Санкт-Петербургский Государственный Университет
Прикладной математики - Процессов Управления

Кафедра Технологии Программирования

Менчуков Павел Александрович

Автодополнение текста на естественном языке с использованием глубокого обучения

Бакалаврская работа

Научный соруководитель:
ст. преп.
Мишенин А. Н.

Научный руководитель:
доцент, кандидат технических наук
Блеканов И. С.

Санкт-Петербург
2022

Оглавление

Введение **3**

Постановка задачи **5**

Обзор литературы **6**

1. Сравнение существующих моделей **9**
1.1. GPT-2 9
1.2. mT5 9
1.3. Дообучение моделей 10
1.4. Сравнение результатов моделей 10
1.4.1. Описание метрик 11
1.4.2. Результаты моделей 13
1.5. Выводы 13

2. Дообучение модели **15**
2.1. Описание датасета 15
2.1.1. Предобработка входных данных 16
2.2. Процесс дообучения 16
2.3. Оценка результатов модели 18
2.3.1. Описание метрик 19
2.3.2. Результаты дообучения 19

3. Дальнейшее улучшение результатов **21**
3.1. Биграммная языковая модель 21
3.2. Комбинированная модель 22
3.3. Результаты комбинированной модели 23

Выводы **24**

Заключение **25**

Список литературы **26**

Введение

В современном мире всё большую значимость приобретает электронный документооборот. Так, в электронном виде могут храниться медицинские данные - о заболеваниях пациента, назначенных процедурах и лекарственных средствах. Корпуса судебных слушаний облегчают юристам поиск похожих дел и прецендентов. Реестры недвижимости позволяют предотвратить мошенничество и спорные ситуации владения имуществом. Это лишь часть примеров - электронный документооборот прочно вошёл в жизнь общества. На то есть несколько причин, одна из которых - удобство. Такой подход не только позволяет обеспечить надёжное и централизованное хранение существующих документов с возможностью быстрого из поиска, но и ускоряет создание и добавление новых. В электронных документах проще оставлять ссылки на связанные тексты, что упрощает навигацию. Ими удобно делиться, делать копии и получать доступ к содержимому вне зависимости от физического расположения читателя.

Тем не менее, несмотря на все свои преимущества и всё удобство, эта система всё ещё содержит простор для оптимизаций. В частности, документы, как правило, строятся по некоторому шаблону, повторяя с некоторыми вариациями не только структуру, но и определённые части текстов. С одной стороны, следование шаблону и использование клишированных фраз не требует творческого участия человека, с другой - усложняет составление документа. Даже обладая релевантным опытом, люди неизбежно тратят какое-то время на подобные участки - как минимум, на выбор и набор нужного шаблона. Такая почти механическая работа является перспективной для автоматизации или, по крайней мере, использования методов, способных ускорить её выполнение человеком.

В то же время, сейчас всё большую популярность приобретают системы, использующие машинное обучение для достижения целей, сложно выразимых алгоритмически. В частности, эта тенденция наблюдается в области обработки естественных языков. Спектр задач, решаемых

такими системами, обширен - в их число входит суммаризация текста, перефразирование, анализ тональности, а также генерация и, в том числе, автодополнение. Скажем, широко известно автодополнение для формальных языков, таких как языки программирования. Одним из наиболее известных примеров систем автодополнения программного кода, использующих машинное обучение, может служить GitHub Copilot [7].

Несмотря на то, что для документов используются естественные языки, они всё же имеют черты, роднящие их с текстами на языках формальных. Так, большинство документов имеют фиксированную форму, строятся по определённому набору правил, содержат одинаковые фрагменты. В свете всего вышеперечисленного кажется разумным использование для их автодополнения тех же методов, которые уже используются для формальных языков.

Постановка задачи

Исходя из сказанного во Введении, требуется упростить написание электронных документов человеком путём сокращения затрат времени на написание присущих данным документам клише, а также конкретных фраз, повторяющихся в рамках одного документа. Обобщить и формализировать эту задачу можно следующим образом: по имеющемуся тексту, представленному в виде набора токенов, требуется предложить наиболее вероятное его продолжение. Таким образом, получаем задачу автодополнения текста. Поскольку в качестве текстов рассматриваются документы - речь идёт о текстах на естественном языке. Для сужения класса возможных подходов к решению будем рассматривать только способы автодополнения текста, основанные на глубоком обучении. Кроме того, для определённости в качестве рассматриваемого типа документов определим юридические тексты, а именно - решения судов. Язык документов - русский.
Данная работа состоит из следующих этапов:

1. Обзор литературы затрагивающей родственные вопросы.
2. Выбор нескольких нейросетевых моделей.
3. Сравнение выбранных моделей и выбор наиболее подходящей.
4. Создание датасета, пригодного для дообучения выбранной модели.
5. Дообучение выбранной модели.
6. Обзор результатов дообученной модели.
7. Дальнейшее улучшение результатов.

