УДК 004.85

СИСТЕМА ОГРАНИЧЕНИЙ ГЕНЕРАЦИИ ДЛЯ УСТРАНЕНИЯ СТРУКТУРНЫХ ОШИБОК ПРИ СОЗДАНИИ ДЕРЕВА ЗАВИСИМОСТЕЙ С ПОМОЩЬЮ БОЛЬШОЙ ЯЗЫКОВОЙ МОДЕЛИ

Е. Д. Шамаева[0009-0002-6233-8958]

Московский государственный университет имени М. В. Ломоносова, г. Москва, Россия

derinhelm@yandex.ru

Аннотация

Для синтаксического анализа естественного языка перспективным направлением является дообучение больших языковых моделей для генерации синтаксической структуры предложения в виде скобочной последовательности формата Grammatical Relation Centered Tree (GRCT). Существующие дообученные модели демонстрируют высокие значения метрик оценки качества синтаксического анализа, однако в некоторых случаях генерируют некорректную скобочную последовательность (например, с несбалансированным числом открывающих и закрывающих скобок). В работе разработан метод, позволяющий уменьшить количество некорректно сгенерированых последовательностей. Для этого создана многоэтапная система ограничений субтокенов, генерируемых на каждом этапе работы большой языковой модели. Реализация этого метода протестирована на 4 моделях (адаптированные и неадаптированные модели размерами 4 и 8 млрд параметров) на датасете деревьев зависимостей SynTagRus. Для всех моделей количество некорректно сгенерированных последовательностей уменьшилось. Дополнительно установлено, что такие ограничения на генерацию влияют на качество собственно синтаксического анализа (может происходить повышение или понижение метрик оценки качества синтаксического анализа). Реализованная система ограничений может быть использована для повышения надежности работы любой большой языковой модели, дообученной для синтаксического анализа в формате скобочной последовательности GRCT. Полученные результаты опубликованы в открытом доступе.

***Ключевые слова**: синтаксический анализ, большие языковые модели, дообучение, ограничение вывода, автоматическая обработка текстов.*

ВВЕДЕНИЕ

Одной из классических задач обработки текстов является автоматический синтаксический анализ, в результате которого определяется синтаксическая структура предложения. В настоящее время синтаксический анализатор применяется как вспомогательный инструмент при разработке интерпретируемых методов решения таких задач, как распознавание именованных сущностей [1–4], оценка сложности текста [5], перефразирование [6], выявление некорректных заимствований [7]. Многие синтаксические анализаторы используют нейронные сети [8–10]. Однако в последние годы для синтаксического анализа начали использовать и большие языковые модели архитектуры декодер (Large Language Model, LLM). Известны работы по применению LLM как без дообучения [11], так и с дообучением [12].

При дообучении LLM для синтаксического анализа успешно показало себя представление синтаксической структуры предложения в виде скобочной последовательности (соответствующей дереву зависимостей) [12]. Однако использование скобочной последовательности стало причиной появления нового класса ошибок синтаксического анализатора – завершения синтаксического анализа с ошибкой. Это происходит при генерации некорректной скобочной последовательности. Примером некорректной скобочной последовательности является последовательность с дисбалансом открывающих и закрывающих скобок.

Для решения этой проблемы в настоящей работе разработана многоэтапная система ограничений генерации. Эта система ограничений на каждом шаге работы LLM запрещает генерацию фрагментов последовательности, нарушающих корректность сгенерированного результата. Система учитывает как сгенерированный фрагмент дерева зависимостей, так и словарь большой языковой модели (система ограничений без донастройки может быть применена к разным большим языковым моделям).

Основной результат работы заключается в создании системы ограничений, ее реализации на языке python3 для библиотеки vllm и тестировании работоспособности этой системы на дообученных больших языковых моделях (двух мультиязычных и двух адаптированных).

1. БЛИЗКИЕ ПО ТЕМАТИКЕ РАБОТЫ

Нам неизвестны исследования по ограничению вывода LLM для синтаксического анализа в формате скобочной последовательности. Однако в сфере применения LLM для задачи синтаксического анализа существуют работы, связанные с дообучением LLM [12] и ее промтированием [11]. В табл. 1 представлена информация о статьях, связанных с применением LLM для синтаксического анализа[1]. В этих работах был использован нестандартный способ вычисления метрик UAS и LAS, поэтому числовые результаты из этих статей не могут быть напрямую сравнены с результатами настоящей статьи.

Табл. 1. Информация об использовании LLM для синтаксического анализа.

Тип	Языки	Деревья зависимостей или составляющие	Источники
FT	26 языков, в том числе русский	Деревья зависимостей	[12]
PT	12 языков, включая русский	Деревья зависимостей	[13]
	Английский, китайский,	Непосредственные составляющие	
PT+FT	17 языков (включая русский)	Деревья зависимостей	[14]
FT	Китайский	Деревья зависимостей	[15]
PT	Английский, китайский, японский	Непосредственные составляющие	[16]
PT+FT	Английский	Непосредственные составляющие	[17]
PT	8 языков (исключая русский)	Деревья зависимостей	[18]
PT	Английский, французский,	Деревья зависимостей	[11]

[1] В этих работах рассмотрены различные способы представления синтаксической структуры предложений (деревья зависимостей и структуры непосредственных составляющих).

	немецкий, хинди		
Другое	Английский, немецкий, французский и другие языки (исключая русский)	Непосредственные составляющие	[19]
PT	Английский, китайский	Деревья зависимостей	[20]

Кроме того, на момент исследования в большинстве работ не проводились эксперименты для русского языка. Как можно увидеть из табл. 1, только в 3 работах учтен русский язык. В них были использованы различные форматы представления дерева зависимостей: в [12] – формат скобочной последовательности GRCT (Grammatical Relation Centered Tree), в [13] и [14] – упрощенная версия формата CoNLL-U [21]. Целью настоящей работы является построение системы ограничений для скобочного формата, поэтому методология дообучения LLM была взята из [12].

