|  |
| --- |
| Пермский филиал федерального государственного автономного образовательного учреждения высшего образования  «Национальный исследовательский университет  «Высшая школа экономики»  *Факультет социально-экономических и компьютерных наук* |
| Чураков Григорий Александрович  **Разработка модели для определения скелетной структуры текста, представленного на русском или английском языке**  *Курсовая работа*  студента образовательной программы «Программная инженерия» по направлению подготовки *09.03.04 Программная инженерия*  Руководитель  Доцент,  НИУ ВШЭ-Пермь, приглашенный преподаватель  Л. А. Мыльников |

Пермь, 2024 год

**Аннотация**

Отчет по практической работе на тему «Разработка модели для определения скелетной структуры текста, представленного на русском или английском языке».

Автор работы: Чураков Григориий Александрович.

Во введении обоснована актуальность работы, поставлена цель работы, определены задачи работы, указаны объект и предмет исследования.

Первая глава обзору научных работ проведенных в над решением проблем обработки естественного языка.

Во второй главе описаны результаты проектирования и обучения модели на предоставленном наборе данных.

Третья глава содержит описание этапа развертывания модели.

Четвертая глава содержит описание экспериментов поставленных в ходе обучения модели с целью повышения качества предсказаний.

Заключение содержит описание результатов, полученных в ходе выполнения работы.

Работа состоит из 35 страниц основного текста и 14 страниц приложений, содержит 21 рисунок, 2 формулы, библиографический список – 23 наименования, 9 приложений.

**Ключевые слова:** частеречная разметка, морфологический анализ словоформ, обработка естественного языка.

Оглавление

[Оглавление 3](#_Toc162042770)

[Глоссарий 5](#_Toc162042771)

[Введение 6](#_Toc162042772)

[Глава 1. Задача частеречной разметки 9](#_Toc162042773)

[Существующие методы частеречной разметки 9](#_Toc162042774)

[Вероятностные 9](#_Toc162042775)

[На основе правил 9](#_Toc162042776)

[Скрытая цепь Маркова 9](#_Toc162042777)

[Метрики оценки качества разметки 10](#_Toc162042778)

[Взвешеная F-1 мера 10](#_Toc162042779)

[Balanced accuracy 11](#_Toc162042780)

[Глава 2. Обзор литературы 12](#_Toc162042781)

[Методы составления векторных представлений 12](#_Toc162042782)

[Understanding bag-of-words model: a statistical framework[17] 12](#_Toc162042783)

[Glove: Global Vectors for Word Representation[15] 12](#_Toc162042784)

[Enriching Word Vectors with Subword Information[2] 12](#_Toc162042785)

[Neural Machine Translation of Rare Words with Subword Units[1] 13](#_Toc162042786)

[Neural Machine Translation with Byte-Level Subwords[6] 14](#_Toc162042787)

[Модели 15](#_Toc162042788)

[Attention is all you need[3] 15](#_Toc162042789)

[BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding[5] 17](#_Toc162042790)

[Performance of Transfer Learning Model vs. Traditional Neural Network in Low System Resource Environment[8] 20](#_Toc162042791)

[A Family of Pretrained Transformer Language Models for Russian[9] 20](#_Toc162042792)

[Глава 2. Разработка модели 22](#_Toc162042793)

[Проектирование модели частеречной разметки 22](#_Toc162042794)

[Анализ входных данных 22](#_Toc162042795)

[Выбор базовой модели 23](#_Toc162042796)

[Перенос обучения 24](#_Toc162042797)

[Обучение модели 26](#_Toc162042798)

[Оценка результатов 27](#_Toc162042799)

[Глава 3. Публикация и развертывание модели 28](#_Toc162042800)

[Глава 4. Эксперименты 29](#_Toc162042801)

[Заключение 32](#_Toc162042802)

[Библиографический список 33](#_Toc162042803)

[Приложение А. Список тегов для русского языка 35](#_Toc162042804)

[Приложение Б. Ссылка на репозиторий модели 37](#_Toc162042805)

[Приложение В. Ссылка на репозиторий набора данных 38](#_Toc162042806)

[Приложение Г. Ссылка на Демо-приложение 39](#_Toc162042807)

[Приложение Д. Техническое задание 40](#_Toc162042808)

[Приложение Е. Программа и методика испытаний 41](#_Toc162042809)

[Приложение Ж. Текст программы 42](#_Toc162042810)

[Приложение И. Оценка качества модели 43](#_Toc162042811)

[Приложение И. Тестирование веб приложения 48](#_Toc162042812)

Глоссарий

NLP – Natural Language Preproccessing, обработка естественного языка.

Token (токен) – атомарный фрагмент последовательности.

Label (target) – класс целевой переменной.

Multiclass classification (мультиклассовая классификация) – задача классификации наблюдений в один из трех или более классов.

Multilabel classification (мультиклассовая классификация) – задача классификации, когда одно наблюдение может быть отнесено к нескольким классам.

Введение

Информация играет важную и интегрированную роль в современном обществе. В связи с быстрым развитием технологий данные накапливаются в значительных объемах, часто в неудобном для понимания формате. Чтобы эффективно использовать эти ресурсы, они должны быть структурированы таким образом, чтобы обеспечить быстрый и эффективный доступ в случае необходимости. Первоначально текстовый документ состоит из неструктурированных символов, поэтому его необходимо обрабатывать и хранить в удобном для пользователя формате.

Процесс управления неструктурированным материалом включает извлечение из текста ценной информации, которая впоследствии преобразуется в структурированную базу данных.

Одним из подвидов задачи по извлечению структурированной информации из текста является задача по частеречной разметке (Part-of-Speech Tagging), также известная как грамматическая разметка, означает процесс аннотирования слова в тексте (корпусе) определенной части речи в зависимости от ее определения и контекста. Этот процесс упрощен для детей школьного возраста, которых учат различать слова как существительные, глаголы, прилагательные, наречия и т. д.

Если изначально маркировка POS выполнялась вручную, то теперь она используется в области компьютерной лингвистики. Алгоритмы используются для связывания отдельных терминов и скрытых частей речи с описательными тегами. Алгоритмы POS-тегеров можно разделить на две отдельные группы: основанные на правилах и стохастические. Разметка частей речи представляет собой более сложную задачу, чем просто составление списка слов с соответствующими частями речи. Это связано с тем, что некоторые слова могут включать в себя несколько частей речи в зависимости от контекста, а некоторые части речи сложны по своей природе. Это нередкое явление в естественных языках, где значительная часть словоформ, в отличие от многих искусственных языков, неоднозначна.

Данная задача решалась различными методами. Первые подходы к решению данного вопроса основывались на составлении правил вручную. Это требовало обширных знаний о грамматике конкретного языка, что делало подобную систему аннотирования ориентированной лишь на некоторое количество языков.

**Объектом данного исследования** служат синтаксические и грамматические признаки, наделенные определенными свойствами, по которым данные лексемы можно распознать в тексте.

**Предметом исследования** является возможность применения алгоритма автоматического аннотирования текстовых данных на русском языке частеречной разметкой.

**Цель исследования** – разработать модель, способную с хорошей точностью выделять из текста на русском языке скелетную частеречевую структуру.

**Задачи исследования:**

1. Рассмотрение существующих методов частеречной разметки текста;

2. Разработка модели для выполнения частеречной разметки текста на русском языке;

3. Анализ качествамодели;

4. Сравнение результатов, полученных при применении использованных методик и проведении экспериментов;

5. Обобщение полученных данных, формулировка вывода.

