СБОРНИК ОТЧЕТОВ О НАУЧНО-ПРОЕКТНОЙ ДЕЯТЕЛЬНОСТИ ВЫПУСКНИКОВ МЕЖДУНАРОДНОЙ ШКОЛЫ ПО ИНФОРМАЦИОННЫМ ТЕХНОЛОГИЯМ «АНАЛИТИКА БОЛЬШИХ ДАННЫХ»

DATA SCIENCE
IT SCHOOL

Министерство образования Московской области

Государственный университет «Дубна»

Объединенный институт ядерных исследований

СБОРНИК ОТЧЕТОВ
О НАУЧНО-ПРОЕКТНОЙ ДЕЯТЕЛЬНОСТИ ВЫПУСКНИКОВ МЕЖДУНАРОДНОЙ ШКОЛЫ ПО ИНФОРМАЦИОННЫМ ТЕХНОЛОГИЯМ «АНАЛИТИКА БОЛЬШИХ ДАННЫХ»

ВЫПУСК 2

Под редакцией В.В. Кореньков, Е.Н. Черемисиной, О.И. Стрельцовой, Д.И. Пряхиной, А.В. Стадника

DATA SCIENCE
IT SCHOOL

ДУБНА

2021

УДК 004.42
ББК 32.97я43
С23

С23 **Сборник отчетов о научно-проектной деятельности выпускников Международной школы по информационным технологиям «Аналитика больших данных»**. Выпуск 2 / под ред. В.В. Коренькова и др. – Дубна : ОИЯИ, 2021. – 48 с.

ISBN 978-5-9530-0560-9

Сборник включает в себя краткие отчеты об итоговой работе студентов государственного бюджетного образовательного учреждения высшего профессионального образования Московской области «Университет «Дубна», обучавшихся в Международной школе по информационным технологиям «Аналитика больших данных» с 2019 по 2021 год.

Студенты в период обучения были включены в реальные перспективные проекты Объединенного института ядерных исследований (ОИЯИ, Дубна, Россия) и других организаций, работу в которых учащиеся осуществляли в рамках дисциплины «Научно-проектная деятельность» под руководством сотрудников ОИЯИ и университета «Дубна».

УДК 004.42
ББК 32.97я43

ISBN 978-5-9530-0560-9

СОДЕРЖАНИЕ

ПРЕДИСЛОВИЕ 4
ВЫПУСКНИКИ 2021 ГОДА 7
ИНТЕЛЛЕКТУАЛЬНЫЙ АНАЛИЗ ДАННЫХ 9
Разработка агентной системы для оценки публикационной активности стран союза БРИКС 10
Сбор данных о проходных баллах в высшие учебные заведения Российской Федерации 14
Интеллектуальное подавление шума 18
Исследование методов идентификации человека по голосу 22
Распознавание дорожной разметки и анализ дорожной ситуации 26
Разработка сервиса на основе нейросетевого подхода для преобразования изображения в аниме персонажа 30
COMPUTING & SOFTWARE ДЛЯ ЭКСПЕРИМЕНТОВ НА УСКОРИТЕЛЬНОМ КОМПЛЕКСЕ NICA 35
Адаптация метода реконструкции треков L1 эксперимента CBM на FAIR для координатных детекторов эксперимента BM@N проекта NICA 36
Численное моделирование дикварков в плотной и горячей ядерной материи 40
Разработка «Облачного Мета-Планировщика» 44

ПРЕДИСЛОВИЕ

Международная школа по информационным технологиям «Аналитика больших данных» (далее ИТ-школа) — совместный образовательный проект Лаборатории информационных технологий им. М.Г. Мещерякова (МЛИТ) Объединенного института ядерных исследований (ОИЯИ) и Института системного анализа и управления (ИСАУ) государственного университета «Дубна», целью которого является подготовка высококвалифицированных ИТ-специалистов для развития компьютинга мегапроектов, аналитики больших данных, цифровой экономики и других перспективных направлений.

Образовательная программа ИТ-школы формируется с учетом кадровых потребностей ОИЯИ и других организаций высокотехнологичного сектора экономики, а также реализуется при их участии. Программа включает изучение таких дисциплин, как:

- Дополнительные главы математики;
- Языки программирования для анализа данных;
- Введение в операционные системы *UNIX*;
- Инструментарий для коллективной разработки программного обеспечения;
- Введение в облачные технологии;
- Аналитика больших данных;
- Прикладные задачи анализа данных;
- Распределенные системы;
- Мультиагентные системы;
- Высокопроизводительные вычисления;
- Современные методологии хранения и обработки данных;
- Квантовая программная инженерия;
- Английский язык в профессиональной деятельности.

Для проведения занятий со студентами в университете «Дубна» и ЛИТ ОИЯИ была создана образовательная инфраструктура, которая состоит из двух компьютерных кабинетов, конференц-зала и аудитории для индивидуальной работы со студентами. Практические занятия проводятся в компьютерных аудиториях с задействованием, в том числе, ресурсов гетерогенной вычислительной платформы *HybriLIT*, которая является частью Многофункционального информационно-вычислительного комплекса ЛИТ ОИЯИ.

В организации учебного процесса и создании программно-информационной среды принимают участие сотрудники университета «Дубна» и сотрудники ОИЯИ.

В сентябре 2019 года состоялся уже второй набор студентов в ИТ-школу. Студенты университета «Дубна», желающие поступить в ИТ-школу, прошли конкурсный отбор, который включал в себя три этапа: анкетирование, письменный экзамен и собеседование. Обучение в ИТ-школе является бесплатным. Образовательная программа осваивается студентами параллельно с основной образовательной программой.

ИТ-школа тесно взаимодействует в своей деятельности с ведущими университетами России, которые готовят квалифицированных ИТ-специалистов. Поэтому за время обучения студенты получили знания и компетенции в области современного компьютинга и аналитики больших данных не только от преподавателей университета «Дубна» и сотрудников ОИЯИ. Лекции также читали преподаватели из таких университетов России, как Национальный исследовательский ядерный университет «МИФИ», Российский экономический университет им. Г.В. Плеханова и др.

Помимо этого, студенты посещали лекции и семинары от ведущих специалистов российских компаний:

- семинар на тему «Архитектуры и технологии *Intel* для высокопроизводительных вычислений и задач машинного/глубокого обучения (*ML/DL*)» от специалистов компаний *Intel* и РСК, на базе ЛИТ ОИЯИ (15.11.2019);
- мастер-класс на тему «Работа со сверточными нейросетями в фреймворке *Keras*» от представителя компании ООО «Видеоинтеллект» (28.11.2019);
- семинар по высокопроизводительным вычислениям на базе Национального исследовательского университета «Высшая школа экономики» (21.01.2020);
- лекция на тему «Тенденции и принципы компьютерных вычислений. Современные базовые компоненты построения компьютерных систем» от директора по развитию корпоративных проектов *HPC*/*Cloud* компании *Intel* (27.11.2020).

Студенты ИТ-школы принимали активное участие в студенческих образовательных и научных мероприятиях:

- Школа молодых ученых «Высокопроизводительные платформы для цифровой экономики и научных проектов класса мегасайенс» (3-4 декабря 2019 г., РЭУ им. Г.В. Плеханова, Москва, Россия);
- II Школа молодых ученых «Высокопроизводительные платформы для цифровой экономики и научных проектов класса мегасайенс» (17-18 ноября 2020 г., РЭУ им. Г.В. Плеханова, Москва, Россия);
- XXVIII научно-практическая конференция студентов, аспирантов и молодых специалистов (12-23 апреля 2021 г., Университет «Дубна», Дубна, Россия).

В связи со сложившейся эпидемиологической ситуацией с начала 2020 года и переводом на дистанционную форму обучения студенты принимали участие в некоторых мероприятиях онлайн.

В июне 2021 года состоялся второй выпуск студентов ИТ-школы. Выпускники в количестве 10 человек успешно завершили обучение и получили документы о дополнительном образовании, удостоверяющие прохождение профессиональной подготовки по программе «Аналитика больших данных».

Одним из главных принципов ИТ-школы является обучение через исследования. Поэтому студенты в период обучения были включены в реальные перспективные проекты ОИЯИ, работу в которых учащиеся осуществляли в рамках дисциплины «Научно-проектная деятельность». В данном сборнике представлены краткие отчеты об их деятельности.

Дирекция ИТ-школы выражает огромную благодарность преподавателям университета «Дубна» и сотрудникам ОИЯИ за плодотворную работу со студентами.

С учащимися ИТ-школы работали:

- Балашов Н.А., инженер-программист МЛИТ ОИЯИ;
- Белов М.А., доц. каф. САУ ИСАУ университета «Дубна»;
- Герценбергер К.В., к.т.н., научно-экспериментальный отдел физики столкновений тяжелых ионов на комплексе *NICA*, начальник группы математического и программного обеспечения ЛФВЭ ОИЯИ;
- Зрелов П.В., к.ф.-м.н., начальник научно-технического отдела программного и информационного обеспечения МЛИТ ОИЯИ;
- Иванцова О.В., ст. преп. каф. САУ ИСАУ университета «Дубна»;
- Кадочников И.С., инженер-программист МЛИТ ОИЯИ;
- Калиновский Ю.Л., д.ф.-м.н., ведущий научный сотрудник МЛИТ ОИЯИ;
- Кошлань Д.И., инженер-программист МЛИТ ОИЯИ;
- Мещерская Ю.В., к.пед.н., доц. каф. САУ ИСАУ университета «Дубна»;
- Ососков Г.А., д.ф.-м.н., главный научный сотрудник МЛИТ ОИЯИ;
- Пелеванюк И.С., научный сотрудник МЛИТ ОИЯИ;
- Решетников А.Г., к.т.н., доц. САУ ИСАУ университета «Дубна»;
- Стадник А.В., к.ф.-м.н., старший научный сотрудник МЛИТ ОИЯИ;
- Сычев П.П., доц. каф. РИВС ИСАУ университета «Дубна»;
- Тятюшкина О.Ю., к.т.н., доц. САУ ИСАУ университета «Дубна»;
- Ульянов С.В., д.ф.-м.н., проф. каф. САУ ИСАУ университета «Дубна».