2. МЕТОДОЛОГИЯ ИССЛЕДОВАНИЯ

Синтаксический анализ

Эксперименты по созданию синтаксического анализаторы различаются несколькими аспектами (помимо собственно архитектуры синтаксического анализатора): способом представления синтаксической структуры предложения, используемым датасетам для обучения и тестирования, а также метриками для оценки качества работы синтаксического анализатора.

Нами, как и в [12], для представления синтаксической структуры предложения использовано дерево зависимостей – ориентированный граф вида дерево, в котором вершины соответствуют токенам предложения, а ребра – синтаксическим связям между токенами. Для каждого ребра предложения указан тип связи между соответствующими токенами. Каждому токену соответствует ровно один главный токен (токен, от которого к данному токену проведено ребро). Главным токеном для корня дерева является вспомогательный токен «root».

Для обучения и тестирования нами использован датасет деревьев зависимостей SynTagRus[2]. Этот датасет входит в проект Universal Dependencies[3], содержащий датасеты деревьев зависимостей для более 100 языков. Датасет SynTagRus основан на художественных и публицистических текстах. Приблизительный его объем составляет около 100 тыс. предложений, тестовая выборка содержит 8800 предложений.

Классическим способом оценки синтаксических анализаторов является вычисление на тестовых предложениях средних значений метрик UAS (Unlabeled Attachment Score) и LAS (Labeled Attachment Score) [21]. Семантически метрика UAS соответствует доле токенов предложения, для которых верно определен главный токен, метрика LAS – доле токенов, для которых верно определены главный токен и тип связи. Для этих метрик вычисляют точность (precision), полноту (recall) и F-меру. В качестве числителя указанных метрик используют количество «корректных» токенов (для метрики UAS – с правильно определенным главным токеном, для метрики LAS – с правильно определенными главным токеном и типом связей). В качестве знаменателя для точности используется количество токенов в построенном дереве зависимостей, для полноты – количество токенов в эталонном дереве зависимостей.

Однако классический способ оценки качества синтаксического анализа не учитывает потенциальное изменение предложения в результате галлюцинирования LLM, поэтому классический способ был модифицирован.

Модифицированный способ оценки качества синтаксических анализаторов

Особенностью оценки качества работы синтаксических анализаторов на основе LLM является принципиальная невозможность использования классического способа их оценки. Причиной этого является то, что синтаксические анализаторы на основе LLM, в отличие от стандартных, могут изменять предложение:

[2] https://universaldependencies.org/treebanks/ru_syntagrus/index.html

[3] https://universaldependencies.org/

удалять, добавлять и переставлять токены. Эта особенность делает невозможным применение стандартного алгоритма выравнивания токенов эталонного и построенных деревьев зависимостей.

В классическом случае оценка качества работы с помощью метрик UAS и LAS производится на наборе тестовых предложений и происходит в несколько этапов.

1. Для каждого тестового предложения осуществляется:
   - выравнивание эталонного предложения и результата синтаксического анализа;
   - вычисление количества «корректных» токенов;
   - вычисление точности, полноты и F-меры для метрик UAS и LAS.
2. Проведится усреднение значений F-меры для метрик UAS и LAS на всех тестовых предложениях.

В классическом алгоритме вычисления метрик на этапе выравнивания производится сопоставление токенов из эталонного дерева зависимостей и дерева зависимостей, построенного анализатором. Стандартным способом сопоставления токенов является сравнение их индексов начала и конца в предложении (для этого требуется, чтобы текст предложения не изменялся при синтаксическом анализе). Однако синтаксические анализаторы на основе LLM могут изменять исходное предложение: переставлять токены, удалять существующие и добавлять новые. Поэтому способ выравнивания на основе индексов неприменим.

Для решения указанной проблемы мы использовали модифицированные метрики UAS и LAS: сравнение ребер происходит не на основе сравнения индексов (как в стандартном варианте), а на основе форм главного и зависимого токенов ребра. В примере на рис. 1 показаны эталонное и построенное деревья зависимостей. В построенном дереве зависимостей изменен текст предложения (пропущен токен «это»). Поэтому индексы начала и конца токенов в предложении нельзя использовать (следовательно, нельзя использовать и классические метрики UAS и LAS). При вычислении модифицированных метрик UAS и LAS между построенным и эталонным деревьями зависимостей с точки зрения совпадения текстов вершин совпадает 7 ребер. С точки зрения совпадения текстов вершин и типов связи ребра совпадают 6 ребер. Поэтому в числителях метрик

UAS и LAS будут значения 7 и 6 соответственно, в знаменателях точности и полноты – соответственно 7 и 8 (токен «root» не учитывается).