Обзор литературы

Вопросы генерации текстовых данных рассматривались в множестве работ. Так, например, авторы [16] в своей работе рассматривают различные нейросетевые модели, предназначенные для генерации текстов - от классических рекуррентных нейронных сетей (RNN) до значительно более необычных состязательных генеративных (GAN). Данная работа ценна тем, что она может быть использована в качестве обзорной статьи, которая не только затрагивает различные подходы к рассматриваемому вопросу, но и касается известных проблем работы с текстом посредством нейросетей - таких как, например, затухание градиентов. Кроме того, в этой же статье упоминается трансферное обучение, более подробно описанное в [18] и играющее значительную роль в современной обработке естественных языков.

Также существенное влияние на данную область оказало появление в 2017 году архитектуры Transformer, представленной в [1]. Главным описанным отличием новой архитектуры от существовавших прежде можно считать то, что она базируется исключительно на механизме внимания, не используя рекуррентность.

После появления данной архитектуры важным этапом стала демонстрация способности нейросетевых моделей осваивать различные задачи обработки естественного языка без непосредственного вмешательства, путём обучения на огромных массивах данных. Подобный результат был продемонстрирован с использованием датасета WebText, состоявшего из более чем миллиона веб-страниц. Данный результат был описан исследователям из OpenAI в работе [11], помимо этого представляющей модель GPT-2, продемонстрировавшую потрясающие для своего времени результаты. Так, без дополнительного обучения она смогла превзойти все существовавшие на тот момент модели в тестах на семи из восьми использованных для её проверки датасетах. Кроме того, подобный подход - обучение на больших массивах данных - сделал особенно востребованным упоминавшееся ранее трансферное обучение. На данный момент оно является одним из наиболее перспективных подхо-

дов к обучению моделей, задействованных в сфере обработки и моделирования естественных языков.

Кроме GPT-2 можно также отметить её улучшенную версию GPT-3 ([10]), а также модели ELMo ([4]) и BERT ([2]). Последняя интересна тем, что является альтернативным взглядом на развитие модели GPT - в то время, как создатели GPT-2 пошли путём увеличения количества параметров модели, в BERT появилось качественное изменение, а именно - двунаправленность, позволяющая модели учитывать не только предшествующие, но и последующие токены, что сделало возможным кроме продолжения текстов ещё и заполнение пропусков.

Помимо фундаментальных результатов, существует также немало исследований, задачей которых ставилось практическое применение нейросетевых подходов к обработке естественного языка в различных областях. В частности, наибольший интерес в рамках данной работы представляют исследования генерации текста нейросетевыми моделями.

Так, для английского языка существуют работы, направленные на улучшение генерации в целом, такое как увеличение длительности сохранения локального контекста и улучшение способности модели планировать большие отрывки теста ([8]) или ускорение дальнейшего дообучения модели для различных доменов ([13]). Кроме того, существуют и работы, ориентированные на конкретные области применения - автодополнение программного кода ([20], [17]), генерация медицинских ([24]) и юридических документов ([14]). Однако несмотря на то, что задачи последней работы определённо близки к задачам данной, следует отметить, что в многоязычной схеме использования большинство моделей на данный момент всё ещё показывают значительно менее хорошие результаты, и несмотря на существование методов использования моделей, обученных для других языков, один из которых, заключающийся в изменении эмбеддингов слов с минимальным дообучением остальной модели, описывается в [21], можно ожидать, что модели, изначально обученные с использованием целевого языка, продемонстрируют лучшие результаты.

Вопросы генерации русскоязычных текстов в рамках ограниченных

доменов также становились предметом исследований ([6], [19]), однако эти исследования заметно более малочисленны и, кроме того, не касаются вопроса генерации и/или автодополнения документов в домене юриспруденции. Таким образом, эта тема требует дополнительного исследования, чему и посвящена данная работа.

1. Сравнение существующих моделей

Поскольку поставленная задача состоит в автодополнении текста, то есть в генерации нескольких последующих токенов на основании набора уже имеющихся, естественным решением является выбор подходов семейства Seq2Seq, поскольку они нацелены на построение одной последовательности токенов по другой. На данный момент наилучшие результаты в области обработки естественного языка показывают модели из семейства Трансформеров, поэтому рассматриваться будут модели именно этого семейства.
В рамках данной работы рассматриваются модели двух архитектур - GPT-2, предложенной в [11], и mT5, предложенной в [22].

1.1. GPT-2

GPT-2 - модель семейства Трансформеров, предложенная в 2019 году исследователями из OpenAL. Несмотря на то, что модель базируется на архитектуре Transformer, от классических Трансформеров её отличает использование только блоков декодера вместо совместного использования блоков декодера и энкодера. Эти блоки включают в себя механизм маскированного самовнимания, который позволяет учитывать контекст лишь до текущего токена. Это делает эту модель архитектурно нацеленной на генерацию продолжений текстов, что соответствует задаче, поставленной в данной работе.
В рамках данной работы применяется модель ruGPT-3, предобученная для русского языка ([27]), в частности - её medium версия.

1.2. mT5

mT5 - многоязыковая версия модели T5, предложенной в 2020 году исследователями из Google Research ([5]). T5, так же как и GPT-2, является представителем семейства Трансформеров, однако в ней уже используются как блоки декодера, так и блоки энкодера. Оригинальная T5 является мультизадачной, позволяя решать 24 различные задачи,

включая задачи:

- суммаризации текста;
- ответа на вопросы по тексту;
- классификации текста;
- перевода.