Научные руководители ИТ-школы:

***В.В. Кореньков**, д.т.н., директор ЛИТ ОИЯИ, заведующий каф. РИВС ИСАУ;*

***Е.Н. Черемисина**, д.т.н., профессор, академик РАЕН, директор ИСАУ.*

Директор ИТ-школы:

***О.И. Стрельцова**, к.ф.-м.н., старший научный сотрудник ЛИТ ОИЯИ, доц. каф. РИВС ИСАУ.*

Ученый секретарь ИТ-школы:

***Д.И. Пряхина**, научный сотрудник ЛИТ ОИЯИ, ст. преп. каф. РИВС ИСАУ.*

itschool.jinr.ru

ВЫПУСКНИКИ 2021 ГОДА

1. Бачинский Станислав Витальевич
2. Колобов Сергей Игоревич
3. Мельник Николай Андреевич
4. Нечушкин Кирилл Антонович
5. Папоян Георгий Владимирович
6. Рыбакина Алена Дмитриевна
7. Сасин Алексей Дмитриевич
8. Соколова Мария Викторовна
9. Ходырев Иван Сергеевич
10. Цегельник Никита Сергеевич

ИНТЕЛЛЕКТУАЛЬНЫЙ АНАЛИЗ ДАННЫХ

РАЗРАБОТКА АГЕНТНОЙ СИСТЕМЫ ДЛЯ ОЦЕНКИ ПУБЛИКАЦИОННОЙ АКТИВНОСТИ СТРАН СОЮЗА БРИКС

Соколова Мария Викторовна[1], Кошлань Диана Игоревна[2], Пряхина Дарья Игоревна[3]

[1] Студент;
Государственный университет «Дубна»;
Международная школа по информационным технологиям
«Аналитика больших данных»;
Направление обучения по основной образовательной программе:
Программная инженерия, группа 4251;
e-mail: somv.17@uni-dubna.ru.

[2] Инженер-программист;
Лаборатория информационных технологий им. М.Г. Мещерякова;
Объединенный институт ядерных исследований.
Аспирант;
Государственный университет «Дубна».

[3] Научный сотрудник;
Лаборатория информационных технологий им. М.Г. Мещерякова;
Объединенный институт ядерных исследований.
Старший преподаватель;
Кафедра распределенных информационно-вычислительных систем;
Государственный университет «Дубна».

Ключевые слова: программа-агент, информационные ресурсы, публикационная активность, союз *БРИКС.*

Введение

В каждой из стран-участниц союза БРИКС (Бразилия, Россия, Индия, КНР, ЮАР) ученые занимаются научной работой в разных направлениях, результаты которой главным образом отражены в статьях. Чтобы проанализировать вклад каждой страны союза в мировую науку, выделить лидирующие научные направления государств, оценить публикационную активность различных организаций и университетов, выделить возможные направления сотрудничества организаций разных стран, необходимо разработать систему, позволяющую собирать информацию из баз данных, которые содержат сведения о научных публикациях, изданных в ведущих журналах, и предоставлять статистическую информацию научно-технического развития стран БРИКС в удобном для анализа виде.

Целью работы является сбор статистических данных публикаций каждой страны союза БРИКС и представление этих данных в виде веб-приложения для их последующего анализа.

1. Сбор данных

В качестве источника данных выбран сайт *Nature Index* [1], агрегирующий ссылки на публикации в различных журналах для каждой страны. Для сбора данных создана программа-агент, которая автоматически сканирует заранее определенные сайты в сети Интернет, собирает информацию по поисковым предписаниям и доставляет ее в соответствующий раздел базы данных.

Для реализации программы-агента выбран язык программирования *Python* и библиотека управления браузером *Selenium WebDriver*.

Задачи программы-агента включают сбор данных публикаций для каждой страны союза БРИКС с использованием информационного ресурса *Nature Index* и доставку собранных данных в систему хранения. Алгоритм работы агента представлен на рисунке 1.

Рис. 1. Алгоритм работы программного агента

Для сбора данных публикации агент должен перейти на страницу публикации по ссылке для нее. Информационный ресурс *natureindex.com* содержит отдельную страницу для каждой страны, где представлен список тем, каждая из которых разворачивается в список журналов, которые публикуют научные статьи. Каждый журнал из списка разворачивается в список ссылок на публикации, в свою очередь ссылки ведут на страницу того же информационного ресурса *natureindex.com* с краткой информацией о публикации и кнопкой перехода на страницу публикации на сайте конкретного журнала.

Для осуществления возможности перехода по ссылкам на публикации, ссылки собираются в список вместе с соответствующими журналами, для каждого из которых создаются отдельные функции для сбора данных, поскольку информационные ресурсы каждого журнала имеют разную структуру. Для корректного сбора данных каждый журнал должен быть связан со своим методом сбора данных. После установления необходимых связей программой-агентом производится непосредственно сбор данных о каждой публикации.

Таким образом, в ходе реализации программы-агента написаны функции для сбора ссылок на публикации каждой страны, функция для перехода по ссылкам из списка, функции для сбора данных со страниц конкретного журнала.

2. Система хранения данных

Собранные агентом данные хранятся в реляционной базе данных. В качестве системы управления базами данных используется *PostgreSQL*.

Для анализа публикационной активности стран союза БРИКС необходимо знать: в каком количестве публикаций каждая страна приняла участие; какие организации и авторы страны в каких публикациях принимали участие; какие авторы с какими организациями связаны; какие ключевые слова относятся к каждой научной публикации; даты научных публикаций.

Для хранения таких данных в реляционной БД используются следующие сущности: публикация; ключевое слово; организация; автор; публикация-организация; публикация-автор; автор-организация; публикация-ключевое слово.

Физическая модель базы данных представлена на рисунке 2.

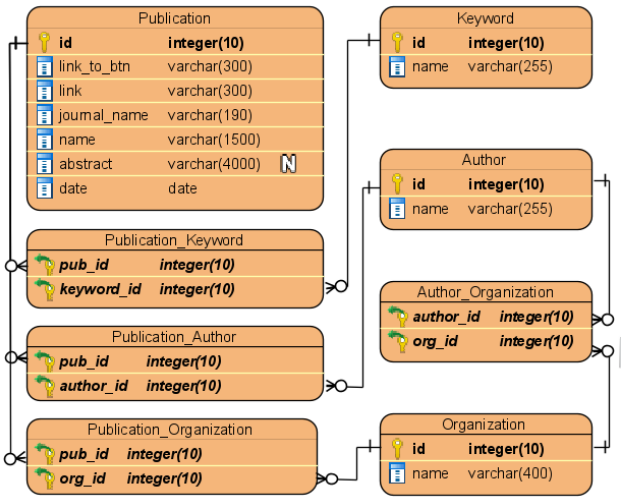


Рис. 2. Физическая модель базы данных

3. Веб-приложение

Для веб-приложения выбрана трехуровневая клиент-серверная архитектура. Такая архитектура подразумевает наличие «тонкого» клиентского приложения для отображения данных, серверного приложения, выполняющего бизнес-логику системы, а также базу данных.

Для реализации серверного приложения выбран язык программирования *Python* и веб-фреймворк *Django*. Это высокоуровневый написанный на языке *Python* веб-фреймворк с открытым исходным кодом [2]. Для реализации клиентского приложения выбраны язык программирования *JavaScript* и библиотека *React*. *JavaScript* — мультипарадигменный язык программирования с динамической типизацией и автоматическим управлением памятью [3].

В системе можно выделить двух пользователей: администратора и клиента. Клиент должен иметь возможность просматривать статистику, а администратор — и просматривать ее, и запускать процесс ее обновления программой-агентом. Для запуска процесса обновления системы администратор должен быть аутентифицирован в системе.

Серверное приложение представляет из себя один проект с подключаемыми и отключаемыми от него приложениями. За работу с данными в нем отвечает приложение *BricsAgentApplication*, за аутентификацию — приложение *AuthenticationApplication*. Программный агент является частью приложения *BricsAgentApplication*.

Для реализации вышеописанных возможностей пользователей описано получение данных в файле *views.py* приложения *BricsAgentApplication*, который представляет собой набор функций, вызывающихся при запросе клиентского приложения (соответствие функций и их *URN* описано в файле приложения *urls.py*). Набор этих функции является *API* серверного приложения, основные его функции: запуска сбора данных публикаций; получения прогресса сбора данных публикаций; получения списка десяти организаций страны с определенной подстрокой в названии и (опционально) определенным количеством статей; получения наиболее часто встречающихся в публикациях страны ключевых слов; получения списка 10 авторов с наибольшим количеством публикаций для конкретной организации; получения публикационной активности стран; получения данных о попарном сотрудничестве стран.

Клиентское приложение посылает запросы серверному приложению, которое посредством вызова этих функций возвращает необходимые данные. Клиентское приложение представляет из себя одностраничное приложение (*Single Page Application*), состоящее из одной страницы с заменяющими друг друга компонентами. Компоненты отображают определенные данные, согласно вариантам использования. Основной класс приложения *App* содержит код, отвечающий за отрисовку всех компонентов, заменяющих друг друга на одной странице.

Каждый компонент, включая *App*, является классом, наследующим класс *Component*, предоставляемый библиотекой *React*. Также каждый компонент содержит функции для обращения к серверу за

определенными данными. Список компонентов включает: компонент для просмотра публикационной активности стран (см. рис. 3), компонент для просмотра организаций страны с наибольшим количеством публикаций, а также с наиболее встречающимися ключевыми словами в публикациях; компонент для просмотра всех организаций страны(см. рис. 4); компонент для просмотра авторов конкретной организации с наибольшим количеством публикаций (см. рис. 4); компонент для аутентификации; компонент для запуска сбора данных, компонент для просмотра данных о попарном сотрудничестве стран.

