DOI 10.37882/2223-2966.2025.02-2.22

АБЛЯЦИОННОЕ ИССЛЕДОВАНИЕ МОДЕЛИ ИЗВЛЕЧЕНИЯ РЕЛЯЦИОННЫХ ТРОЕК RESC

AN ABLATIVE STUDY OF THE RELATIONAL TRIPLE EXTRACTION MODEL RESC

A. Kuzmenko
V. Kireev

Summary. Knowledge graphs are an important tool for improving recommendation systems, information retrieval systems, and question-and-answer systems. One of the ways to build them based on the corpus of natural language texts is to extract relational triples — entities and relationships between them. This article explores the RISC model previously proposed by the authors, based on a BERT-like encoder and allowing to solve the problem of extracting triples with high performance. The authors conducted a study of the ablation of the RESC model on a key set of NYT 11 texts. The optimal architecture parameters were determined and the most effective learning strategy was identified.

Keywords: relational triple, neural network, natural language processing, transformers, knowledge graphs.

Кузьменко Андрей Владимирович
аспирант, Национальный исследовательский ядерный университет «МИФИ»
andrey_kuzmenko2907@mail.ru
Киреев Василий Сергеевич
кандидат технических наук, доцент, Национальный исследовательский ядерный университет «МИФИ»
vskireev@mephi.ru

Аннотация. Важным инструментом совершенствования рекомендательных систем, систем информационного поиска и вопрос-ответных систем являются графы знаний. Одним из способов их построения по корпусу текстов на естественном языке является извлечение реляционных троек — сущностей и связей между ними. В данной статье исследуется ранее предложенная авторами модель RESC, основанная на BERT-подобном кодировщике и позволяющая решать задачу извлечения троек с высокой производительностью. Авторами было проведено исследование абляции модели RESC на ключевом наборе текстов NYT 11. Были определены оптимальные параметры архитектуры и выявлена наиболее эффективная стратегия обучения.

Ключевые слова: реляционная тройка, нейронная сеть, обработка естественного языка, трансформеры, графы знаний.

Актуальность

Обработка текстов на естественном языке является одним из самых бурно растущих направлений в области искусственного интеллекта. Базовой задачей обработки текстов является извлечение информации. Один из наиболее удобных форматов представления извлеченной текстовой информации является граф знаний [1]. Представимая в такой форме информация является надежной основой для построения и усовершенствования рекомендательных систем, систем информационного поиска, вопросно-ответных систем.

Граф знаний состоит из вершин — набора сущностей, как правило, представимых в виде субъекта и объекта, а также ребер — связей между сущностями. Реконструкция графа знаний по коллекции текстовых документов, обычно, сводится к поиску минимальных подграфов — реляционных троек. Реляционная тройка — это набор из двух сущностей и отношения между ними.

Синтез современной системы извлечения реляционных троек включает в себя большое количество факторов, ключевыми из которых являются выбор модели, настройка модели под текущий домен, сбор данных. В данной работе проводится абляционное исследования архитектуры RESC, предложенной авторами, и предлагается оптимизация, ускоряющая работу RESC.

Цель работы

Целью данной работы является синтез оптимальной архитектуры модели RESC, основанной на нейросетевой трансформерной архитектуре и предназначенной для извлечения реляционных троек из корпуса текстов на естественном языке, а также поиск стратегии обучения.

Состояние проблемы

Задачу извлечения реляционных троек или совместного извлечения сущностей и отношений можно сформулировать как поиск набора тройных множеств $Y = \{(s_1, r_1, o_1), (s_2, r_2, o_2), \ldots, (s_k, r_k, o_k)\}$, извлеченного из текстового документа $X = \{x_1, x_2, \ldots, x_m\}$, в результате решения уравнения:

$$p(Y|X,\theta) = p(k|X)\prod_{i=1}^{K} p(Y_i|X, Y_{i\neq j}, \theta), \quad (1)$$

где $s_i = \left[x_{s_i}^{start}, \ldots, x_{s_i}^{end}\right]$ — субъект отношения; $o_i = \left[x_{o_i}^{start}, \ldots, x_{o_i}^{end}\right]$ — объект отношения; r_i — тип семантической связи между s_i и o_i из фиксированного набора $r = \{r_1, r_2, \ldots, r_n\}$; $p(k|x)$ — моделирует размер набо-

ра реляционных троек; $p(Y_i \mid X, Y_{i\neq j}, \theta)$ — моделирует тройку Y_i, при условии, что она связана не только с предложением X, но и с другими тройками $Y_{i\neq j}$; θ — параметры модели.

Авторы в [2] предлагают следующую классификацию методов извлечения реляционных троек: (1) конвейерные методы — технология, состоящая из двух четко разграниченных этапов: извлечение сущностей, определение отношений [3–5]; (2) табличные методы — поиск сущностей трансформируется в задачу заполнения квадратной матрицы, оси которой — слова в предложении, а их пересечение — тип отношения [6]; (3) методы использующие дополнительную маркировку [8]; (4) генеративные методы, среди которых отдельно выделены авторегрессионные [9,10].

Исследуемая модель RESC (см. рис. 1), находится на пересечении конвейерных и генеративных подходов. В RESC стадия поиска сущностей является вспомогательной, а в качестве основного вычислительного модуля используется одна модель — BERT [10], что существенно увеличивает скорость отклика системы. RESC обладает гибкостью в настройке модели выражается в простой отладке каждой из компонент: общего кодировщика, модуля извлечения сущностей, модуля идентификации отношений. В случае обнаружения проблемных мест пользователь может регулировать результирующее качество модели путем влияния на каждый из модулей, изменяя архитектуру или процесс обучения.

