Оглавление

[ВВЕДЕНИЕ 2](#_Toc169905307)

[ГЛАВА 1. ПОГРУЖЕНИЕ В ПРОЕКТ 4](#_Toc169905308)

[1.1 ВЫБОР ТЕМЫ ПРАКТИКИ 4](#_Toc169905309)

[1.2. ОПИСАНИЕ ПРОЕКТА 4](#_Toc169905310)

[1.3. ЗАДАНИЕ НА ПРАКТИКУ 10](#_Toc169905311)

[ГЛАВА 2. РАЗРАБОТКА 11](#_Toc169905312)

[2.1. АРХИТЕКТУРА ПРИЛОЖЕНИЯ 11](#_Toc169905313)

[2.2. ФОРМИРОВАНИЕ ДАТАСЕТА 12](#_Toc169905314)

[2.3. СРАВНИТЕЛЬНЫЙ АНАЛИЗ МОДЕЛЕЙ 14](#_Toc169905315)

[2.4 ОПИСАНИЕ РАБОТЫ WEB-ПРИЛОЖЕНИЯ 15](#_Toc169905316)

[ГЛАВА 3. РЕЗУЛЬТАТЫ РАБОТЫ 17](#_Toc169905317)

[3.2. ПОЛУЧЕННЫЕ МЕТРИКИ 17](#_Toc169905318)

[3.3. WEB-ПРИЛОЖЕНИЕ 17](#_Toc169905319)

[3.3. ТЕСТИРОВАНИЕ ПРИЛОЖЕНИЯ 18](#_Toc169905320)

[ГЛАВА 4. ОПИСАНИЕ КОМАНДНОЙ РАБОТЫ 19](#_Toc169905321)

[4.1 ОПИСАНИЕ КОМАНДЫ 19](#_Toc169905322)

[4.2 ОРГАНИЗАЦИЯ КОМАНДНОЙ РАБОТЫ 19](#_Toc169905323)

[ЗАКЛЮЧЕНИЕ 21](#_Toc169905324)

[БИБЛИОГРАФИЧЕСКИЙ СПИСОК 22](#_Toc169905325)

[ПРИЛОЖЕНИЯ 24](#_Toc169905326)

|  |  |  |
| --- | --- | --- |
| Ответственный за соответствие содержания и оформления текста отчёта предъявляемым требованиям | (подпись) | Ступников Андрей Анатольевич |

# ВВЕДЕНИЕ

Тип практики: технологическая (проектно-технологическая)

Дата начала: 26.02.2024

Дата окончания: 26.06.2024

Цель практики – получение первичных профессиональных умений и навыков, в том числе первичных умений и навыков научно-исследовательской деятельности, обучающихся по направлению подготовки 02.03.03 Математическое обеспечение и администрирование информационных систем (уровень бакалавриата).

Задачи учебной практики:

* Закрепление теоретических и практических знаний, умений и навыков, полученных на первом и втором курсах обучения;
* Выполнение заданий, предусмотренных программой практики и назначенных руководителем;
* Освоение отдельных компьютерных программ и информационных систем, используемых в профессиональной деятельности;
* Получение навыков работы с периодическими, реферативными и справочными информационными изданиями по прикладной математике и информационным системам;
* Подготовка к осознанному и углубленному изучению дисциплин старших курсов;
* Подготовка и защита в установленный срок отчета по практике.

В результате выполнения технологической (проектно-технологической) учебной практики студент должен обладать следующими компетенциями:

* Анализ и обработка больших данных
* Machine learning
* Создание веб-приложения с помощью фреймворка Flask

# ГЛАВА 1. ПОГРУЖЕНИЕ В ПРОЕКТ

## 1.1 ВЫБОР ТЕМЫ ПРАКТИКИ

Участникам нашей команды была интересна сфера машинного и, в частности, глубокого обучения. Данный проект основывается обработке естественного языка (NLP) – эта тема является одной из наиболее актуальных в сфере машинного обучения. Задача проекта была ясна. Мы владели навыками алгоритмического и объектно-ориентированного программирования, управления проектами, однако мы не владели компетенциями машинного обучения. В ходе практики мы стремились на освоить машинное обучение, веб-разработку, анализ данных.

## 1.2. ОПИСАНИЕ ПРОЕКТА

Связность текста – это признак текста, при котором следующее предложение строится на базе предыдущего с помощью языковых средств [План, средства связи в тексте]. Связность также является важным содержательным критерием, от которого зависит качество его реферирования [Белогорская, Резанова, с. 16] и сложность восприятия читателем [Соловьев, Вольская, Андреева, Заикин, с. 539]. Инструменты для оценки данного критерия могут найти применение в области обработки естественного языка, в частности при анализе сложности восприятия текста читателем.

Существующие сервисы для английского языка используют различные подходы, включающие семантические сети (WordNet), дистрибутивную семантику (LSA, word2vec), эвристические подходы подсчета повторяющихся слов. При этом сервисы для русского языка с открытым описанием их принципа работы используют только последний подход, не учитывая словосочетания производных слов, родовидовые и ссылочные отношения, которые также являются средствами связности.