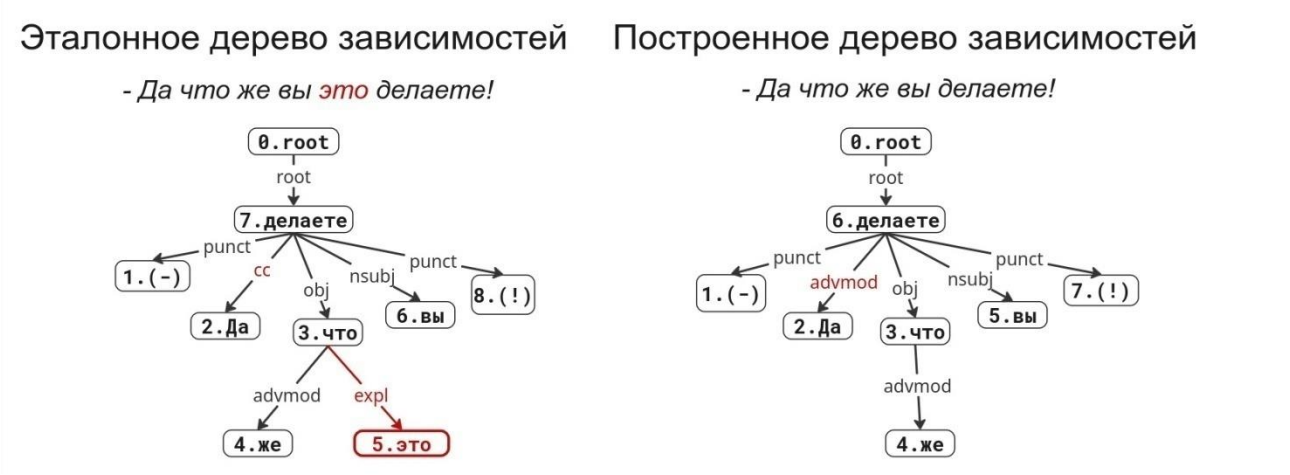

Рис. 1. Выравнивание эталонного дерева зависимостей и дерева зависимостей, построенного с помощью LLM.

Скобочные форматы представления дерева зависимостей

Синтаксические анализаторы, основанные на большой языковой модели, сводят задачу синтаксического анализа к задаче генерации последовательности. Для представления дерева зависимостей в виде текстовой последовательности используют несколько форматов. Более известным является формат CoNLL-U, в котором хранятся датасеты деревьев зависимостей (пример предложения в этом формате представлен на рис. 2). В этом текстовом формате каждому токену соответствует одна строка, в которой для этого токена указаны его индекс, словоформа, лемма, морфологические характеристики, а также индекс главного токена и тип связи с главным токеном. Однако в данном формате связь между главным и зависимым токенами осуществляется через индекс главного токена. Таким образом, LLM должна корректно связать между собой не только индексы главного и зависимого токенов, но и индекс главного токена с главным токеном. Это усложняет работу LLM и влияет на качество ее работы.

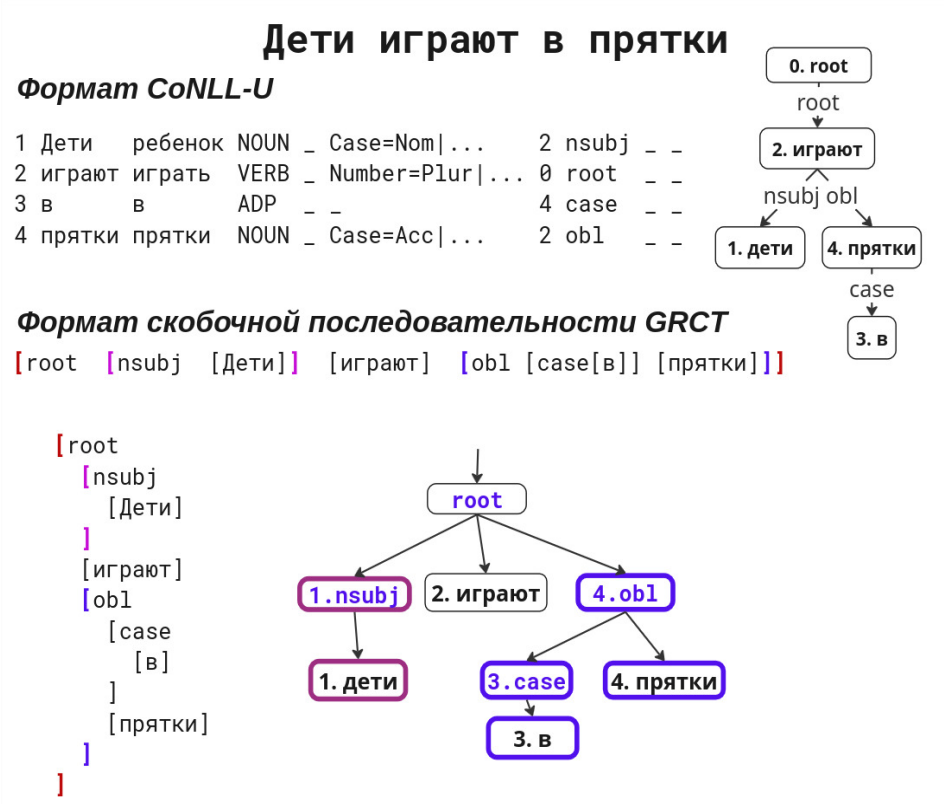

Рис. 2. Форматы текстовой записи дерева зависимостей.

Проблему соединения токенов на основе индекса решает скобочной формат дерева зависимостей. В частности, в [12] для генерации дерева зависимостей с помощью LLM использован формат GRCT (Grammatical Relation Centered Tree). Пример представления дерева зависимостей в формате GRCT приведен на рис. 2. Построение такого текстового представления происходит в два этапа: преобразование дерева зависимостей в дерево без меток на ребрах и создание скобочной последовательности с помощью обхода в глубину дерева без меток. Для преобразования дерева зависимостей в дерево без меток добавляются в дерево вершины с метками на ребрах, при этом вершина типа связи становится главной для вершины исходного токена. При генерации с помощью LLM дерева зависимостей в формате скобочной последовательности GRCT связь между вершинами дерева зависимостей выражается через границы поддеревьев (открывающие и закрывающие квадратные скобки).

При использовании скобочной последовательности синтаксический анализ на основе LLM происходит в два этапа:

1) генерация дерева зависимостей в формате скобочной последовательности;

2) декодирование дерева зависимостей из скобочного формата.