Другим свойством исследуемой архитектуры является возможность обучения каждой из компонент модели по отдельности, используя разные наборы данных. На-

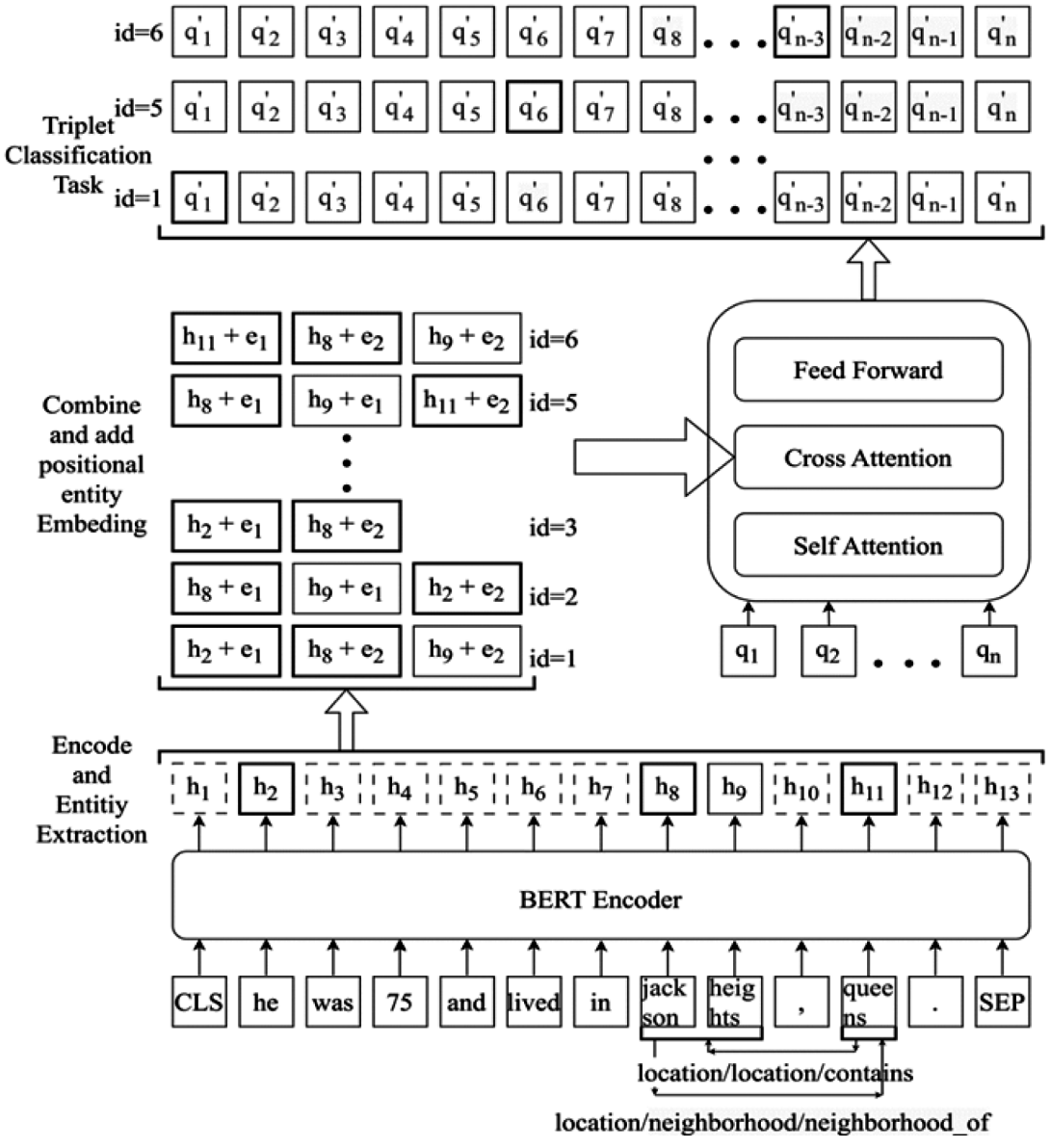


Рис. 1. Архитектура RESC

пример, если есть размеченный набор данных с реляционными тройками, то можно обучать с его помощью как модуль извлечения сущностей, так и модуль идентификации отношений. Однако такие наборы довольно дорогостоящие, поскольку требуют от асессоров следовать довольной сложной инструкции. RESC же дает возможность обучать модуль идентификации сущностей, используя наборы данных, в которых размечены только сущности. Во-первых, таких публичных наборов данных существенно больше, а во-вторых, разметка сущностей существенно проще, чем разметка реляционных троек.

Стадии RESC

В RESC можно выделить три стадии: отбор кандидатов — фильтрация заведомо неверных кандидатов; формирование кандидатов — этап формирования пар сущностей; классификационные отношения — идентификационные отношения между сущностями.

Формирование кандидатов. Последовательность текстовых токенов $x = \{x_0, x_1, \ldots, x_k\}$ кодируется предобученной моделью типа BERT в последовательность векторных представлений $h = \{h_0, h_1, \ldots, h_k\}$. Из векторных представлений формируются все возможные подстроки — сущности-кандидаты. Каждая из сущностей — это потенциальный субъект $\{h^{sub}_{start}, \ldots, h^{sub}_{end}\}$ или объект $\{h^{obj}_{start}, \ldots, h^{obj}_{end}\}$ отношения. Операция сжатия последовательности осуществляет конкатенацию субъекта-кандидата и объекта-кандидата $\{h^{sub}_{start}, \ldots, h^{sub}_{end}, h^{obj}_{start}, \ldots, h^{obj}_{end}\}$ для создания кандидатов-триплетов. Она позволяет извлекать отношение между этими сущностями напрямую, путем решения простейшей классификационной задачи.

