Московский государственный университет имени М.В.Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра алгоритмических языков

Кудисов Артём Аркадьевич

Адаптация к изменению типов сущностей в задаче извлечения отношений

Выпускная квалификационная работа

Научный руководитель:

Майоров Владимир Дмитриевич
:

Москва, 2023

Аннотация

Адаптация к изменению типов сущностей
в задаче извлечения отношений

Кудисов Артём Аркадьевич

Задача извлечения бинарных отношений между сущностями является одной из классических проблем компьютерной лингвистики и имеет большое прикладное значение в современных системах извлечения информации.

Несмотря на важность задачи, подавляющее большинство существующих решений являются неуниверсальными: модели, обученные на одном наборе типов сущностей, не способны работать при их изменении.

Работа посвящена сравнению различных методов адаптации к изменяющимся типам сущностей в задаче извлечения отношений. Рассмотрены существующие решения, выделены их ключевые недостатки и проведен сравнительный анализ качества.

Abstract

Adaptation to entity types change in relation extraction task

Kudisov Artem

Relation extraction (RE) is one of the fundamental tasks of computer linguistics and it has widespread applications in the modern information extraction systems.

Despite the importance of RE task, the vast majority of solutions are non-universal: models trained on one set of entity types are not able to work when set changes.

This thesis investigates and compares various methods of adaptation to entity types change in RE task. The existing solutions are considered, their key disadvantages are highlighted and a comparative analysis of quality is performed.

Содержание

1 Введение

1.1 Задача извлечения отношений

Одной из классических проблем компьютерной лингвистики является задача извлечения отношений (REC, relation extraction and classification). Основная ее суть заключается в установлении и классификации бинарных отношений между именованными сущностями (named entities) в тексте. На рисунке 1 представлен пример REC задачи, где каждое предложение содержит по 2 сущности, связь между которыми необходимо обнаружить. Каждая сущность харатеризуется не только тем, какой объект (выраженный словами в тексте) она представляет собой, но и определенным типом. Так в первом предложении примера сущность «ООН» относится к типу ORG (организация), а «Нью-Йорк» - к типу CITY (город).

Формально описать задачу можно следующим образом: дан документ d, содержащий текст с некоторым выделенным набором из n сущностей $\{e_i\}_{i=1}^{n}$, между которыми необходимо установить бинарные отношения из множества $\mathcal{R}$. Каждая сущность e_i относится к одному из $\mathcal{T}$ типов и может упоминаться в тексте несколько раз. В таких случаях говорится о том, что данные упоминания ссылаются на единый объект и образует одну кореферентную цепочку. Сами типы t из множества $\mathcal{T}$ являются словами естественного языка, максимально подробно описывающими некоторую предметную область.

Рис. 1: Пример задачи извлечения и классификации отношений. Отдельные сущности в тексте выделены с указанием их типа. Отношения указаны в виде стрелок.

Решения задачи извлечения отношений имеют множество практических применений, которые впоследствии можно использовать в вопросно-ответных системах, в интеллектуальном анализе текста, в системах извлечения информации и знаний, при построении семантических баз знаний и т.д. В целом, существует большое количество

подходов к решению REC задачи , но все из них можно разделить на 2 большие категории:

- решения на базе правил, когда отношения выделяются и классифицируются на основании некоторых заранее выявленных (вручную) правил и паттернов;
- решения на базе машинного обучения, когда модель обучается на большом корпусе подготовленных данных и самостоятельно выявляет определенные статистические правила, в соответствии с которыми и действует в дальнейшем.

Далее речь будет идти только о решениях на базе машинного обучения (а конкретнее о построенных на архитектуре трансформеров [1] нейронных сетях, таких как BERT [2], RoBERTa [3] или BART [4]), т.к данный подход на текущий момент демонстрирует лучшее качество по F1-мере на большинстве основных наборов данных, посвященных задаче извлечения отношений (таблица 1).

Набор данных	Лучшая модель	Архитектура	F1-мера
DocRED	DREEAM	RoBERTa$_{large}$	67.53
CoNLL04	REBEL	BART	76.65
NYT	REBEL	BART	93.4

Таблица 1: Лучшие модели на указанных наборах данных с приведенной архитектурой и качеством[1].

В качестве мер оценивания того, насколько хорошо работает тот или иной подход, в задаче извлечения отношений применяются 3 классические оценки:

- точность (precision) - доля верно классифицированных отношений от общего числа найденных отношений;
- полнота (recall) - доля верно классифицированных отношений от общего числа отношений;
- F1-мера (F1-score) - среднее гармоническое между точностью и полнотой; оценка общего качества решения.

[1]Данные взяты с сайта https://paperswithcode.com/task/relation-extraction

1.2 Изменение типов сущностей

Каждая сущность характеризуется определенным типом. Однако модели зачастую конструируются и обучаются с тем допущением, что типы сущностей не будут изменяться в ходе дальнейшей работы. поэтому явным образом используют информацию о фиксированных типах. Это весьма смелое предположение, если учесть, что актуальность проблемы изменения наборов типов в данных можно обнаружить на практике. Возьмем, к примеру, систему извлечения информации из новостей (рисунок 2), где в обрабатываемом потоке данных могут регулярно видоизменяться типы сущностей из-за изменения используемого множества NERC-моделей (named entity recognition and classification[2]). Изменение данного набора может происходить по разными причинам, но среди основных можно выделить:

- добавление нового языка для обработки, а следовательно и добавление новых NERC-моделей;
- добавление более тонко настроенных обработчиков, выделяющих сущности более узких (специфичных) типов (например, типы QUANTIY, ORDINAL и CARDINAL вместо одного единого NUMBER).

Естественно это не полный список ситуаций, когда в обрабатываемых документах начинают встречаться новые типы сущностей. В то же время реализация новых REC-моделей может быть сопряжена с рядом трудностей (недостаток новых обучающих данных, частое изменение типов сущностей, дороговизна переобучения и т.п.). В результате было бы полезно разработать и реализовать некоторые методы, расширяющие обобщающие возможности моделей по извлечению отношений и позволяющие данным моделям эффективно и быстро адаптироваться к новым данным. Под скоростью и эффективностью подразумеваются следующие 3 пункта:

- Независимость от наличия дополнительных обучающих данных (т.к. на их сбор и обработку уходят ресурсы и время). Иными словами, методы должны работать с уже имеющимися данными;

[2]NERC - задача по извлечению сущностей из текста; https://paperswithcode.com/task/named-entity-recognition-ner

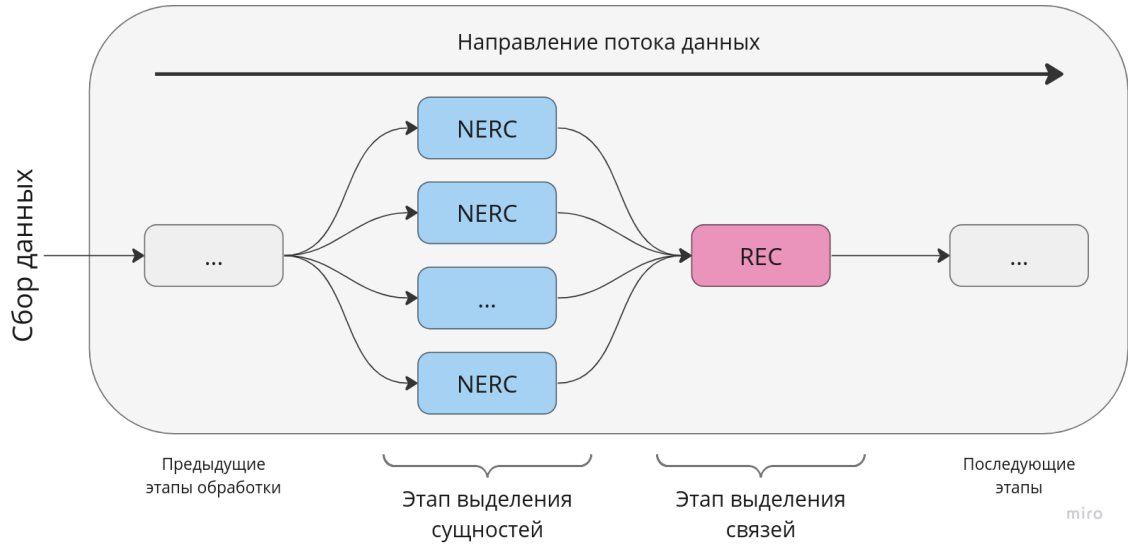

Рис. 2: Упрощенная схема системы по извлечению информации. Каждый блок представляет собой отдельную модель/алгоритм. NERC - модели по извлечению именованных сущностей. REC - модель по извлечению отношений.