На втором этапе может возникнуть ошибка, если сгенерирована последовательность, которую невозможно корректно отобразить в дерево зависимостей (например, нарушен баланс скобок). Такое завершение синтаксического анализа с ошибкой архитектурно невозможно для классических нейросетевых синтаксических анализаторов. Поэтому при вычислении средних значений метрик UAS и LAS рассматривают два вида средних значений: на множестве предложений с корректным результатом и множестве всех предложений (для некорректных результатов значения метрик считаются равными 0).

Применение больших языковых моделей для синтаксического анализа в формате скобочной последовательности

Генерация текста с помощью LLM происходит на основе генерации фрагментов текстов, субтокенов. Для каждой большой языковой модели заранее определен словарь субтокенов. Субтокенами могут быть как фрагменты слов, так и последовательности символов (например, «игра», «ют», «[color», «]])\n\n», «]=»).

В настоящей работе рассмотрены LLM – архитектуры «авторегрессионный декодер», которые обучены для продолжения текста. Генерация текста с помощью такой архитектуры происходит итеративно. На каждой итерации LLM обрабатывает последовательность субтокенов и на ее основе предсказывает номер следующего субтокена для этой последовательности. На следующей итерации LLM обрабатывает последовательность субтокенов, дополненную новым субтокеном. Пример генерации дерева зависимостей в скобочном формате представлен на рис. 3 («Дети играют в прятки»). При генерации скобок использованы не только субтокены «[» и «]», но и субтокены «][», «]]».

Рис. 3. Пример поэтапной генерации дерева зависимостей в скобочном формате.

Обработка последовательности субтокенов происходит в несколько этапов:

1) применение тензорных преобразований, описываемых архитектурой и весами LLM. На этом шаге для каждого субтокена из словаря субтокенов берется числовое значение, логит. На основе логитов в дальнейшем будет происходить выбор субтокена;

2) применение ограничений (изменение значений логитов);

3) выбор следующего субтокена на основе измененных значений логитов.

Выбор следующего субтокена осуществляется разными стратегиями. Нами рассмотрена стратегия выбора субтокена с максимальным значением логита (семплирование не применяется).

При использовании LLM для генерации дерева зависимостей существенной проблемой является генерация некорректных скобочных последовательностей (в частности, последовательностей с дисбалансом открывающих и закрывающих скобок). Нами выделены две основные причины генерации некорректных скобочных последовательностей:

1) генерация некорректного субтокена (например, субтокена «]bool», в котором после закрывающей скобки стоит текст);

2) генерация некорректной последовательности субтокенов (например, субтокены «[» и «]» вместе дают некорректный фрагмент последовательности «[ ]»).

Для решения указанных проблем была разработана многоэтапная система ограничения логитов. Эти ограничения применяются на каждой итерации генерации. С их помощью для логитов субтокенов, из-за которых скобочная последовательность становится некорректной, устанавливается значение минус бесконечность. Вследствие этого такие субтокены не будут выбраны на этой итерации.

Система ограничений

Рассмотрены два вида ограничений: принудительная генерация и запрет генерации. При использовании ограничений принудительной генерации для логитов всех субтокенов, кроме заданных, устанавливаются значения минус бесконечность (на этом этапе генерации может быть сгенерирован только токен из заданного набора субтокенов). При использовании ограничения вида запрет генерации для всех субтокенов из заданного набора устанавливается значение логитов, равное минус бесконечности (т. е. могут быть сгенерированы все субтокены, кроме заданных).

Табл. 2. Система ограничений генерации.

	Название	Тип	Ограничение	Применимость
1.	Генерация первого символа [	прин.	Разрешена генерация только субтокена [	Не сгенерировано ни одного субтокена
2.	Генерация символа конца последовательности	прин.	Разрешена генерация только субтокена завершения последовательности	Количество [равно количеству]

3.	Ограничение на генерацию [	огр.	Запрещена генерация открывающей скобки	Количество [превысило заданное значение
4.	Ограничение на генерацию субтокенов с запрещенными символами	огр.	Запрещена генерация субтокенов из набора субтокенов с запрещенными символами	Всегда
5.	Ограничение генерации символов после [	огр	Запрещена генерация [и]	В конце сгенерированной последовательности стоит [
6.	Ограничение на генерацию символа конца строки при дисбалансе скобок	огр	Запрещена генерация субтокена конца последовательности	Количество [меньше, чем количество]
7.	Ограничение на дисбаланс скобок	огр	Запрещена генерация субтокенов, содержащих такое количество [и], что после добавления этого субтокена в сгенерированной последовательности станет] больше, чем [	Всегда

Информация о всех ограничениях, использованных в настоящем исследовании, представлена в табл. 2. Для каждого ограничения дополнительно указаны требования к его применимости. Использованы следующие обозначения: «прин» – ограничение типа принудительная генерация, «огр» – ограничение запрета генерации.

Ограничения применялись в порядке следования в таблице. Если было применено одно из ограничений 1–3, то дальнейшие ограничения не применялись.

В ограничении 3 использовано количество скобок, заданное пользователем. В данной работе это значение равно количеству токенов в эталонном дереве зависимостей, увеличенному на 10. Это ограничение также может быть отключено.

В ограничении 7 проверка сбалансированности скобок происходит для каждого префикса нового субтокена, содержащего квадратные скобки. Это по-

могает избежать некорректных скобочных последовательностей вида «[Остается][», в которой при добавлении субтокена «][» произошла ошибка (все открывающие скобки были закрыты).

В ограничении 4 учтено несколько типов запрещенных субтокенов:

1) субтокены, содержащие последовательность [] (например, субтокены «[].», «[][»);

2) субтокены, в которых перед символом [присутствует символ, отличный от символа] (например, субтокены «[[», «:[»);

3) субтокены, в которых после символа] присутствует символ, отличный от символов [или] (например, субтокены «]bool», «].», «].[»);

4) субтокены, в которых присутствуют символы \n, \r и другие (например, субтокены «)]\n\n», «)]\r\n»);

5) технические субтокены «<|im_start|>», «<think>», «</think>».