Важной частью сжатия последовательности является позиционное кодирование сущностей, в качестве способа учета позиций последовательности векторов. Расположение сущностей предлагается учитывать при помощи двух обучаемых вспомогательных векторов e_0, e_1, преобразуя сжатую последовательность $\{h^{sub}_{start} + e_0, \ldots, h^{sub}_{end} + e_0, h^{obj}_{start} + e_1, \ldots, h^{obj}_{end} + e_1\}$. Так, для субъекта «he» и объекта «Jackson heights» последовательность векторных представлений выглядит как $\{h_1 + e_0, h_7 + e_1, h_8 + e_1\}$.

Вектора позиционного кодирования являются также обучаемыми, поскольку они имеют такую же размерность, как и размерность позиционированного кодирования в модели BERT, их можно инициализировать не случайно, а использовать ранее предобученные в модели BERT вектора, отвечающие за связность текстовых фрагментов.

Отбор кандидатов. Критичной проблемой жадного перебора являются составные сущности, которые включают два и более токенов. В таком случае, количество сущностей-кандидатов будет иметь сложность $O(S*N)$, где S длина текста в токенах, а N — максимальная длина подстроки. При этом количество комбинаций кандидатов-триплетов будет вычисляться, как число размещений $\frac{n!}{(n-2)!}$, где n — количество найденных сущностей. Нетрудно видеть, что при $S \to N$ оценка сложности составляет $O(N^4)$. Такая сложность мотивирует использовать вспомогательный модуль идентификации сущностей. Он решает простую задачу, которая является практически полным аналогов известной задачи извлечения именованных сущностей.

Каждый вектор h_i преобразуется при помощи линейной проекции и решается классическая задача классификации на три класса: 0 — не является частью сущности, 1 — начало сущности, 2 — является частью сущности, но не первый элемент. В результате решения этой задачи сложность преобразуется из $O(N^4)$ в $O(N^2)$. Этап определения границ сущностей иллюстрирован на рисунке 1. Если вектор выделен h_i в сплошной жирной обводке (2) — модель определила его как начало сущности, если в сплошной обводке (1) — продолжение сущности, пунктир (0) — не является фрагментом сущности.

Классификация отношений. Для определения типа отношений используется подход, предложенный в [11, 12]. Инициализируется набор обучаемых векторов запросов $\boldsymbol{q} = \{q_0, q_1, \ldots, q_n\}$, где n количество отношений. Эти вектора передаются в трансформер блок, который связан механизмом перекрестного внимания с кандидатами-триплетами. Под действием механизма перекрестного внимания вектора-запросы преобразуются в вектора сигналы $\boldsymbol{q'} = \{q'_0, q'_1, \ldots, q'_n\}$. Для каждого вектора-сигнала решается бинарная задача классификации. Такой подход решает проблему множественных отношений, когда между одной парой сущности существует несколько типов отношений.

Обучение модели

RESC обладает двумя механизмами принятия решений: модуль извлечения сущностей и модуль идентификации отношения. Для обучения такой системы была использована комбинированная функция потерь:

$$\mathcal{L} = \mathcal{L}_E + \lambda \mathcal{L}_R \to min \qquad (2)$$

Компонента $\mathcal{L}_E$ отвечает за решения задачи определения сущностей-кандидатов. Минимизируя это слагаемое, определяется принадлежность каждого из токенов

к одному из трех классов: 0 — не сущность, 1 — начало сущности, 2 — продолжение сущности. Вычисление этого слагаемого можно записать в следующем виде:

$$\mathcal{L}_E = -\frac{1}{B_e * S} \times \times \sum_{i}^{B_e} \sum_{j}^{S} log \frac{\exp\left(h_i^e\right) * 1\left[y_i^e \neq ignore\_index\right] * 1\left[y_i^e \neq pad\_index\right]}{\sum_{k=0}^{2} \exp\left(h_k^e\right)} \quad (3)$$

где $1\left[y_i^e \neq ignore\_index\right]$ — игнорирование результатов на целевых метках; $1\left[y_i^e \neq pad\_index\right]$ — игнорирование результатов на дополненных последовательностях; B_e — размер пакета; S — длина максимальной последовательности в пакете.

Компонента $\mathcal{L}_R$ отвечает за решения задачи идентификации типа отношения преобразованной последовательности из двух сущностей. Это задача бинарной мультиклассовой (англ. multi-label) классификации, которую можно записать в следующем виде

$$\mathcal{L}_R = -\sum_{i}^{B_r} \sum_{j}^{N_i} \sum_{k}^{N_r} \{y_k^r \log \hat{y}_k^r + \left(1 - y_k^r\right) \log\left(1 - \hat{y}_k^r\right) \quad (4)$$

где B_r — размер пакета текстов; N_i — количество размещений, которые можно вычислить как $\frac{n!}{(n-2)!}$, где n — количество сущностей, определенных в тексте; N_r — количество отношений; y_k^r — метка класса k: 0 — сущности не обладают данной связью, 1 — сущности обладают данной связью; $\hat{y}_k^r$ — оценка полученных значений вероятности принадлежности к классу k.

Для контроля значимости каждой из компонент вводится дополнительный гиперпараметр λ.

Результаты исследования и их обсуждение

Фреймворк RESC состоит из трех основных обучаемых компонент: общий кодировщик, модуль извлечения сущностей и модуль идентификации отношений. План экспериментов предполагал изменение архитектуры и стратегии обучения. При модификации каждого из трех обучаемых модулей, использованы компоненты двух других из базового сценария архитектуры модели. Как показали эксперименты, обучение модели RESC довольно стохастичный процесс. В первую очередь это связано со случайной инициализацией группы весов модели. Чтобы повысить достоверность наших наблюдений, каждый из экспериментов был проведен три раза, полученные результаты усреднялись.