- Универсальность адаптационных методов, иными словами модели должны работать на любом наборе типов сущностей (постоянное переобучение модели после каждого изменения типов сущностей влечет за собой большие накладные расходы в виде времени и вычислительных ресурсов);
- Неизменность скорости работы моделей (неизменность вычислительной сложности).

В целом, задачу адаптации можно рассматривать в следующей формулировке: необходимо реализовать некоторые адаптационные методы $\mathcal{A}$, использование которых позволило бы REC-моделям, обученным устанавливать бинарные отношения $r \in \mathcal{R}$ между сущностями типов $\mathcal{T}_1$, выделять те же отношения (когда это уместно) между сущностями типов $\mathcal{T}_2$ ($\mathcal{T}_1 \neq \mathcal{T}_2$).

2 Постановка задачи

Целью представленной выпускной квалификационной работы является разработка новых методов адаптации к изменению типов сущностей в задаче извлечения отношений и их сравнение с уже существующими методами.

В ходе работы необходимо:

1. Рассмотреть основные методы адаптации, выделив их ключевые особенности, и разработать собственные;
2. Выбрать несколько популярных наборов данных для задачи извлечения отношений и реализовать модели машинного обучения (при необходимости адаптировав и модифицировав код), совместимых с ними;
3. Провести эксперименты для троек вида «модель, данные, метод», получив оценки качества в виде точности, полноты и F1-меры;
4. Провести анализ полученных результатов.

Основные требования, выдвигаемые к методам адаптации, следующие:

- Методы должны работать с уже имеющимися данными;
- Методы должны быть универсальными, иными словами, модели должны работать на любом наборе типов сущностей;
- Скорость работы моделей должна остаться прежней (т.е. вычислительная сложность не должна изменяться).

3 Обзор существующих решений

В этой главе рассматриваются основные наборы данных и нейронные модели для задачи извлечения отношений, а также рассматриваются уже существующие методы адаптации.

3.1 Существующие методы адаптации

Обозначенную выше проблему с изменением типов сущностей можно попытаться решить следующими способами, встречающимися в литературе.

3.1.1 Дообучение

Первым распространенным решением является дообучение. Данный метод часто используется, когда необходимо адаптировать модель, обученную на одних данных, для работы с другими [5], [6]. При наличии обучающих данных можно попытаться дообучить (переобучить) модель для работы с новыми типами. Это простое решение, но здесь сразу же возникают 2 главных ограничения:

1. Тренировочные данные нужно каким-то образом собрать (или быть может сгенерировать) и подготовить, выделив в них сущности и отношения между ними;
2. В случае нового изменения типов вновь необходимо переобучать модель. Если же типы изменяются достаточно часто, то переобучение и вовсе может оказаться бессмысленным (если не успевать переобучаться), поэтому встаёт вопрос целесообразности данного метода.

В одной из работ [6] авторы попытались частично решить первую проблему - избавиться от необходимости в разметке новых данных. Для этого они предложили обучать модель одновременно на уже известных D_1 и новых D_2 неразмеченных данных сразу под две задачи:

1. Задача извлечения отношений: обучение происходит только на известном наборе данных D_1.

2. Задача классификации примеров: модель учится различать между собой примеры $d_1 \in D_1$ и $d_2 \in D_2$ из известного и нового источника соответственно. Причем градиенты берутся с обратным знаком, из-за этого модель выявляет не уникальные для обоих наборов данных признаки, а напротив их общие черты.

Таким образом, на первой подзадче модель учится находить отношения, а на второй - выявлять общие закономерности различных данных. Предложенный авторами способ доказал свою состоятельность, однако несмотря на это, он не решает оставшиеся проблемы дообучения. Кроме того, несмотря на то, что он делает модели устойчивыми к изменениям данных, это всё равно не позволяет работать им с измененными типами.

В конечном счёте, возможности данного метода упираются в проблему сбора дополнительных данных и ограниченности времени. Однако, этот способ не требует каких-либо дополнительных модификаций кода, что является несомненным плюсом.

3.1.2 Игнорирование

Достаточно кардинальным методом можно назвать полный отказ от использования типов сущностей $\mathcal{T}$. Тем самым теряется часть ценной информации, полученной с предыдущих этапов обработки текста. Так, в работах [7] и [8] было показано, что простое выделение сущностей в тексте показывет более низкое качество по сравнению с одновременным их выделением и указанием их типов. В таблице 2 представлены полученные результаты из работы [7].

Способ	Пример	$\text{BERT}_{\text{base}}$
Выделение	[E1] Bill [/E1] was born in [E2] Seattle [/E2]	68.4
Выделение (пункт.)	@ Bill @ was born in # Seattle #	68.7
Выделение с типом	<S:PERSON> Bill </S:PERSON> was born in <O:CITY> Seattle </O:CITY>	71.5
Выделение (пункт.) с типом	@ * person * Bill @ was born in # ∧ city ∧ Seattle #	70.9

Таблица 2: Результаты (F1-мера, %) модели $\text{BERT}_{\text{base}}$ на данных TACRED при разных способах выделения сущностей в тексте. Данные взяты из исследования [7].

Однако отказ от типов полностью избавляет от проблем с обработкой новых из них. Более того, подобное игнорирование легко реализовывается (не так сложно добавить соответсвующие модификацию в код уже готовых моделей) и не накладывает никаких других ограничений, будь то увеличение времени работы или же необходимость в сборе дополнительных данных.

3.2 Наборы данных

Прежде чем перейти к экспериментам, необходимо определиться с набором данных, на котором они будут поставлены. Существует достаточно большое число корпусов, посвященных задаче извлечения отношений[3]. Но наиболее популярными являются следующие 2:

- DocRED - ежегодно выпускается порядка 30 публикаций с использованием данного корпуса[4];
- TACRED - аноличным образом порядка 30 публикаций в год[5].

Именно DocRED и исправленная версия TACRED (далее Re-TACRED) будут использоваться в дальнейшем.

3.2.1 DocRED

DocRED (Document-level Relation Extraction Dataset) - один из основных наборов данных, предложенный в 2019 году группой исследователей из Китая [9] и быстро завоеваший свою популярность. К качестве источников использовались статьи из Wikipedia[6] и Wikidata[7]. Главной особенностью примеров (также называемых документами) из DocRED является их размер. В большинстве корпусов каждый документ состоит всего из 1-ого предложения, таким образом связи ищутся лишь в небольшой локальной

[3]С основным списком можно ознакомиться на сайте https://paperswithcode.com/task/relation-extraction, раздел Benchmarks.

[4]Приблизительная статистика по версии сайта https://paperswithcode.com/dataset/docred, раздел Usage.

[5]Приблизительная статистика по версии сайта https://paperswithcode.com/dataset/tacred, раздел Usage.

[6]https://en.wikipedia.org/wiki/Main_Page

[7]https://www.wikidata.org/wiki/Wikidata:Main_Page

области. В то же время согласно исследованию авторов [9], порядка 40.7% всех отношений могут быть обнаружены лишь на уровне нескольких предложений. Это крайне большое число связей, которое просто нельзя игнорировать. Именно поэтому был создан DocRED, в котором каждый из примеров состоит из нескольких предложений и отношения между сущностями ищутся на уровне всего документа.

Корпус состоит из 5053 вручную размеченных примеров, содержащих в общей сложности 132375 именнованных сущностей и 56354 отношений между ними[8]. В каждом документе авторы выделили сами сущности, их типы и упоминания, кореферентные цепочки и семантические отношения. Дополнительно к основным примерам авторы набора также предоставили 101873 документов, размеченных автоматическими методами и содержащих 2558350 сущностей и 881298 отношений между ними[8]. Основную статистику по корпусу можно найти в таблице 4.