Цель работы – разработать веб-приложение для оценки локальной связности перевода на русский язык с помощью модели машинного обучения, учитывающей синтаксические, лексические и морфологические средства связи.

Задачи проекта:

* Сформировать и разметить датасет.
* Провести сравнительный анализ моделей машинного обучения
* Создать приложение, предоставляющее пользователю оценку связности введённого им текста.
* Внедрить модель для оценки локальной связности текста.

1.2.1. ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ

Предметная область нашего проекта – средства связи в русском языке. В русском языке выделяют следующие средства связи:

1. **Лексические:**
   * синонимы – слова схожие по смыслу
   * антонимы – слова противоположные по смыслу
   * слова одной тематической группы – это слова, обладающие общностью лексического значения и обозначающие сходные, но не одинаковые понятия.
   * лексический повтор – намеренный повтор одинаковых словоформ
   * однокоренные слова – слова, которые имеют одинаковый корень
2. **Морфологические:**
   * Союз — это служебная часть речи, которую используют, чтобы связать однородные члены предложения, части сложного предложения или части текста.
   * Союзные слова — слова, которые связывают части сложноподчинённого предложения и при этом выступают в роли одного из членов предложения
   * Частица — это служебная часть речи, которая служит для выражения оттенков значений слов, словосочетаний, предложений и для образования форм слов
   * **Местоимение — это самостоятельная часть речи, которая объединяет слова, которые не содержат конкретного лексического значения, не называют предметы, признаки или количество, а лишь указывают на них**
   * Наречие — неизменяемая самостоятельная часть речи, обозначающая признак действия, качества, другого признака или предмета
   * **Форма слова** — это:
     1. Морфологическая разновидность слова, несущая комплекс тех или иных значений, присущих данной части речи грамматических категорий.
     2. Совокупность морфологических и фонологических характеристик слова (формальных показателей, в том числе нулевых), указывающих на его грамматическое значение — принадлежность к определённым грамматическим категориям.
   * единство видовременных форм глаголов - использование одинаковых форм грамматического времени, которые указывают на одновременность или последовательность ситуаций.
3. **Синтаксические:**
   * Параллелизм — это схожее или одинаковое построение расположенных рядом предложений или отрезков речи,
   * Парцелляция - стилистический прием расчленения в поэтическом произведении фразы на части или даже на отдельные слова; цель П. — придать речи интонационную экспрессию путем ее отрывистого произнесения,
   * вводные слова — это **слово (или словосочетание), которое входит в состав предложения, но не вступает с его членами в синтаксическую связь**

1.2.2. ПОСТАНОВКА ЗАДАЧИ

Дана выборка , элементы которой представляют пары предложений , где – два неповторяющихся предложения и бинарные оценки . D разделяется на На входе модель для обучения получает набор векторных представлений для , dim() = m, , m и набор оценок . На выходе модель выдает набор вещественных оценок , определяющих характер взаимосвязи между значениями целевых переменных и векторных представлений , в котором каждая оценка принадлежит отрезку от 0 до 1. Необходимо построить модель машинного обучения, находящую неизвестное отображение

1.2.3. ОПИСАНИЕ ТЕХНОЛОГИЙ

Для проведения сравнительного анализа рассматривались модели машинного обучения на базе архитектуры transformer BERT (rubert-base-cased [Kuratov, Arkhipov] и sbert\_large\_nlu\_ru [Обучение модели естественного языка с BERT и Tensorflow]), градиентного спуска (CatBoostRegressor, CatBoostClassifier [CatBoostClassifier]), байесовского подхода (Bayessian Ridge), метода LASSO (LassoLarsCV) и сетей с долгой краткосрочной памятью LSTM.

Мы освоили использование этих моделей. Среди них были модели (CatBoostRegressor, LassoLarsCV), архитектура которых изучалась в рамках дисциплины АДиОМО. Также встречались модели незнакомой нам архитектуры BERT, LSTM, принципы работы которых достаточно сложны и их изучение выходит за рамки практики.

Для векторизации данных обучающей и тестовой выборки применялись разные виды представлений, такие как: эмбеддинги sentence transformers [Reimers N., Gurevych I], tf-idf, усредненные по эмбеддингам слов векторы elmo, fasttext [Bojanowski P., Grave E., Joulin A., Mikolov T.] и word2vec.

Ембеддинг слова (word embedding) – общее название для векторного представления слова из естественного языка. Самый простой способ векторизации – one-hot encoding. Вектор некоторого слова i из словаря S – вектор длинны , где на i-том месте стоит 1, а на остальных позициях 0. Недостаток данного способа заключается в том, что полученные векторы обладают большой размерностью, также вектор слова никак не связан со смыслом слова. Мы не использовали этот способ векторизации напрямую, но он применяется в других методах.

Способ векторизации word2vec использует нейронные сети и основывается на контексте слова в предложении. Для обучения word2vec используются предобработанный текст (или набор текстов). С помощью скользящего окна размера 2n+1 слов, для каждого слова в тексте определяются другие слова, стоящие к нему на расстоянии не более n. Такие слова для исходного слова считаются близкими по контексту и расстояние между их векторами будет невелико. На основании такой разметки обучается нейронная сеть.