На данный момент даже при использовании разработанной системы может быть сгенерирована некорректная последовательность, если на каком-то уровне дерева зависимостей в виде скобочной последовательности не будет вершины с формой токена. Примером такой последовательности является [root[nsubj]].

Дообучение больших языковых моделей для генерации дерева зависимостей в формате скобочной последовательности

Разработанная система ограничений была протестирована на LLM, дообученных для генерации дерева зависимостей в скобочном формате. Для дообучения была использована методология из [12]. Рассмотрены инструктивные модели, ориентированные на взаимодействие с пользователем в формате чата. Пример работы LLM, дообученной для «<|im_start|>» синтаксического анализа, показан на рис. 4 (технические субтокены и «<|im_end|>» соответствуют началу и концу сообщения, «user» и «assistant» – сообщению от пользователя и LLM, «<think>» и «</think>» – началу и концу фрагмента размышлений LLM, в данном эксперименте пустому). В дообученную LLM передавался промпт,

содержащий исходное предложение. В результате работы LLM возвращался исходный промпт, дополненный текстовой записью дерева зависимостей. Примеры исходного и результирующего промптов также показаны на рис. 4.

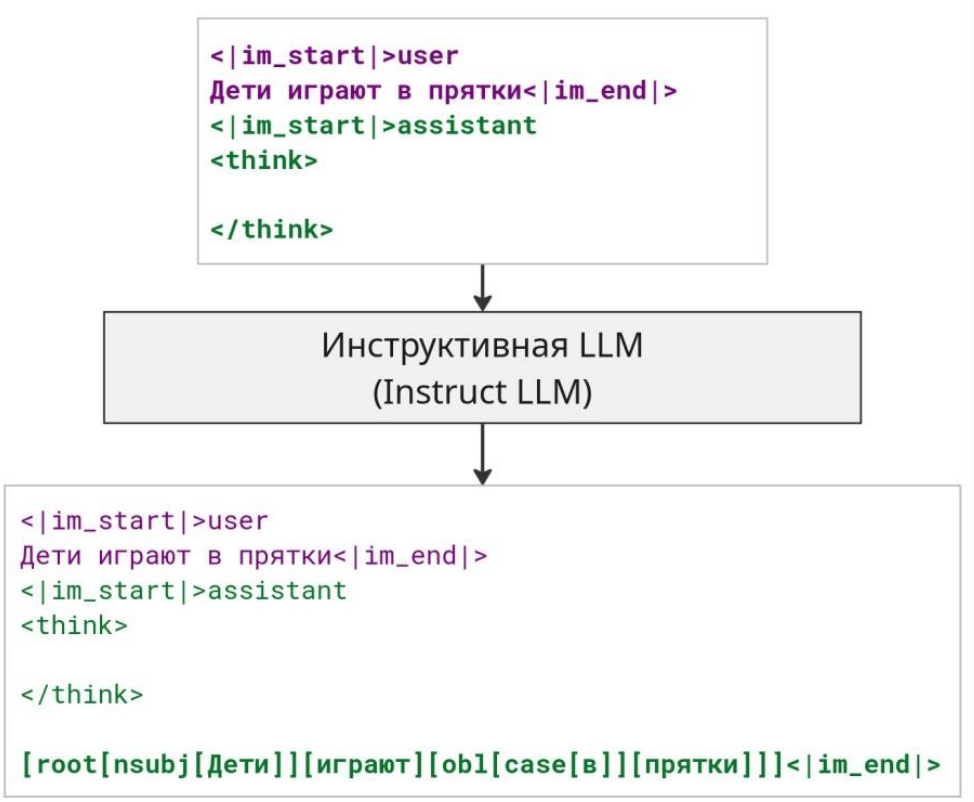

Рис. 4. Пример работы дообученной инструктивной большой языковой модели.

В настоящем исследовании дообучение производилось для двух мультиязычных моделей (Qwen3-4B и Qwen3-8B) и двух адаптаций этих моделей для русского языка (RuadaptQwen3-4B-Hybrid и RuadaptQwen3-8B-Hybrid). Размеры этих моделей составляют 4 и 8 млрд параметров соответственно. Авторами адаптированных моделей [22] был изменен словарь субтокенов и соответствующим образом скорректированы веса моделей.

Как и в [12], нами для дообучения был использован метод LoRA с квантованием. Дообучение производилось на датасете русскоязычных деревьев зависимостей SynTagRus.

Для каждой дообученной модели проводилось тестирование в двух режимах: с использованием ограничений генерации и без их использования.

3. РЕЗУЛЬТАТЫ

При дообучении больших языковых моделей было проведено несколько экспериментов для каждой модели и была выбрана дообученная модель с наилучшим результатом. Далее каждая модель была запущена с ограничением генерации. Результаты тестирования приведены в табл. 3.

Табл. 3. Пример поэтапной генерации дерева зависимостей в скобочном формате.

Модель	Ограничения	Количество некорректных предложений	UAS (корр)	UAS (все)	LAS (корр)	LAS (все)
DeepPavlov	—	*0*	*90.76%*	*90.76%*	*87.19%*	*87.19%*
Stanza	—	*0*	*92.83%*	*92.83%*	*89.70%*	*89.70%*
Qwen3-4B	Без ограничений	19	93.09%	92.89%	91.30%	91.11%
	С ограничениями	15	93.08%	92.92%	91.29%	91.14%
RuadaptQwen3-4B-Hybrid	Без ограничений	6	93.28%	93.22%	91.52%	91.45%
	С ограничениями	3	93.28%	93.24%	91.51%	91.48%
Qwen3-8B	Без ограничений	11	94.48%	94.37%	92.82%	92.70%
	С ограничениями	2	94.48%	94.46%	92.81%	92.70%
RuadaptQwen3-8B-Hybrid	Без ограничений	0	93.95%	93.95%	92.30%	92.30%
	С ограничениями	0	93.95%	93.95%	92.30%	93.30%

В табл. 3 представлены два вида значений средних значений модифицированных метрик UAS и LAS: на множестве предложений с корректной скобочной последовательностью (пометка корр.) и на множестве всех тестовых предложений (пометка все). И с ограничениями, и без ограничений результаты применения дообученных LLM превосходят результаты наилучших классических нейросетевых синтаксических анализаторов Stanza и DeepPavlov (которые считаются наилучшими в этом классе [23]). При использовании ограничений значения метрик UAS и LAS не были существенно изменены, поскольку относительная доля предложений с изначально некорректно сгенерированным деревом зависимостей невелика.