**Материалом исследования** является корпус, из 100 предложений на русском языке, размеченных экспертом.

**Методы исследования**: интуитивное выделение необходимых признаков для определения отнесенности слова части речи – представление слов в закодированном виде, моделирование синтаксического уровня при представлении предложения в виде закодированной последовательности, аугментация текстовых данных.

Результатом работы является компьютерная программа, позволяющая распознавать в тексте части речи. Практическая значимость заключается в том, что тегирование текста частями речи (P-о-S) играет важную роль в обработке естественного языка и является фундаментальным компонентом на этапе подготовки данных. Отсутствие этой лингвистической аннотации затрудняет дальнейший анализ текста, в первую очередь из-за неоднозначности значений слов. Этот вычислительный алгоритм используется для решения множества задач, включая машинный перевод, понимание семантики, проверку пунктуации и обнаружение речевых ошибок[10,13,14,16,18].

Глава 1. Задача частеречной разметки

Существующие методы частеречной разметки

Существующие алгоритмы частеречного аннотирования делятся на несколько категорий.

Вероятностные

Метод аннотирования заключается в наблюдении за словами в текстах и подсчету вероятности отнесения его к определенной категории, и при необходимости выполнения предсказания части речи алгоритм обращается к таблице, на основании которой делает выводы. Недостаток метода – необходим большой обьем информации для выявления закономерностей, метод не понимает контекст.

На основе правил

Этот метод основан на установленных правилах, которые были определены эвристически. Первоначально всем терминам присваиваются потенциальные обозначения, основанные на лексиконе. Затем, если термин связан с несколькими названиями, конкретное обозначение определяется в соответствии с заданными правилами. Необходимо разработать множество инструкций, позволяющих устранить все неясности и охватить все возможные сценарии.

Скрытая цепь Маркова

Скрытая цепь Маркова включает в себя скрытые состояния, теги (морфемы), которые приводят к наблюдаемым событиям — терминам. С математической точки зрения цель состоит в том, чтобы идентифицировать серию меток (C), максимизирующих условную вероятность P (C | W), где C = C1, C2,... CT и W = W1, W2,... Применяя теорему Байеса, можно сделать вывод, что максимизация последующего выражения имеет решающее значение: p (C1, C2,... CT) ⋅p (W1, W2,... ВЕС | С1, С2,... ().

Метрики оценки качества разметки

Так как в поставленной задаче мы классифицируем слова, строим сюръективное отображение естественного языка в пространство тегов для оценки качества модели можно использовать метрики пригодные для оценки многоклассовой классификации.

Учитывая также природу языка следует заметить дисбаланс классов в выборке (то есть некоторые части речи встречаются на порядок чаще других) в этих условиях предпочтение стоит следующим метрикам:

Взвешеная F-1 мера

Данная метрика в задаче многоклассовой классификации вычисляется с точки зрения «один-против всех», когда к True Positive - относятся все классифицированные токены, а объекты других классов относятся к False Positive. Формула ниже описывает вычисление F1 меры для отдельно взятого класса.

Формула 1 – Вычисление F-1 меры.

В задаче мульт классовой классификации используется взвешеная F 1 мера, она вычисляется, как взвешеная сумма значений меры на каждом отдельновзятом классе, русунок ниже демонстрирует процесс ее вычисления.

A math equations and formulas on a white background

Description automatically generated

Рисунок 1– Скриншот с онлайн ресурса[20]: F 1.

Balanced accuracy

Balanced accuracy используется как для двоичной, так и для многоклассовой классификации. Он представляет собой среднее математическое значение чувствительности и специфичности. Его можно применять при работе с непропорциональными данными, например, когда одна из объективных категорий преобладает значительно больше, чем другая. Рисунок ниже демонстрирует процесс вычисления метрики.

A white paper with black text and numbers

Description automatically generated

Рисунок 2– Скриншот с онлайн ресурса[20]: Balanced Accuracy.

Глава 2. Обзор литературы

Методы составления векторных представлений

Understanding bag-of-words model: a statistical framework[17]

3/21/24 12:06:00 PMМодель Bag-of-Words (BoW) упрощает текст, представляя слова без учета их порядка или синтаксиса. Каждый документ преобразуется в числовой вектор в зависимости от частоты слов. Преимущества BoW заключаются в простоте, эффективности и возможности извлечения ключевых слов. Однако в нем отсутствует семантическое и контекстное понимание, что снижает насыщенность текста. Он испытывает трудности со словами, выходящими за рамки общепринятого словарного запаса, что затрудняет анализ синтаксиса и грамматики.

Glove: Global Vectors for Word Representation[15]

Модель GloVE решает редкие словарные проблемы, расширяя атрибуты слов и улучшая словесные ассоциации. Она включает контекстные характеристики, улучшающие словесные связи и способствующие конвергенции.

Авторы публикации Glove представили новую структуру, использующую двунаправленные GRU ячейки (реккурентные блоки) для обработки входного текста и контекста. Благодаря встраиванию токенов слов и тегов POS в двунаправленную структуру модель обеспечивает полное представление входных данных. Окончательные скрытые состояния двойных двунаправленных GRU агрегируются и распределяются по слоям для точного прогнозирования типов транзакций.

Преимущества модели GloVE заключаются в устранении редких словарных пробелов, добавлении атрибутов контекста слов и использовании двунаправленных GRU для улучшения конвергенции при встраивании слов.

Enriching Word Vectors with Subword Information[2]

В авторской методологии используется модель скипграммы для представления слов в виде n-граммовых символов, что улучшает векторную информацию слов. Метод быстрый, независимый от предварительной обработки и эффективен при обработке естественного языка. Эта методология эффективна при поиске сходств и аналогий между языками и превосходит другие методы, в которых отсутствует информация о подсловах. Она успешно используется при межъязыковом анализе слов с учетом подслов и нюансов лингвистической структуры. Этот метод особенно полезен на языках с обширным словарным запасом и редкими словами. Он позволяет создавать словесные репрезентации для слов, которых нет в обучающих данных. Эта функция полезна для точного представления недостаточно представленных слов в академическом контексте.

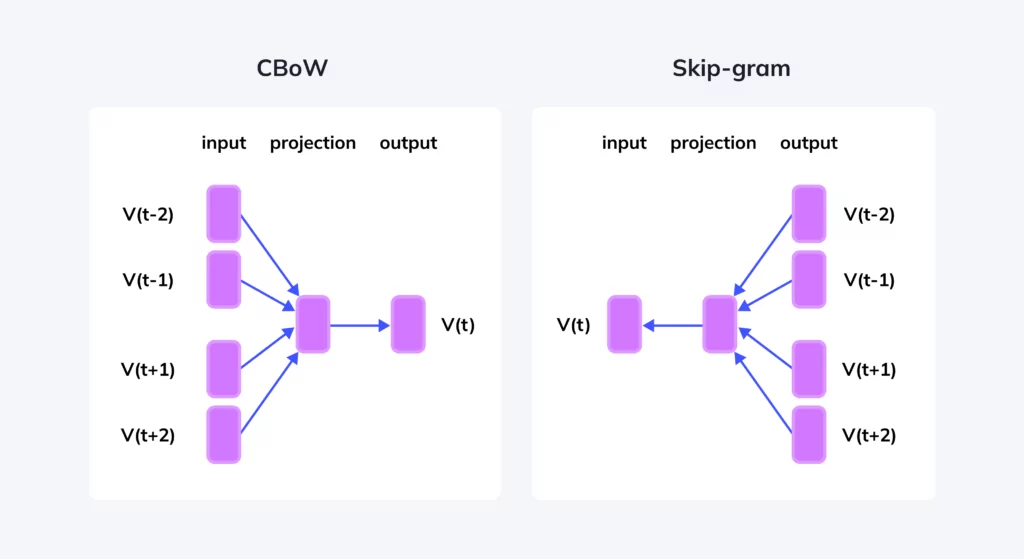


Рисунок 3– Скриншот с интернет ресурса[23]: архитектура механизмов внимания.