3.2.2 TACRED

TACRED (TAC Relation Extraction Dataset) - один из основных корпусов, предложенный в 2017 году группой ученных из Стэнфордского Университета [10]. В качестве исходных данных были взяты новости и текста из интернета, используемые в ежегодных задачах TAC Knowledge Base Population (TAC KBP)[9]. В таблице 3 представлена чуть более подробная информация о происхождении TACRED.

В целом, корпус содержит порядка 106264 примеров, разделенных на обучающую, валидационную и тестовую часть. Каждый пример состоит из 1-ого предложения, в котором выделены всего 2 сущности и указаны их типы. Именно между ними необходимо определить наличие связи и классифицировать ее (41 вид отношений). Основную статистику по набору данных также можно найти в таблице 4.

3.2.3 Re-TACRED

В 2020 году вышла статья немецких исследователей [11], в которой они попытались ответить на вопрос, почему модели по извлечению отношений до сих пор имеют такой высокий уровень ошибок (порядка 30 процентных пунктов по F1-мере) и достигли ли модели своего потолка. В ходе исследования ученые рассмотрели 5000 наиболее слож-

[8]Без учета отношений, обозначающих отсутсвие какой-либо связи между сущностями

[9]https://tac.nist.gov/2017/KBP/index.html

Часть	Число примеров	Оригинальный корпус
Обучающая	68124	TAC KBP 2009–2012
Валидационная	22631	TAC KBP 2013
Тестовая	22631	TAC KBP 2014

Таблица 3: Оригинальные корпуса, на базе которых собирался TACRED.

ных примеров (на которых модели ошибаются наиболее часто) из валидационной и тестовой части набора TACRED и выяснили, что более 50% из данных примеров нуждаются в переразметке. На основании данного исследования была предложена новая версия TACRED под названием RE-TACRED, в которой исправлена часть примеров из тестовой и валидационной части. Именно эта версия и будет использоваться в дальнейшей работе.

Наборы данных	Число примеров			Число типов	Число отношений
	Обучающий	Валидационный	Тестовый		
DocRED	3053	1000	1000	6	97*
TACRED	68124	22631	15509	17	42*
Re-TACRED	68124	22631	15509	17	42*

Таблица 4: Основные характеристики используемых корпусов. * - с учетом отдельного отношения, обозначающего отсутствие связи.

3.3 Модели

3.3.1 SSAN-Adapt

Модель SSAN-Adapt была представлена в мае 2021 года группой ученых из Китая [12]. На момент своего выхода модель показывала лучший результат на корпусах DocRED [9], GDA [13] и CDR [14], а на 2023 год модель занимает на них 3, 7 и 7 места соответственно[10]. В целом, подход авторов к решению задачи по извлечению отношений можно назвать классическим - в качестве базовой архитектуры использовалась BERT модель (с

[10]Данные взяты с сайта https://paperswithcode.com/paper/entity-structure-within-and-throughout

небольшим изменением механизма внимания), создающая для каждой пары сущностей вектор признаков, который впоследствии соотносится с одним из 97 типов отношений[11].

Структурная матрица

Из интересных особенностей SSAN-Adapt можно выделить представленный авторами измененный механизм внимания, который позволяет модели в своих рассчетах учитывать не только глобальный семантический контекст всего текста, но и его структуру. Так авторы вводят целых шесть видов структурных отношений в тексте между словами на базе трех характеристик:

- Корреферентность - являются ли два слова упоминаниями одной сущности;
- Сущность - являются ли слова сущностями или нет;
- Локальность - находятся ли два слова в одном и том же предложении.

Пользуясь ими, каждую пару слов можно охарактеризовать одним из следующих 4 типов:

- intra+coref: находятся в одном предложении и ссылаются на одну сущность;
- inter+coref: находятся в разных предложениях и ссылаются на одну сущность;
- intra+relate: находятся в одном предложении и ссылаются на разные сущности;
- inter+relate: находятся в разных предложениях и ссылаются на разные сущности.

Стоит сразу пояснить, что данные структруные типы применимы только для пар, в которых оба слова являются упоминаниями какой-либо сущности. Однако авторы вводят ещё два дополнительных типа:

- intraNE: оба слова находятся в одном предложении и первое из них является сущностью;
- NA: все остальные случаи.

[11] С учетом отдельного отношения, обозначающего отсутсвие связи.

Пользуясь данными типами, можно представить полную структуру всего текста в виде матрицы $S = (s_{ij})$, пример которой представлен на рисунке 3, где сущности обозначены символом S (с соответсвующим индексом), а обычные слова - символом N. Элемент s_{ij} данной матрицы в i-ой строке и j-ом столбце обозначает структурную связь между i-ым и j-ым словами. Вообще говоря, каждому виду структурной связи соответсвует свой линейный оператор $A \in \mathbb{R}^{d_{out} \times 1 \times d_{out}}$, но на картинке они просто обозначены цветами. Стоит обратить внимание, что матрица не является симметрической - это свойство нарушается отношением intraNE, которое рассчитывается только для сущностей, но не для обычных слов.

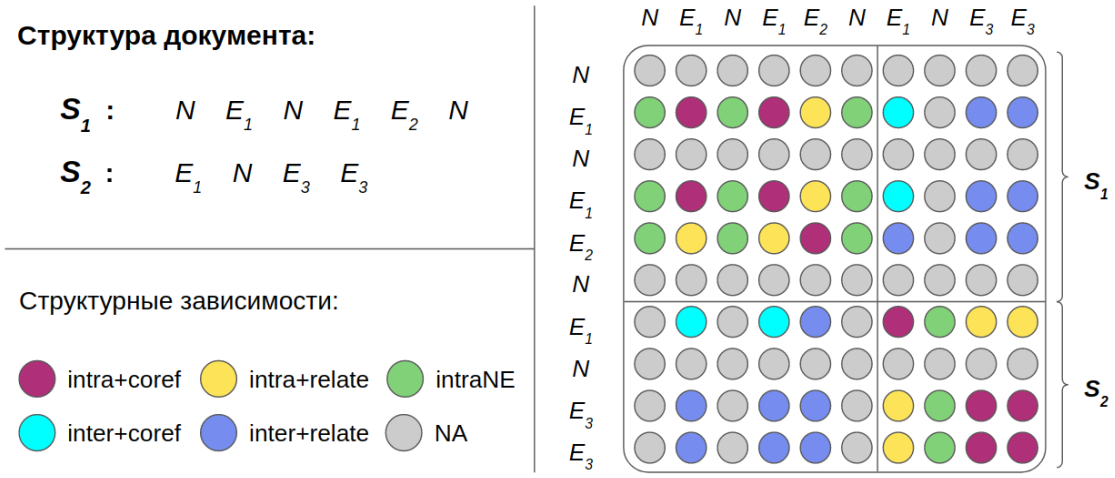

Рис. 3: Пример структурной матрицы S для текста, состоящего из двух предложений S_1, S_2 и трех сущностей E_1, E_2 и E_3.

Реализация

1. Первым этапом работы модели является токенизация текста, в результате которой документ d разбивается на серию токенов $\{w_i\}_{i=1}^{N}$. Каждому токену w_i соответствует свой собственный эмбеддинг (вектор вещественных значений) $v_i \in \mathbb{R}^{d_{emb}}$, в численном виде кодирующий всю информацию о данном токене. Для кодирования информации о типах сущностей в модели SSAN-Adapt используется дополнительный набор эмбеддингов - по одному вектору на каждый тип. На рисунке 4 схематично изображен данный процесс: после токенизации текста к каждому

из эмбеддингов, соответствующих токену какой-либо сущности прибавляется дополнительный эмбеддинг типа данной сущности. В результате получается набор векторов $\{x_i\}_{i=1}^{N}$.