Однако многоязыковая модель mT5 обучалась исключительно для заполнения пропусков в тексте. Это не вполне отвечает поставленной в данной работе задаче, но эта проблема может быть устранена в ходе дообучения модели на сформированных особым образом обучающих данных.
В рамках данной работы применяется модель rut5-base - уменьшенная версия модели mT5, из словаря которой исключены все токены, кроме тех, которые относятся к русскому и английскому языкам. Процесс такой адаптации описан в [3].

1.3. Дообучение моделей

Для привнесения контекста задачи обе модели дообучались на данных из датасета, описанного в Разделе 2.1. Однако поскольку задачей данного раздела является лишь сравнение результатов моделей для выбора более подходящей, а не получение полностью работоспособной модели, решающей глобальную задачу автодополнения текстов из заданного домена наиболее полным образом и показывающей наилучшие результаты в соответствии с используемыми метриками, для дообучения использовалось лишь малое подмножество всего датасета, а именно - 10 тыс. текстов. Кроме того, по тем же причинам дообучение производилось в течение двух эпох.

1.4. Сравнение результатов моделей

В данном разделе сравнивается эффективность рассматриваемых моделей в задаче генерации текстов, принадлежащих указанному до-

мену. Несмотря на то, что глобальной задачей данной работы является автодополнение текста, а не его генерация, такой подход позволяет получить представление о пригодности модели и для автогенерации, поскольку эти задачи родственны.

Оценка производилась с использованием 1000 текстов из упомянутого выше датасета в качестве эталонных. Ни один из используемых текстов не входил в подмножество, использованное для дообучения моделей.

1.4.1. Описание метрик

Для оценки качества генерируемых моделью продолжений текста использовалась группа метрик ROUGE. Эта группа, предложенная в [15], состоит из концептуально схожих метрик, среди которых, в частности, можно выделить ROUGE-N, ROUGE-L и ROUGE-Lsum.

Идея метрики ROUGE-N заключается в сравнении количества n-грамм в сгенерированной строке, которые совпадают с n-граммами в эталонной строке, с общим количеством n-грамм в сгенерированной строке. Обозначая за n_i i-ую n-грамму сгенерированной строки, за G - сгенерированную строку, за R - эталонную строку, а за $Count_{n_i}(K)$ - количество вхождений n-граммы n_i в множество K, формулу для метрики ROUGE-N можно записать в следующем виде:

$$ROUGE_N = \frac{\sum\limits_{n_i \in G} \min(Count_{n_i}(G), Count_{n_i}(R))}{\sum\limits_{n_i \in G} Count_{n_i}(G)}$$

В частности, в данной работе использовались метрики ROUGE-1 и ROUGE-2.

Идея метрики ROUGE-L заключается в вычислении F-меры по отношению к наибольшей общей подпоследовательности двух строк. Введём понятие подпоследовательности: последовательность $Y = [y_1; y_2; ...; y_m]$ называется подпоследовательностью последовательности $X = [x_1; x_2; ...; x_n]$, если существует строго возрастающая последовательность $[i_1; i_2; ...; i_k]$ индексов элементов последовательности X такая, что $\forall j = 1; ...; k$:

$x_{i_j} = y_j$. Понятие наибольшей общей подпоследовательности тогда тривиально. Обозначая за $LCS(A, B)$ длину наибольшей общей подпоследовательности строк A и B, за n длину строки G, а за m - длину строки R, формулу для метрики ROUGE-L можно записать в следующем виде:

$$P = \frac{LCS(G,R)}{n}$$
$$S = \frac{LCS(G,R)}{m}$$
$$\beta = \frac{P}{S}$$
$$ROUGE_L = \frac{(1+\beta^2)PS}{S+\beta^2 P}$$

Данная метрика вычисляет F-меру на уровне предложений, т.е. под строками G и R в формулах выше понимаются отдельные предложения из сгенерированной и эталонной строк соответственно.

Метрика ROUGE-Lsum является обобщением метрики ROUGE-L на уровень строк вместо предложений. Пусть строка G состоит из предложений g_i, а строка R состоит из предложений r_i. Обозначим за $LCS_\cup(g_i, R)$ длину объединения наибольших общих подпоследовательностей предложения g_i с каждым из предложений r_i. Тогда формулу для метрики ROUGE-Lsum можно записать в следующем виде:

$$P_{sum} = \frac{\sum\limits_{g_i \in G} LCS_\cup(g_i,R)}{n}$$
$$S_{sum} = \frac{\sum\limits_{g_i \in G} LCS_\cup(g_i,R)}{m}$$
$$\beta_{sum} = \frac{P_{sum}}{S_{sum}}$$
$$ROUGE_{Lsum} = \frac{(1+\beta_{sum}^2)P_{sum}S_{sum}}{S_{sum}+\beta_{sum}^2 P_{sum}}$$

Следует отметить, что метрики группы ROUGE предназначены для оценки результатов в задачах генерации текста и плохо подходят для задач автодополнения текста, поскольку для них характерна генерация малого количества токенов на каждой итерации. Так, например, метрики ROUGE-N очевидным образом неприменимы, если число генерируемых токенов меньше N. Однако поскольку в данном разделе эффективности моделей предварительно сравнивается на задаче генерации текста, использование метрик группы ROUGE оправданно. Мет-

рики, лучше подходящие для использования с задачей автодополнения текста, предложены в Разделе 2.3.1.