Для LLM RuadaptQwen3-8B-Hybrid количество некорректных результатов равно 0 даже без использования ограничений генерации. Для всех остальных LLM количество некорректных результатов уменьшилось при использовании ограничений генерации, однако не стало равным 0. При использовании ограничений некорректные результаты появлялись по одной из двух причин: генерации скобочной последовательности, не соответствующей GRCT-дереву, и бесконечной генерации формы слова или типа связи в дереве зависимостей. Статистика по случаям некорректного выполнения синтаксического анализа приведена в табл. 4.

Табл. 4. Причины некорректного завершения синтаксического анализа

Модель	Не GRCT-дерево	Бесконечная генерация
Qwen3-4B	10	5
RuadaptQwen3-4B-Hybrid	1	2
Qwen3-8B	2	0

Причиной появления скобочных последовательностей, которые не могут быть преобразованы в дерево зависимостей, является нарушение количества вершин типа связи и формы слова или нарушение порядка следования этих вершин. Например, для одного из предложений была сгенерирована последовательность `[Остается]`, которая корректна с точки зрения баланса скобок, но не может быть преобразована в дерево зависимостей из-за отсутствия в ней типа связи. Для другого предложения была сгенерирована последовательность `[root[obl…]]`, в которой у вершины типа связи нет соответствующей вершины формы слова.

Причиной бесконечной генерации является продолжение генерации в случае, когда LLM считает нужным завершить ее. Для нескольких предложений без использования системы ограничений LLM генерировала последовательность с дисбалансом открывающих и закрывающих скобок. В случае использования системы ограничений LLM не может завершить генерацию в этом месте, поскольку дисбаланс скобок запрещен. В некоторых случаях после принудительного продолжения генерации LLM начинает генерировать в качестве типа связи или формы слова бесконечную последовательность токенов, например последовательность `orphan:float=0.5,orphan:float=0.5,orphan:float=0.5,…`

в качестве типа связи. Для завершения генерации в таком случае используется ограничение по времени генерации для одного предложения (в экспериментах 10 мин).

В дальнейшем планируется дополнение системы ограничений ограничениями для запрета генерации некорректной GRCT-последовательности и некорректных типов связей и форм слов.

ЗАКЛЮЧЕНИЕ

В работе рассмотрена проблема генерации структурно некорректного ответа большой языковой моделью, дообученной для синтаксического анализа русского языка. Ограничение генерации проведено на основе запрета генерации определенных субтокенов. Для этого разработана система многоэтапных ограничений.

Реализованная система ограничений протестирована на четырех дообученных LLM: двух мультиязычных (Qwen3-4B и Qwen3-8B) и двух адаптированных для русского языка (RuadaptQwen3-4B-Hybrid и RuadaptQwen3-8B-Hybrid). В результате экспериментов установлено, что при применении системы ограничений уменьшилось количество сгенерированных последовательностей со структурной ошибкой.

В будущем предполагается расширить разработанную систему ограничений и добавить ограничения для устранения галлюцинаций при генерации синтаксической структуры предложения (с точки зрения текста токенов дерева зависимостей и типов связи). Кроме того, предусматривается добавление ограничений для проверки порядка следования типов связи и форм слов в скобочной последовательности.

Реализация кода исследования представлена в открытом доступе по ссылке:
https://github.com/Derinhelm/syntax_finetuning/tree/demo_structural_constraints

Благодарности

Выражаю благодарность Наталье Валентиновне Лукашевич за проявленный интерес к исследованию и значимые советы при подготовке статьи.

Исследование выполнено в рамках государственного задания МГУ имени М. В. Ломоносова.

Работа выполнялась с использованием суперкомпьютера «МГУ-270» МГУ имени М. В. Ломоносова.

СПИСОК ЛИТЕРАТУРЫ

1. *Li L., Chen Z., Liao S. et al.* Event Extraction in Complex Sentences Based on Dependency Parsing and Longformer // Proc. of 2024 International Conference on Machine Learning and Intelligent Computing, Wuhan, China. 2024. P. 1–7.

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

3. *Alonso M. A., Gómez-Rodríguez C., Vilares J.* On the use of parsing for named entity recognition // Applied sciences. 2021. Vol. 11 (3). 1090. https://doi.org/10.3390/app11031090

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

5. *Morozov D. et al.* Exploring the Feature Space for Cross-Domain Assessing the Complexity of Russian-Language Texts // 2024 Ivannikov ISPRAS Open Conference, Moscow. 2024. P. 1–8. https://doi.org/10.1109/ISPRAS64596.2024.10899137

6. *Liu T., Sun Y., Wu J. et al.* Unsupervised Paraphrasing under Syntax Knowledge // Proc. of the AAAI Conference on Artificial Intelligence. 2023. P. 13273–13281. https://doi.org/10.1609/aaai.v37i11.26558

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

8. *Chen D., Manning C.D.* A fast and accurate dependency parser using neural networks // Proc. of the Conference on Empirical Methods in Natural Language Processing, Doha, Qatar. 2014. P. 740–750.

https://doi.org/10.1016/j.eswa.2023.11957910.3115/v1/D14-1082

9. *Andor D., Alberti C., Weiss D. et al.* Globally normalized transition-based neural networks // Proc. of the 54th Annual Meeting of the Association for Computational Linguistics (ACL), Berlin, Germany. 2016. P. 2442–2452. https://doi.org/10.1016/j.eswa.2023.11957910.18653/v1/P16-123

10. *Dozat T., Manning C.D.* Deep Biaffine Attention for Neural Dependency Parsing // International Conference on Learning Representations. 2017.