Neural Machine Translation of Rare Words with Subword Units[1]

Модели нейронного машинного перевода сталкиваются с трудностями при использовании открытых словарей, но они могут эффективно кодировать редкие термины для повышения эффективности перевода. Различные методы сегментации слов, включая BPE, показывают лучшие результаты по сравнению с базовыми переводами лексиконов. Предложенный подход упрощает перевод, позволяя системе машинного перевода анализировать и синтезировать знания для лучшего перевода необычных терминов. Авторы подтверждают свое предположение о том, что редкие термины можно эффективно сегментировать. Основное внимание уделяется решению проблем перевода открытой лексики путем кодирования редких терминов в виде подслов для повышения производительности и предоставления эмпирических данных в поддержку предлагаемой методологии. Обсуждаются различные методы сегментации слов. Доказано, что модели подслов, использующие сегментацию BPE, превосходят базовые переводы в резервном лексиконе.

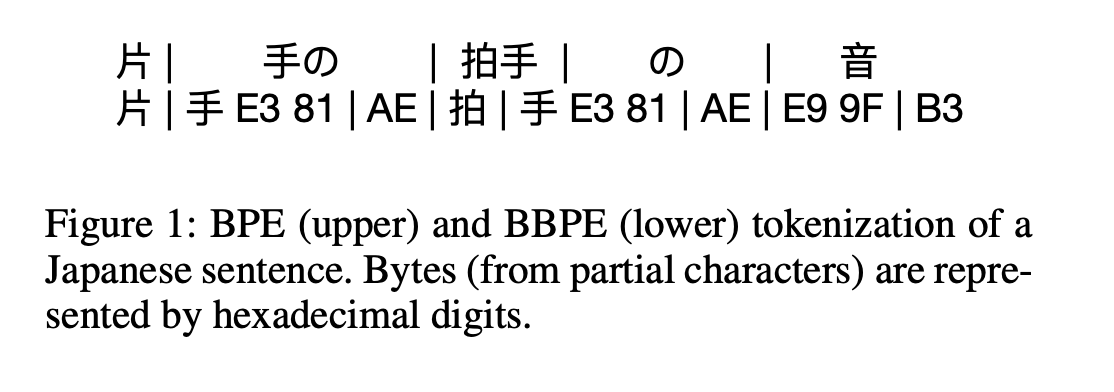
A screenshot of a computer

Description automatically generated

Рисунок 4– Скриншот из онлайн ресурса[22] с примером работы токенизатора BPE.

Neural Machine Translation with Byte-Level Subwords[6]

Подслова на уровне байтов, в частности BPE на уровне байтов (BBPE), представляют собой потенциальное решение проблем, связанных с символьными словарями в машинном переводе. Словарь BBPE сжат сильнее, чем словари символов, и в нем нет недостающих токенов, что повышает эффективность по сравнению с использованием чистых байтов. BBPE позволяет более эффективно распределять словарный запас между языками в многоязычной среде, улучшая качество перевода и повышая производительность. Кроме того, BBPE облегчает перенос моделей между языками с разными наборами символов, расширяя универсальность и применимость моделей машинного перевода. Крайне важно контекстуализировать вложения BBPE с помощью сверточных или повторяющихся слоев, чтобы оптимизировать производительность и добиться точных переводов. Экспериментальные результаты показывают, что BBPE работает так же, как и BPE, но имеет значительно меньший размер, что приводит к снижению вычислительных затрат и повышению эффективности. Символы BBPE обычно более детализированы и короче на уровне байтов, что приводит к сокращению токенизированных последовательностей и ускорению обучения и вывода. Кроме того, BBPE обеспечивает более сбалансированное распределение словарного запаса по частотам, тем самым повышая общее качество перевода. Будущие исследования будут сосредоточены на устранении дисбаланса между исходными и целевыми длинами предложений, оценке BBPE в различных сценариях перевода и изучении усовершенствованных алгоритмов сегментации подслов на уровне байтов.

 Рисунок 5– Картинка из оригинальной статьи[6]: сравнение преимущества токенизации подслов на байтововм уровне.

Модели

Attention is all you need[3]

Архитектура Transformer, предложенная в статье, представляет собой новую нейросетевую архитектуру, которая заменяет традиционные рекуррентные или сверточные слои механизмами вниманияб чтто показало заметный прирост качества и производительности при решении задачи машинного перевода.

A diagram of a multi-head attention

Description automatically generated

Рисунок 6– Картинка из оригинальной статьи[3]: архитектура механизмов внимания.

Модель состоит из 2-х ключевых блоков Энкодер и Декодер, далее рассмотрим внутреннее устройство энкодера. Энкодер Transformer состоит из ряда одинаковых блоков, каждый из которых оснащен механизмом самовнимания с несколькими головами и полносвязными слоями, предназначенной для конкретной позиции. Благодаря этому механизму самовнимания каждое положение энкодера может совпадать со всеми предыдущими позициями слоя, фиксируя взаимозависимости между различными элементами входной последовательности.

A diagram of a process flow

Description automatically generated

Рисунок 7– Картинка из оригинальной статьи[3]: архитектура сети трансформер.

Благодаря оценке показателей внимания с помощью механизма самовнимания, использующего векторные сходства, ключи и значения запросов, полученные на основе выходных данных предыдущего слоя, модель может эффективно оценивать важность различных входных элементов и собирать информацию различных типов.

Затем енкодер генерирует серию непрерывных представлений, инкапсулирующих информацию из входной последовательности, которую декодер использует для генерации выходной последовательности.

В отличие от архитектур, использующих повторяющиеся или свёрточные уровни, эта модель отличается высокой степенью параллелизации, что позволяет ускорить обучение, а продолжительность обучения значительно меньше, чем у предыдущих моделей, что делает её эффективной для сложных задач обучения.

Рисунок ниже представляет собой Heat-map (от черного к белому по возрастанию) демонстрирующий зависимости между токенами последовательности:

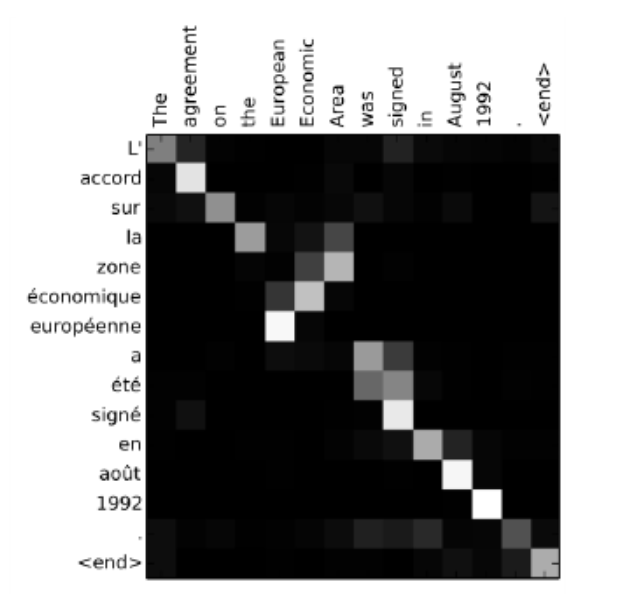


Рисунок 8–Взаимосвязи токенов в предложении на heatmap при переводе текста с английского на французский в задаче машинного перевода.