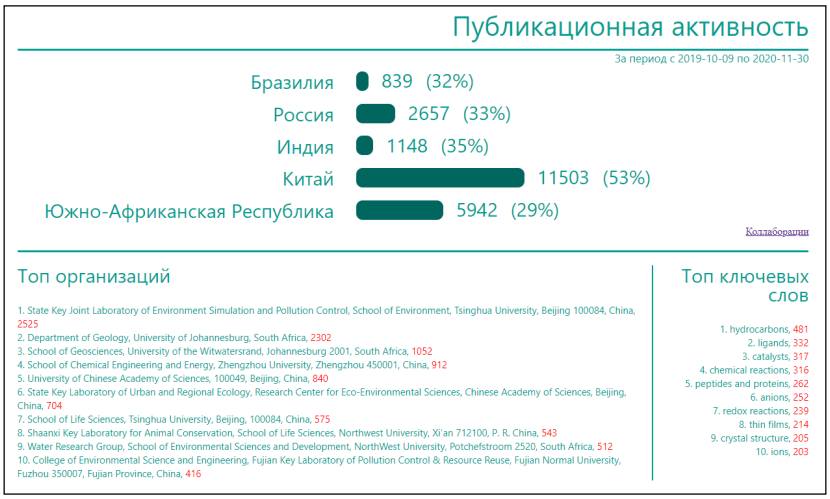


Рис. 3. Компонент для просмотра публикационной активности

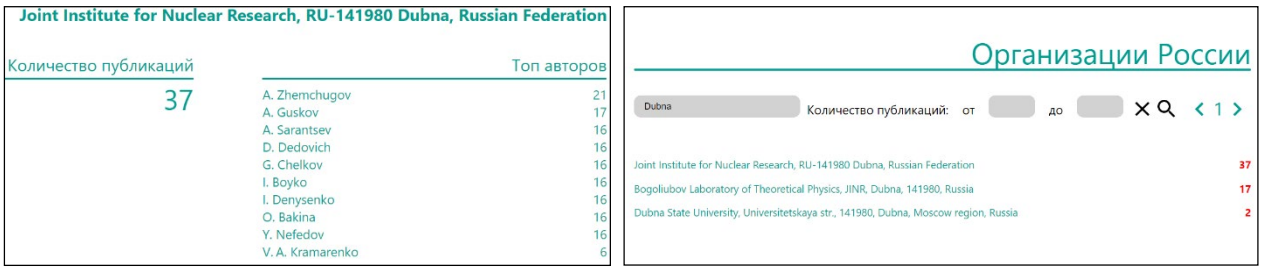


Рис. 4. Компоненты для просмотра информации об организации (слева) и всех организаций (справа) страны

Заключение

Таким образом, в ходе работы собраны данные публикаций стран союза БРИКС и создано веб-приложения для их отображения и возможности анализа публикационной активности стран данного союза.

В дальнейшем планируется совершенствование дизайна компонентов клиентского приложения, а также развертывание веб-приложения на виртуальной машине со статическим *IP*-адресом, предоставляемой облачным сервисом ОИЯИ.

Список литературы

1. База данных научных публикаций Nature Index / Электрон. дан. – [Электронный ресурс]. URL: https://natureindex.com.
2. Дронов В.А. Django 2.1. Практика создания веб-сайтов на Python. — СПб.: БХВ-Петербург, 2019.
3. Хавербеке М. Выразительный JavaScript. Современное веб-программирование. 3-е изд. — СПб.:Питер, 2019.

СБОР ДАННЫХ О ПРОХОДНЫХ БАЛЛАХ В ВЫСШИЕ УЧЕБНЫЕ ЗАВЕДЕНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Сасин Алексей Дмитриевич[1], **Пряхина Дарья Игоревна**[2]

[1] *Студент;*
Государственный университет «Дубна»;
Международная школа по информационным технологиям
«Аналитика больших данных»;
Направление обучения по основной образовательной программе:
Информационные системы и технологии, группа 3282;
e-mail: sad.18@uni-dubna.ru.

[2] *Научный сотрудник;*
Лаборатория информационных технологий им. М.Г. Мещерякова;
Объединенный институт ядерных исследований.
Старший преподаватель;
Кафедра распределенных информационно-вычислительных систем;
Государственный университет «Дубна».

Ключевые слова: сбор данных, извлечение данных с веб-страниц, многопоточные алгоритмы.

Введение

После получения среднего образования перед выпускниками школ встает вопрос выбора учебного заведения для получения высшего или средне-специального образования. В сети Интернет имеются ресурсы, на которых расположена информация о проходных баллах в высшие учебные заведения (вузы). В качестве примеров можно привести сайт «Учеба.ру» [1] и «Поступи онлайн» [2]. Главными недостатками таких ресурсов являются: высокая стоимость за размещение информации и необходимость самостоятельной регистрации вузов. Бесплатное же размещение информации на подобных сайтах не гарантирует достоверность этих данных. Таким образом существует необходимость разработки сервиса, который будет предоставлять такую информацию обо всех высших и средне-специальных учебных заведениях Российской Федерации, как направления обучения, проходные баллы, стоимость обучения, объявления о мероприятиях для студентов и многое другое. Однако данная информация должна быть достоверной, поэтому необходимо получать ее из проверенных источников, которыми являются документы, расположенные на официальных сайтах учебных заведений.

Задачей данной работы является разработка алгоритма для сбора приказов о зачислении в вузы Российской Федерации с целью дальнейшего извлечения информации о проходных баллах.

1. Извлечение информации о вузах

Информация об учебных заведениях была получена с официального сайта Федеральной службы по надзору в сфере образования и науки, где она представлена в виде файла формата *xml* [3]. В данном файле содержится информация о 922 учебных заведениях, из них: 492 головных заведений и 430 филиалов. Из файла с данными об учебных заведениях были извлечены их названия и ссылки на официальные сайты при помощи программы, написанной на языке программирования *Python* [4] с использованием библиотеки *ElementTree* [5]. Полученные данные были сохранены в файл формата *JSON* (см. рис. 1).

```
Name,Links
Московский городской университет управления Правительства Москвы,http://mguu.ru/
города Москвы Московский городской педагогический университет,http://mgpu.ru/
города Москвы Московский государственный институт индустрии туризма имени Ю.А.Сенкевича,http://mgiit.ru/
Московской области Технологический университет,http://unitech-mo.ru
Московской области Университет Дубна,http://uni-dubna.ru
```

Рис. 1. Данные об учебных заведениях

2. Сбор приказов о зачислении

Программа для сбора приказов о зачислении в вузы разработана на языке программирования *Python* [4] с использованием фреймворка для автоматизированного управления браузерами *selenium* [6]. Поскольку путь к приказам на сайте каждого вуза уникален, необходимо воспользоваться поисковой системой, например, «Яндекс» для поиска страницы, на которой может располагаться нужная информация. Для поиска задается запрос, который включает название вуза, год и словосочетание «приказ о зачислении». Названия вузов извлекаются из *JSON* файла (см. рис. 1). После получения ссылок из поисковой системы для сбора приказов о зачислении запускается алгоритм, представленный на рисунке 2.

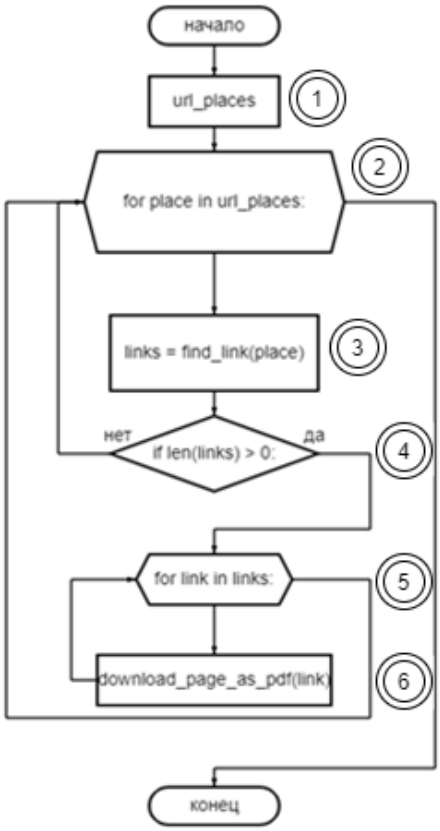


Рис. 2. Алгоритм сбора приказов о зачислении

Работу алгоритма можно описать следующим образом.

1. Переход по каждой ссылке, полученной из поисковой системы. На открывшейся странице запускается поиск и запись всех ссылок, где встречаются выражения: «зачисл», «Зачисл», «ЗАЧИСЛ».
2. Проход в цикле по всем ссылкам из списка, сформированного в п.1.
3. Запись в новый список всех ссылок, которые находились в данных местах или ниже по структурному дереву веб-страницы.
4. Проверка длины нового списка на наличие ссылок для скачивания документов.
5. Проход в цикле по всем полученным ссылкам.
6. Извлечение файлов формата *.pdf* со всех ссылок с помощью фреймворка *selenium* [6].

К недостаткам данного алгоритма можно отнести следующее.

- Привязка к ключевым словам («зачисл», «Зачисл», «ЗАЧИСЛ»).
- Отсутствие проверки *url* сайта, полученного в поисковой системе, на соответствие с официальным сайтом учебного заведения.

- Последовательный перебор всех учебных заведений.
- Время сбора данных 48 часов.

Из-за вышеописанных недостатков первого алгоритма возникла необходимость в разработке оптимизированного алгоритма (см. рис. 3).