11. *Ezquerro A., Gómez-Rodríguez C., Vilares D.* Better Benchmarking LLMs for Zero-Shot Dependency Parsing // Proc. of the Joint 25th Nordic Conference on Computational Linguistics and 11th Baltic Conference on Human Language Technologies (NoDaLiDa/Baltic-HLT 2025), Tallinn, Estonia. 2025. P. 121–135.

12. *Hromei C. D., Croce D., Basili R.* U-DepPLLaMA: Universal Dependency Parsing via Auto-regressive Large Language Models // IJCoL. Italian Journal of Computational Linguistics. 2024. Vol. 10 (1). https://doi.org/10.1016/j.eswa.2023.11957910.4000/125nm

13. *Lin B. et al.* ChatGPT is a potential zero-shot dependency parser // arXiv:2310.16654. https://doi.org/10.48550/arXiv.2310.16654

14. *Matsuda H., Ma C., Asahara M.* Step-by-step Instructions and a Simple Tabular Output Format Improve the Dependency Parsing Accuracy of LLMs // Proc. of the 18th International Conference on Parsing Technologies (IWPT, SyntaxFest 2025), Ljubljana, Slovenia. Association for Computational Linguistics: 2025. P. 11–19.

15. *Zhou H., Chersoni E., Hsu Y.Y.* Branching Out: Exploration of Chinese Dependency Parsing with Fine-tuned Large Language Models // Conference on Recent Advances in Natural Language Processing (RANLP 2025), Varna, Bulgaria. INCOMA Ltd: 2025. P. 1437–1445.

16. *Tian Y., Xia F., Song Y.* Large language models are no longer shallow parsers // Proc. of the 62nd Annual Meeting of the Association for Computational Linguistics, Thailand. Association for Computational Linguistics: 2024. P. 7131–7142.

17. *Bai X., Wu J., Chen Y., Wang Z., Zhang Y.* Constituency parsing using llms // IEEE Transactions on Audio, Speech and Language Processing IEEE.2025. Vol. 33. P. 3762–3775. https://doi.org/10.1109/TASLPRO.2025.3600867

18. *Ginn M., Palmer A.* LLM Dependency Parsing with In-Context Rules // Proc. of the 1st Joint Workshop on Large Language Models and Structure Modeling, Vienna, Austria. Association for Computational Linguistics. 2025. P. 189–196.

19. *Kim T.* Revisiting the practical effectiveness of constituency parse extraction from pretrained language models // Proc. of the 29th International Conference on Computational Linguistics, Gyeongju, Republic of Korea. International Committee on Computational Linguistics. 2022. P. 5398–5408.

20. *Sun X. et al.* Pushing the Limits of ChatGPT on NLPTasks // arXiv: 2306.09719. 2023. https://doi.org/10.48550/arXiv.2306.09719

21. *Zeman D., Hajic J., Popel M. et al.* CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies // Proc. of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, Vancouver, Canada. 2017. P. 1–19. https://doi.org/10.18653/v1/K17-3001

22. *Tikhomirov M., Chernyshev D.* Facilitating large language model russian adaptation with learned embedding propagation // Journal of Language and Education. 2024. Vol. 10, No. 4 (40). P. 130–145. https://doi.org/10.17323/jle.2024.22224

23. *Шамаева Е.Д.* Сравнение нейросетевых синтаксических анализаторов для русского языка // Компьютерная лингвистика и вычислительные онтологии. 2025. № 1. С. 26–47. https://doi.org/10.17586/3033-5582-2025-9-26-47

CONSTRAINED DECODING FOR STRUCTURAL ERROR SUPPRESSION IN DEPENDENCY TREE GENERATION WITH LARGE LANGUAGE MODELS

E. D. Shamaeva[0009-0002-6233-8958]

Lomonosov Moscow State University, Moscow, Russia

derinhelm@yandex.ru

Abstract

In the field of syntactic parsing, fine-tuning large language models to generate a sentence's syntactic structure in bracket form is a promising approach. Existing fine-tuned models demonstrate high values of syntax parsing metrics; however, in some cases they generate an incorrect bracket sequence (for example, with an unbalanced number of opening and closing parentheses). Therefore, this study aims to develop a

method to reduce the number of incorrectly generated sequences. A system of constraints has been introduced on the set of tokens generated at each stage of the large language model. The implemented method was tested on four fine-tuned models (adapted and non-adapted with 4 and 8 billion parameters) on the SynTagRus dependency tree dataset. For all models, the number of incorrectly generated sequences decreased. However, some sentences were not parsed, because the LLMs generated an excessively long token sequence and were interrupted. The code for this study is publicly available.

***Keywords**: dependency parsing, large language model, fine-tuning, generation constraing, natural language processing.*

REFERENCES

1. *Li L., Chen Z., Liao S. et al.* Event Extraction in Complex Sentences Based on Dependency Parsing and Longformer // Proc. of 2024 International Conference on Machine Learning and Intelligent Computing, Wuhan, China. 2024. P. 1–7.