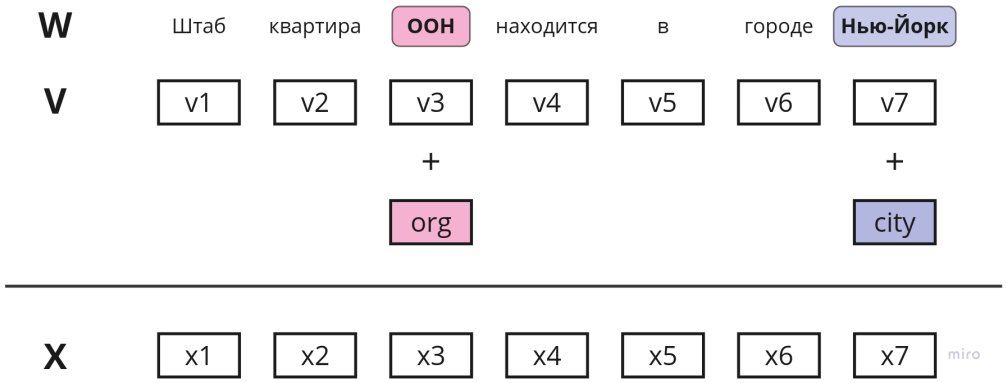

Рис. 4: Упрощенный пример этапа кодирования типов, где вместо токенов используются сами словам. *ООН* и *Нью-Йорк* являются сущностями типов «org» (organization, организация) и «city» (город) соответственно.

2. Модель SSAN-Adapt состоит из кодирующей и классифицирующей частей. На рисунке 5 представлена общая структура кодирующей части модели. Как видно, она представляет собой набор из N одинаковых блоков трансформера [1], в которых применяется механизм внимания (self-attention + кодирование структурной матрицы), полносвязная нейронная сеть, нормализация и skip connections [15]. В результате работы кодирующей части для каждого из токенов $\{x_i\}_{i=1}^{N}$, на которые был разбит исходный текст, создается свой вектор $y_i \in \mathbb{R}^d$. Для дальнейшего кодирования сущностей берется среднее арифметическое векторов тех токенов, которые соответствуют упоминаниям этих самых сущностей. Таким образом, для каждой сущности $e_i \in \{e_i\}_{i=1}^{n}$ строится свой кодирующий вектор $m_i \in \mathbb{R}^d$.

3. В классифицирующей части модели для каждой упорядоченной пары сущностей (e_h, e_t) вычисляется вероятность наличия между ними отношения $r \in \mathcal{R}$:

$$P_r(m_h, m_t) = sigmoid(m_h W_r m_t)$$

где m_h и m_t - соответсвующие сущностям вектора, $W_r \in \mathbb{R}^{d \times d}$ - соответсвующая отношению r матрица.

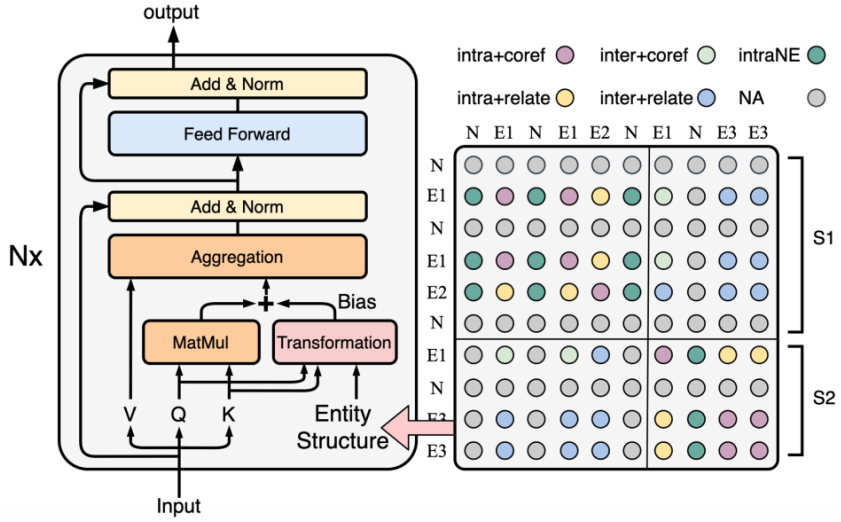

Рис. 5: Архитектура модели SSAN-Adapt. Иллюстрация взята из оригинальной статьи, посвященной SSAN-Adapt модели [12].

3.3.2 DocUNet

В 2021 году была представлена модель DocUNet [16] ((архитектура которой представлена на рисунке 6)), частично представляющая задачу извлечения отношений как задачу семантической сегментации изображений. Суть заключается в том, что модель для документа с M упоминаниями сущностей:

1. Этап кодирования:

    1.1 Строит для каждого i-ого упоминания вектор вещественных значений $m_i \in \mathbb{R}^d$ с помощью BERT-модели;

    1.2 Строит для каждой пары упоминаний (i, j) вектор внимания $a_{ij} \in \mathbb{R}^{d_{att}}$ [1];

2. Этап семантической сегментации:

    2.1 Представляет весь документ в виде изображения m на m пикселей и глубиной d_{att}, при этом каждый пиксель ассоциируется с одной парой упоминаний. В результате получается матрица $U \in \mathbb{R}^{m \times m \times d_{att}}$ из векторов, полученных на этапе 1.2;

2.2 Применяет UNet-модель [17] для классификации каждого пикселя на d_{out} классов[12]. В результате получается матрица $V \in \mathbb{R}^{m \times m \times d_{out}}$

3. Этап классификации:

3.1 Для каждой пары упоминаний (i, j) объединяет соответсвующие вектора m_i, m_j и v_{ij}, полученные с предыдущих этапов, и классифицирует на наличие одного из $\mathcal{R}$ отношений.

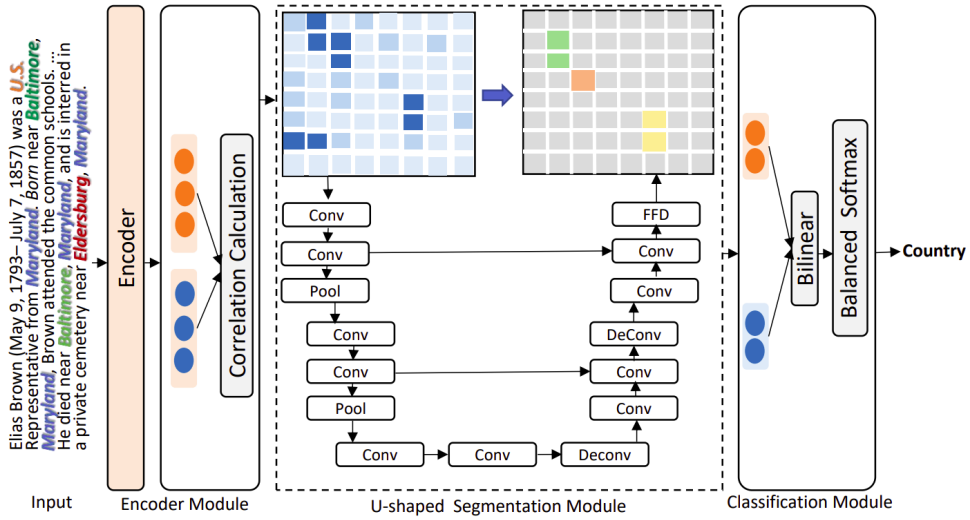

Рис. 6: Архитектура модели DocUNet. Иллюстрация взята из оригинальной статьи, посвященной DocUNet модели [16].

Для кодирования типов модель обрамляет с двух сторон каждую сущность специальными зарезервированными символами [unused...], где вместо многоточий указывается порядковый номер конкретного типа. Для каждого такого символа существует свой собственный эмбеддинг.

3.3.3 $\text{BERT}_{\text{base}}$

Модель $\text{BERT}_{\text{base}}$ была представлена в 2021 году [7] как улучшенное стандартное решение задачи извлечения отношений на уровне предложений. Главным достоинством

[12] d_{out} - некоторое заранее выбранное число, вообще говоря не равное $|\mathcal{R}|$

модели является её внутренняя простота и высокое качество работы. Так на момент своего выхода на корпусе Re-TACRED $\text{BERT}_{\text{base}}$ занимала 2-ое место.

Реализация

Модель состоит из кодирующей и классифицирующей частей. Кодирующая часть представляет собой обычную предтренированную BERT модель [2], создающую эмбеддинги для сущностей в тексте. Классифицирующая часть является простой комбинацией линейного слоя, функции активации ReLU, dropout слоя и снова линейного слоя.