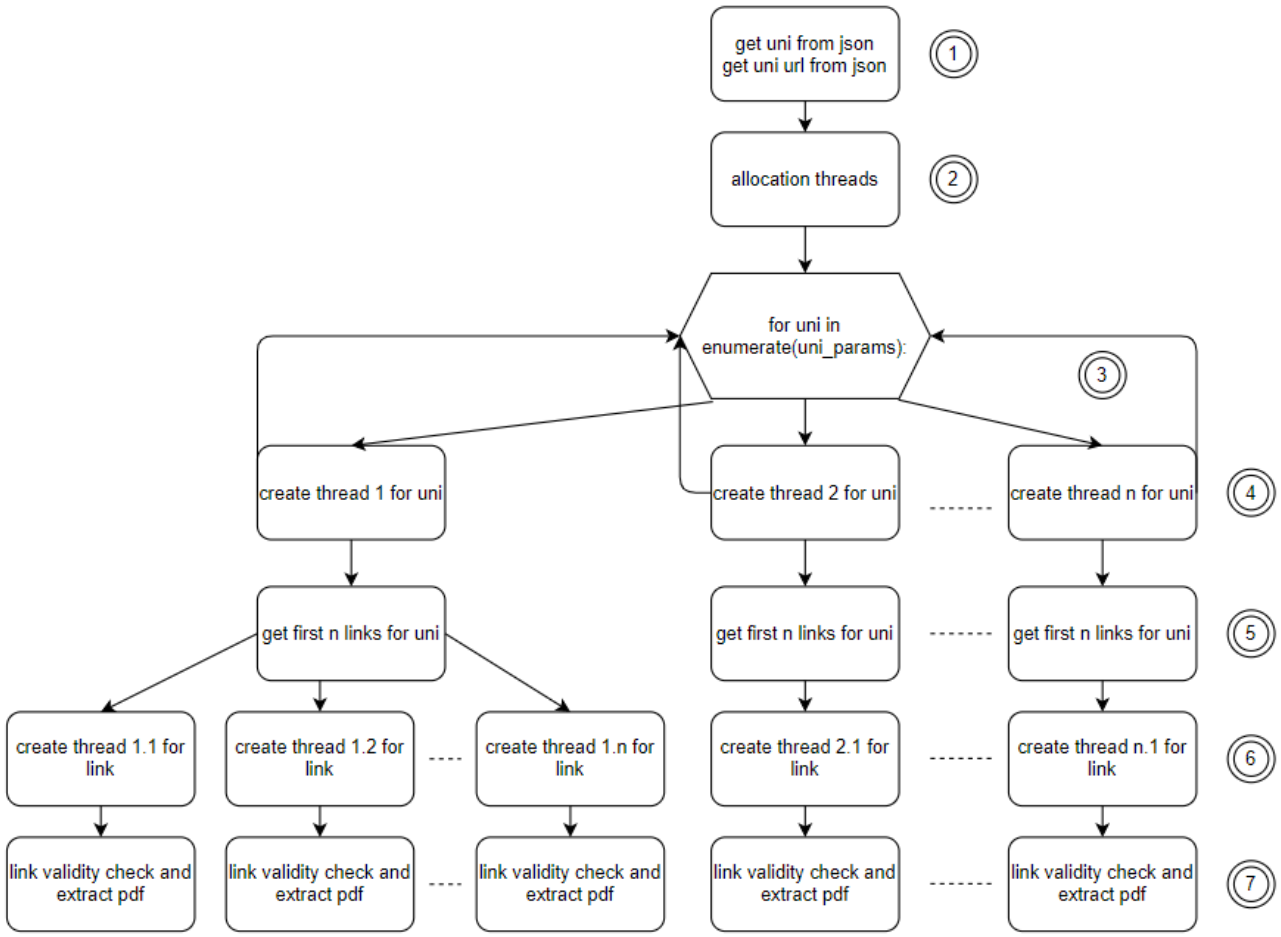


Рис. 3. Оптимизированный алгоритм сбора приказов о зачислении

Работу оптимизированного алгоритма можно описать следующим образом.

1. Извлечение названий и ссылок учебных заведений из *JSON* файла.
2. Установка максимального числа потоков (12) с помощью модуля *threading*.
3. Проход в цикле по каждому названияю-ссылке.
4. Выделение отдельного потока для каждого учебного заведения.
5. Получение первых *N* ссылок из поисковой системы «Яндекс».
6. Проход в цикле по всем полученным из поисковой системы ссылкам с выделением отдельного потока для каждой ссылки.
7. Проверка актуальности полученных ссылок (сравнение url ссылки и url вуза) и последующее извлечение .pdf файлов с помощью фреймворка selenium [6].

Также стоит отметить, что время работы алгоритма составило 6 часов.

3. Результаты сбора данных

С помощью оптимизированного алгоритма были получены и сохранены на локальный компьютер приказы о зачислении в формате *pdf* за 2020 год в 922 учебных заведения, из которых 492 головные заведения и 430 филиалы. Каждый файл сохранен в директорию, имя которой совпадает с названием вуза.

Заключение

В результате работы были разработаны два алгоритма для сбора приказов о зачислении в вузы Российской Федерации, которые размещены на официальных сайтах учебных заведений, с целью дальнейшего извлечения информации о проходных баллах. Тестирование, которое осуществлялось на компьютере с процессором *AMD RYZEN 5 3600X* и пропускной способностью сети 100 Мбит/с, показало, что первый алгоритм собирает приказы о зачислении за 48 часов, а оптимизированный алгоритм – за 6 часов. Следовательно, оптимизированный алгоритм работает быстрее в 8 раз и за счет проверки актуальности ссылки не собирает лишние данные. В планы на дальнейшее развитие входит разработка методов для извлечения данных из собранных документов.

Список литературы

1. Учеба.ру. – [Электронный ресурс]. URL: https://www.ucheba.ru/.
2. Поступи онлайн. – [Электронный ресурс]. URL: https://postupi.online/.
3. Федеральная служба по надзору в сфере образования и науки. – [Электронный ресурс]. URL: http://obrnadzor.gov.ru/otkrytoe-pravitelstvo/opendata/7701537808-raoo/.
4. Доусон М. Программируем на Python. — СПб: Питер, 2016.
5. Официальная документация библиотеки ElementTree. – [Электронный ресурс]. URL: https://docs.python.org/3/library/xml.etree.elementtree.html.
6. Avasarala S. Selenium WebDriver Practical Guide. – Packt Publishing, 2014.

ИНТЕЛЛЕКТУАЛЬНОЕ ПОДАВЛЕНИЕ ШУМА

Рыбакина Алена Дмитриевна[1], Климанова Дарья Дмитриевна[2]

[1] Студент;
Государственный университет «Дубна»;
Международная школа по информационным технологиям
«Аналитика больших данных»;
Направление обучения по основной образовательной программе:
Информатика и вычислительная техника, группа 4015;
e-mail: lena.ribackina@yandex.ru.

[2] Системный аналитик департамента Machine Learning;
Общество с ограниченной ответственностью
«Центр искусственного интеллекта МТС».

Ключевые слова: шумоподаление, сверточные нейронные сети, глубокое обучение.

Введение

В период пандемии инструменты для совместной работы на расстоянии стали необходимым средством для организации рабочего процесса. Именно в это время так необходимы аудиозвонки с хорошим и отличным качеством речи. Такие звонки особенно востребованы на совещаниях в крупных корпорациях, для организации занятий во время учебного процесса в школах, университетах, на развлекательных мероприятиях, например, при трансляции выступления симфонического оркестра и т.п. Однако окружающая среда, в которой может находится аудиоустройство, воспринимающее входящие сигналы, может воспроизводить и воспринимать различные виды фоновых шумов. Сами участники аудиозвонков не могут повлиять на то, чтобы внешние шумы не искажали переданный сигнал. Например, это такие шумы, как лай собаки, плач ребенка, шум кондиционера, движение транспорта, шум столовых приборов на кухне и т.п.

Задача улучшения речи (*speech enhancement*) состоит в том, чтобы с помощью алгоритмов повысить разборчивость и качество речевого сигнала, загрязненного шумом.

Разработку алгоритмов шумоподавления усложняют такие факторы, как: низкое соотношение сигнала к шуму, быстро меняющиеся распределения вероятностей, нелинейные комбинации различных типов шумов и т. д. В таких неблагоприятных условиях существующие подходы шумоподавления могут улучшить соотношение сигнала к шуму, но, к сожалению, при этом будут генерировать новые искажения: неестественные искажения речи, колеблющиеся остаточные шумы.

Потому задача разработки алгоритма эффективного подавления шума на аудиосигналах остается актуальной проблемой и темой исследования, которая до сих пор рассматривается во многих статьях. В 80-х годах данная задача решалась с использованием классических алгоритмов улучшения речи, таких, как: спектральное вычитание, фильтрация Винера, а также алгоритмы, основанные на статистических моделях.

Данные методики основаны на вычислении спектральной оценки и полагаются на спектральную оценку шума, которая, в свою очередь, работает при помощи детектора голосовой активности (*VAD*). Каждые из трех алгоритмов требуют аккуратной подгонки параметров, а данные параметры трудно настраивать.

Нейронные сети, в отличие от вероятностных моделей, поддерживают поток информации только в одном направлении, от входа к выходу. Объектом исследования является изучение методов, которые используются для решения задач шумоподавления, субъектом исследования является алгоритмы нейронных сетей. В работе были исследованы такие алгоритмы рекуррентной нейронной сети, генеративно-состязательной нейронной сети, сверточных нейронных сетей.

1. Обзор алгоритмов для шумоподавления

Исследование состояло из нескольких этапов: обзор существующих методов для решения поставленной задачи, сбор и подготовка данных и обучение выбранной модели нейронной сети.

Этап обзора существующих методов для решения задачи шумоподавления заключался в исследовании статей с описанием распространенных архитектурных решений в области глубокого обучения.

Первая модель основана на рекуррентной нейронной сети, где основная часть подавления шума выполняется на спектральной огибающей низкого разрешения с использованием коэффициентов улучшения. Данные коэффициенты вычисляются с помощью рекуррентной нейронной сети. Коэффициенты улучшения получаются путем вычисления квадратного корня из маски идеального соотношения (*ideal ratio mask*) (*IRM*).

Такой подход имеет ограничение – невозможно смоделировать более мелкие детали в спектре. В качестве решения используется гребенчатый фильтр (*comb pitch filter*) для подавления межгармонического шума. Поскольку периодичность речевого сигнала сильно зависит от частоты, то гребенчатый фильтр работает в частотной области на основе коэффициента полосовой фильтрации [1].

Следующий алгоритм основан на генеративно-состязательной модели. Генеративные модели учатся сопоставлять семплы *z* из некоторого предшествующего распределения *Z*, с семплами x из другого распределения *X*. *X* — набор данных обучающих примеров (например, изображения, аудио и т. д.). Компонент в структуре *GAN*, выполняющий отображение, называется генератором (*G*), и его основная задача — изучить эффективное отображение, которое может имитировать реальное распределение данных для генерации новых семплов.