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

3. *Alonso M. A., Gómez-Rodríguez C., Vilares J.* On the use of parsing for named entity recognition // Applied sciences. 2021. Vol. 11 (3). 1090. https://doi.org/10.3390/app11031090

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

5. *Morozov D. et al.* Exploring the Feature Space for Cross-Domain Assessing the Complexity of Russian-Language Texts // 2024 Ivannikov ISPRAS Open Conference, Moscow. 2024. P. 1–8. https://doi.org/10.1109/ISPRAS64596.2024.10899137

6. *Liu T., Sun Y., Wu J. et al.* Unsupervised Paraphrasing under Syntax Knowledge // Proc. of the AAAI Conference on Artificial Intelligence. 2023. P. 13273–13281. https://doi.org/10.1609/aaai.v37i11.26558

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

8. *Chen D., Manning C.D.* A fast and accurate dependency parser using neural networks // Proc. of the Conference on Empirical Methods in Natural Language Processing, Doha, Qatar. 2014. P. 740–750. https://doi.org/10.1016/j.eswa.2023.11957910.3115/v1/D14-1082

9. *Andor D., Alberti C., Weiss D. et al.* Globally normalized transition-based neural networks // Proc. of the 54th Annual Meeting of the Association for Computational Linguistics (ACL), Berlin, Germany. 2016. P. 2442–2452. https://doi.org/10.1016/j.eswa.2023.11957910.18653/v1/P16-123

10. *Dozat T., Manning C.D.* Deep Biaffine Attention for Neural Dependency Parsing // International Conference on Learning Representations. 2017.

11. *Ezquerro A., Gómez-Rodríguez C., Vilares D.* Better Benchmarking LLMs for Zero-Shot Dependency Parsing // Proc. of the Joint 25th Nordic Conference on Computational Linguistics and 11th Baltic Conference on Human Language Technologies (NoDaLiDa/Baltic-HLT 2025), Tallinn, Estonia. 2025. P. 121–135.

12. *Hromei C. D., Croce D., Basili R.* U-DepPLLaMA: Universal Dependency Parsing via Auto-regressive Large Language Models // IJCoL. Italian Journal of Computational Linguistics. 2024. Vol. 10 (1). https://doi.org/10.1016/j.eswa.2023.11957910.4000/125nm

13. *Lin B. et al.* ChatGPT is a potential zero-shot dependency parser // arXiv:2310.16654. https://doi.org/10.48550/arXiv.2310.16654

14. *Matsuda H., Ma C., Asahara M.* Step-by-step Instructions and a Simple Tabular Output Format Improve the Dependency Parsing Accuracy of LLMs // Proc. of the 18th International Conference on Parsing Technologies (IWPT, SyntaxFest 2025), Ljubljana, Slovenia. Association for Computational Linguistics: 2025. P. 11–19.

15. *Zhou H., Chersoni E., Hsu Y. Y.* Branching Out: Exploration of Chinese Dependency Parsing with Fine-tuned Large Language Models // Conference on Recent Advances in Natural Language Processing (RANLP 2025), Varna, Bulgaria. INCOMA Ltd: 2025. P. 1437–1445.

16. *Tian Y., Xia F., Song Y.* Large language models are no longer shallow parsers // Proc. of the 62nd Annual Meeting of the Association for Computational Linguistics, Thailand. Association for Computational Linguistics: 2024. P. 7131–7142.

17. *Bai X., Wu J., Chen Y., Wang Z., Zhang Y.* Constituency parsing using llms // IEEE Transactions on Audio, Speech and Language Processing IEEE. 2025. Vol. 33. P. 3762–3775. https://doi.org/10.1109/TASLPRO.2025.3600867

18. *Ginn M., Palmer A.* LLM Dependency Parsing with In-Context Rules // Proc. of the 1st Joint Workshop on Large Language Models and Structure Modeling, Vienna, Austria. Association for Computational Linguistics. 2025. P. 189–196.

19. *Kim T.* Revisiting the practical effectiveness of constituency parse extraction from pretrained language models // Proc. of the 29th International Conference on Computational Linguistics, Gyeongju, Republic of Korea. International Committee on Computational Linguistics. 2022. P. 5398–5408.

20. *Sun X. et al.* Pushing the Limits of ChatGPT on NLP Tasks // arXiv: 2306.09719. 2023. https://doi.org/10.48550/arXiv.2306.09719

21. *Zeman D., Hajic J., Popel M. et al.*CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies // Proc. of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, Vancouver, Canada. 2017. P. 1–19. https://doi.org/10.18653/v1/K17-3001

22. *Tikhomirov M., Chernyshev D.* Facilitating large language model Russian adaptation with learned embedding propagation // Journal of Language and Education. 2024. Vol. 10, No.4 (40). P. 130–145. https://doi.org/10.17323/jle.2024.22224

23. *Shamaeva E.D.* Sravnenie neirosetevih sintaksicheskih analyzatorov dlya ruskogo yazika // Komp'uternaya lingvistika i vichislitelnie ontologii. 2025. No. 1. P. 26–47. https://doi.org/10.17586/3033-5582-2025-9-26-47

СВЕДЕНИЯ ОБ АВТОРЕ

ШАМАЕВА Елена Денисовна. В 2022 году закончила факультет вычислительной математики и кибернетики (ВМК) Московского государственного университета (МГУ) имени М. В. Ломоносова и поступила в аспирантуру ВМК МГУ. Работает на факультете ВМК с 2024 г. на кафедре алгоритмических языков. Область научных интересов: синтаксический анализ текстов, большие языковые модели, графовые нейронные сети.

Elena Denisovna SHAMAEVA. In 2022 she graduated from Lomonosov Moscow State University (the Faculty of Computational Mathematics and Cybernetics) and enrolled in the graduate school of the Faculty. Since 2024 she has been working at the Faculty of Computational Mathematics and Cybernetics (Department of Algorithmic Languages). Research interests: syntactic parsing, large language models, graph neural networks.

email: derinhelm@yandex.ru

ORCID 0009-0002-6233-8958

Материал поступил в редакцию 11 апреля 2026 года