Для кодирования типов модель вставляет в текст с обеих сторон сущности специальные зарезервированные метки, для которых существуют собственные эмбеддинги:

- **Игнорирование**

  Сущности в тексте выделяются при помощи символов [E1], [\E1] и [E2], [\E2].

- **С указанием типов**

  В случае, если типы сущностей должны быть указаны, то используются следующие символы: [SUBJ-...], [\SUBJ-...], [OBJ-...], [\OBJ-...], где вместо многоточий вставляется тип соответсвующих сущностей.

4 Исследование и построение решения задачи

В этой главе рассматриваются 2 дополнительных метода адаптации, не встречающихся в литературе, а также описывается постановка экспериментов вместе с полученными результатами.

4.1 Методы адаптации

4.1.1 Соотнесение

Зачастую (таблица 5) типы сущностей в разных корпусах концептуально не отличаются друг от друга и единственной разницей между ними являются их названия. Так, например, в обоих корпусах DocRED и Re-TACRED есть сущности типа человек, но в первом случае данный тип обозначается как «PER», а во втором - «PERSON». В таких случаях можно попытаться соотнести типы («переименовать», «отобразить» их), построив некоторое отображение $f : \mathcal{T}_2 \to \mathcal{T}_1$ названий из нового множества $\mathcal{T}_2$ в старое $\mathcal{T}_1$ и переименовав новые названия типов в соответствии с этим отображением.

Названия типов	Корпуса		
	DocRED	Re-TACRED	OntoNotes
Человек	PER	PERSON	PERSON
Организация	ORG	ORGANIZATION	ORGANIZATION
Место	-	LOCATION	GEO_LOCATION

Таблица 5: Примеры одинаковых типов сущностей в разных корпусах.

Однако не всегда возможно построить однозначное отображение. Так, например, одному типу из нового множества $\mathcal{T}_2$ может соответствовать несколько названий из старого $\mathcal{T}_1$. На рисунке 7 изображен пример подобного сюръективного отображения f, когда типу «NUMBER» $\in T_2$ одновременно соответсвует сразу 5 типов из T_1[13].

Подобные конфликтные ситуации можно попытаться решить несколькими способами:

[13] «NUMBER» - используемый тип сущностей из корпуса DocRED. Остальные типы взяты из корпуса OntoNotes.

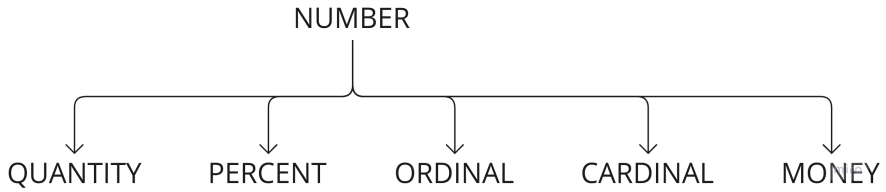

Рис. 7: Пример сюръективного отображения, когда новому типу «NUMBER» потенциально соответсвуют 5 старых.

- Соотносить $t_2 \in \mathcal{T}_2$ с одним конкретным типом $t_1 \in \mathcal{T}_1$. Но вполне очевидно, что, например, не все числа являются процентами, поэтому данный способ не подходит;
- Соотносить $t_2 \in \mathcal{T}_2$ сразу со всеми подходящими типами из $\mathcal{T}_1$. Однако не все модели способны кодировать сущности более чем одним типом. Кроме того, подобный подход вносит в систему шум.

Таким образом, необходим некий дополнительный механизм автоматического построения наиболее подходящих отображений (в особенности, если данные с новыми типами поступают довольно часто и вручную их не обработать). Однако стоит учитывать, что если, например, в тексте присутствует N сущностей, типы которых необходимо переименовать, то при наличии m вариантов того, как это сделать, механизму потребуется обработать N^m возможных решений. Самой же главной проблемой здесь является разработка некоторой оценки, на основании которой, мы будем предпочитать одно возможное соотнесение типов другому (или одни предсказанные отношения другим).

Подводя итог, можно сказать, что метод отображений может давать свой результат, но необходимо дополнительно реализовывать механизм построения отображений и разрабатывать некоторую оценку, что является совсем не тривиальной задачей, так как вполне возможно, что она будет зависеть от конкретной предметной области, в которой работает модель.

4.1.2 Диверсифицированное обучение

Последним предлагаемым методом в данной работе будет диверсифицированное обучение, суть которого заключается в том, чтобы во время обучения в тренировочных данных истинные типы именованных сущностей с определенной вероятностью заменять на синонимичные названия, либо же в 15% случаев специально зашумлять (т.е.

в качестве типа использовать случайное слово). Похожий механизм зашумления присутствует во время обучения BERT модели [2], когда порядка 10% выбранных токенов в примере заменяются на совершенно случайные. В обоих случаях это делается для того, чтобы модель училась больше полагаться в своих предсказаниях на окружающий контекст и тем самым становилась более устойчивой.

Подобный процесс обучения требует некоторых дополнительных данных, но всё сводится к тому, что нужно лишь составить тот самый список синонимичных типов (таблица 6 и приложение А.1, таблицы 9 и 10). Однако и данный процесс можно автоматизировать, если воспользоваться возможностями открытых семантических сетей (например, Wikidata[14] или WordNET[15]), в которых имеются иерархические связи вида «гипоним» и «гипероним». Так при замене исходного типа во время обучения можно с определенной вероятностью брать гипоним типа или его гипероним.

Оригинальные типы	Типы во время обучения			Типы во время тестирования
ORG	ORG	INSTITUTE	GROUPING	ORGANIZATION
LOC	LOC	AREA	DISTRICT	LOCATION
TIME	TIME	MOMENT	STAGE	DATE
PER	PER	CHARACTER	HUMAN	PERSON
MISC	MISC	ETC	DIFFERENT	MISCELLANEOUS
NUM	NUM	DIGIT	QUANTITY	NUMBER

Таблица 6: Пример синонимичных замен. Типы взяты из корпуса DocRED.

4.1.3 Сравнение

Из всего вышесказанного можно составить предварительную сводную таблицу 7 основных слабых мест всех разобранных методов, выделив в ней наиболее принципиальные моменты.

[14]Викиданные — это свободная и отрытая база знаний, в которой хранятся структурированные данные Википедии, см. https://www.wikidata.org/wiki/Wikidata:Main_Page.

[15]WordNET - это большая лексическая база данных английского языка, см. https://wordnet.princeton.edu.

Методы	Недостатки		
	Требуются дополнительные данные	Требуются дополнительные запуски	Требуется дополнительная оценка
Дообучение†	+	−	−
Соотнесение	−	+	+
Игнорирование†	−	−	−
Диверсифицированное обучение	−	−	−

Таблица 7: Основные недостатки методов адаптации. Символом † обозначены методы, описанные в литературе, остальные методы являются предложенными в этой работе.

Прежде чем перейти к сравнению данных методов на практике, нужно заметить, что:

- главным недостатком дообучения является необходимость в дополнительных данных и его неуниверсальность, что противоречит двум основным требованиям данной работы. Поэтому этот метод в дальнейшем рассматриваться не будет;
- также не будет рассматриваться метод отображений из-за экспоненциального роста числа вычислений (относительно числа возможных вариантов построения данного отображения) и из-за нетривиальности задачи по построению оценки.

Что касается оставшихся методов, то они полностью удовлетворяют выдвигаемым требованиям.

4.2 Адаптация существующих моделей

Для проведения экспериментов по сравнению методов адаптации пришлось несколько видоизменить исходные структуры моделей так, чтобы они могли кодировать типы сущностей различными способами.

4.2.1 SSAN-Adapt

Игнорирование типов

При игнорировании типов (глава 3.1.2) модель никаким образом не оперирует информацией о них. Всё, о чём знает модель, что то или иное слово является сущностью, но не более того. Поэтому нет никакой необходимости в эмбеддингах для типов. Вместо этого используется лишь один отдельный эмбеддинг для обозначения в тексте сущностей в целом (рисунок 8).