1.4.2. Результаты моделей

Ниже приведена таблица, содержащая значения метрик для сравниваемых моделей.

Архитектура модели	ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-Lsum
GPT-2	0,6142	0,3134	0,5186	0,5987
mT5	0,5940	0,2867	0,4952	0,5651

Как можно видеть, модель архитектуры GPT-2 превзошла своего конкурента в данной задаче.

1.5. Выводы

Модель на основе архитектуры GPT-2 показала лучшие результаты, чем модель на основе архитектуры mT5. Это может быть следствием нескольких причин.

1. Лучшее предобучение. Модель ruGPT-3 проектировалась и обучалась целенаправленно для данных на русском языке, в то время как модель ruT5-base была получена из модели, обученной на многоязыковом корпусе, путём исключения токенов из её словаря и соответствующих строк из матриц входных и выходных эмбеддингов.

2. Меньший размер модели. Поскольку обучение было прекращено после фиксированного количества итераций, существует вероятность, что этого времени обучения не хватило, чтобы модель ruT5-base, обладающая большим количеством обучаемых параметров, чем модель ruGPT-3, смогла обучиться достаточно, чтобы обойти своего конкурента.

3. Более подходящая для задачи архитектура. В то время, как ruT5-base изначально предназначалась для заполнения пропусков в тексте, ruGPT-3 изначально позиционировалась как модель, нацеленая на генерацию токенов, следующих за данными. Заполнение пропусков подразумевает, в том числе, возможность наличия контекста после пропуска, а в решаемой задаче этот контекст был недоступен, что могло сыграть свою роль в превосходстве модели, этот контекст не использующей.

2. Дообучение модели

Несмотря на то, что выбранная модель изначально обучена для работы с текстами на русском языке, что заведомо даёт ей преимущество по сравнению с оригинальной GPT-2, обученной с использованием текстов преимущественно на английском языке, задача привнесения контекста рассматриваемых документов всё ещё остаётся актуальной, поскольку исходный массив данных, на котором обучалась модель, состоит из русской литературы, различных интернет-сайтов, а также датасета Omnia Russica, и не является репрезентативным относительно судебных решений. Задачи подобного типа (имеется модель, обученная на данных общего характера, которую требуется использовать в рамках конкретного частного домена) относятся к задачам трансферного обучения, основные принципы которого описаны в [12], и решаются путём дообучения модели на данных принадлежащих рассматриваемому домену - то есть несущих в себе контекст и специфику конкретной области.

2.1. Описание датасета

Как известно, результаты нейросетевых моделей существенно зависят от данных использованных для их обучения. Поэтому выбор обучающего датасета играет важную роль в достижении поставленной цели. Поскольку, как было указано выше, задача автодополнения судебных решений на русском языке до сих пор не становилась предметом масштабных исследований, подходящего датасета в общем доступе ожидаемо не оказалось. Тем не менее, публичность судебных заседаний и практика обнародования судебных решений позволила собрать такой датасет на основании общедоступных данных. Так, например, судебные решения публикуются на сайтах [29], [26], [28] и [25]. В общей сложности для составления датасета было собрано более 700 тыс. решений общим объёмом 9 Гб в текстовом формате.

2.1.1. Предобработка входных данных

Исходные данные кроме непосредственно текста решения содержали дополнительные метаданные, как то: название судебного органа, вынесшего решение, заголовок рассматриваемого дела, различные даты (подачи заявления, судебных заседаний, вынесения решения и т.д.), список фигурантов дела, представители сторон и прочее. Очевидно, что значительная часть этих данных бесполезна для решения поставленной задачи, а некоторые из них, кроме того, дублируются непосредственно в тексте решения, поэтому все они были исключены из результирующего набора данных. Несмотря на то, что в исходном виде тексты судебных решений были представлены в формате гипертекста и содержали разметку, потенциально способную предоставить большее количество информации и расширить контекст, при составлении датасета было решено очистить тексты от разметки, поскольку в данной задаче объём предоставляемой ей дополнительной информации был сочтён не оправдывающим увеличение объёма данных и, соответственно, замедление и большую ресурсоёмкость процесса обучения. Кроме того, все тексты были сконкатенированы с использованием разделяющего токена <|endoftext|>, интерпретируемого моделью как ограничитель логически связанного фрагмента и используемого в дальнейшем в качестве индикатора завершения генерации.

2.2. Процесс дообучения

В целях дообучения перед непосредственной подачей на вход модели данные были дополнительно обработаны и приведены в удобный для обучения вид путём нескольких последовательных операций:

1. Токенизация - текстовые данные были разбиты на токены и каждому токену было поставлено в соответствие целочисленное значение.