S – множество слов из текста (текстов). – множество слов, находящихся в тексте от слова i на расстоянии не более n, где . На вход нейросети word2vec подаётся one-hot encoding вектор слова i, а также правильный ответ в виде one-hot encoding вектора слова . Нейросеть word2vec устроена следующим образом:

Где A – матрица размера B – матрица размера ; j – искомый вектор, состоящий из вещественных чисел в диапазоне [0, 1], отражающий степень контекстной близости слов. Во время обучения . Задача обучения сводится к нахождению коэффициентов матриц A и B. Для каждого эта операция производится раз.

Недостатком word2vec является то, что с его помощью не могут быть представлены слова, не встречающиеся в обучающей выборке. Fasttext построен на схожих идеях, но призван решить проблемы word2vec.

В fasttext слова делятся на n-граммы по буквам скользящим окном, которые поступают на вход нейросети. Вектором слова в таком случае является сумма векторов его n-грамм. Однако такой подход требует больше вычислительных ресурсов.

Word2vec и fasttext не учитывают порядок слов в предложении, в попытке это исправить к задаче векторизации применили идею рекуррентных нейронных сетей. Elmo – модель для векторизации, основанная на архитектуре LSTM. Elmo учитывает не только контекст слова, но и его место в предложении, что помогает разделить векторы многозначных слов.

Более продвинутая технология transformer, применяемая в моделях архитектуры BERT, состоит из пары кодировщик – декодировщик, в каждом из которых используется механизм внимания – метод поиска связи между входными и выходными данными.

TF-IDF – метод векторизации, основывающийся на выделении ключевых

слов в тексте. TF (Частота термина) обозначает, насколько часто определенное слово появляется в данном тексте. IDF (Обратная частота документа) измеряет, насколько уникально слово является по всей коллекции текстов.

- Частота термина для слова "t" в тексте "d".

- Обратная частота текста (документа) для слова "t".

Подход TF-IDF учитывает важность слова в конкретном тексте, относительно других текстов, а также устраняет шум – слова, встречаемые очень часто во всех текстах. Однако такой метод не учитывает семантической информации, контекста и чувствителен к размерам текста.

Для предобработки теста использовались библиотеки NLTK, ruwordnet. NLTK обладает большим функционалом для обработки естественного языка. Мы применяли эту библиотеку для токенизации – выделения предложений в тексте. Библиотека ruwordnet содержит в себе базу синонимов, гипонимов гиперонимов, а также производных слов русского языка.

Для взаимодействия с пользователем мы решили использовать web-приложение. Чтобы его реализовать, мы использовали HTML, CSS, JavaScript с технологией ajax, Python c библиотекой Flask.

Flask – микрофреймворк для создания веб-приложений, предоставляющий базовые возможности: сервер и отладчик, поддержка cookie и сессий, RESTful routing.

HTML – язык разметки гипертекста, предназначенный для описания структуры документа.

CSS – формальный язык для описания внешнего вида документа, предназначен для гибкого и красивого представления структуры документа, а также для разделения логической структуры страницы и ее внешнего вида.

JavaScript – мультипарадигменный язык программирования. В основном используется в браузерах для того, чтобы пользователь мог комфортно взаимодействовать со страницей. Технология ajax используется для фонового обмена данными между пользователем и сервером, что позволяет клиентам отправлять и получать данные без обновления страниц.

## 1.3. ЗАДАНИЕ НА ПРАКТИКУ

Сформировать и разметить датасет. Требуется найти тексты переводов на русский язык, разбить их на предложения, перемешать предложения из разных текстов и разбить предложения на пары. Для каждой пары предложений вычислить значение целевой переменной.

Провести сравнительный анализ. Обучить модели, вычислить метрики, выявить модель с лучшими метриками. При проведении сравнительного анализа требуется использовать различные способы представлений. Найти оптимальные гиперпараметры модели.

Создать приложение, предоставляющее пользователю оценку связности введённого им текста. Требуется описать структуру и стили документа, разработать backend и script для отправки формы.

Внедрить модель оценки локальной связности текста, для последующего использования в приложении. Требуется интегрировать обученную модель в веб-приложение.

# ГЛАВА 2. РАЗРАБОТКА

# 2.1. АРХИТЕКТУРА ПРИЛОЖЕНИЯ

Файловая структура приложения:

* static
  + css
    - app.css
    - bootstrap.min.css
  + js
    - app.js
    - jquery-3.4.1.min.js
* templates
  + index.html
* app.py
* config.py
* process\_sent\_pairs.py
* best\_model

app.css – файл с описанием используемых стилей для index.html.

bootstrap.min.css – библиотека стилей и изображений.

app.js – файл, реализующий технологию ajax для отправки формы с предложениями (документа, текста), без обновления страницы.

jquery-3.4.1.min.js – библиотека ajax.

index.html – файл отвечающий за структуру основного документа.