Способ, которым *G* учится делать отображение, заключается в состязательном обучении, где у нас есть еще один компонент, называемый дискриминатором (*D*). *D* обычно является бинарным классификатором, и его входными данными являются либо реальные семплы, поступающие из набора данных, который имитирует *G*, либо поддельные семплы, составленные *G*.

G может слегка скорректировать свою выходную форму сигнала в сторону правильного распределения. Он избавляется от шумных сигналов, поскольку они определяются *D*, как фальшивые. В этом смысле *D* можно понимать, как функцию потерь для *G*.

В задаче шумоподавления улучшение выполняет *G*-сеть. Его входы — шумный речевой сигнал *x* вместе с латентным представлением *z*, а выход-улучшенная версия $x' = G(x)$. *G* — полностью сверточная нейросеть без полносвязных слоев.

Работа *G* состоит из двух этапов. Во-первых, происходит кодирование зашумленной речи в скрытое представление, и последующее декодирование скрытого представления и латентного вектора *z* в очищенный от шумов сигнал. На этапе кодирования входной сигнал сжимается через ряд расширенных (*dilated*) сверточных слоев. Сжатие выполняется до тех пор, пока не получается скрытое представление. На этапе декодирования архитектура сети симметрична описанной выше [2].

Третья рассматриваемая модель отличается от предыдущих тем, что не использует частотно-временные представления или амплитудные спектрограммы в качестве входных данных для уменьшения большой размерности необработанных сигналов. Тем самым она не сталкивается с недостатками, связанными с отбрасыванием потенциально ценной информации (по фазе) и недостатками при использовании универсальных методов для извлечения признаков (например, анализ амплитудных спектрограмм). Модель конкретно изучает признаки у поданного ей сигнала.

Архитектура модели также основана на свертках. Это делает ее гибкой во временном измерении. Примечательно, что модель использует контекстную информацию для обработки аудио. Для того, чтобы обработать один фрагмент аудиосигнала, модель смотрит также на части аудиозаписи до и после этого фрагмента. Это позволяет собрать полезную информацию для дальнейшей обработки текущих рассматриваемых фрагментов [3].

Четвертая модель – *TCNN*. Это полностью сверточная нейронная сеть (*CNN*) для улучшения речи, работающая в реальном времени. Архитектура модели построена на основе энкодера-декодера

с дополнительным временным сверточным модулем (*TCM*). *TCM* также использует расширенные (*dilated*) слои с разным шагом в блоке энкодера.

Архитектура декодера симметрична архитектуре энкодера и имеет *skip*-соединения со слоями энкодера.

Энкодер принимает последовательность зашумленных фреймов в качестве входных данных. Первый слой в энкодере увеличивает количество каналов с 1 до 16. Следующие семь слоев сжимают размер аудио, используя обычные свертки с размером шага 2. Выход энкодера преобразуется в одномерный сигнал. Этот сигнал подается на вход блоку *TCM. TCM* состоит из трех блоков. Блок в свою очередь состоит из шести *dilated* сверток, которые имеют размеры: 1, 2, 4, 8, 16 и 32. Это позволяет алгоритму охватить большую область аудио и извлечь признаки из все более возрастающей рассматриваемой области. Таким образом, он получает глобальную информацию о поступившем аудиосигнале.

То есть, сначала первая свертка получила локальные признаки аудиосигнала, вторая свертка с шагом два, берет область шире (в два раза больше семплов, чем первая и увеличивает размер шага в два раза больше первой, т.е. рассматривает соседние семплы, по сравнению с предыдущим слоем), соответственно, получает информацию больше первой. Таким образом в самом последнем слое, в конце блока, будет собрана самая общая информация об аудиосигнале [4].

Последняя модель *DEMUCS* также состоит из сверточного энкодера и декодера и рекуррентной нейронной сети между ними (*LSTM*) с четырьмя слоями. Стоит добавить, что модель основана на идее архитектуры *U-net*, которая чаще всего применяется в задачах сегментации изображений.

Энкодер принимает в качестве входа необработанный сигнал и выводит его скрытое представление. Далее новое представление сигнала проходит через обработку в сверточных слоях. Затем это представление на вход подается *LSTM*, который в последствии обработки передает новое представление на сеть декодера. После обработки, декодер выдает чистый сигнал. По структуре, декодер является симметрией энкодера и имеет скип-соединения [5].

2. Обучение модели

Вторая часть исследования состояла в подготовке данных для обучения нейронной сети.

Russian Open Speech To Text [6] большой открытый корпус устной русской речи, который содержит в себе 20 тысяч часов устной речи, из разных предметных областей – звонки, *youtube*, лекции, телефонные разговоры, книги. Для разработки набора данных были взяты только аудокниги, так как записи без шумов и достаточно много голосов начитывали книги. Так как аудио здесь разной длительности от менее секунды до 17 минут, то была написана функция по нарезки аудио и сохранения ее в папку. Не брались файлы менее, чем 2 секунды. Сама нарезка производилась по 2 секунды.

Далее был произведен поиск шумов. Было сформулировано 200 типов шумов, где на каждый тип было найдено по 5 аудиофайлов. Аудио были найдены в сети Интернет [7].

Все эти шумы можно разделить на следующие классы: аудиофайлы в виде хлопка, хруста, шипения, жевания, бурления, отрывания, клацанья, скрежета, короткого шума, бьющиеся предметы, скрип, шумы, связанные с водой – капание, береговые, шуршание, музыкальные, стук, шумы, издаваемые людьми – кашель, смех, сморкание, лепет, шумы от техники, звонкие шумы – от сирен, будильников, шумы, которые относятся к месту (окружающая среда) – в стадионе, с кафе, ресторане и т.д. Для аугментации – наложения шума на аудиозаписи был разработан конвейер при помощи библиотеки *Apache Beam*. Здесь была реализована задача стриминга в некоторых кругах такой процесс известен как *ETL*, т.е. извлечение, преобразование и загрузка информации.

Шаги написанного конвейера состояли из чтения табличного набора данных с описанием чистых файлов, хранимых в формате *.csv*, затем наложения шума и запись в память зашумленного файла. Выбор шума происходит путем объявления объекта класса *Provider*. В самом классе происходит чтение табличного файла в формате *.csv*, и из полученного списка выбирается один случайный файл, который будет участвовать в зашумление аудиосигнала. Для зашумленного файла сохраняются его характеристики, которые включают путь, где он храниться, путь файла с чистой речью, путь файла с

шумом, средняя мощность и отношение чистого сигнала к шуму, так как он может быть случайным от -5 до 5. Последним шагом является запись зашумленного файла.

Чтобы количество зашумленных файлов по типам было равномерным, был код доработан, чтобы он делил общий набор данных на части по введенному количеству типов шумов на отдельные наборы данных и сохранял их в формате *.csv*. Далее идет перебор файлов с шумами и процесс аугментации проходит также, только в процессе выбирается определенный файл, а после отработки конвейера сбор всех файлов.

Полученные наборы данных были протестированы на всех описанных выше моделях и в качестве дальнейшего этапа подбора конфигурации модели, был выбран *Demoucs*, так как занимал меньшее время обучения модели и подавлял шумы на большинстве частот. В качестве метрики качества, были использованы такие метрики, как *MOS* и *STOI*.

Качество аудио файлов в основном проводилось с помощью прослушивания аудио из разных классов и выставления оценки от 1 до 5 по параметрам разборчивости речи, слышимости шума и качества речи, так как субъективные метрики не всегда отражали верную оценки. Было замечено, что, если число выводимой метрикой *STOI* менее 0.8, то оценка качества аудио для нас была низкая (3 балла). Если более 0.8, то довольно ставили высокую оценку. Если оценка была не более 0.6, то качество было совсем плохое (1 или 2 балла). В целом, всегда плохие результаты появлялись у колокольных шумов колокола или сирен, в остальных типах были стабильные хорошие результаты.

Сам проект и его результаты представлен здесь [8].

Заключение

В ходе исследовательской работы было рассмотрено несколько архитектур и подходов в решении задач адаптивного шумоподавления, но и также были изучены принципы методы решения различных задач, с потоковой обработкой данных. Оптимизированная модель *Demoucs* может быть в дальнейшем использована в разработке продукта для подавления шума при проведении конференций, звонков, как с андроид устройства, так и с компьютера. Также были изучены возможности представления аудиофайлов в виде спектрограмм, которые при сравнении показывали частоты, которые нейросеть по той или иной причине пропускает и убирает в них шум. В ходе изучения были изучены алгоритм Фурье, возможности библиотеками *librosa, pandas*, *matplotlib*.

Работа с технологией *DataFlow* наглядно показала быстрый и удобный процесс аугментации данных.

Список литературы

1. Jean-Marc Valin. A Hybrid DSP/Deep Learning Approach to Real-Time Full-Band Speech Enhancement // IEEE Multimedia Signal Processing (MMSP) Workshop, 2018.
2. Pascual S., Bonafonte A., Serra J. SEGAN: Speech Enhancement Generative Adversarial Network // Cornel University, 2017.
3. Rethage D., Pons J., Serra X. A WaveNet for Speech Denoising // Cornel University, 2017.
4. Pandey A., Wang D.. TCNN: temporal convolutional neural network for real-time speech enhancement in the time domain // ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2019.
5. Defossez A., Synnaeve G., Adi Y. Real Time Speech Enhancement in the Waveform Domain // Cornel University, 2020.
6. Russian Open Speech To Text. – [Электронный ресурс]. URL: https://azure.microsoft.com/en-us/services/open-datasets/catalog/open-speech-to-text/#AzureNotebooks.
7. ZVUKOGRAM. – [Электронный ресурс]. URL: https://zvukogram.com.
8. Intelligent-noise-reduction. – [Электронный ресурс]. URL: https://github.com/Alena0704/intelligent-noise-reduction.