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding[5]

BERT — это система лингвистических репрезентаций, позволяющая взглянуть на немаркированный текст с глубоких двунаправленных точек зрения, учитывая левый и правый контексты на всех уровнях. Структура BERT основана на архитектуре Transformer, который представляет собой многослойный двунаправленный кодировщик. Обучение BERT включает в себя два неконтролируемых задания: маскированное лингвистическое моделирование (MLM) и предсказание последующих предложений (NSP).

A diagram of training and fine-tuning

Description automatically generated

Рисунок 9– Картинка из оригинальной статьи[5]: архитектура механизмов внимания.

A diagram of a single sentence

Description automatically generated

Рисунок 10– Картинка из оригинальной статьи[5]: дообучения BERT для задачи Token Classification.

A diagram of a training model

Description automatically generated

Рисунок 11– Картинка из оригинальной статьи[5]: архитектура сети BERT.

BERT готов работать с большим количеством немаркированных документов, таких как Wikipedia и BookScorpus, и может быть адаптирован для выполнения определенных последующих задач с использованием помеченных данных. Подготовленные точки зрения BERT можно перенести на последующие задачи, что снижает необходимость в фреймворках, ориентированных на конкретные задачи. BERT обеспечивает исключительную эффективность при выполнении целого ряда задач по обработке естественного языка, включая ответы на запросы и лингвистическую продукцию. Эффективность BERT в снижении необходимости в фреймворках, ориентированных на выполнение задач, и достижении максимальной эффективности была подтверждена в научной литературе. В целом, структура модели BERT обеспечивает всестороннее представление о контексте и коннотации языка, а процедура обучения обеспечивает возможность переноса знаний и высокую эффективность при выполнении разнообразных задач.

Performance of Transfer Learning Model vs. Traditional Neural Network in Low System Resource Environment[8]

A diagram of a learning model

Description automatically generated

Рисунок 12– Картинка из оригинальной статьи[8]: архитектура решения задачи с переносом обучения

В статье сравниваются эффективность и стоимость более легких архитектур трансферного обучения и традиционных нейронных сетей для категоризации текста и идентификации именованных сущностей (NER). Исследование показало, что более легкие модели трансферного обучения, примером которых являются модели, разработанные на основе BERT, XLnet и GPT, способствуют сокращению затрат вычислительных ресурсов и необходимых данных для обучения.

A Family of Pretrained Transformer Language Models for Russian[9]

В статье представлены 13 моделей русского языка трансформеров (LM) для русского языка, включая модели Rubert, RuRoberta, Ru-Electra, ruGPT-3, RuT5 и FRED-T5.

- Модели предварительно обучаются на разнообразном корпусе, включая тексты из общедоступных ресурсов, таких как Википедия, OpenSubtitles.

Эффективность моделей оценивается на основе данных по пониманию русского естественного языка и генерации наборов данных и тесты SuperGlue и RuCola.

A table with numbers and text

Description automatically generated

Рисунок 13– Картинка из оригинальной статьи[9]: сравнение обученных моделей по архитектуре, числу обучаемых параметров, использованным токенизаторам.

Глава 2. Разработка модели

Проектирование модели частеречной разметки

Далее рассмотрим шаги, предприняты на этапе проектирования архитектуры модели для решения поставленной задачи. Основываясь на знаниях, полученных из статей рассмотренных в этой 1 главе, был сделан вывод, о том что использование модели уже предобученой на различных текстах является наиболее перспективным вариантом для построения решения задачи. State-of-art моделями в области языкового моделирования и понимания естественного языка являются модели с архитектурой BERT, что еще более важно, модель должна понимать языковой контекст (должна быть обучена на корпусах целевого языка), наличие этих факторов значительно снижает количество вариантов моделей, подходящих для решения задач.[4,5]

Анализ входных данных

Данные от эксперта получены в виде текстового файла с расширением “.docx”. В таком виде данные не могут быть использованы для обучения, поэтому было принято решение о их переформатировании, где последовательность слов на русском языке остается неизменной, а частеречная разметка конвертируется в последовательность тегов. Такая последовательность может быть обработана и переведена в векторное представление.

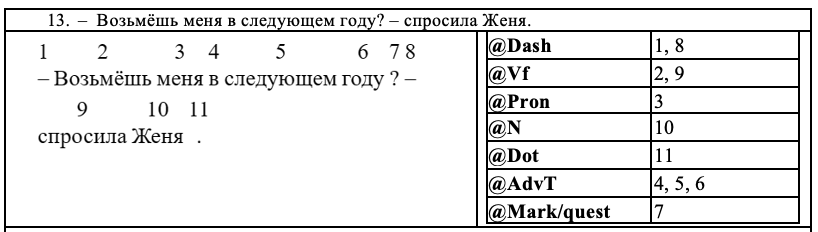


Рисунок 15– Пример входных данных, полученных от эксперта.



Рисунок 16– Пример входных данных.

Выбор базовой модели

Основываясь на утверждении о важности сегментации морфем на словарном уровне (источник) для русского языка, а также минимизируя потерю информации на этапе предобработки данных и приведения элементов последовательности к единой форме был сделан выбор в пользу семейства моделей предобученых на корпусах текстов на русском (источник).

В качестве базовой модели решения выбрана RuBERT-base[9] с токенизатором BPE. Среди особенностей модели хочется выделить ее устройство:

- предобученый Токенизатор – BPE.

- Модель обучена на задаче маскированого языкового моделирования, в процессе обучения модель извлекает из последовательности фиксированной длинны признаки необходимые для предсказания замаскированого токена, расположенного на случайной позиции в последовательности.

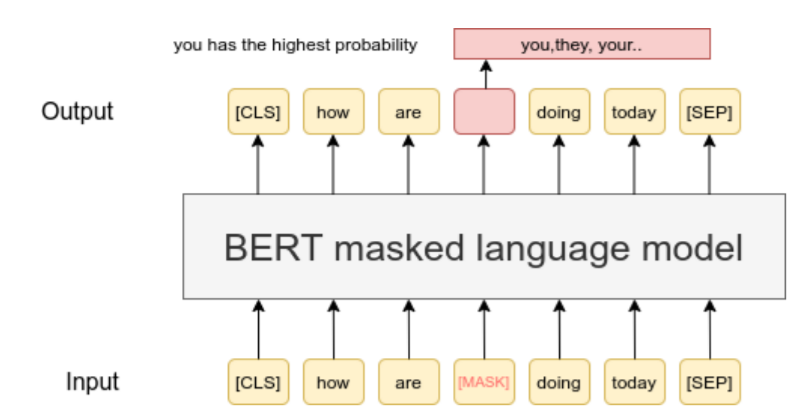


Рисунок 17 – Картинка с онлайн рисунка[21]: Маскированное языковое моделирование.