2. Каждый набор токенов, соответствующий отдельному тексту, был разделён на последовательности длиной 128 токенов, формируя

таким образом набор векторов указанной длины. Поскольку по очевидным причинам кратность длины каждого текста значению 128 не гарантируется, последний из векторов набора может иметь меньшее количество элементов. Чтобы избежать таких ситуаций, последний вектор дополнялся разделительными токенами <|endoftext|> до нужной длины. Таким образом, указанный токен также играл роль токена выравнивания. Такое разбиение основано, во-первых, на технических ограничениях, связанных с увеличивающимися с ростом длины последовательности требованиями к доступному объёму видеопамяти, а во-вторых - на эмпирическом предположении о том, что контекст, на который ссылаются части текста, является для этого текста либо глобальным, либо локальным, который находится близко к ссылающейся на него части. Поскольку выделение из текста глобального контекста является отдельной задачей, выходящей за рамки данной работы, было принято решение ограничиться локальным контекстом.

3. Полученные на предыдущем шаге наборы векторов объединялись и разбивались на батчи размером 32.

Модель обучалась с использованием оптимизатора Adam [9] с коэффициентом скорости обучения $5e-05$ в течение пяти эпох. Как можно видеть на графике значений функции ошибки на тренировочном и тестовом множествах, представленном на Рисунке 1, после значения в 330-340 тыс. итераций, примерно соответствующего концу четвёртой эпохи, наблюдается картина, типичная для переобучения: колебания и даже рост ошибки для тестового множества при относительно устойчивом снижении ошибки для тренировочного множества. Из этого можно сделать вывод, что к указанному моменту модель достигла предела своего обучения на используемых данных с используемыми гиперпараметрами, поэтому для получения окончательных результатов был использован набор весов, сохранённый после окончания четвёртой эпохи. Однако для процесса расчёта метрик, описанного в Разделе 2.3.1, также использовались веса и с более поздних этапов обучения.

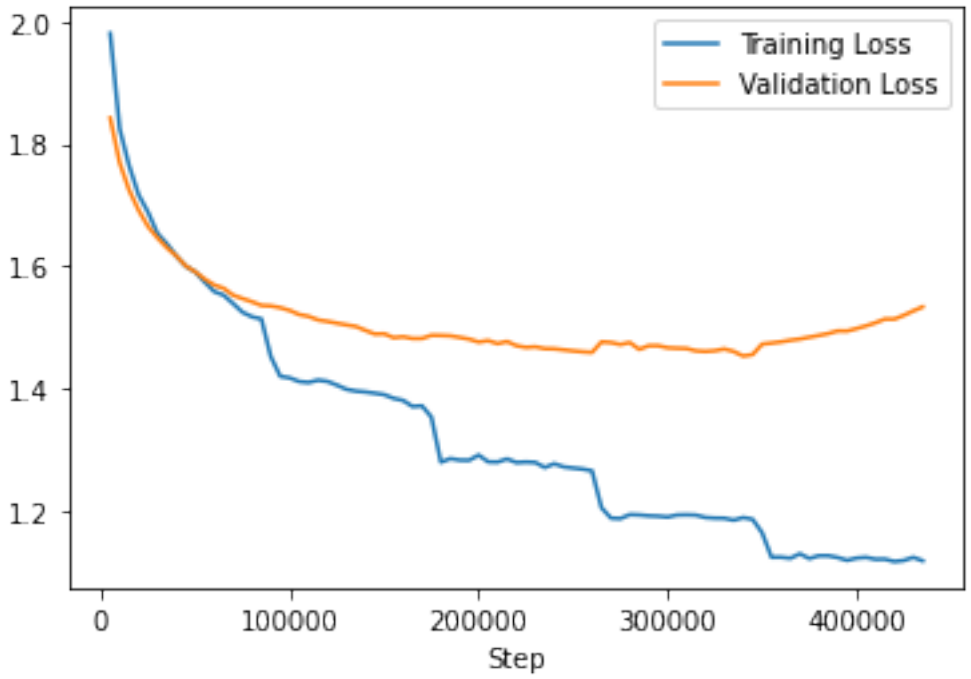

Рис. 1: Графики изменения значений ошибки в течение обучения

Всё описанное выше было реализовано в виде программного кода на языке Python версии 3.9 с использованием библиотек Huggingface Transformers и PyTorch версии 1.10.0. Дообучение производилось на машине, оснащённой GPU NVIDIA RTX 3080 с 10 Гб видеопамяти и 8704 ядрами CUDA с использованием технологии NVIDIA CUDA.

2.3. Оценка результатов модели

После окончания дообучения несколько наборов сохранённых весов использовались для расчёта метрик, позволяющих сравнить качество делаемых моделью предсказаний. В качестве начальных строк использовались фрагменты документов, являющихся частью описанного в Разделе 2.1 датасета, но не присутствовавших ни в тренировочном, ни в тестовом множестве. Для вычисления значений метрик использовалась 1000 таких документов.

2.3.1. Описание метрик

Для задач автодополнения текста существенную роль играет точность предсказания следующих нескольких токенов, поэтому в качестве метрики качества можно использовать вероятность правильно предсказать следующие n токенов. По очевидным причинам величину n следует выбирать не слишком большой, поскольку в противном случае предыдущие предсказания модели всё сильнее будет влиять на точность последующих, что приведёт к существенному снижению вероятности верного предсказания. В рамках данной работы для оценки качества модели использовались метрики со значениями $n = 1$ и $n = 2$ (в дальнейшем обозначаются как $prob_1$ и $prob_2$ соответственно). Ещё одной возможной метрикой является ранг верного продолжения, то есть позиция эталонного токена в наборе предложенных моделью, отсортированном по убыванию вероятности (в дальнейшем обозначается как $rank$). Значения указанных метрик были последовательно рассчитаны для всех подстрок, после чего агрегированы. В случае с вероятностями в качестве агрегационной функции использовалось среднее арифметическое, а в случае с рангом - 0.75 квантиль. С учётом этого итоговое значение метрики $rank$ следует трактовать как ранг эталонного продолжения в не менее, чем $\frac{3}{4}$ случаев.