ИССЛЕДОВАНИЕ МЕТОДОВ ИДЕНТИФИКАЦИИ ЧЕЛОВЕКА ПО ГОЛОСУ

Нечушкин Кирилл Антонович[1], Стрельцова Оксана Ивановна[2]

[1] Студент;
Государственный университет «Дубна»;
Международная школа по информационным технологиям
«Аналитика больших данных»;
Направление обучения по основной образовательной программе:
Информатика и вычислительная техника, группа 4013;
e-mail: nka.17@uni-dubna.ru.

[2] к.ф.-м.н., старший научный сотрудник;
Лаборатория информационных технологий им. М.Г. Мещерякова;
Объединенный институт ядерных исследований.
Доцент;
Кафедра распределенных информационно-вычислительных систем;
Государственный университет «Дубна».

Ключевые слова: нейронная сеть, голос, распознавание.

Введение

В данной работе предметом исследования служит устная речь. Устная речь индивидуальна по звучанию. Индивидуальные особенности речи – совокупность характеристик, которые определяются особенностями строения голосовых органов и индивидуальных особенностей человека. Наиболее яркими внешними проявлениями индивидуальных особенностей являются такие характеристики как тембр голоса и темп речи.

Основными задачами в области обработки речи являются распознавание сказанного (*text-to-speech*) и идентификация по голосу, которая является частью задачи полной биометрической аутентификации. Также, есть другие задачи, связанные с аудио, например, удаление шумов из звукозаписи или набирающее популярность клонирование голоса, позволяющее обучить программу говорить любые слова и предложения любым голосом, данные о котором имеются в системе.

В настоящее время существуют решения по биометрии голоса, которые используются в *call*-центрах для идентификации мошенников или в банковской среде для биометрии.

Целью данного проекта является разработка алгоритма на базе нейросетевого подхода для решения задачи идентификации человека по его голосу. Из этого следуют следующие подзадачи:

- формирование обучающей выборки (включая методику сбора данных, преобразование данных, выбор формата и др.);
- проектирование архитектуры нейронной сети;
- проведение исследований по применимости выбранной архитектуры для решения поставленной задачи.

1. Аудио файл

При создании нейронной сети, работающей с аудио файлами, важно понимать устройство аудио файла. Для того, чтобы обучить нейросетевую модель, необходимо создать набор данных, который в данном случае будет полностью состоять из аудиофайлов.

В случае с идентификацией по голосу аудиофайлы представляют собой запись человеческого голоса. Эти записи получены с использованием микрофона и являются аналоговыми по своей природе, однако их хранение и обработка производится в цифровом виде. Это возможно благодаря аналогово-цифровому преобразованию звука. Аналоговый аудиосигнал можно представить в виде функции, для перевода его в цифровой вид производится дискретизация, то есть вычисляется значение функции

для каждого аргумента, что можно увидеть на рисунке 1. В случае с аудио файлом вместо аргумента мы имеем значение времени, а в качестве значения функции – значение амплитуды. Множество значений аргументов (значений времени) называется частотой дискретизации, а шагом дискретизации называется разница между значениями двух соседних аргументов, то есть временной интервал между двумя измерениями [1].

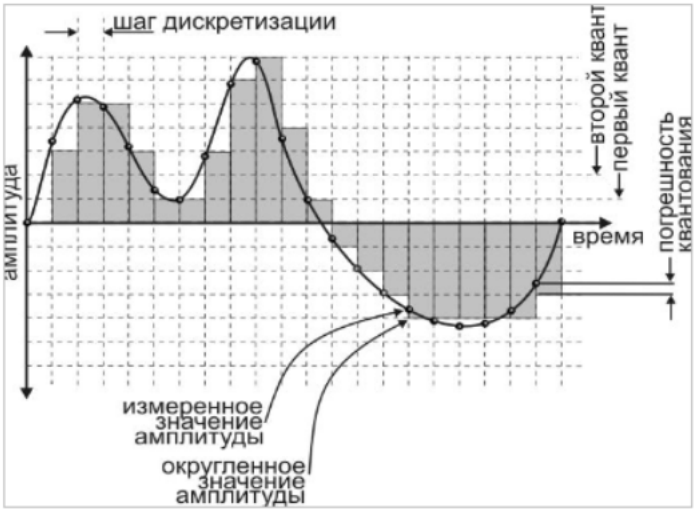


Рис. 1. Аналогово-цифровое преобразование звука

2. Сбор данных

Для идентификации человека по его голосу необходимо собрать данные о голосе. В этом проекте было принято решение не прибегать к использованию уже имеющихся наборов данных, а сгенерировать свой.

Было принято решение использовать аудио данные из открытых источников, после чего преобразовывать их так, как лучше всего подходит для нейросетевых моделей. Экспериментальным путем было выяснено, что лучше других себя показывают одноканальные аудио с частотой дискретизации 16 000 Гц, длиной файла в одну секунду и отсутствием в нем пауз в речи. Паузы в речи могут достигать нескольких секунд и так как длина файла была выбрана одна секунда – будут получатся файлы в которых нет информации о голосе.

После получения аудио данных из открытых источников было необходимо удалить все паузы из аудио. Так как аудио файлы могут быть записаны в разных условиях, необходимо использовать способ удаления пауз речи, который динамически будет определять эти паузы, а не будет зависеть от выставленных пороговых значений децибел. Для этого была использована библиотека *pyAudioAnalysis* [2]. Эта библиотека создана специально для создания наборов данных для нейросетей и в нее включено множество необходимых методов, но в рамках этого проекта было достаточно только одного – метода удаления тишины. Работу метода можно увидеть на рисунке 2.

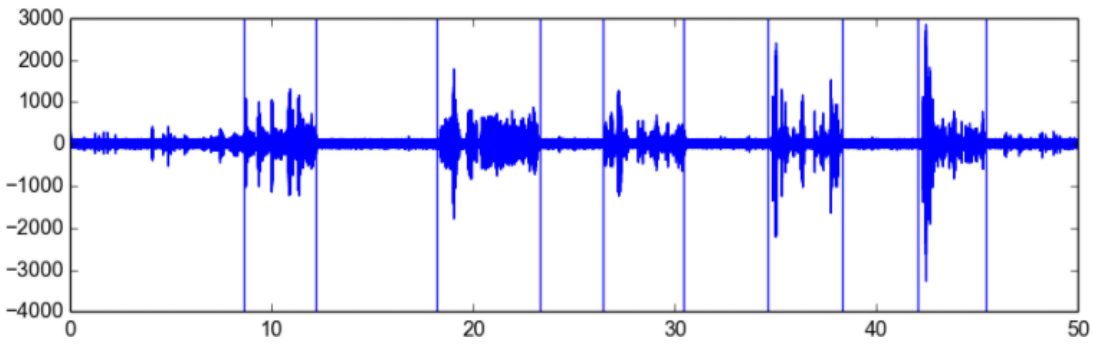


Рис. 2. Работа метода удаления тишины (SilenceRemoval)

После удаления пауз при помощи библиотеки *pyAudioAnalysis* были получены аудио файлы различной длины, с которыми в последствии необходимо было произвести дальнейшее преобразование. Для дальнейших операций над аудиосигналом было решено использовать библиотеку *pydub* [3], так как в этой библиотеке есть все необходимые для дальнейшей работы инструменты: разделение аудио файла, изменение его частоты дискретизации и изменение количества каналов аудио файла. С использованием библиотеки *pydub* был создан класс, позволяющий разделять аудио файл на множество файлов с заданным шагом.

Для анализа полученных аудио данных была использована библиотека *torchAudio* и практическое руководство по работе с данной библиотекой [4]. Так, например, с помощью этой библиотеки было выяснено, что до преобразования аудио имело двухканальный звук и частоту дискретизации 48 000.

3. Преобразование данных

Затем было принято решение использовать мел-спектрограмму в качестве метода представления аудио файла, так как мел шкала лучше отражает особенности речи, а спектрограмма дает больше информации об аудио файле. Вид спектрограмме можно увидеть на рисунке 3.

Мел – единица высоты звука, основанная на восприятии этого звука органами слуха. Она также часто используется в анализе аудио. Высота звука воспринимается человеческим ухом нелинейно. Эта зависимость описывается математической формулой: $m = 1125\ln(1 + f/700)$.

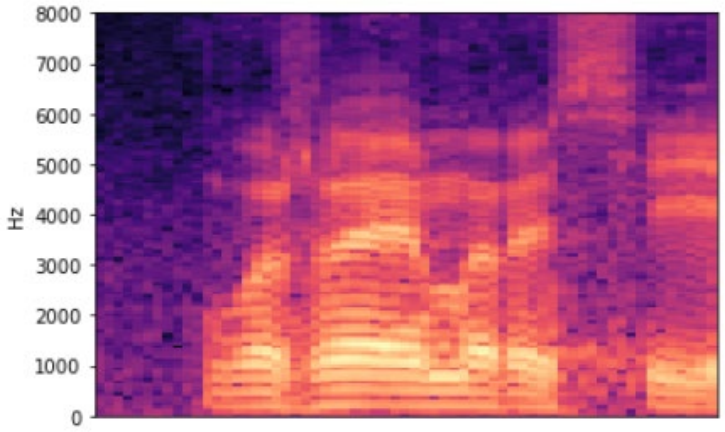


Рис. 3. Вид мел-спектрограммы

Таким образом, был получен готовый набор данных для обучений нейронных сетей разного типа и архитектуры. Полученный набор данных создавался для идентификации человека по голосу, но может также применятся и для задачи клонирования голоса.

4. Создание нейросетевой модели

Для создания нейросетевой модели был использован фреймворк *tensorflow* [5]. Архитектура нейронной сети представляет собой сверточную сеть, состоящую из пяти слоев: входной слой, сверточный слой, слой подвыборки по максимальному значению, слой свертки, слой, который трансформирует матрицу в вектор и обычный полносвязанный слой. Общий вид сверточной сети можно увидеть на рисунке 4. Выходные данные представляют собой число, которое обозначает номер диктора, которому принадлежит голос из аудио файла.

Работая со спектрограммами можно не ограничиваться в выборе нейросетевой архитектуры, потому как спектрограмма очень схожа с черно-белыми изображениями по своей структуре, в отличии от аудио в исходном виде, который больше напоминает одномерный массив [6].