app.py – основной файл откуда запускается web-приложение, здесь описаны get запросы, для получения пользователем основной страницы и скачивания csv файла с ответом на запрос пользователя. Также здесь описаны post запрос, для обработки формы с предложениями, отправленных пользователем.

config.py – файл с описанием параметров приложения.

process\_sent\_pairs.py – файл с функциями оценки пар предложений по критериям связности.

best\_model – файл, содержащий данные обученной модели *rubert\_base\_cased .*

## 2.2. ФОРМИРОВАНИЕ ДАТАСЕТА

Нами был сформирован собственный датасет на основе 3503 текстов публицистической и информационной направленности корпуса учебных переводов [Kutuzov, Kunilovskaya, c. 362]. Исходные тексты с помощью токенизатора библиотеки NLTK были разбиты на предложения, затем предложения были перемешаны в случайном порядке. Такой подход нужен: во-первых, потому что мы собираемся вычислять связность для пар предложений; во-вторых, потому что в исходном тексте пары предложений будут связаны всегда, тогда как для обучения моделей требуются и не связанные предложения тоже. На основе полученного набора предложений был составлен список пар соседних предложений объемом 91169 элементов.

Для получения столбца целевой переменной были реализованы 5 методов поиска отношений связности между словами внутри одной пары предложений на языке Python. Каждый метод возвращает 1 при успешном поиске и 0 в случае неуспешного. Признаками наличия отношений являются случаи, когда одно из слов для другого является:

1. гипонимом () — понятие, выражающее частную сущность по отношению к другому, более общему понятию;
2. гиперонимом () — слово с более широким значением, выражающее общее, родовое понятие;
3. производным () — слово, образованное, произведённое от какого-либо другого слова или словосочетания;
4. формой слова ();
5. отсылкой (ссылкой) для другого ().

Общим этапом в работе реализованных методов является разбиение предложений на токены.

Методы для нахождения признаков 1–3 используют библиотеку ruwordnet [9], предоставляющую интерфейс для работы с одноименным тезаурусом. Он хранит информацию о синсетах – группах синонимичных слов, бинарно соответствующих другим группам посредством разметки: «гипонимы–гиперонимы», «производные–производящие» и т. д. Для поиска синсетов токены были приведены к нормальным формам, исключены знаки пунктуации. Условием окончания работы методов являлось наличие нормальной формы конкретного токена в плоском списке синсетов соседнего предложения.

Признаки 4 – 5 рассчитываются с использованием модуля stem библиотеки NLTK и морфологического анализатора pymorphy2. Для токенов соседних предложений создаются 2 списка стемов (основ слов) . Если , то метод возвращает 1, в противном случае – 0. Метод поиска возможной анафорической связи с помощью экземпляра морфологического анализатора MorphAnalyzer находит тэги частей речи, числа и рода для токенов без знаков пунктуации. Если в списке токенов второго предложения есть местоимение, совпадающее по роду и числу с существительным из первого списка, метод возвращает 1, в противном случае 0.

На основе признаков по формуле: формируется значение целевой переменной оценки связности. Два предложения считаются связными, если целевая переменная равна 1, и не являются связными, если целевая переменная равна 0.

Итоговый набор содержит 13002 пары предложений ввиду того, что при отборе учитывались только пары, в которых представлен только один из признаков или ни одного. Таким образом, в датасете 6501 пара имеет хотя бы одно средство связности, другая половина – нет. Данные обучающей и тестовой выборок делятся в пропорции 80:20, для избежания предвзятости модели в пользу одного из классов, классы в выборках были сбалансированы.

Для обучения моделей требовалось перевести пары предложений в векторы. Эмбеддинги sentence transformers, tf-idf предоставляли готовый функционал для векторизации предложений, тогда как модели elmo, fasttext, word2vec предоставляли готовый функционал только для векторизации слов из предложения. Здесь перед нами возникла задача: требуется из эмбеддингов каждого слова в предложении получить эмбеддинг для предложения. Эмбеддинг предложения должен отражать смысл предложения. Есть два подхода к решению этой проблемы: суммировать эмбеддинги отдельных слов, усреднять эмбеддинги отдельных слов. Мы реализовали оба этих способа и заметили, что суммирование хуже работает для предложений разной длинны. Таким образом модуль вектора длинного предложения будет в разы больше, чем вектор короткого предложения. Требуется, чтобы векторы предложений с одним смыслом были близки к друг другу (для определения дистанции используется косинусное расстояние). В следствии чего был выбран метод усреднения эмбеддингов отдельных слов для получения эмбеддингов предложений. При таком подходе векторы схожих по смыслу предложений будут близки, независимо от длинны этих предложений.

## 2.3. СРАВНИТЕЛЬНЫЙ АНАЛИЗ МОДЕЛЕЙ

Нам требуется оценить степень локальной связности текста с помощью модели машинного обучения. Для оценки локальной связности текста нужно оценить связность для каждых двух идущих подряд предложений. Можно выделить два различных подхода:

1. оценивать степень связности вещественным числом в промежутке [0, 1] и решать задачу регрессии;

2. давать бинарную оценку связности предложений и решать задачу классификации.