Базовый сценарий архитектуры. В качестве базовой модели кодировщика использован bert-base-uncased [10]. В качестве модуля извлечения сущностей — линейная проекция, размерности которой согласуются с выходом кодировщика. Компонента идентификации отношений включает несколько составляющих: вектора запросов $\boldsymbol{q}$, вектора позиционного кодирования $\boldsymbol{e}$, трансформерный блок, классификатор, представленный в виде линейной проекции. Все ключевые размерности блока также согласованы с размерностью базового кодировщика. Инициализация векторов $\boldsymbol{q}$ производилась случайным образом, а вектора позиционного кодирования e инициализированы векторами позиционного кодирования из кодировщика типа BERT.

Базовый сценарий обучения. Обучение RESC производится в две стадии. На первой веса кодировщика не корректируются, обучаются модули извлечения сущностей и идентификации отношений. В таком режиме сигналы каждой из задач никак не влияют друг на друга. Каждая из частей учится независимо от другой. Авторами было сделано предположение, что это помогает избежать сильных всплесков градиента на ранних стадиях обучения модели и достигнуть более устойчивого состояния, что в свою очередь позволяет более бережно работать с информацией, которую выучил кодировщик на стадии предварительного обучения на задачу маскированного языкового моделирования (англ. Masked language modeling).

Обучение первой стадии выполнялось до тех пор, пока значение f меры каждой из задач не выходит на плато. На наборе данных NYT это достигалось за менее 10 эпох со скоростью обучения 1e–4. На второй стадии корректируются все параметры модели в течении 5 эпох со скоростью обучения 1e–4, после скорость обучения уменьшалась до 1e–5 модель обучается 5 эпох.

На всех этапах обучения был использован хорошо показавший себя на практике оптимизатор AdamW [13]. В качестве гиперпараметров использованы следующие значения: B_e: 32, B_r: 32, λ:1, Learning rate (1 стадия): 1e–4, Learning rate (2 стадия): 1e–4, 1e–5, B1, B2: 0.9; 0.999, Weight decay: 0.01.

Данные и методы исследования

Для экспериментов был использован открытый набор данных NYT [14] адаптированный под задачу извлечения реляционных троек публичный набор данных [15]. В наборе данных 24 типа уникальных отношений. Он содержит более 60 тыс. текстов, более 177 тыс. отношений и более 121 тыс. сущностей, и разбит на тренировочную, валидационную и тестовую части.

Модуль извлечения сущностей

Модуль извлечения сущностей — это механизм принятия решений, задача которого определить принадлеж-

ность токена к одному из трех классов: 0 — токен не является частью сущности, 1 — токен является началом сущности, 2 — токен является продолжением сущности. Эксперименты проводились с двумя, архитектурами: многослойная полносвязная сеть с нелинейностью GELU (англ. Gaussian Error Linear Units) [16] и аналогичной сетью, используемой в кодировщике, трансформерной.

Полносвязные сети позволяют наиболее просто вносить изменение в архитектуру модуля извлечения сущностей за счет наращивания линейных слоев в комбинации с нелинейностями. Эксперименты проводились с типами архитектур из следующего набора: {768,3; 768,10,3; 768,50,3; 768,100,3; 768,768,3; 768,768,3; 768,768,10,3; 768, 768,100,3; 768,768,768,3}. Первое число каждой последовательности — размерность входа данного модуля, последнее число — размер выхода, совпадающий с количеством классов, промежуточные числа — размерности скрытых представлений. Так, последовательность 768,768,10,3 означает, что данный модуль содержит три полносвязных слоя с размерностями [768x768], [768x10], [10x3].

Усложнение архитектуры через наращивание числа слов и вместе с тем, числа параметров, позволяет существенно улучшить качество как отдельно задачи извлечения сущностей, так и финальной задачи — извлечения реляционных троек.

Трансформерные слои. Другая ветка наших экспериментов основана на более сложной архитектуре трансформер. Трансформерные архитектуры, как правило, удобно описывать блоками, состоящими из ряда слоев. В качестве такого блока использован блок, идентичный тому, который используется в кодировщике RESC. Схема данного блока показана на рисунке 2. Он включает блок самовнимания (англ. Self-attention), ряд линейных проекций, нормирований векторов (англ. Layer norm), прореживания (англ. Dropout), а также нелинейности GELU [16,17].

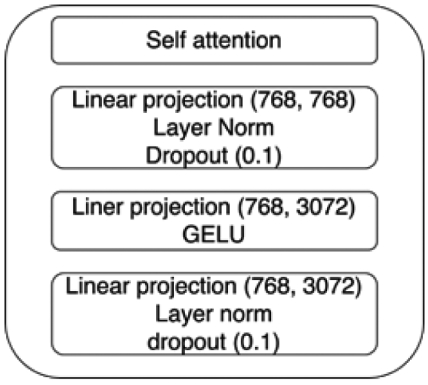


Рис. 2. Трансформерный блок

Авторы проводили свои эксперименты, наращивая число таких блоков от 1 до 3 включительно. Гомогенность кодировщика и модуля извлечения сущностей позволяет производить инициализацию параметров с блока кодировщика, а не начинать обучение со случайных весов, что предположительно существенно могло бы улучшить сходимость модели. Инициализация весов проводилась в аналогичном порядке, как это реализовано в кодировщике. В случае двух блоков, первый блок будет инициализирован весами предпоследнего блока, а второй блок — последним блоком кодировщика RESC.