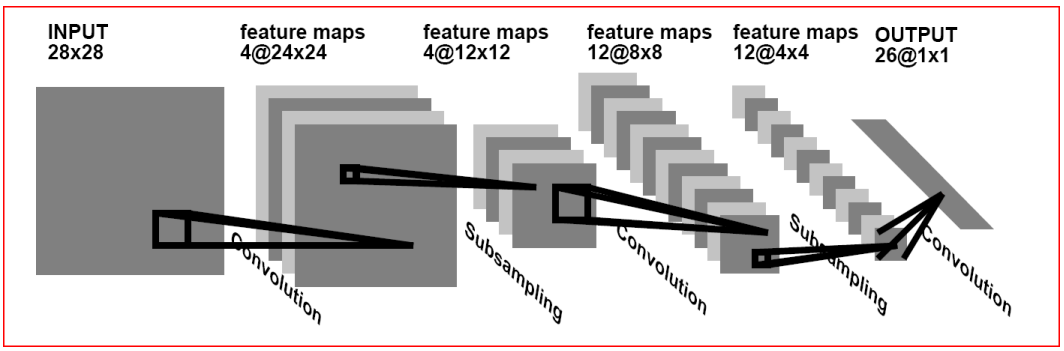


Рис. 4. Архитектура обобщенной сверточной нейроной сети

Заключение

В работе проведен начальный этап исследования возможностей нейросетевых моделей в задаче идентификации по голосу. Собрана пробная обучающая выборка, содержащая голосовые сэмплы двух дикторов. Работа является первым этапом в рамках исследования возможностей нейросетевого подхода к задаче идентификации по голосу и формированию рабочего окружения (*workflow*) для исследований подобного рода. В рамках дальнейшего исследования идет добавление большего количества голосов людей в обучающую выборку и соответствующие этому доработки скриптов анализа аудио данных.

На текущий момент с использованием сверточной нейронной сети была достигнута точность в 99.9% для распознавания двух человек. В качестве пробного результата такая эффективность является признаком того, что выбранное направление исследований является достаточно интересным для дальнейшей проработки.

Список литературы

1. Щеголев А.В. Распознавание звука при помощи нейронных сетей. Курсовая работа // Кубанский государственный университет, 2018.
2. Библиотека pyAudioAnalysis для извлечения признаков, классификации, сегментации. – 2019. – [Электронный ресурс]. URL: https://github.com/tyiannak/pyAudioAnalysis.
3. Джеймс Р. Библиотека обработки аудио Pydub. – 2016. – [Электронный ресурс]. URL: https://github.com/tyiannak/pyAudioAnalysishttps://github.com/jiaaro/pydub/releases.
4. Манипуляции с аудио с помощью torchAudio, руководство для начинающих. – [Электронный ресурс]. URL: https://pytorch.org/tutorials/beginner/audio_preprocessing_tutorial.html.
5. Фреймворк tensorflow. – [Электронный ресурс]. URL: https://www.tensorflow.org/.
6. LeCun Y., Bengio Y. Convolutional Networks for Images, Speech, and Time-Series, in Arbib, M. A., The Handbook of Brain Theory and Neural Networks. – MIT Press, 1995.

РАСПОЗНАВАНИЕ ДОРОЖНОЙ РАЗМЕТКИ И АНАЛИЗ ДОРОЖНОЙ СИТУАЦИИ

Мельник Николай Андреевич[1], Стрельцова Оксана Ивановна[2]

[1] Студент;
Государственный университет «Дубна»;
Международная школа по информационным технологиям
«Аналитика больших данных»;
Направление обучения по основной образовательной программе:
Информатика и вычислительная техника, группа 4011;
e-mail: columbiysky@uni-dubna.ru.

[2] к.ф.-м.н., старший научный сотрудник;
Лаборатория информационных технологий им. М.Г. Мещерякова;
Объединенный институт ядерных исследований.
Доцент;
Кафедра распределенных информационно-вычислительных систем;
Государственный университет «Дубна».

Ключевые слова: компьютерное зрение, глубокое обучение, нейронные сети.

Введение

В данной статье исследуется возможность анализа дорожной ситуации с использованием компьютерного зрения, включающая в себя распознавание объектов и участников дорожного движения. Задачей данной работы является изучение технологий компьютерного зрения для распознавания участников дорожного движения, таких как пешеходы, машины, дорожные знаки, светофоры с использованием нейронных сетей.

1. Выбор набора данных

В сети Интернет существует множество наборов данных, содержащих размеченные фотографии машин, дорожных знаков и т.д. Был выбран набор данных *BDD*100*k*, так как он содержит всю необходимую информацию, а также достаточно популярен в сообществе, из-за чего в сети достаточно информации о том, как работать с этим набором данных. Пример (см. рис. 1) представляет собой размеченное изображение, где для маски различных типов выбраны условные цвета, иллюстрирующие принадлежность сегмента изображения к конкретному типу: синим цветом отмечены автомобили, красным пешеходы, желтым светофоры, фиолетовым дорожные знаки.

Рис. 4. Пример разметки объектов на фотографии в наборе данных

2. Выбор средств реализации

При реализации данной работы использовались: язык программирования *Python*, библиотека компьютерного зрения *OpenCV*, фреймворк глубокого обучения *Darknet*. Архитектурой нейронной сети является *YOLOv*3.

Изначально была попытка использования архитектуры *YOLOv*4. Для обучения сети на выбранном наборе данных необходимо в файле конфигурации изменить значение атрибута *classes* на количество распознаваемых объектов, в данной работе это значение равно 10. Далее во всех слоях, идущих перед теми, в которых был изменен атрибут *classes* необходимо изменить атрибут *max_batches*, который равен произведению количества классов: 20000. Но после тренировки сети распознавание оказалось неточным, многие объекты не распознавались вообще, а те, что распознавались, отмечались сетью в неверном месте. При использовании *YOLOv*3 такой проблемы не возникло.

3. Обучение нейронной сети

В качестве рабочего окружения был выбран *Google Colaboratory*, в связи с простотой установки необходимых библиотек. Перед началом обучения необходимо подготовить данные, а именно, создать *.names* файл, заполненный именами классов распознаваемых объектов; *.train* и *.val* файлы, содержащие пути до фотографий из тренировочного и валидационного наборов; *.txt* файлы для каждой фотографии из тренировочного набора, в файлах содержится информация о классе объекта на изображении, а также его позиция; *.data* файл, содержащий информацию о количестве классов из файла *.names* пути до файлов *.train*, *.val*, *.names* и путь к директории, в которую будут сохраняться резервные копии текущей конфигурации нейронной сети при обучении (см. рис 2) [1].

bdd100k.data

```
classes = 10
train  = /content/train.txt
valid = /content/val.txt
names = /content/bdd100k.names
backup = /content/drive/MyDrive/edu/4k/Школа/НПД/backup
```

bdd100k.names

```
traffic light
traffic sign
car
person
bus
truck
rider
bike
motor
train
```

Рис. 5. Заполненные .data и .names файлы

Обучение нейронной сети запускается командой «!./*darknet detector train *.data *.cfg weights -dont_show*» Обучение длилось около двух часов, была пройдена 18671 эпоха обучения, средняя ошибка распознавания составила 7,391626 [2].

4. Результаты обучения

Если распознать изображение из тестового набора, то верно будут распознаны 15 из 16 объектов на изображении (см. рис. 3).

Рис. 6. Результат распознавания с использованием фотографии из тестового набора

При использовании случайной фотографии из интернета, результат также удовлетворительный 8 верно распознанных объектов из 13 (см. рис. 4).

Рис. 7. Результат распознавания с использованием фотографии из интернета

При распознавании объектов на видео результат несколько хуже, около 60% верно распознанных объектов в кадре, при этом средняя частота обработки равна 33 кадра в секунду.

Заключение

Данная работа представляет собой начальный этап исследования, и полученные результаты несут оценочный характер. В ходе дальнейшего исследования одним из следующих этапов является этап получения статистически достоверной оценки эффективности полученного решения. Также одним из направлений, которое необходимо отметить – это разработка алгоритма распознавания дорожной разметки и сегментации полос движения, что может оказать существенное влияние на итоговую эффективность решения задачи. Также в планах разработки исследовать эффективность работы ансамбля из нейронных сетей.

Список литературы

1. Официальная документация к фреймворку Darknet. – [Электронный ресурс]. URL: https://github.com/AlexeyAB/darknet#how-to-train-to-detect-your-custom-objects.
2. Шолле Ф. Глубокое обучение на Python. – СПб.: Питер, 2018.

РАЗРАБОТКА СЕРВИСА НА ОСНОВЕ НЕЙРОСЕТЕВОГО ПОДХОДА ДЛЯ ПРЕОБРАЗОВАНИЯ ИЗОБРАЖЕНИЯ В АНИМЕ ПЕРСОНАЖА

Бачинский Станислав Ватальевич[1], Колобов Сергей Игоревич[2], Стрельцова Оксана Ивановна[3]

[1] Студент;
Государственный университет «Дубна»;
Международная школа по информационным технологиям
«Аналитика больших данных»;
Направление обучения по основной образовательной программе:
Автоматизация технологических процессов и производств, группа 4231;
e-mail: staz26@yandex.ru.

[2] Студент;
Государственный университет «Дубна»;
Международная школа по информационным технологиям
«Аналитика больших данных»;
Направление обучения по основной образовательной программе:
Программная инженерия, группа 4251;
e-mail: s.kolobov99@mail.ru.

[2] к.ф.-м.н., старший научный сотрудник;
Лаборатория информационных технологий им. М.Г. Мещерякова;
Объединенный институт ядерных исследований.
Доцент;
Кафедра распределенных информационно-вычислительных систем;
Государственный университет «Дубна».

Ключевые слова: аниме, селфи, передача стиля, машинное обучение, нейронные сети.

Введение

На текущий момент огромной популярностью пользуются приложения для обработки и стилизации изображений, получаемых при съемках на цифровую камеру: всевозможные фильтры, маски, а также представления одного изображения в стиле другого.