Мы реализовали оба подхода. При реализации первого подхода были обучены модели: rubert-base-cased и sbert\_large\_nlu\_ru, CatBoostRegressor, Bayessian Ridge, LassoLarsCV, LSTM. Для оценки качества моделей, обученных для решения регрессионной задачи, мы использовали метрики MSE и MAE. Оказалось, что полученные модели по немного лучше константной и обладают низкой обобщающей способностью. Поэтому, мы отказались от использования первого подхода.

При реализации второго подхода были обучены модели: rubert-base-cased, sbert\_large\_nlu\_ru, CatboostClassifier, SupportVectorMachine, LogisticRegression, LSTM. Для оценки качества моделей, обученных для решения задачи бинарной классификации, мы выбрали метрику F1 score, как основную. Так как, исходя из данной предметной области, нам не требуется особым образом контролировать ложноположительные или ложноотрицательные результаты. Полученные модели показали неплохие метрики на тестовой выборке. Было принято решение использовать второй подход.

Такие результаты можно объяснить тем, что в качестве правильных оценок моделям были поданы на вход бинарные оценки, но, при этом, в первом подходе мы ожидали получить от моделей вещественные оценки. Это приводило к тому, что модели во время обучения некорректно рассчитывали функцию потерь, в следствии чего модуль вычисленного ими градиента был неверен, что приводило к неадекватной корректировке весов нейронов.

## 2.4 ОПИСАНИЕ РАБОТЫ WEB-ПРИЛОЖЕНИЯ

Основной файл app.py, в котором описаны все маршруты, также благодаря которому пользователь взаимодействует с приложение. Здесь описано три роутера:

1. Роутер, который отправляет основную страницу. Эту функция возвращает отрендеренный файл index.html
2. Роутер, который обрабатывает POST запрос. Здесь происходит проверка текста, его обработка, токенизация, а потом обработка с помощью модели, которая подключается при запуске приложения.
3. Роутер, который отправляет csv файл с отчетом пользователю. Здесь идет формирование отчета и его отправка с помощью форм.

Для удобства пользователя была использована технология ajax, которая позволяет серверу и клиенту обмениваться данными в фоновом режиме, благодаря этому при отправке формы или документа с текстом, пользователю не нужно ждать загрузки страницы вновь, лишь подождать обработку текста. Для ее работы также в этой же функции реализована отправка текста из документа, а не полного документа, что облегчает нагрузку сети. Чтобы это сделать, нужно из страницы пользователя выгрузить файл или текст. И отправить, используя метод POST на адрес, указанный в роутере, далее программа обработает данные и вернет ответ клиенту, функция обработает полученную информацию и отобразит ответ на странице пользователя в виде таблицы.

# ГЛАВА 3. РЕЗУЛЬТАТЫ РАБОТЫ

## 3.2. ПОЛУЧЕННЫЕ МЕТРИКИ

В таблице 1 приведены 7 моделей, показавших лучшие результаты валидации на тестовой выборке.

Таблица 1

Результаты валидации моделей на тестовой выборке

|  |  |  |  |
| --- | --- | --- | --- |
| Модель | F1 | Recall | Precision |
| DeepPavlov/rubert-base-cased | 0,738 | 0,731 | 0,745 |
| ai-forever/sbert\_large\_nlu\_ru | 0,722 | 0,727 | 0,717 |
| CatboostClassifier + fasttext | 0,66 | 0,681 | 0,641 |
| SVC + fasttext | 0,66 | 0,661 | 0,664 |
| LogisticRegression + sentence-transformers | 0,63 | 0,63 | 0,63 |
| SVC + elmo | 0.65 | 0.65 | 0.65 |
| LSTM + word2vec | 0,666 | 0,673 | 0,704 |

## 3.3. WEB-ПРИЛОЖЕНИЕ

Для демонстрации работы модели было разработано веб-приложение с использованием фреймворка Flask.

Фронтенд написан с использованием HTML, CSS и JS. Модель интегрируется в приложение с помощью библиотеки simpletransformers Взаимодействие с пользователем осуществляется через точки API посредством HTTP GET и POST запросов. Для получения предсказаний пользователь выбирает текстовый файл посредством диалогового окна, по кнопке «Загрузить файл» файл сохраняется в хранилище для его дальнейшего использования моделью. После нажатия “Получить предсказание” начинается процесс обработки данных моделью, после завершения на экран также выводятся средние значения признаков. Пользователь также может скачать отчет в формате csv. Интерфейс приложения представлен на рис. 1.

A screenshot of a computer

Description automatically generated

*Рис. 1.* Интерфейс приложения

## 3.3. ТЕСТИРОВАНИЕ ПРИЛОЖЕНИЯ

Во время тестирования мы обнаружили, что модель *rubert\_base\_cased* при оценке связности не учитывает значение многозначных слов. Так предложения «Как приготовить автомобиль к поездке?» и «Нужно его пожарить.» модель считает связными, не учитывая другое значение слова «приготовить». Так произошло из-за того, что библиотеки, применяемые нами для разметки, не учитывали различия в значении многозначных слов.