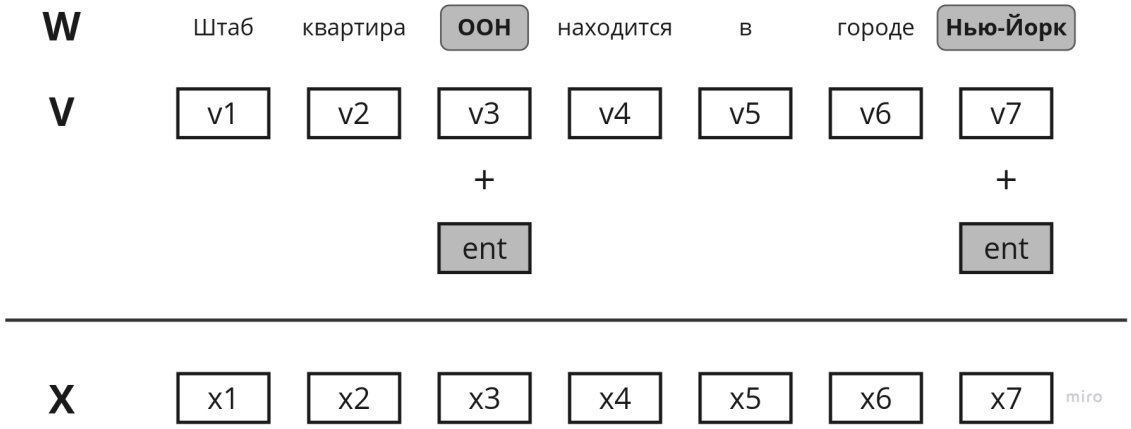

Рис. 8: Упрощенный пример этапа кодирования типов, где вместо токенов используются сами словам. *ООН* и *Нью-Йорк* являются сущностями (ent, entities), при этом никакой информации об их типах нет.

Кодирование в тексте

Метод диверсифицированного обучения (глава 4.1.2) в ходе своей работы постоянно изменяет истинные типы сущностей на синонимичные. Кроме того, в ходе дальнейшей эксплуатации модели на обработку могут поступать документы с новыми неизвестными типами. Как уже говорилось в главе 1.2, использование отдельных эмбеддингов для типов в таком случае совершенно не допустимо. Именно поэтому типы указываются в самом тексте в виде специальных меток. Для обозначения сущностей в тексте используется лишь один отдельный эмбеддинг (рисунок 9).

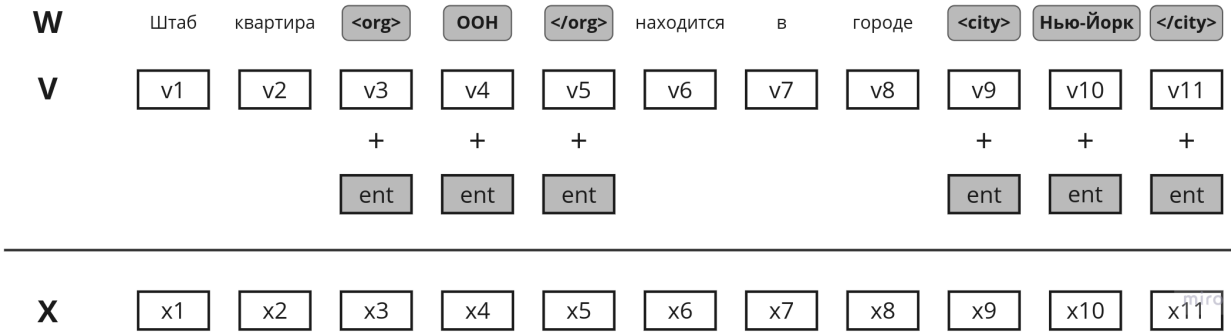

Рис. 9: Упрощенный пример этапа кодирования типов, где вместо токенов используются сами словам. *ООН* и *Нью-Йорк* являются сущностями (ent, entities), при этом их типы кодируются в самом тексте в виде специальных меток.

4.2.2 DocUNet

В ходе реализации модели DocUNet в нее были внесены минимальные изменения:

- Механизм игнорирования типов не оперирует информацией о них. Поэтому на этапе кодирования текста специальные символы [unused...] не используются;
- Для реализации механизма диверсифицированного обучения пришлось заменить исходные символы вида [unused...] на [...] и [/...], где вместо многоточий указаны типы сущностей (или их синонимичные замены).

4.2.3 $\text{BERT}_{\text{base}}$

В модель $\text{BERT}_{\text{base}}$ были внесены также минимальные изменения:

- Механизм игнорирования типов уже был встроен в модель, поэтому здесь ничего не изменялось;
- Метод диверсифицированного обучения требует постоянного изменения истинных типов сущностей на синонимичные. Поэтому в специальных символах [SUBJ-...], [/SUBJ-...], [OBJ-...], [/OBJ-...], использущихся для кодирования типов сущностей, вместо истинных типов вставляются синонимичные.

4.2.4 Замечание

В ходе работы метода диверсифицированного обучения метки, использующиеся для кодирования типов сущностей в тексте, не являются зарезервированными из-за их случайной природы. Это означет, что для них не существует отдельных собственных эмбеддингов, и на этапе токенизации текста каждый из этих символов распадается на серию токенов.

4.3 Результаты

Основные результаты, полученные в ходе работы, представлены в таблице 8. Каждая оценка качества была получена как среднее арифметическое 3-ех экспериментов с различными начальными инициализациями сетей. Кроме того, указаны стандартные отклонения результатов.

Как видно из таблицы, метод диверсифицированного обучения значительно уступает простому игнорированию типов сущностей в 2 случаях из 3. Для модели DocUNet методы показывают сопоставимое качество.

Методы адаптации	Результат, F1-мера (%)		
	DocRED		Re-TACRED
	SSAN-Adapt	DocUNet	$\text{BERT}_{\text{base}}$
Игнорирование	54.32 ± 0.05	60.24 ± 0.08	76.64 ± 0.37
Диверсифицированное обучение	51.62 ± 0.16 †	60.13 ± 0.08 †	74.96 ± 0.20 ‡

Таблица 8: Результаты применения выбранных методов адаптации. Символом † обозначены результаты, где вероятнось зашумления была равна 0.15, а символом ‡ обозначены результы, где вероятность равнялась 0.9

Для модели $\text{BERT}_{\text{base}}$ были дополнительно проведены эксперименты по исследованию зависимости результатов (при использовании метода диверсифицированного обучения) от вероятности зашумления. Как видно на рисунке 10, с ростом вероятности зашумления качество модели на данных, в которых выделены ранее неизвестные типы сущностей, растет, постепенно выходя на плато, но так и недостигает результатов,

полученных при полном игнорировании типов. Подобное поведение свидетельствует о следующем:

1. Изначальный способ кодирования типов сущностей в тексте для модели $\text{BERT}_{\text{base}}$ оказался неподходящим (т.к. при малых значениях вероятности зашумления наблюдается большой разрыв в результатах между полным игнорированием типов и методом диверсифицированного обучения). При этом модель переобучается на известном наборе типов;

2. С ростом же вероятности зашумления, модель постепенно начинает игнорировать информацию о типах, принимая ее за шум. Благодаря этому уменьшается разрыв в качестве между двумя методами.