2.3.2. Результаты дообучения

Для вычисления значений метрик использовались веса модели, сохранённые на итерациях с шагом 50 тыс., а также веса после 340 тыс. итераций, как наилучшее состояние модели до возникновения переобучения (конец четвёртой эпохи), и 435 тыс. итераций, как состояние модели в конце процесса обучения (конец пятой эпохи). Результаты вычислений сведены в таблицу ниже:

Номер итерации	$prob_1$	$prob_2$	$rank$
0	0,3550	0,1168	12
50 тыс.	0,3742	0,1361	10
100 тыс.	0,3871	0,1398	9
150 тыс.	0,3980	0,1496	8
200 тыс.	0,4051	0,1572	7
250 тыс.	0,4075	0,1601	7
300 тыс.	0,4362	0,1755	6
340 тыс.	0,4403	0,1881	5
350 тыс.	0,4403	0,1832	5
400 тыс.	0,4384	0,1763	5
435 тыс.	0,4361	0,1721	6

Наблюдаемые в таблице значения подтверждают высказанное ранее предположение о том, что наиболее качественно модель была обучена в момент, близкий к концу четвёртой эпохи, т.е. к итерации 340 тыс.

3. Дальнейшее улучшение результатов

Несмотря на то, что нейросетевая модель показала хорошие результаты и обеспечила достаточно высокую точность предсказаний, её результаты могут быть улучшены. Так, в частности, несмотря на то, что эта модель предсказывает следующий токен с использованием информации о токенах, ему предшествующих, она недостаточно учитывает контекст документа. Например, каждое судебное решение содержит свои, характерные именно для него словосочетания. Примером такого словосочетания может служить "истец <фамилия>", где <фамилия> специфична для данного конкретного документа. Учёт таких словосочетаний при взвешивании вариантов следующего токена для нейросетевой модели может быть затруднителен.

Решением этой проблемы может стать добавление в схему механизма, учитывающего исключительно контекст документа, предшествующего генерируемому токену. Комбинируя тем или иным образом предсказания нейросетевой модели и этого механизма можно добиться повышения точности предсказаний.

3.1. Биграммная языковая модель

Для учёта контекста конкретного текста можно использовать биграммную языковую модель. Она моделирует текст исходя из предположения, что вероятность каждого последующего токена зависит лишь от вероятности предыдущего. Таким образом, имея токен w_i, можно определить вероятность того, что за ним последует некоторый токен w_{i+1} как $p(w_i|w_{i+1}) = \frac{p(w_i,w_{i+1})}{p(w_i)}$. При этом обозначая за $Count(a)$ количество последовательностей токенов a в тексте, а за l - длину текста (в токенах), можно записать $p(w_i, w_{i+1}) = \frac{Count(w_i w_{i+1})}{l}$ и $p(w_i) = \frac{Count(w_i)}{l}$. Тогда для вероятности следующего токена w_{i+1} при заданном предыдущем w_i окончательно имеем

$$p(w_i|w_{i+1}) = \frac{Count(w_i w_{i+1})}{Count(w_i)}$$

Биграммная модель является частным случаем n-граммной языковой модели, описанной в [23].

3.2. Комбинированная модель

Результирующие данные нейросетевой модели можно интерпретировать как распределение вероятностей следующего токена с учётом последовательности предыдущих. Биграммная модель также предоставляет вероятности для каждого варианта следующего токена. Тогда в качестве итогового распределения вероятности можно использовать линейную комбинацию этих двух распределений, предоставленных моделями. Очевидно, что для того, чтобы линейную комбинацию распределений вероятностей можно также было рассматривать как распределение вероятностей, необходимо, чтобы все коэффициенты этой комбинации были неотрицательными, а их сумма была равна единице. Другими словами, комбинация должна быть выпуклой. Коэффициенты этой выпуклой комбинации для уже обученной нейросетевой модели можно искать оптимизационными методами.

Таким образом, при добавлении биграммной модели в схему автодополнения, можно прийти к следующему алгоритму:

1. На основе уже набранной части текста построить биграммную модель.

2. Используя нейросетевую модель получить распределение вероятностей для следующего токена на основании последовательности предшествующих.

3. Используя биграммную модель получить распределение вероятностей для следующего токена на основании предыдущего.

4. Составить выпуклую комбинацию двух распределений и трактуя её как новое распределение выбрать предлагаемый следующий токен.

5. Получить следующий токен (пользователь может или согласиться с предложенным, или ввести свой).

6. Добавить следующий токен в биграммную модель, обновив вероятности.

7. Повторить шаги 2-7.

3.3. Результаты комбинированной модели

В таблице ниже приведены значения метрик, описанных в Разделе 2.3.1, для комбинированной модели, сочетающей нейросетевую и биграммную, в сравнении с результатами отдельно нейросетевой модели. Метрики были рассчитаны на том же наборе данных, что и в Разделе 2.3.