# ГЛАВА 4. ОПИСАНИЕ КОМАНДНОЙ РАБОТЫ

## 4.1 ОПИСАНИЕ КОМАНДЫ

|  |  |  |  |
| --- | --- | --- | --- |
| ФИО | Дрожащих Григорий Алексеевич | Зимин Андрей Валерьевич | Жилин Андрей Игоревич |
| Фото |  |  |  |
| Интересы | Разработка чат-ботов, Data science, NLP, Computer Vision | Machine Learning, Deep Learning, NLP, CV | Web-разработка, CV |

## 4.2 ОРГАНИЗАЦИЯ КОМАНДНОЙ РАБОТЫ

Командная работа была организована в гибридном формате, иногда проводились очные встречи, иногда онлайн встречи. На оффлайн встречах, наш куратор – Григорий, ставил задачи, объяснял детали реализации поставленных задач и осуществлял общее руководство процессом разработки. На онлайн встречах у нас была возможность обсудить дальнейшие действия, а также поработать удаленно в комфортной обстановке. Командная работа была организована с помощью таких средств как Discord, Google Collab и GitHub. Discord – платформа для связи, Google Collab – облачная платформа для создания и выполнения кода на Python, GitHub – сервис для совместной разработки.

Совместные задачи:

1. Формирование датасета

Задачи, назначенные Андрею Зимину:

1) Выполнение предобработки данных с помощью различных средств векторизации текста;

2) Обучение ML-моделей, проведение сравнительного анализа.

Задачи, назначенные Андрею Жилину:

1) Разработка web-приложения;

2) Интеграция ML-модели в web-приложение.

Календарный план был организован следующим образом, было составлено общее расписание и в свободное время были назначены общие встречи.

Григорий Дрожащих, отведенная роль – тимлид, благодаря его руководству, удалость достигнуть поставленных задач в срок, а также изучить новую для нас на тот момент область.

Андрей Зимин, отведенная роль – ML-разработчик, при выполнении задач он показал глубокие познания в смежных сферах его задач, а также быстро освоил необходимый материал.

Андрей Жилин, отведенная роль – web-разработчик, показал наличие практических и теоретических навыков в своей сфере, также показал заинтересованность в сфере ML-разработки.

# ЗАКЛЮЧЕНИЕ

Основные задачи, которые были выполнены:

1. Проведен сравнительный анализ моделей, по результатам которого модель rubert-base-cased показала наилучшие метрики
2. Найдены оптимальные гиперпараметры модели rubert-base-cased
3. Разработано web-приложение, в которое была встроена модель rubert-base-cased.

Планы для улучшения проекта:

1. Дообучение модели на новых данных
2. Использование дополнительных видов связи в предложении, для оценки связности текста
3. Добавление новых возможностей в веб-приложение, таких как авторизация и хранение текстов на длительных срок

# БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. План, средства связи в тексте // Foxford. URL: https://foxford.ru/wiki/russkiy-yazyk/plan-sredstva-svyazi-v-tekste (дата обращения: 20.06.2024).
2. Белогорская Д.В., Резанова 3.И. Лингвистическая оценка автоматически сгенерированных рефератов новостных текстов // Язык и культура. 2023. № 61. C. 15-28. doi: 10.17223/19996195/61/2.
3. Соловьев В.Д., Вольская Ю.А., Андреева М.И., Заикин А.А. Словарь русского языка с индексами конкретности/абстрактности // Вестник РУДН. Серия: Лингвистика. 2022. № 2. URL: https://cyberleninka.ru/article/n/slovar-russkogo-yazyka-s-indeksami-konkretnosti-abstraktnosti (дата обращения: 20.06.2024).
4. Kuratov Y., Arkhipov M. Adaptation of deep bidirectional multilingual transformers for Russian language // Komp'juternaja Lingvistika i Intellektual'nye Tehnologii. 2019. С. 333-339.
5. Обучение модели естественного языка с BERT и Tensorflow // Habr. URL: https://habr.com/ru/companies/sberdevices/articles/527576/ (дата обращения: 09.06.24).
6. CatBoostClassifier // catboost.ai. URL: https://catboost.ai/en/docs/concepts/python-reference\_catboostclassifier (дата обращения: 09.06.24).
7. Reimers N., Gurevych I. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. URL: https://arxiv.org/abs/1908.10084 (дата обращения: 09.06.24)
8. Bojanowski P., Grave E., Joulin A., Mikolov T. Enriching Word Vectors with Subword Information. URL: https://arxiv.org/abs/1607.04606 (дата обращения: 09.06.24)
9. Kutuzov A.B., Kunilovskaya M.A., Chepurkova A.Y., Oschepkov A.Y. Russian Learner Parallel Corpus as a Tool for Translation Studies // Компьютерная лингвистика и интеллектуальные технологии: труды XVIII Международной конференции «Диалог 2012»: в 2-х томах, Бекасово, 30 мая – 03 июня 2012 года. Выпуск 11 (18). Том 1. Бекасово: Российский государственный гуманитарный университет, 2012. Pp. 362-369.

# ПРИЛОЖЕНИЯ

Приложение 1.