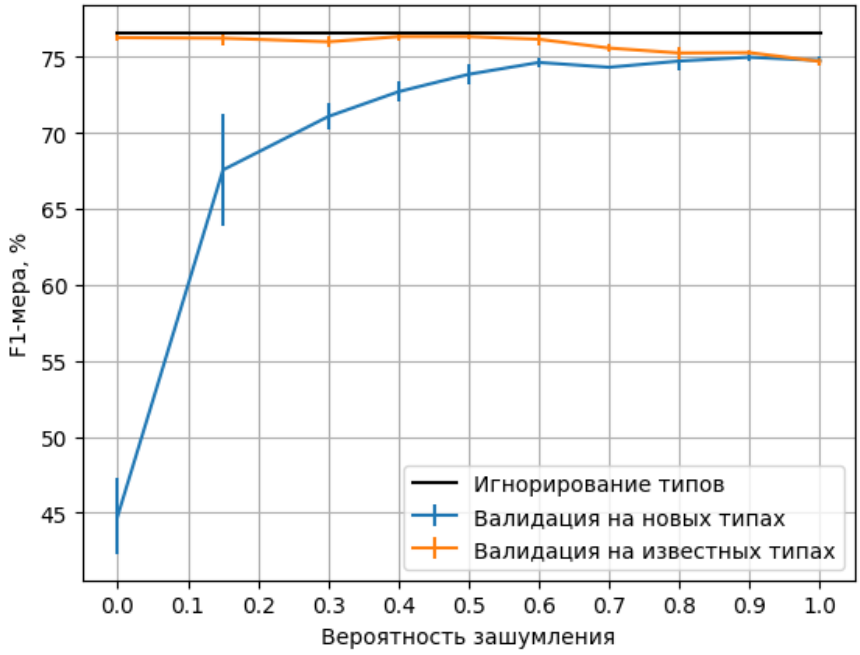

Рис. 10: Зависимость валидационных результатов модели $\text{BERT}_{\text{base}}$ от используемой во время обучения вероятности зашумления. На графике отображены средние значения со стандратным отклонением на базе 3-ех запусков модели. Оранжевым цветом выделены результаты, полученные на валидационном корпусе с использованием уже известных для модели типов (таблица 9). Синим - результаты на валидационном корпусе с использованием новых типов (таблица 10). Черным - результаты на валидационном корпусе при полном игнорировании типов.

5 Описание практической части

5.1 Инструментальные средства

Основным языком программирования был выбран Python версии 3.8.10. Причиной тому послужили 2 вещи: наличие большого числа библиотек для машинного обучения (часть из которых была использована и в данной работе), а также факт того, что Python является одним из самых используемых языков в области МО[16].

В работе использовались следующие вспомогательные библиотеки (указаны лишь самые основные):

- numpy (1.21.6) - необходима для работы с многомерными векторами [18];
- torch (1.13.0) - необходима для реализации глубоких нейронных моделей и для тензорных вычислений с автоматическим дифференцированием функций [19];
- transformers (4.23.1) - библиотека от Huggingface, необходимая для использования уже предобученных нейронных моделей [20];
- jupyter (1.0.0) - для работы с интерактивным кодом [21];
- pandas (2.0.0) - для работы с табличными данными [22];
- matplotlib (3.7.1) - для отрисовки графиков [23].

Все модели обучались на видеокарте NVIDIA TESLA A100 80GB с использованием драйвера CUDA Toolkit версии 11.7.

5.2 Архитектура решения

В рамках работы требовалось поставить серию экспериментов на различных наборах данных, моделей и адаптационных методах. Для этого был реализован CLI интерфейс, код которого лежит в открытом доступе на платформе *github.com*[17].

[16]Так в январе 2019 года хостинг GitHub опубликовал свой рейтинг самых популярных языков программирования в МО, в котором Python занимал 1-ое место (см. https://github.blog/2019-01-24-the-state-of-the-octoverse-machine-learning/).

[17]https://github.com/phos-phophy/creed

На рисунке 11 представлена диаграмма основных классов программы, где выделены отношения наследования, агрегации и т.п..

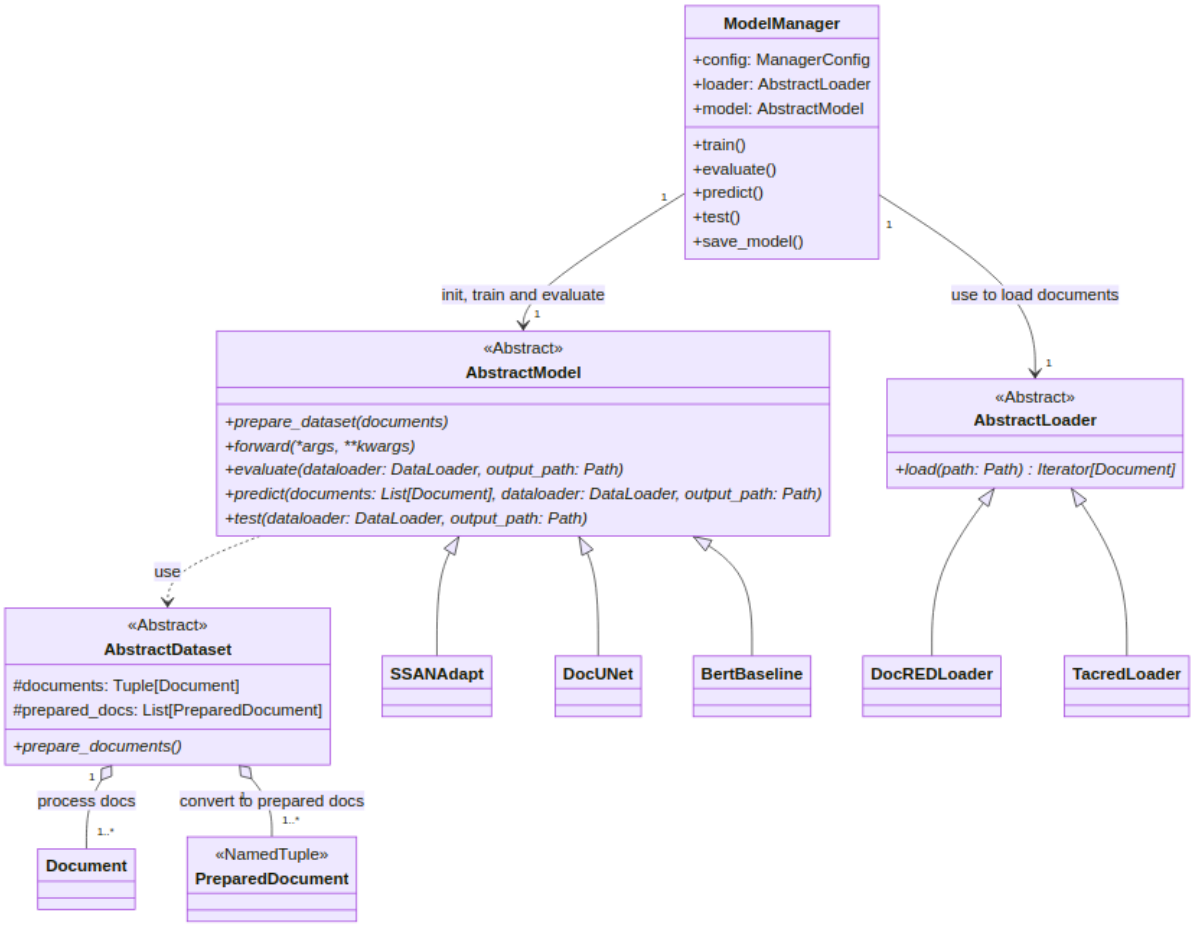

Рис. 11: Диаграмма классов.

- **ModelManager** - класс-одиночка, инициализирующий модель, подготавливающий для нее данные, обучающие и тестирующий ее;
- **AbstractModel** - абстрактный базовый класс для всех моделей, реализуемых в рамках данной работы; является родительским для таких классов, как **SSANAdapt**, **BertBaseline** и **DocUNet**;
- **AbstractLoader** - абстрактный базовый класс для всех загрузчиков корпусов данных; является родительским для таких классов, как **DocREDLoader** и **TacredLoader**;

- **Document** - вспомогательный класс, представляющий собой 1 пример из любого набора данных;
- **PreparedDocument** - вспомогательный класс, представляющий собой набор тензоров, относящихся к 1-ому примеру;
- **AbstractDataset** - абстрактный базовый класс, представляющий собой отдельный корпус/набор данных; хранит в себе оригинальные необработанные примеры (в виде наборе **Document**'ов), которые впоследствии переводятся в набор тензоров (в виде **PreparedDocument**'ов). Каждая модель в ходе работы использует свой собственный набор подходящих ей **AbstractDataset**'ов. Так, например, для модели **SSANAdapt** специально реализованы **BaseDataset**, **IETypesDataset** и **WOTypesDataset**.

5.3 Схема функционирования

Запуск программы осуществляется из командной строки при помощи bash-скрипта scripts/main.sh, на вход которому через флаги -c, -v, -o, -s передаются путь до конфигурационного файла модели, идентификатор используемой видеокарты, путь до директории, где будет сохранена модель, и seed - стартовое число для генератора псевдослучайных чисел. Далее автоматически вызывается код из файла main.py, который на базе переданного конфигурационного файла создает менеджер модели (экземпляр класса ModelManager). Данный менеджер инициализирует саму модель и приступит к её обучению, тестированию и т.д. Важно отметить, что поведение менеджера полностью определяется содержимым конфигурационного файла. Так, если в нем отсутсвует путь до обучающего корпуса данных, модель обучаться не будет, а менеджер сразу перейдет к следующей стадии.