На рисунках 3а, 3б иллюстрированы эксперименты в конце первой стадии обучения. По оси *x* отображено количество параметров модуля извлечения сущностей в логарифмическом масштабе. По оси *y* — значение *f* меры для задач извлечения сущностей и реляционных троек соответственно. Результаты этих экспериментов отображены в следующем формате, указано число блок K (BERT x K), а также способ инициализации. Если используется инициализация при помощи весов кодировщика добавляется суффикс init.

На рисунках явно наблюдается зависимость качества от числа параметров, особенно сильно это проявляется для финальной задачи извлечения реляционных троек. На рисунках 3в, 3г показаны результаты второй стадии обучения. На эти рисунках уже отсутствует какая-либо закономерность качества извлечения сущностей, так и качества извлечения реляционных троек.

Результаты экспериментов демонстрируют сильную зависимость *f* меры извлечения отношений и реляционных троек на первой стадии обучения. Однако, эта положительная тенденция полностью исчезает во время второй стадии обучения, или даже вовсе приводит к небольшой деградации качества. Видна целесообразность изменения архитектуры этого блока только в случае, когда есть жесткое ограничение на корректировку весов кодировщика. В остальных случаях оптимальной архитектурой является наиболее простая — линейная проекция.

Модуль идентификации отношений

Модуль идентификации отношений обладает более сложным строением в сравнении с модулем извлечения сущностей. Он включает вектора запросов $\boldsymbol{q}$, вектора позиционного кодирования $\boldsymbol{e}$, трансформерный блок — преобразователь информации из сжатой последовательности в вектора сигналов $\boldsymbol{q}'$, механизм принятия решений. Авторы проводили эксперименты над каждой составляющей этого модуля, сравнивая два ключевых показателя: качество идентификации отношения при условии, что в модель переданы истинные сущности; качество извлечения реляционных троек от начала до конца.

Наиболее очевидный этап — наращивание трансформерных блоков. Однако, эксперименты показали,

