root _ _
4 again _ _ _ _ 3 advmod _ _
5 Nov. _ _ _ _ 3 obl:tmod _ _
6 28 _ _ _ _ 5 nummod _ _
7 . _ _ _ _ 3 punct _ _
Now return the CoNLL format for the sentence: <What if Google
Morphed Into GoogleOS ?>
```

Figure 10. A prompt for the simplified CoNLL-U format [5]