Зимин А.В, программный код на языке Python:

import string

import pandas as pd

import numpy as np

from gensim.models.doc2vec import Doc2Vec, TaggedDocument

from nltk import word\_tokenize

from sklearn.svm import SVR

from sklearn.preprocessing import StandardScaler

from sentence\_transformers import SentenceTransformer

from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error, r2\_score

import warnings

import warnings

warnings.filterwarnings('ignore')

import nltk

import numpy as np

df = pd.read\_csv(‘.\\data\\balanced\_dataset.csv')

np.random.seed(42)

msk = np.random.rand(len(df)) < 0.8

df\_train = df[msk]

df\_eval = df[~msk]

train\_sents = []

for i in df\_train.sents\_l:

    train\_sents.append(i)

for j in df\_train.sents\_r:

    train\_sents.append(j)

test\_sents = []

Приложение 2.

for i in df\_eval.sents\_l:

    test\_sents.append(i)

for j in df\_eval.sents\_r:

    test\_sents.append(j)

tagged\_data = [

    TaggedDocument(words=[w.lower() for w in word\_tokenize(\_d) if w not in string.punctuation], tags=[str(i)]) for i, \_d

    in enumerate(train\_sents)]

model250 = Doc2Vec(vector\_size=250)

model250.build\_vocab(tagged\_data)

model300 = Doc2Vec(vector\_size=300)

model300.build\_vocab(tagged\_data)

X250\_train.columns = list(range(500))

X250\_test.columns = list(range(500))

X300\_train.columns = list(range(600))

X300\_test.columns = list(range(600))

from sklearn.utils import all\_estimators

def get\_all\_regressors\_sklearn():

    estimators = all\_estimators(type\_filter='regressor')

    all\_regs = []

    for name, RegClass in estimators:

        print('Appending', name)

        try:

            if name != 'QuantileRegressor':

                reg = RegClass()

                all\_regs.append(reg)

        Приложение 3.

except Exception as e:

            pass

    return all\_regs

all\_regs = get\_all\_regressors\_sklearn()

for reg in all\_regs:

    try:

        reg.fit(X250\_train, y\_train)

        print(type(reg), reg.score(X250\_test, y\_test))

    except Exception:

        print(type(reg), 'no')

sentence\_transformer = SentenceTransformer("DeepPavlov/rubert-base-cased")

train\_sents\_embs = sentence\_transformer.encode(train\_sents)

test\_sents\_embs = sentence\_transformer.encode(test\_sents)

train\_sents\_embs\_df = pd.concat([pd.DataFrame(train\_sents\_embs[:len(train\_sents\_embs) // 2]),

                 pd.DataFrame(train\_sents\_embs[len(train\_sents\_embs) // 2:])], axis=1)

test\_sents\_embs\_df = pd.concat([pd.DataFrame(test\_sents\_embs[:len(test\_sents\_embs) // 2]),

                 pd.DataFrame(test\_sents\_embs[len(test\_sents\_embs) // 2:])], axis=1)

from sklearn.linear\_model import BayesianRidge

from sklearn.linear\_model import LassoLarsCV

br = BayesianRidge()

lr = LassoLarsCV()

lr.fit(train\_sents\_embs\_df, y\_train)

lr.score(test\_sents\_embs\_df, y\_test)

br.fit(train\_sents\_embs\_df, y\_train)

br.score(test\_sents\_embs\_df, y\_test)

import logging

Приложение 4.

from simpletransformers.classification import ClassificationModel, ClassificationArgs

logging.basicConfig(level=logging.INFO)

transformers\_logger = logging.getLogger("transformers")

transformers\_logger.setLevel(logging.WARNING)

train\_df = pd.DataFrame(df\_train)

train\_df.columns = ["text\_a", "text\_b", "labels"]

eval\_df = pd.DataFrame(df\_eval)

eval\_df.columns = ["text\_a", "text\_b", "labels"]

model\_args = ClassificationArgs(num\_train\_epochs=1)

model\_args.save\_best\_model = True

model\_args.learning\_rate = 3e-5

model\_args.train\_batch\_size = 8

model\_args.eval\_batch\_size = 8

model\_args.num\_train\_epochs =  1

model\_args.regression = True

model\_args.use\_early\_stopping = True

model\_args.early\_stopping\_delta = 0.01

model\_args.early\_stopping\_metric = "r2"

model\_args.early\_stopping\_metric\_minimize = False

model\_args.early\_stopping\_patience = 5

model\_args.evaluation\_during\_training = True

model\_args.evaluate\_during\_training\_steps = 100

model\_args.overwrite\_output\_dir = True

model\_args.evaluate\_during\_training = True

model\_args.manual\_seed = 42

model\_args.overwrite\_output\_dir = True

model = ClassificationModel("bert", #roberta

                            Приложение 5.