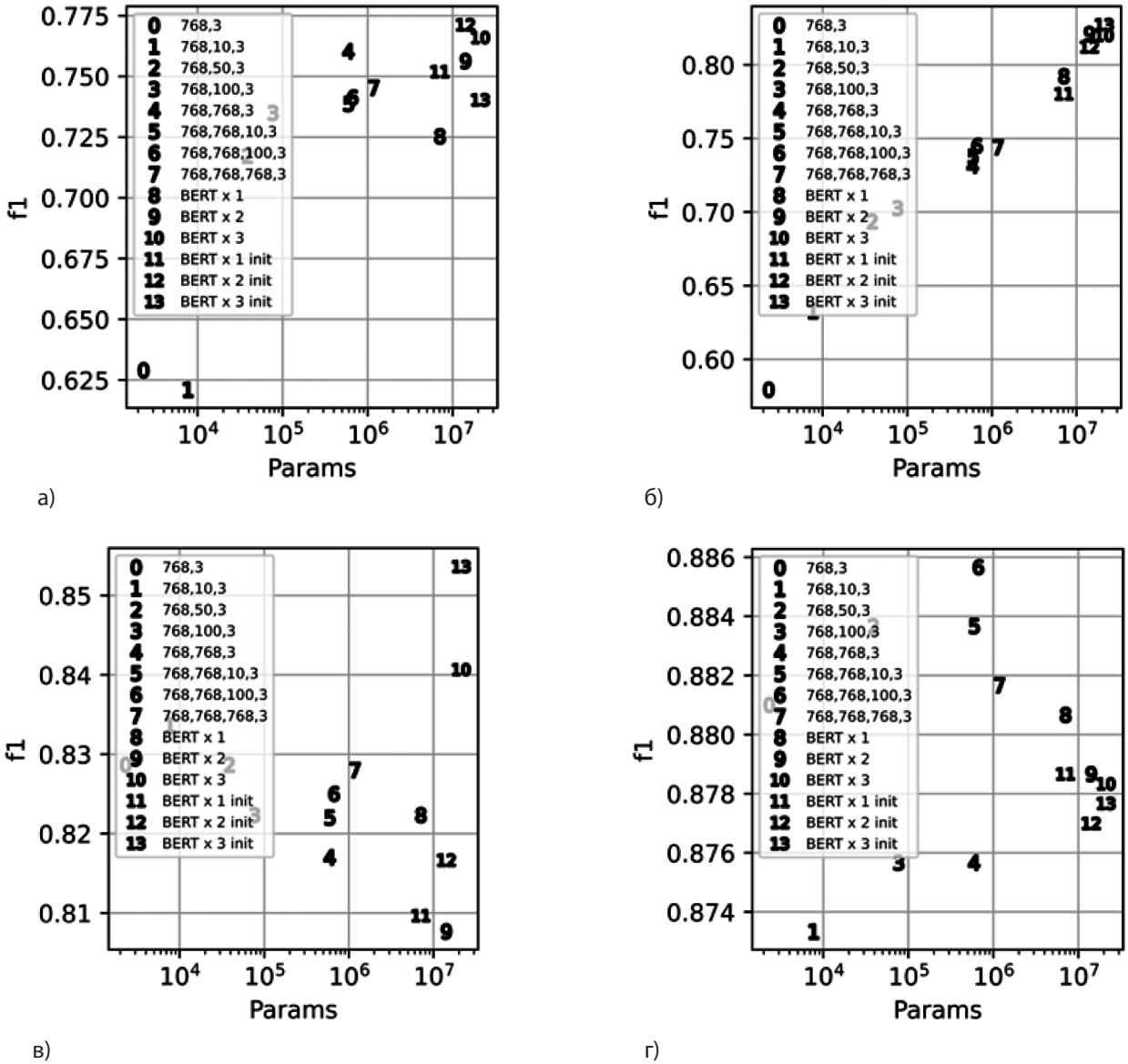
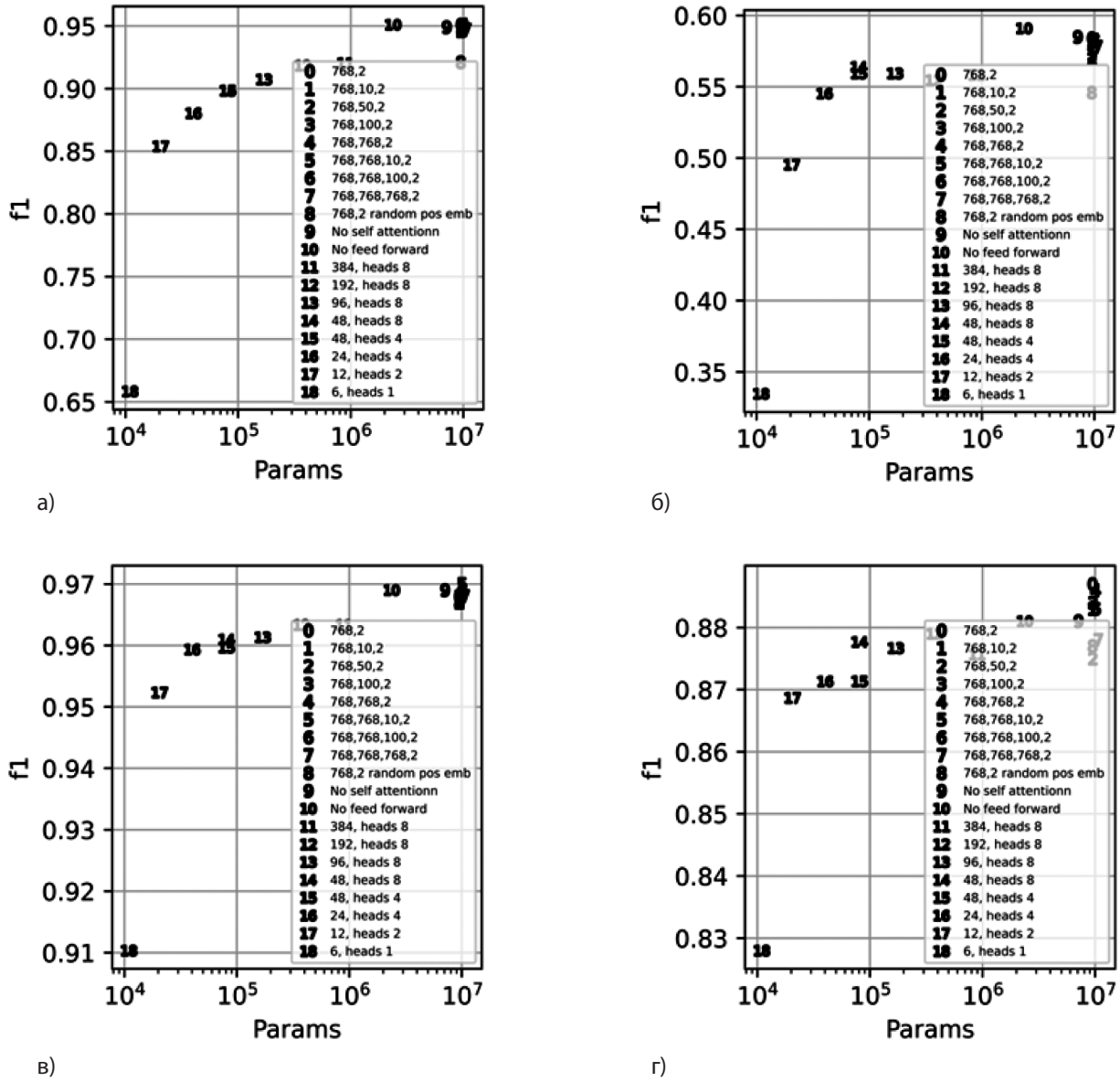


Рис. 3. Модификация модуля извлечения сущностей:
а) качество извлечения сущностей после первой стадии обучения; б) качество извлечения реляционных троек после первой стадии обучения; в) качество извлечения сущностей после второй стадии обучения; г) б) качество извлечения реляционных троек после второй стадии обучения

что их увеличение существенно ухудшает качество модели. При двух блоках модель во время обучения на первой стадии достигала максимальное качество на третьей эпохе обучения, далее же существенно теряла свою прогнозную способность с каждой эпохой. При наличии трех и более блоков наблюдалась, что неработоспособность модели и нулевые показатели *f* меры.

Аналогично проводимым экспериментам с механизмом принятия решений модуля извлечения сущностей, усложнялась архитектура механизма принятия решения модуля идентификации отношения наращивая полносвязные слои, с использованием того же ряда архитектур. В отличие от модуля извлечения сущностей, статистически значимых изменений от сложности механизма принятия решений не наблюдается, как в конце первой стадии обучения, так и в конце второй. Предположительно, это было связано с существенно большим набором обучаемых параметров данного модуля. Их количества достаточно, чтобы учесть всю информацию из обучающего набора данных. Именно поэтому, изменения качества не происходит даже после первой стадии обучения. Это поведение иллюстрировано на рисунках 4а, 4б, 4в, 4г.