6 Заключение

В ходе работы были достигнуты следующие результаты:

1. Реализован гибкий CLI интерфейс для проведения экспериментов с REC-моделями на различных наборах данных;
2. Реализованы и модифицированы REC-модели;
3. Выделены сильные и слабые стороны 4-ых методов адаптации, 2 из которых ранее не упоминались в литературных источниках;
4. Проведена серия экспериментов по сравнению метода игнорирования с методом диверсифицированного обучения. Было показано, что простое игнорирование типов показывает лучший результат.

Список литературы

[1] Attention is All you Need / Ashish Vaswani, Noam M. Shazeer, Niki Parmar et al. // *ArXiv*. — 2017. — Vol. abs/1706.03762.

[2] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding / Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova // *arXiv preprint arXiv:1810.04805*. — 2018.

[3] RoBERTa: A Robustly Optimized BERT Pretraining Approach / Yinhan Liu, Myle Ott, Naman Goyal et al. // *arXiv preprint arXiv:1907.11692*. — 2019.

[4] BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension / Mike Lewis, Yinhan Liu, Naman Goyal et al. // *arXiv preprint arXiv:1910.13461*. — 2019.

[5] *Jiang, Jing*. Instance Weighting for Domain Adaptation in NLP / Jing Jiang, ChengXiang Zhai. — 2007. — 01.

[6] Domain Adaptation for Relation Extraction with Domain Adversarial Neural Network / Lisheng Fu, Thien Huu Nguyen, Bonan Min, Ralph Grishman // Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers). — Taipei, Taiwan: Asian Federation of Natural Language Processing, 2017. — . — Pp. 425–429. https://aclanthology.org/I17-2072.

[7] *Zhou, Wenxuan*. An Improved Baseline for Sentence-level Relation Extraction. — 2022.

[8] *Ni, Jian*. A Generative Model for Relation Extraction and Classification. — 2022.

[9] DocRED: A Large-Scale Document-Level Relation Extraction Dataset / Yuan Yao, Deming Ye, Peng Li et al. // Proceedings of ACL 2019. — 2019.

[10] Position-aware Attention and Supervised Data Improve Slot Filling / Yuhao Zhang, Victor Zhong, Danqi Chen et al. // Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017). — 2017. — Pp. 35–45. https://nlp.stanford.edu/pubs/zhang2017tacred.pdf.

[11] *Alt, Christoph.* TACRED Revisited: A Thorough Evaluation of the TACRED Relation Extraction Task / Christoph Alt, Aleksandra Gabryszak, Leonhard Hennig // Proceedings of ACL. — 2020. https://arxiv.org/abs/2004.14855.

[12] Entity Structure Within and Throughout: Modeling Mention Dependencies for Document-Level Relation Extraction / Benfeng Xu, Quan Wang, Yajuan Lyu et al. // *Proceedings of the AAAI Conference on Artificial Intelligence*. — 2021. — May. — Vol. 35, no. 16. — Pp. 14149–14157. https://ojs.aaai.org/index.php/AAAI/article/view/17665.

[13] RENET: A Deep Learning Approach for Extracting Gene-Disease Associations from Literature / Ye Wu, Ruibang Luo, Henry C. M. Leung et al. // Research in Computational Molecular Biology / Ed. by Lenore J. Cowen. — Cham: Springer International Publishing, 2019. — Pp. 272–284.

[14] Overview of the BioCreative V chemical disease relation (CDR) task / C.H. Wei, Yifan Peng, Robert Leaman et al. — 2015. — 09. — Pp. 154–166.

[15] *He, Kaiming.* Deep Residual Learning for Image Recognition. — 2015.

[16] *Zhang, Ningyu.* Document-level Relation Extraction as Semantic Segmentation. — 2021.

[17] *Ronneberger, Olaf.* U-Net: Convolutional Networks for Biomedical Image Segmentation. — 2015.

[18] Array programming with NumPy / Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt et al. // *Nature*. — 2020. — . — Vol. 585, no. 7825. — Pp. 357–362. https://doi.org/10.1038/s41586-020-2649-2.

[19] Automatic differentiation in PyTorch / Adam Paszke, Sam Gross, Soumith Chintala et al. // NIPS-W. — 2017.

[20] Transformers: State-of-the-Art Natural Language Processing / Thomas Wolf, Lysandre Debut, Victor Sanh et al. // Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations. — Online: Association for Computational Linguistics, 2020. — . — Pp. 38–45. https://www.aclweb.org/anthology/2020.emnlp-demos.6.

[21] Jupyter Notebooks - a publishing format for reproducible computational workflows / Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez et al. // Positioning and Power in Academic Publishing: Players, Agents and Agendas / Ed. by Fernando Loizides, Birgit Scmidt. — Netherlands: IOS Press, 2016. — Pp. 87–90. https://eprints.soton.ac.uk/403913/.

[22] *Wes McKinney.* Data Structures for Statistical Computing in Python / Wes McKinney // Proceedings of the 9th Python in Science Conference / Ed. by Stéfan van der Walt, Jarrod Millman. — 2010. — Pp. 56 – 61.

[23] *Hunter, J. D.* Matplotlib: A 2D graphics environment / J. D. Hunter // *Computing in Science & Engineering.* — 2007. — Vol. 9, no. 3. — Pp. 90–95.

A Приложение

A.1 TACRED: изменение типов

Оригинальные типы	Типы во время обучения		
ORGANIZATION	ORGANIZATION	ORG	GROUPING
PERSON	PERSON	PER	HUMAN
NUMBER	NUMBER	NUM	QUANTITY
DATE	DATE	MOMENT	STAGE
NATIONALITY	NATIONALITY	RACE	FOLK
LOCATION	LOCATION	AREA	LOC
TITLE	TITLE	HEADING	HEADER
CITY	CITY	TOWN	URBAN
MISC	MISC	ETC	MISCELLANEOUS
COUNTRY	COUNTRY	NATION	REALM
CRIMINAL_ CHARGE	CRIMINAL_ CHARGE	CRIME	CHARGE
RELIGION	RELIGION	CULT	FAITH
DURATION	DURATION	PERIOD	LENGTH
URL		URL	
STATE_OR_ PROVINCE	STATE_OR_ PROVINCE	REGION	AREA
IDEOLOGY	IDEOLOGY	PHILOSOPHY	WORLDVIEW
CAUSE_OF_ DEATH	CAUSE_OF_ DEATH	DEATH	MORTAL

Таблица 9: Пример синонимичных замен во время обучения. Типы взяты из корпуса TACRED.

В таблице 9 представлено изменение типов корпуса TACRED во время обучения. Каждое из слов в строке имеет одинаковый шанс p_c быть выбран в качестве замены, где $p_c = (1 - p_n)/3$, p_n - вероятность зашумления типа.

В таблице 10 представлены типы сущностей, используемые во время валидации и тестирования.

Оригинальные типы	Типы во время валидации	Типы во время тестирования
ORGANIZATION	INSTITUTE	ESTABLISHMENT
PERSON	CHARACTER	INDIVIDUAL
NUMBER	DIGIT	SIGN
DATE	TIME	DAY
NATIONALITY	ETHNIC	NATION
LOCATION	DISTRICT	POSITION
TITLE	NAME	APPELLATION
CITY	PLACE	MUNICIPAL
MISC	DIFFERENT	VARIOUS
COUNTRY	STATE	POLITY
CRIMINAL_CHARGE	PENAL	ACCUSATION
RELIGION	CONFESSION	CREED
DURATION	TIME	VALIDITY
URL	URL	URL
STATE_OR_PROVINCE	TERRITORY	DOMAIN
IDEOLOGY	DOCTRINE	OUTLOOK
CAUSE_OF_DEATH	DIE	PERDITION

Таблица 10: Пример синонимичных замен во время валидации и тестирования. Типы взяты из корпуса TACRED.