Тип модели	$prob_1$	$prob_2$	$rank$
Нейросетевая	0,4403	0,1881	5
Комбинированная	0,4514	0,1927	5

Как видно из приведённых выше данных, предположение о возможном улучшении результатов нейросетевой модели путём учёта контекста конкретного документа подтвердилось. Использование даже простой биграммной модели положительно влияет на вероятность предложения верного варианта дополнения текста.

Выводы

Подводя итог, можно сделать следующие выводы.

1. Наилучшие результаты в задачах генерации и автодополнения текстов на естественных языках показывают нейросетевые модели Seq2Seq-архитектуры, в первую очередь - принадлежащие семейству Transformer.

2. Модели семейства Transformer пригодны для использования их с целью автодополнения текстов документов на русском языке.

3. Из двух исследованных моделей (ruGPT-3 и ruT5-base) первая показывает лучшие результаты в решении поставленной задачи.

4. После дообучения на релевантном задаче наборе документов модель ruGPT-3 показывает хорошие результаты и может быть использована в целях автодополнения клишированных текстов юридической направленности на русском языке и решений судов в частности.

5. Результаты работы модели могут быть улучшены путём учёта контекста конкретного документа, что было реализовано посредством комбинирования предсказаний нейросетевой модели с предсказаниями биграммной языковой модели, построенной на предшествующей части документа.

Заключение

В данной работе был сформулирован и достигнут ряд целей. Был проведён обзор литературы, так или иначе имеющей отношение к решению задач NLP методами машинного обучения и в частности - задач генерации и автодополнения текста на естественном языке. На основании данного обзора была выбрана архитектура, наилучшим образом подходящая для решения выдвинутой задачи, а именно - Transfomer. Кроме того, был произведён сравнительный анализ двух предобученных моделей выбранной архитектуры - ruGPT-3 и ruT5-base, сделаны выводы и выдвинуты гипотезы, объясняющие результаты. Также с использованием открытых источников был собран датасет решений российских судов. На данных из этого датасета была дообучена модель ruGPT-3. Были выбраны метрики, которые позволили оценить качество итоговой модели. Кроме того, был предложен и реализован вариант улучшения результатов модели путём учёта контекста конкретного документа с помощью использования биграммной языковой модели совместно с нейросетевой. Таким образом, поставленные изначально задачи были выполнены в полном объёме.

Список литературы

[1] Vaswani Ashish, Shazeer Noam, Parmar Niki, Uszkoreit Jakob, Jones Llion, Gomez Aidan N, Kaiser Ł ukasz, and Polosukhin Illia. Attention is All you Need // Advances in Neural Information Processing Systems / ed. by Guyon I., Luxburg U. Von, Bengio S., Wallach H., Fergus R., Vishwanathan S., and Garnett R. — Curran Associates, Inc. — 2017. — Vol. 30. — Access mode: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[2] Devlin Jacob, Chang Ming-Wei, Lee Kenton, and Toutanova Kristina. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. — 2018. — Access mode: https://arxiv.org/abs/1810.04805.

[3] Dale David. — How to adapt a multilingual T5 model for a single language : 2021. — Access mode: https://towardsdatascience.com/how-to-adapt-a-multilingual-t5-model-for-a-single-language-b9f9

[4] Peters Matthew E., Neumann Mark, Iyyer Mohit, Gardner Matt, Clark Christopher, Lee Kenton, and Zettlemoyer Luke. Deep contextualized word representations. — 2018. — Access mode: https://arxiv.org/abs/1802.05365.

[5] Raffel Colin, Shazeer Noam, Roberts Adam, Lee Katherine, Narang Sharan, Matena Michael, Zhou Yanqi, Li Wei, and Liu Peter J. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. — 2020. — 1910.10683.

[6] Alexandr Nikolich, Irina Osliakova, Tatyana Kudinova, Inessa Kappusheva, and Arina Puchkova. Fine-Tuning GPT-3 for Russian Text Summarization // Data Science and Intelligent Systems / ed. by Silhavy Radek, Silhavy Petr, and Prokopova Zdenka. — Cham : Springer International Publishing. — 2021. — P. 748–757.

[7] GitHub Copilot. Your AI pair programmer. — 2021. — Access mode: https://copilot.github.com/.

[8] Kang Dongyeop and Hovy Eduard. Plan ahead: Self-Supervised Text Planning for Paragraph Completion Task. — 2020. — Access mode: https://arxiv.org/abs/2010.05141.

[9] Kingma Diederik P. and Ba Jimmy. Adam: A Method for Stochastic Optimization. — 2014. — 1412.6980.