Теперь вернемся к обсуждению предобработки текста, учитывая выбранную модель в качестве базовой. Так как RuBERT-base использует токенизатор с открытым словарем (предоставляет возможность закодировать любую последовательность, а в задаче нанесения частеречной разметки должны быть предсказаны классы для все токенов входящей последовательности необходимо сохранить максимум информации для каждого токена входящей последовательности, учитывая эти условия было принято решение отказаться от нормализации и очистки входной последовательности, за исключением ручной проверки результатов экспертной разметки (на этом этапе были устранены опечатки и , что вполне обосновано использованием предобученой модели с токенизотором BPE.

A screenshot of a computer screen

Description automatically generated

Рисунок 18– Пример обработки текста.

Перенос обучения

Как описано в оригинальной статье, BERT можно легко дообучить под решение узкой задачи, для этого добавим к предобученой русскоязычной модели полносвязный слой и определим функцию активации Softmax, на выходе модели получим закодированные значения классов токенов, которые затем раскодируем во время постобработки.

Softmax - многомерная логистическая функция преобразует вектор z размера K в сигма-вектор одинакового размера, где каждый компонент sigma\_i результирующего вектора обозначается действительным числом в диапазоне от 0 до 1, а сумма компонентов равна 1, что позволяет интерпретировать значения как на выходе модели, как вероятность отнесения к конкретному классу.

Формула 2 – Функция активации Softmax.

Далее, на рисунке ниже изображена упрощенная схема работы модели, на рисунке – подробное устройство модели.

A diagram of software components

Description automatically generated

Рисунок 19– Картинка из оригинальной статьи[17]: дообучение BERT для задачи Token Classification.



Рисунок 20– Архитектура модели.

Обучение модели

Далее обучим модель на задачу Token Classification с учителем, будем оценивать качество после каждой эпохи и при помощи оптимизатора Adam корректировать веса модели методом обратного распространения ошибки.

С тренировочными параметрами:

* Количество эпох обучения: 45;
* Learning rate: 2\*10^-5;
* Batch size: 64.

Оценка результатов

В результате обучение на выборке с размерами:

- train: 60 наблюдений;

- validation: 20 наблюдений;

- test: 20 наблюдений.

Были получены следующие результаты, при вычислении при помощи уже разработанного модуля[11]:

**Balanced Accuracy = 0.212**

**Weighted F-1 = 0.014**

Вывод: модель недообучена, данных для обучения недостаточно, некоторые из классов не встретились в тренировочной выборке и даже увеличения количества эпох недосточно для получения удовлетворительного качества.

A graph with a line and numbers

Description automatically generated

Рисунок 21– Оценка работы модели.

Глава 3. Публикация и развертывание модели

Модель опубликована на бесплатном облачном хостинге Hugging Face (см. Приложение Б), спроектировано и развернуто демонстрационное веб-приложение, с которым можно взаимодействовать в браузере, перейдя по ссылке в Приложении Г.

Кроме того, качество модели было оценено по метрикам, описанным во Введении, с результатами тестирования можно ознакомиться в Приложении И.

Демо приложение разработано при помощи Python фреймворка Gradio, позволявшим разрабатывать удобные веб интерфейсы для взаимодействия с моделями. Приложение протестировано при помощи юнит и интеграционных тестов (см. Приложение К), так как использует REST запросы к модели, развернутой на бесплатном облачном хостинге.

Глава 4. Эксперименты

В ходе обучения модели были получены неудовлетворительные результаты, выявлен факт недообучения модели, который можно решить либо увеличением выборки. Однако, производить аннотирование данных вручную весьма трудоемкий процесс, значит, что необходимо менять метод обучения, применять методы reinforcement-learning или semi-supervised[12] или увеличить выборку при помощи техник аугментации[7].

Среди техник аугментации при работе с текстовыми данными имеются:

- замена слов синонимами (этот метод применим к задаче PoS, так как синонимы – слова одной части речи);

- добавление возможных опечаток во входные данные, тем самым добавляя случайный шум;

- нарезка текста на фрагменты (chunking).

Для начала был выбран последний подход, предложения тестовой выборки были нарезаны на фрагменты при помощи скользящего окна размером [1; количество слов в предложении].

Таким образом удалось увеличить тренировочную выборку в более чем в 300 раз, на рисунке ниже приведена статистика по полученным данным, а также балансе частей речи.

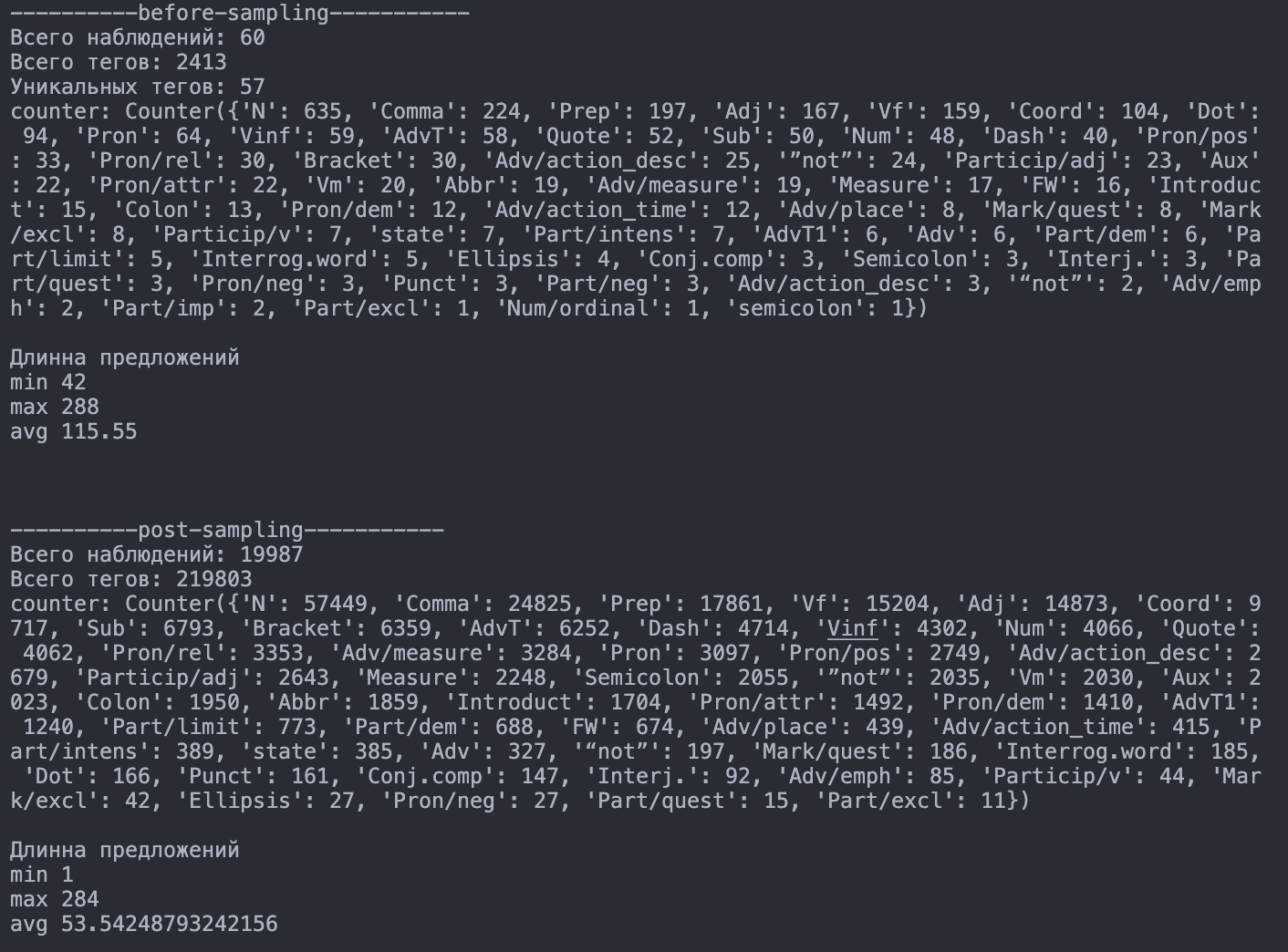


Рисунок 22 – Анализ входных данных, сравнение наблюдений до нарезки и после.