Проведя эксперимент со случайной инициализацией векторов позиционного кодирования $\boldsymbol{e}$ «768,2 random pos emb», наблюдалось незначительно ухудшение качества в конце первой стадии обучения базового эксперимента



а) б)

в) г)

Рис. 4. Модификация модуля идентификации отношений:
а) качество идентификации отношений после первой стадии обучения; б) качество извлечения троек после первой стадии обучения; в) качество идентификации отношений после второй стадии обучения;
г) качество извлечения реляционных троек после второй стадии обучения

«768,2». Однако на второй стадии обучения изменения пропадают. Это иллюстрируют рисунки 4а, 4б, 4в, 4г.

Второе направление исследований модуля идентификации сущностей было направлено на оптимизацию архитектуры. В первую очередь проводились эксперименты исключая блок самовнимания. Как показали эксперименты, его наличие или отсутствие никак не влияет на конечный результат, что иллюстрируют рисунки 4а, 4б, 4в, 4г. Все дальнейшие эксперименты проводились без данного блока. Следующая оптимизация архитектуры была связана с наиболее параметросодержащим блоком feed-forward. При устранении данного блока не наблюдалось значимых изменений.

Заключительный этап оптимизаций применяется к блоку cross-attention. Этот блок представляет собой блочный механизм внимания (англ. Multi-head attention) [16] Почти вся масса обучаемых параметров в данном блоке сосредоточена в четырех матрицах весов: W_{α} — матрица контекста, W_q — матрица ключей, W_v — матрица запросов, W_{out} — матрица выхода, финальная проекция из скрытого представления в вектор ответов. Операции в блоке cross-attention можно записать в следующем виде:

$$CrossAttention(q,H) = [head_i,\ldots,head_n] \times W_{out} \quad (5),$$

где $head_i = Attention\left(q \times W_c^i, h_q W_q^i, h_q W_v^i\right)$ — выход с i блока механизма внимания; $h_q = \left\{h_{start}^{sub} + e_0, \ldots, h_{end}^{sub} + e_0, h_{start}^{obj} + e_1, \ldots, h_{end}^{obj} + e_1\right\}$ — сжатая последовательность векторов.

Размеры матриц W_q и W_v равны размерности выхода кодировщика и скрытому представлению соответственно, W_v — размерности векторов запросов $\boldsymbol{q}$ и скрытой размерности, W_{out} — скрытой размерности и размерности векторов запросов соответственно. Авторами сделано допущение, что размерность $\boldsymbol{q}$ совпадает с скрытой размерностью, и далее в экспериментах проводится калибровка ее значения.

Эксперименты показали, что при размерности в несколько десятков деградации качества практически не происходит, что иллюстрируют рисунки 4а, 4б, 4в, 4г. Однако, если количество параметров становится ниже десяти, наблюдалось явное ухудшение показателей точности. Такие результаты мы связываем с довольно небольшим набором обучающих данных.

Кодировщик

Заключительным этапом исследования является анализ кодировщика модели. Для сравнения были использованы пять различных кодировщиков. Поскольку сущности как правило представимы в виде именованных объектов, исследуются как модели инвариантные к регистру, так и модели, использующие эту информацию. В таблицах 1 и 2 сведены результаты обучения в конце первой стадии и второй стадии соответственно.

Кроме экспериментов с архитектурой, исследуются различные стратегии второй стадии обучения. За основу была взята модель Bert-base-uncased, с которой производилась первая стадия тренировки. Авторы предлагают четыре направления для второй стадии. Первая — обучение всех весов кодировщика со скоростью обучения 1e–5. Вторая — обучение всех весов модели 13 эпох со скоростью обучение 1e–4 и 5 эпох со скоростью обучения 1e–5. Третья — поблочное обучение, при котором на каждой N эпохе корректируются N последних трансформерных блоков кодировщика начиная последнего со скоростью обучения 1e–4. При N=13 одну эпоху корректируются все веса кодировщика со скоростью обучения 1e–4. После 13 эпох скорость обучения снижается до 1e–5 и модель обучается еще 5 эпох. Заключительная стратегия аналогична третьей, но используется постоянная скорость обучения 1e–5.

На рисунке 5 продемонстрированы изменение качества извлечения реляционных троек в зависимости от эпохи обучения. Можно пронаблюдать, что наиболее эффективной оказалась первая стратегия.

Таблица 1.

Результаты обучения в конце первой стадии

	Params	F1 entity extraction	F1 Relation identification	F1 Triplet extraction
Bert-tiny-uncased	4.4	0.455	0.828	0.283
Bert-base-uncased	109.5	0.629	0.92	0.571
Bert-large-uncased	333.5	0.467	0.902	0.454
Bert-base-cased	108.3	0.563	0.918	0.574
Bert-large-cased	333.5	0.497	0.921	0.538

Таблица 2.

Результаты обучения в конце второй стадии

	Params	F1 entity extraction	F1 Relation identification	F1 Triplet extraction
Bert-tiny-uncased	4.4	0.838	0.928	0.735
Bert-base-uncased	109.5	0.805	0.962	0.879
Bert-large-uncased	333.5	0.896	0.954	0.878
Bert-base-cased	108.3	0.761	0.959	0.885
Bert-large-cased	333.5	0.888	0.963	0.884