[10] Brown Tom, Mann Benjamin, Ryder Nick, Subbiah Melanie, Kaplan Jared D, Dhariwal Prafulla, Neelakantan Arvind, Shyam Pranav, Sastry Girish, Askell Amanda, Agarwal Sandhini, Herbert-Voss Ariel, Krueger Gretchen, Henighan Tom, Child Rewon, Ramesh Aditya, Ziegler Daniel, Wu Jeffrey, Winter Clemens, Hesse Chris, Chen Mark, Sigler Eric, Litwin Mateusz, Gray Scott, Chess Benjamin, Clark Jack, Berner Christopher, McCandlish Sam, Radford Alec, Sutskever Ilya, and Amodei Dario. Language Models are Few-Shot Learners // Advances in Neural Information Processing Systems / ed. by Larochelle H., Ranzato M., Hadsell R., Balcan M.F., and Lin H. — Curran Associates, Inc. — 2020. — Vol. 33. — P. 1877–1901. — Access mode: https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[11] Radford Alec, Wu Jeffrey, Child Rewon, Luan David, Amodei Dario, and Sutskever Ilya. Language Models are Unsupervised Multitask Learners. — 2019. — Access mode: https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

[12] Golovanov Sergey, Kurbanov Rauf, Nikolenko Sergey, Truskovskyi Kyryl, Tselousov Alexander, and Wolf Thomas. Large-Scale Transfer Learning for Natural Language Generation // Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. — Florence, Italy : Association for

Computational Linguistics. — 2019. — July. — P. 6053–6058. — Access mode: https://aclanthology.org/P19-1608.

[13] Lee Dong-Ho, Hu Zhiqiang, and Lee Roy Ka-Wei. Improving Text Auto-Completion with Next Phrase Prediction. — 2021. — Access mode: https://arxiv.org/abs/2109.07067.

[14] Peric Lazar, Mijic Stefan, Stammbach Dominik, and Ash Elliott. Legal Language Modeling with Transformers // Proceedings of the Fourth Workshop on Automated Semantic Analysis of Information in Legal Text (ASAIL 2020) held online in conjunction with te 33rd International Conference on Legal Knowledge and Information Systems (JURIX 2020) December 9, 2020 / ed. by Ashley Kevin D., Atkinson Katie, Branting L. Karl, Francesconi Enrico, Grabmair Matthias, Walker Vern R., and Waltl Bernhard. — s.l. : CEUR-WS. — 2020-12. — Vol. 2764. — 4th Workshop on Automated Semantic Analysis of Information in Legal Text (ASAIL 2020); Conference Location: online; Conference Date: December 9, 2020; Due to the Coronavirus (COVID-19) the conference was conducted virtually.

[15] Lin Chin-Yew. ROUGE: A Package for Automatic Evaluation of Summaries. — 2004. — Access mode: https://aclanthology.org/W04-1013.pdf.

[16] Lu Sidi, Zhu Yaoming, Zhang Weinan, Wang Jun, and Yu Yong. Neural Text Generation: Past, Present and Beyond. — 2018. — Access mode: https://arxiv.org/abs/1803.07133.

[17] Paik Incheon and Wang Jun-Wei. Improving Text-to-Code Generation with Features of Code Graph on GPT-2 // Electronics. — 2021. — Vol. 10, no. 21. — Access mode: https://www.mdpi.com/2079-9292/10/21/2706.

[18] Pan Sinno Jialin and Yang Qiang. A Survey on Transfer Learning //

IEEE Transactions on Knowledge and Data Engineering. — 2010. — Vol. 22, no. 10. — P. 1345–1359.

[19] Shatalov O. Ryabova N. Towards Russian Text Generation Problem Using OpenAI's GPT-2. — 2021. — Access mode: https://openarchive.nure.ua/handle/document/19040.

[20] Sobania Dominik, Schweim Dirk, and Rothlauf Franz. A Comprehensive Survey on Program Synthesis with Evolutionary Algorithms // IEEE Transactions on Evolutionary Computation. — 2022. — P. 1–1.

[21] de Vries Wietse and Nissim Malvina. As Good as New. How to Successfully Recycle English GPT-2 to Make Models for Other Languages // Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021. — Association for Computational Linguistics. — 2021. — Access mode: https://doi.org/10.18653%2Fv1%2F2021.findings-acl.74.

[22] Xue Linting, Constant Noah, Roberts Adam, Kale Mihir, Al-Rfou Rami, Siddhant Aditya, Barua Aditya, and Raffel Colin. mT5: A massively multilingual pre-trained text-to-text transformer. — 2021. — 2010.11934.

[23] Brown Peter F, Cocke John, Della Pietra Stephen A, Della Pietra Vincent J, Jelinek Frederick, Lafferty John, Mercer Robert L, and Roossin Paul S. A statistical approach to machine translation // Computational linguistics. — 1990. — Vol. 16, no. 2. — P. 79–85.

[24] von Davier Matthias. Training Optimus Prime, M.D.: Generating Medical Certification Items by Fine-Tuning OpenAI's gpt2 Transformer Model. — 2019. — Access mode: https://arxiv.org/abs/1908.08594.

[25] Архив судебных решений. — 2022. — Access mode: https://sudrf.cntd.ru/rospravo/.

[26] ГАС РФ "Правосудие". — Access mode: https://sudrf.ru/index.php?id=300.

[27] Марков Сергей. Сбер выложил русскоязычную модель GPT-3 Large с 760 миллионами параметров в открытый доступ. — 2020. — Access mode: https://habr.com/ru/company/sberbank/blog/524522/.

[28] Судебные и нормативные акты РФ. — 2022. — Access mode: https://sudact.ru/regular/.

[29] Судебные решения РФ. — Access mode: http://судебныерешения.рф/.