Далее модель была обучена на новых тренировочных данных, с теми же фрагментами для тестирования и валидации.

Тренировочные параметры были скорректированы из-за выросшей трудоемкости вычислений, а именно было сокращено число эпох обучения до 5.

Результаты, полученные после обучения такой модели, показали значительный прирост качества решения.

**F1: 0.8642**

**Accuracy: 0.8822**

A graph on a white background

Description automatically generated

Рисунок 23– Оценка работы модели, обученной на нарезанных данных.

Такой результат можно считать хорошим, так как во время экспертного аннотирования текста тоже случаются ошибки, как уже было отмечено в Главе 2.

Кроме того, был поставлен эксперимент со сменой базовой модели, в качестве новой была выбрана модель с бОльшим количеством параметров, обученая на той же архитектуре: ruBERT-large[9]. Результаты модели обученной с теми же параметрами на рисунке ниже.

A screenshot of a graph

Description automatically generated

Рисунок 24– Оценка работы модели: ruBERT-large.

Результат эксперимента – увеличение модели (рост числа обучаемых гиперпараметров) не дал ощутимого прироста к точности модели и не имеет смысла в данной реализации.

Заключение

Целью данной работы разработка модели, способной с хорошей точностью выделять из текста на русском языке скелетную частеречевую структуру.

На первом этапе работы была описана литература и методы использованные на этапе разработки модели.

Оценка качества работы модели проводилась при помощи метрик классификации Weighted F-1 и Balanced Accuracy. Важно признать, что обучение и экзамен проводились на относительно ограниченном наборе данных, поэтому результаты могут отличаться в зависимости от набора данных.

Во время второго этапа разработки модели были обучены две модели глубокого обучения. Была выдвинута гипотеза: модель недообучена из-за наличия малого объема размеченных данных, в ходе эксперимента с нарезкой тестовой выборки скользящим окном гипотеза была подтверждена, модель обучена снова и качество значительно улучшено. Была выдвинута и другая гипотеза: качество предсказаний можно улучшить за счет увеличения количества обучаемых гиперпараметров, однако в ходе второго эксперимента гипотеза была опровергнута.

Данную работу можно продолжать за счёт увеличения объёма тренировочных данных и подбора гиперпараметров нейронной сети для повышения качества классификации, а также: разработать аналогичную модель для мультиязычной классификации частей речи, использования модели в качестве ядра приложения, анализирующего скелетную структуру предложений при интеллектуальном анализе текстовых данных.

Библиографический список

1. Sennrich R., Haddow B., Birch A. Neural Machine Translation of Rare Words with Subword Units // 2016.

2. Bojanowski P. [и др.]. Enriching Word Vectors with Subword Information // 2017.

3. Vaswani A. [и др.]. Attention Is All You Need // 2023.

4. Devlin J. [и др.]. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding // 2019.

5. Tenney I., Das D., Pavlick E. BERT Rediscovers the Classical NLP Pipeline // 2019.

6. Wang C., Cho K., Gu J. Neural Machine Translation with Byte-Level Subwords // 2019.

7. Thakur N. [и др.]. Augmented SBERT: Data Augmentation Method for Improving Bi-Encoders for Pairwise Sentence Scoring Tasks // 2021.

8. Hui W. Performance of Transfer Learning Model vs. Traditional Neural Network in Low System Resource Environment // 2020.

9. Zmitrovich D. [и др.]. A Family of Pretrained Transformer Language Models for Russian // 2023.

10. Ailamazyan Program Systems Institute of RAS, Pereslavl-Zalessky, 152020, Russian Federation, Trofimov I. V. Automatic Morphological Analysis for Russian: Application-Oriented Survey // PROGRAMMNAYA INGENERIA. 2019. № 9–10 (10). C. 391–399.

11. Hiroki Nakayama Seqeval: A Python framework for sequence labeling evaluation [Электронный ресурс]. URL: https://github.com/chakki-works/seqeval.

12. Liao W., Veeramachaneni S. A simple semi-supervised algorithm for named entity recognition Boulder, Colorado: Association for Computational Linguistics, 2009.C. 58–65.

13. Mylnikova A. V., Trusov V. A., Mylnikov L. A. Use of Text Skeleton Structures for the Development of Semantic Search Methods // Automatic Documentation and Mathematical Linguistics. 2023. № 5 (57). C. 301–307.

14. Novikova A. Direct Machine Translation and Formalization Issues of Language Structures and Their Matches by Automated Machine Translation for the Russian-English Language Pair 2018.

15. Pennington J., Socher R., Manning C. Glove: Global Vectors for Word Representation Doha, Qatar: Association for Computational Linguistics, 2014.C. 1532–1543.

16. Sapin A. S. Building neural network models for morphological and morpheme analysis of texts // Proceedings of the Institute for System Programming of the RAS. 2021. № 4 (33). C. 117–130.

17. Wei Q. [и др.]. Relation Extraction from Clinical Narratives Using Pre-trained Language Models // AMIA ... Annual Symposium proceedings. AMIA Symposium. 2019. (2019). C. 1236–1245.

18. Zhang Y., Jin R., Zhou Z.-H. Understanding bag-of-words model: a statistical framework // International Journal of Machine Learning and Cybernetics. 2010. № 1–4 (1). C. 43–52.

19. INTELLECTUAL ANALYSIS OF DATA ON THE BASIS OF STANFORD CoreNLP FOR POS TAGGING OF TEXTS IN THE RUSSIAN LANGUAGE // Systems and Means of Informatics. 2018.

20. Classical ML Equations in LaTeX [Электронный ресурс]. URL: https://blmoistawinde.github.io/ml\_equations\_latex/.

21. MLM [Электронный ресурс]. URL: https://www.sbert.net/examples/unsupervised\_learning/MLM/README.html.

22. Summary of the tokenizers [Электронный ресурс]. URL: https://huggingface.co/docs/transformers/tokenizer\_summary.

23. what-is-word2vec-and-how-does-it-work [Электронный ресурс]. URL: https://swimm.io/learn/large-language-models/what-is-word2vec-and-how-does-it-work.