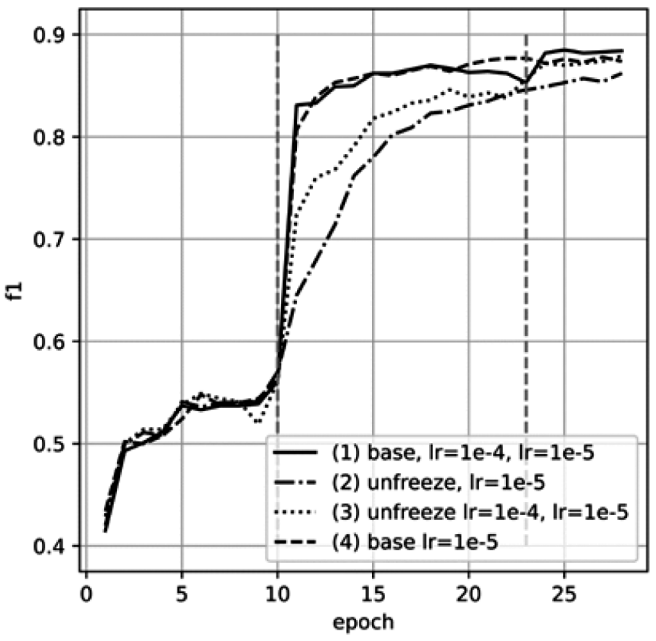


Рис. 5. Сравнение стратегий обучения

Заключение

В данной работе проведено абляционное исследование трансформерной архитектуры RESC. Проведены эксперименты с каждым из ее ключевых модулей. В ре-

зультате исследования были предложены корректировки модели, позволяющие повысить качество извлечения реляционных троек из текстов на естественном языке. Кроме того, были предложены корректировки, упрощающие архитектуру модели, что позволяет ускорить модель без потери качества. Исследованы различные стратегии обучения RESC. В настоящей работе все эксперименты проводились с фиксированным публичным набором данных NYT11, однако на практике, частая проблема — поиск достаточного набора данных. В своих будущих работах авторы планируют исследовать RESC в различных сценариях относительно набора данных.

ЛИТЕРАТУРА

1. Zhang J.C., et al. A review of recommender systems based on knowledge graph embedding // Expert Systems with Applications. — 2024. — Vol. 250. — P. 123876.
2. Кузьменко А.В., Киреев В.С. Классификация методов извлечения реляционных троек из текстов на естественном языке // Сб. науч. тр. XXV Междунар. науч.-техн. конф. «Нейроинформатика-2023». — М., 2023. — С. 302–311.
3. Zenelnko D., Aone C., Richardella A., et al. Kernel methods for relation extraction // Journal of Machine Learning Research. — 2003. — Vol. 3. — P. 1083–1106.
4. Chan S.Y., Roth D. Exploiting syntactico-semantic structures for relation extraction // Proc. 49th Annu. Meet. Assoc. Comput. Linguistics: Human Lang. Technol. — 2011. — P. 551–560.
5. Zhong Z., Chen D.A. A frustratingly easy approach for entity and relation extraction // Proc. 2021 Conf. North Amer. Chapter Assoc. Comput. Linguistics: Human Lang. Technol. — 2021. — P. 50–61.
6. Zhang M., Zhang Y., Fu G. End-to-End Neural Relation Extraction with Global Optimization // Proc. 2017 Conf. Empirical Methods in Natural Lang. Process. — 2017. — P. 1730–1740.
7. Wei Z., Su J., Wang Y., et al. A Novel Cascade Binary Tagging Framework for Relational Triple Extraction // Proc. 58th Annu. Meet. Assoc. Comput. Linguistics. — 2020. — P. 1476–1488.
8. Sui D., Chen Y., Liu K., et al. Joint entity and relation extraction with set prediction networks // IEEE Trans. Neural Networks Learn. Syst. — 2024. — P. 12784–12795.
9. Yuan C., Xie Q., Ananiadou S. Zero-shot Temporal Relation Extraction with CharGPT // 22nd Workshop Biomedical Natural Lang. Process. BioNLP Shared Tasks. — 2023. — P. 92–102.
10. Delvin J., Chang M.-W., Lee K., Toutanova K. BERT: pre-training of deep bidirectional transformer for language understanding // Proc. 2019 Conf. North Amer. Chapter Assoc. Comput. Linguistics: Human Lang. Technol. — 2019. — P. 4171–4186.
11. Li J., Li D., Savarese S., Hoi S. BLIP-2: bootstrapping language-image Pre-Training with frozen image encoders and large language models // Proc. 40th Int. Conf. Mach. Learn. — 2023. — P. 19730–19742.
12. Zhao H., Xin Y., Yu Z., et al. SLIT: boosting audio-text pre-training via multi-stage learning and instruction tunning. — 2024. — URL: https://www.catalyzex.com/paper/slit-boosting-audio-text-pre-training-via. (дата обращения: 20.01.2025).
13. Loshchilov I., Hutter F. Decoupled weight decay regularization // Proc. 7th Int. Conf. Learn. Represent. — 2018. — URL: https://openreview.net/pdf?id=Bkg6RiCqY7. (дата обращения: 20.01.2025).
14. Riedal S., Yao L., McCallum A. Modeling relations and their mentions without labeled text // Mach. Learn. Knowl. Discov. Databases. — 2010. — P. 148–163.
15. Zeng X., Zeng D., He S., et al. Extracting relational facts by an end-to-end neural model with copy mechanism // Proc. 56th Annu. Meet. Assoc. Comput. Linguistics. — 2018. — Vo1. 1. — P. 506–514.
16. Hendrycks D., Gimpel K. Gaussian error linear units (GELUs). — URL: https://arxiv.org/pdf/1606.08415. (дата обращения: 20.01.2025).
17. Ni S., Yang M., Xu R., Li C., Hu X. Layer-wise Regularized Dropout for Neural Language Models // arXiv preprint. — 2024. — eprint 2402.16361. — URL: https://arxiv.org/abs/2402.16361 (дата обращения: 20.01.2025).

Журнал «Современная наука: актуальные проблемы теории и практики»