"DeepPavlov/rubert-base-cased", # ai-forever/ruRoberta-large, ai-forever/sbert\_large\_nlu\_ru,

                            num\_labels=1,

                            args=model\_args,

                            use\_cuda=True,

                            ignore\_mismatched\_sizes=True,

                            )

model.train\_model(train\_df=train\_df,

                  eval\_df=eval\_df,

                  mae=mean\_absolute\_error,

                  mse=mean\_squared\_error,

                  r2=r2\_score,

                  )

model = RepresentationModel("bert", "DeepPavlov/rubert-base-cased",

                            num\_labels=1,

                            args=model\_args,

                            use\_cuda=True,

                            ignore\_mismatched\_sizes=True)

train\_sents\_embs = model.encode\_sentences(train\_sents, combine\_strategy="mean")

test\_sents\_embs = model.encode\_sentences(test\_sents, combine\_strategy="mean")

train\_sents\_embs\_df = pd.concat([pd.DataFrame(train\_sents\_embs[:len(train\_sents\_embs) // 2]),

                 pd.DataFrame(train\_sents\_embs[len(train\_sents\_embs) // 2:])], xaaxis=1)

test\_sents\_embs\_df = pd.concat([pd.DataFrame(test\_sents\_embs[:len(test\_sents\_embs) // 2]),

                 pd.DataFrame(test\_sents\_embs[len(test\_sents\_embs) // 2:])], axis=1)

test\_sents\_embs\_df = pd.concat([pd.DataFrame(test\_sents\_embs[:len(test\_sents\_embs) // 2]),

                 pd.DataFrame(test\_sents\_embs[len(test\_sents\_embs) // 2:])], axis=1)

Приложение 6.

Жилин А.И, программный код на языке Python:

import copy

import csv

import os

import re

import string

import time

import numpy as np

from nltk import pos\_tag, SnowballStemmer

from ruwordnet import RuWordNet

# import spacy\_streamlit

os.environ["TOKENIZERS\_PARALLELISM"] = "false"

import flask

from flask import Flask, request, render\_template, send\_from\_directory

import json

import nltk

# import ru\_core\_news\_sm

# nltk.download('punkt')

import pymorphy2

from nltk.corpus import stopwords

from simpletransformers.classification import ClassificationModel, ClassificationArgs

from process\_sent\_pairs import \*

app = Flask(\_\_name\_\_)

app.config['UPLOAD\_FOLDER'] = f'{app.root\_path}/reports'

@app.route('/')

Приложение 7.

def index():

    return render\_template('index.html')

@app.route('/predict', methods=['POST'])

def predict():

    if 'sentence\_from\_file' in request.json:

        text = request.json['sentence\_from\_file']

        if text != '':

            sentences = nltk.sent\_tokenize(text, language='russian')

            if len(sentences) > 1:

                sentences = [s.replace('\n', '') for s in sentences]

                tmp\_report = []

                for i in range(1, len(sentences)):

                    tmp\_preds = []

                    tmp\_preds.append(float(MODEL.predict([[sentences[i - 1], sentences[i]]])[0]))

                    for p in predictors:

                        tmp\_preds.append(p(sentences[i - 1], sentences[i]))

                    tmp\_report.extend([[sentences[i - 1], sentences[i], \*tmp\_preds]])

                    preds.append(tmp\_preds)

                preds\_ = np.array(preds)

                reports.append(tmp\_report)

                response = {}

                response['response'] = {

                    'mean\_score': float("{:.5f}".format(np.mean(preds\_[:, 0]))),

                   Приложение 8.

'mean\_is\_wordfrom\_rep\_in\_sent\_pairs': np.mean(preds\_[:, 1]),

                    'mean\_is\_deriv\_in\_sent\_pairs': np.mean(preds\_[:, 2]),

                    'mean\_is\_hyponym\_in\_sent\_pairs': np.mean(preds\_[:, 3]),

                    'mean\_is\_hypernym\_in\_sent\_pairs': np.mean(preds\_[:, 4]),

                    'mean\_is\_anph\_cand\_in\_sent\_pairs': np.mean(preds\_[:, 5])

                }

                preds.clear()

                return flask.jsonify(response)

            else:

                return flask.jsonify("Not enough sentences")

        else:

            res = dict({'message': 'Empty review'})

            return app.response\_class(response=json.dumps(res), status=500, mimetype='application/json')

    else:

        print('Please, choose file to process!')

@app.route('/download\_csv/')

def download\_csv():

    print(reports[-1])

    if len(reports[-1]) > 0:

        fn = f'report{time.strftime("%Y%m%d%H:%M:%S")}.csv'

        path\_fn = f'reports/{fn}'

        with open(path\_fn, 'w', newline='') as file:

            writer = csv.writer(file)

            for line in reports[-1]:

                writer.writerow(line)

        # report.clear()

        Приложение 9.

return send\_from\_directory(app.config['UPLOAD\_FOLDER'], fn, as\_attachment=True)

    else:

        print("No data for report!")

    return ('', 204)

if \_\_name\_\_ == '\_\_main\_\_':

    MODEL = ClassificationModel(

        "bert", '/best\_model/',

        use\_cuda=False

    )

    # models = ["ru\_core\_web\_sm"]

    # default\_text = "Москва столица России"

    # spacy\_streamlit.visualize(models, default\_text)

    app.run(debug=True, use\_reloader=True)