Приложение А. Список тегов для русского языка

not - Неизменяемая частица при отрицании

Abbr - Аббревиатуры

Adj - Прилагательное

Adv - Наречие

Adv/action\_desс - Наречие образа действия

Adv/action\_time - Наречие времени действия

Adv/measure - Наречие меры

Adv/place - Наречие места

Adv/emph - Усилительные наречия

AdvT - Наречия определенного и неопределенного временного периода

AdvT1 - Наречия определенной и неопределенной частоты

Aux - Вспомогательный и связочный глаголы

Bracket - Скобки

Colon - Двоеточие

Comma - Запятая

Conj.comp - Союз сравнения

Coord - Сочинительные союзы

Dash - Тире

Dot - Точка

Ellipsis - Троеточие

FW - Слово на иностранном языке

Interj. - Междометие

Interrog.word - Вопросительное слово

Introduct - Вводные слова

Mark/excl - Знак восклицания

Mark/quest - Вопросительный знак

Measure - Слова меры

N - Существительное

Num - Числительное

Num/ordinal - Порядковое числительное

Part/dem - Указательные частицы

Part/excl - Восклицательные частицы

Part/limit - Ограничительные частицы

Part/imp - Побудительные частицы

Part/intens - Усилительные частицы

Part/neg - Отрицательные частицы

Part/quest - Вопросительные частицы

Part/soft - Частицы смягчения требования

Prep - Предлог

Pron - Местоимение

Pron/attr - Местоимение определительное

Pron/dem - Местоимение указательное

Pron/neg - Местоимение отрицательное

Pron/pos - Местоимение притяжательное

Pron/rel - Местоимение относительное

Particip/adj - Причастие

Particip/v - Деепричастие

Punct - Пунктуационный знак

Quote - Кавычки

semicolon - Двоеточие

state - Слова категории состояния

Sub - Подчинительные союзы

Vf - Изменяемая часть смыслового глагола

Vinf - Описание предиката с помощью инфинитивов

Vm - Описание предиката с помощью глаголов модального значения

Приложение Б. Ссылка на репозиторий модели

Ссылка на репозиторий модели: <https://hf.co/disk0dancer/ru-bert-base-pos>

Приложение В. Ссылка на репозиторий набора данных

Ссылка на репозиторий: <https://hf.co/datasets/disk0dancer/ru_sentances_pos>

Приложение Г. Ссылка на Демо-приложение

Ссылка на Демо-приложения: <https://huggingface.co/spaces/disk0dancer/demo-rubert-base-finetuned-pos>

Приложение Д. Техническое задание

См. Приложения «Техническое задание».

Приложение Е. Программа и методика испытаний

См. Приложения «Программа и методика испытаний».

Приложение Ж. Текст программы

Ссылка на репозиторий с кодом программы: <https://github.com/disk0dancer/rubert-finetuned-pos>

Приложение И. Оценка качества модели

**Результаты тестирования модели ruBert-base обученой на оригинальном датасете.**

{'\_': {'precision': 0.375, 'recall': 0.028037383177570093, 'f1': 0.05217391304347826, 'number': 107}, 'ark/excl': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 1}, 'ark/quest': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 1}, 'art/excl': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 0}, 'art/imp': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 0}, 'art/intens': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 3}, 'art/limit': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 0}, 'art/neg': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 1}, 'art/quest': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 1}, 'articip/adj': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 2}, 'articip/v': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 1}, 'ash': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 3}, 'bbr': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 2}, 'dj': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 38}, 'dv': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 1}, 'dv/action\_desс': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 3}, 'dv/action\_time': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 0}, 'dv/emph': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 1}, 'dv/measure': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 5}, 'dv/place': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 1}, 'dvt': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 5}, 'dvt1': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 0}, 'easure': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 5}, 'emicolon': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 0}, 'f': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 34}, 'inf': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 12}, 'm': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 4}, 'nterj.': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 0}, 'nterrog.word': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 1}, 'ntroduct': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 1}, 'olon': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 3}, 'omma': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 38}, 'onj.comp': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 0}, 'oord': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 17}, 'ot': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 21}, 'racket': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 8}, 'rep': {'precision': 0.15384615384615385, 'recall': 0.04081632653061224, 'f1': 0.06451612903225806, 'number': 49}, 'ron': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 17}, 'ron/attr': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 4}, 'ron/dem': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 3}, 'ron/neg': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 0}, 'ron/pos': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 6}, 'ron/rel': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 8}, 'tate': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 1}, 'ub': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 4}, 'um': {'precision': 0.07692307692307693, 'recall': 0.13333333333333333, 'f1': 0.0975609756097561, 'number': 15}, 'um/ordinal': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 1}, 'unct': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 1}, 'uote': {'precision': 0.125, 'recall': 0.16666666666666666, 'f1': 0.14285714285714288, 'number': 6}, 'ux': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 3}, 'w': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 5}, 'overall\_precision': 0.018691588785046728, 'overall\_recall': 0.01805869074492099, 'overall\_f1': 0.018369690011481053, 'overall\_accuracy': 0.022964509394572025}

**Результаты тестирования модели ruBert-base обученой на датасетеб нарезаном при помощи метода скользящего окна.**

{'\_': {'precision': 0.92, 'recall': 0.8598130841121495, 'f1': 0.888888888888889, 'number': 107}, 'ark/excl': {'precision': 1.0, 'recall': 1.0, 'f1': 1.0, 'number': 1}, 'ark/quest': {'precision': 1.0, 'recall': 1.0, 'f1': 1.0, 'number': 1}, 'art/dem': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 0}, 'art/intens': {'precision': 1.0, 'recall': 0.3333333333333333, 'f1': 0.5, 'number': 3}, 'art/neg': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 1}, 'art/quest': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 1}, 'articip/adj': {'precision': 0.5, 'recall': 0.5, 'f1': 0.5, 'number': 2}, 'articip/v': {'precision': 1.0, 'recall': 1.0, 'f1': 1.0, 'number': 1}, 'ash': {'precision': 0.75, 'recall': 1.0, 'f1': 0.8571428571428571, 'number': 3}, 'bbr': {'precision': 0.5, 'recall': 1.0, 'f1': 0.6666666666666666, 'number': 2}, 'dj': {'precision': 0.7272727272727273, 'recall': 0.8421052631578947, 'f1': 0.7804878048780488, 'number': 38}, 'dv': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 1}, 'dv/action\_desс': {'precision': 0.2, 'recall': 0.3333333333333333, 'f1': 0.25, 'number': 3}, 'dv/emph': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 1}, 'dv/measure': {'precision': 0.4444444444444444, 'recall': 0.8, 'f1': 0.5714285714285714, 'number': 5}, 'dv/place': {'precision': 1.0, 'recall': 1.0, 'f1': 1.0, 'number': 1}, 'dvt': {'precision': 0.6666666666666666, 'recall': 0.8, 'f1': 0.7272727272727272, 'number': 5}, 'dvt1': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 0}, 'easure': {'precision': 0.3333333333333333, 'recall': 0.4, 'f1': 0.3636363636363636, 'number': 5}, 'f': {'precision': 0.9428571428571428, 'recall': 0.9705882352941176, 'f1': 0.9565217391304348, 'number': 34}, 'inf': {'precision': 1.0, 'recall': 1.0, 'f1': 1.0, 'number': 12}, 'm': {'precision': 1.0, 'recall': 0.5, 'f1': 0.6666666666666666, 'number': 4}, 'nterrog.word': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 1}, 'ntroduct': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 1}, 'olon': {'precision': 1.0, 'recall': 1.0, 'f1': 1.0, 'number': 3}, 'omma': {'precision': 1.0, 'recall': 1.0, 'f1': 1.0, 'number': 38}, 'oord': {'precision': 1.0, 'recall': 1.0, 'f1': 1.0, 'number': 17}, 'ot': {'precision': 0.9545454545454546, 'recall': 1.0, 'f1': 0.9767441860465117, 'number': 21}, 'racket': {'precision': 1.0, 'recall': 1.0, 'f1': 1.0, 'number': 8}, 'rep': {'precision': 0.9787234042553191, 'recall': 0.9387755102040817, 'f1': 0.9583333333333333, 'number': 49}, 'ron': {'precision': 1.0, 'recall': 1.0, 'f1': 1.0, 'number': 17}, 'ron/attr': {'precision': 0.6666666666666666, 'recall': 0.5, 'f1': 0.5714285714285715, 'number': 4}, 'ron/dem': {'precision': 1.0, 'recall': 0.6666666666666666, 'f1': 0.8, 'number': 3}, 'ron/pos': {'precision': 1.0, 'recall': 1.0, 'f1': 1.0, 'number': 6}, 'ron/rel': {'precision': 0.875, 'recall': 0.875, 'f1': 0.875, 'number': 8}, 'tate': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 1}, 'ub': {'precision': 0.5, 'recall': 1.0, 'f1': 0.6666666666666666, 'number': 4}, 'um': {'precision': 0.8, 'recall': 0.8, 'f1': 0.8000000000000002, 'number': 15}, 'um/ordinal': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 1}, 'unct': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 1}, 'uote': {'precision': 1.0, 'recall': 1.0, 'f1': 1.0, 'number': 6}, 'ux': {'precision': 1.0, 'recall': 1.0, 'f1': 1.0, 'number': 3}, 'w': {'precision': 1.0, 'recall': 0.4, 'f1': 0.5714285714285715, 'number': 5}, 'overall\_precision': 0.873589164785553, 'overall\_recall': 0.873589164785553, 'overall\_f1': 0.873589164785553, 'overall\_accuracy': 0.8893528183716075}

**Результаты тестирования модели ruBert-large обученой на датасетеб нарезаном при помощи метода скользящего окна.**

{'\_': {'precision': 0.900990099009901, 'recall': 0.8504672897196262, 'f1': 0.875, 'number': 107}, 'ark/excl': {'precision': 1.0, 'recall': 1.0, 'f1': 1.0, 'number': 1}, 'ark/quest': {'precision': 1.0, 'recall': 1.0, 'f1': 1.0, 'number': 1}, 'art/intens': {'precision': 1.0, 'recall': 0.3333333333333333, 'f1': 0.5, 'number': 3}, 'art/limit': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 0}, 'art/neg': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 1}, 'art/quest': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 1}, 'articip/adj': {'precision': 0.25, 'recall': 0.5, 'f1': 0.3333333333333333, 'number': 2}, 'articip/v': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 1}, 'ash': {'precision': 1.0, 'recall': 1.0, 'f1': 1.0, 'number': 3}, 'bbr': {'precision': 0.3333333333333333, 'recall': 0.5, 'f1': 0.4, 'number': 2}, 'dj': {'precision': 0.7948717948717948, 'recall': 0.8157894736842105, 'f1': 0.8051948051948051, 'number': 38}, 'dv': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 1}, 'dv/action\_desс': {'precision': 0.14285714285714285, 'recall': 0.3333333333333333, 'f1': 0.2, 'number': 3}, 'dv/emph': {'precision': 1.0, 'recall': 1.0, 'f1': 1.0, 'number': 1}, 'dv/measure': {'precision': 0.6666666666666666, 'recall': 0.8, 'f1': 0.7272727272727272, 'number': 5}, 'dv/place': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 1}, 'dvt': {'precision': 0.8, 'recall': 0.8, 'f1': 0.8000000000000002, 'number': 5}, 'dvt1': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 0}, 'easure': {'precision': 0.3333333333333333, 'recall': 0.4, 'f1': 0.3636363636363636, 'number': 5}, 'f': {'precision': 0.9428571428571428, 'recall': 0.9705882352941176, 'f1': 0.9565217391304348, 'number': 34}, 'inf': {'precision': 1.0, 'recall': 1.0, 'f1': 1.0, 'number': 12}, 'm': {'precision': 1.0, 'recall': 0.5, 'f1': 0.6666666666666666, 'number': 4}, 'nterrog.word': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 1}, 'ntroduct': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 1}, 'olon': {'precision': 1.0, 'recall': 1.0, 'f1': 1.0, 'number': 3}, 'omma': {'precision': 1.0, 'recall': 1.0, 'f1': 1.0, 'number': 38}, 'oord': {'precision': 1.0, 'recall': 1.0, 'f1': 1.0, 'number': 17}, 'ot': {'precision': 0.9545454545454546, 'recall': 1.0, 'f1': 0.9767441860465117, 'number': 21}, 'racket': {'precision': 1.0, 'recall': 1.0, 'f1': 1.0, 'number': 8}, 'rep': {'precision': 0.9795918367346939, 'recall': 0.9795918367346939, 'f1': 0.9795918367346939, 'number': 49}, 'ron': {'precision': 1.0, 'recall': 1.0, 'f1': 1.0, 'number': 17}, 'ron/attr': {'precision': 0.75, 'recall': 0.75, 'f1': 0.75, 'number': 4}, 'ron/dem': {'precision': 1.0, 'recall': 0.6666666666666666, 'f1': 0.8, 'number': 3}, 'ron/pos': {'precision': 1.0, 'recall': 1.0, 'f1': 1.0, 'number': 6}, 'ron/rel': {'precision': 0.875, 'recall': 0.875, 'f1': 0.875, 'number': 8}, 'tate': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 1}, 'ub': {'precision': 0.5, 'recall': 1.0, 'f1': 0.6666666666666666, 'number': 4}, 'um': {'precision': 0.8, 'recall': 0.8, 'f1': 0.8000000000000002, 'number': 15}, 'um/ordinal': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 1}, 'unct': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 1}, 'uote': {'precision': 1.0, 'recall': 1.0, 'f1': 1.0, 'number': 6}, 'ux': {'precision': 1.0, 'recall': 1.0, 'f1': 1.0, 'number': 3}, 'w': {'precision': 1.0, 'recall': 0.4, 'f1': 0.5714285714285715, 'number': 5}, 'overall\_precision': 0.8792710706150342, 'overall\_recall': 0.871331828442438, 'overall\_f1': 0.8752834467120182, 'overall\_accuracy': 0.8914405010438413}

Приложение И. Тестирование веб приложения

Код интеграционных тестов:

|  |
| --- |
| import os |
| import pytest |
| import requests |
| from dotenv import load\_dotenv |
|  |
| load\_dotenv() |
|  |
| API\_URL = os.environ.get("API\_URL") |
| AUTH\_TOKEN = os.environ.get("AUTH\_TOKEN") |
| headers = {"Authorization": f"Bearer {AUTH\_TOKEN}"} |
|  |
|  |
| def test\_api(): |
| responce = requests.post(API\_URL, headers=headers, json='{"существительное"}') |
| assert responce.status\_code > 0 |

Код модульных тестов:

|  |
| --- |
| import pytest |
| from app import postproccess |
|  |
| def test\_postprocess(): |
| assert postproccess([]) == '' |
| assert postproccess([{"entity\_group": '1'}, {"entity\_group": '2'}]) == "1 2" |