В данной статье рассматриваются методы переноса стиля с одного изображения на другое для решения задачи синтеза изображения аниме персонажа, на основе изображения реального человека, для дальнейшего создания сервиса преобразования изображения человека в персонажа аниме.

1. Архитектура нейронной сети VGG-19

Перенос стиля с одного изображения на другое можно считать проблемой переноса текстуры. При передаче текстуры цель состоит в том, чтобы синтезировать текстуру исходного изображения, так, чтобы сохранить семантическое содержание целевого изображения [1].

Для передачи стилей используем сеть *VGG-19*, которая состоит из ряда сверточных слоев, слоев пулинга и нескольких полностью связанных слоев. *VGG-19* — это нейронная сеть, способная классифицировать более тысячи различных объектов. Эта сеть была обучена на огромном наборе данных *ImageNet*. На каждом из своих слоев, нейронная сеть извлекает некоторые элементы из изображения, представляющие собой его характеристику. В конце концов, все характеристики складываются в итоговую классификацию.

VGG-19 делится на две части:

- *VGG-19 feachures*, которые представляют все сверточные и субдискретизирующие слои;
- *VGG-19 classifier* — три линейных слоя классификатора.

В работе использовано только часть сверточных и субдискретизирующих слоев предварительно обученной модели *VGG-19* (см. рис. 1).

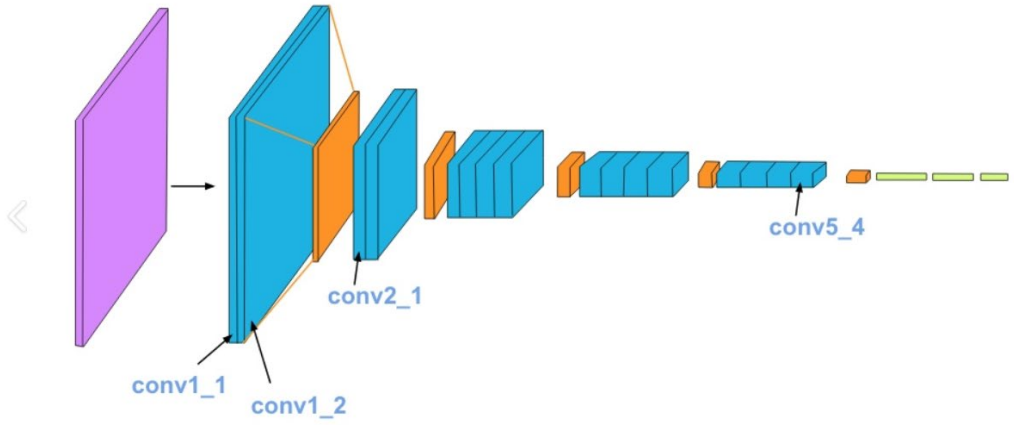


Рис. 1. Структура нейронной сети VGG-19

Результирующее изображение получается итерационным способом. На каждой итерации подаем на вход нейронной сети текущее изображение и получаем набор признаков для него. Вычисляем потери от исходного изображения и стилистического изображения. Используя полученные потери и соответствующие им веса, вычисляем общую потерю и используем любой оптимизатор для выполнения градиентного спуска, чтобы изменить сгенерированное изображение так, чтобы оно уменьшало его потерю после каждой итерации (см. рис. 2).

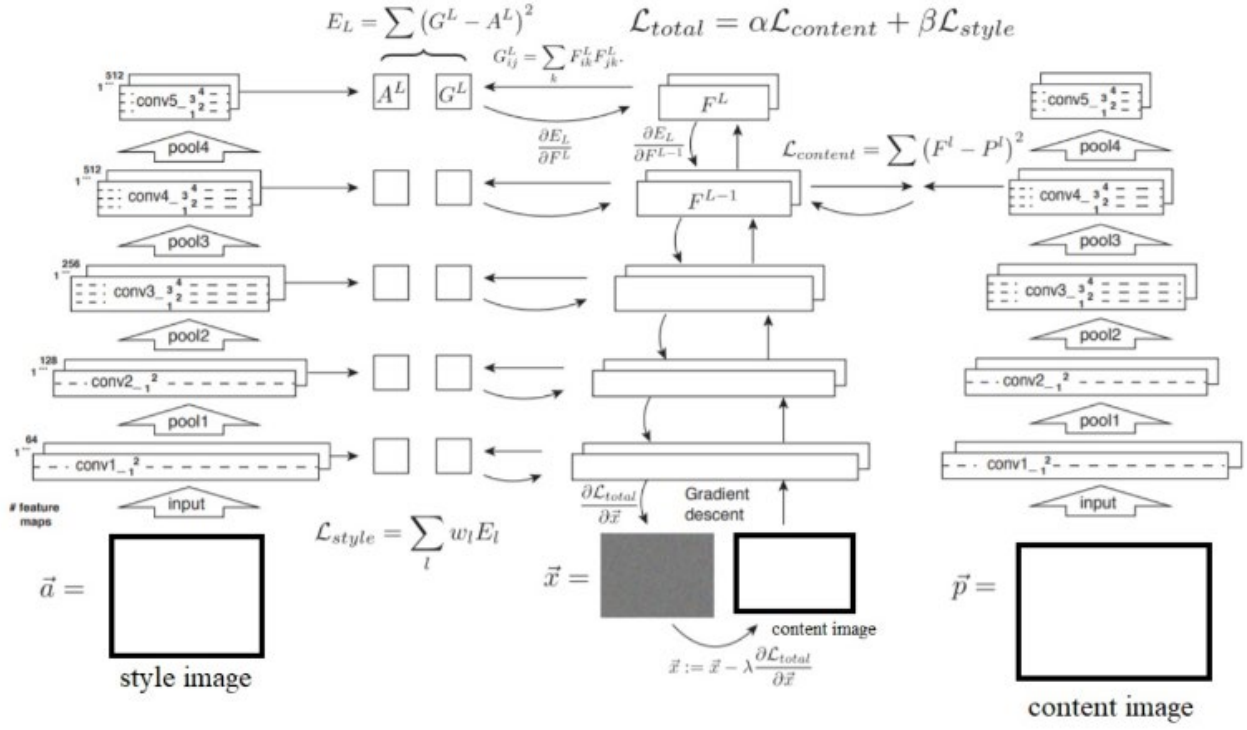


Рис. 2. Схема процесса получения результирующего изображения

2. Междоменный перенос с помощью CycleGAN

CycleGAN — это тип генерирующей сети, используемой для передачи стиля изображения. Обучение осуществляется без учителя, что означает, что невозможно однозначно сопоставить изображе-

ния из обоих этих доменов. Сеть способна распознавать объекты на изображениях исходного домена и выполнять необходимые преобразования, чтобы соответствовать внешнему виду объекта на изображениях целевого домена. Первоначальная реализация этого алгоритма была обучена «превращать» лошадей в зебр, яблоки в апельсины, а фотографии - в картины. Используя изображения человеческих лиц, взятые из набора данных *Selfie2Anime* [2], обучим пару генеративных состязательных сетей, одна из которых изучает визуальный стиль селфи, а другая изучает аниме.

Процесс обучения начинается с оригинального изображения лица человека. Обучим две глубокие сети, один генератор и один дискриминатор. Дискриминатор со временем научится различать реальные и смоделированные изображения лица. Генератор будет обучен преобразовывать входное изображение из исходного домена в целевой домен с использованием случайных изображений персонажей аниме из обучающего набора [3].

Чтобы убедиться, что это преобразование имеет смысл, вводим условие восстановления. Это означает, что одновременно обучаем другой набор генератора/дискриминатора, который восстанавливает изображение в исходном домене из целевого домена. Соблюдаем условие, что эта реконструкция должна быть «похожей» на исходное изображение, вычисляя значение функции потерь, которую стремимся минимизировать в процессе обучения (см. рис. 3, 4). Это похоже на автоэнкодер, за исключением того, что ищем кодировку не в скрытом слое для промежуточного шага, а для всего изображения в целевом домене [4].

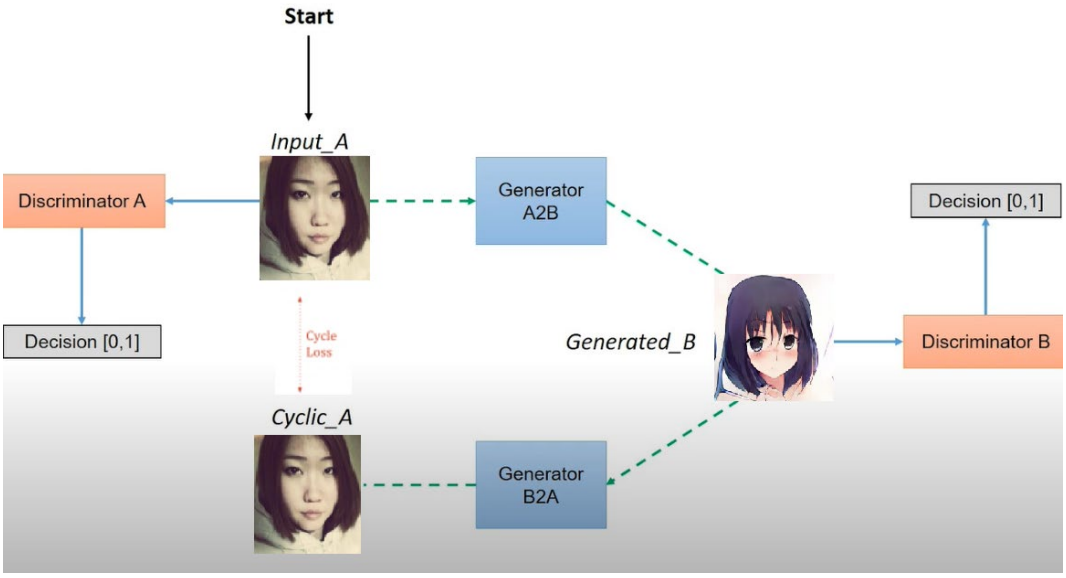


Рис. 3. Схема работы сети CycleGAN

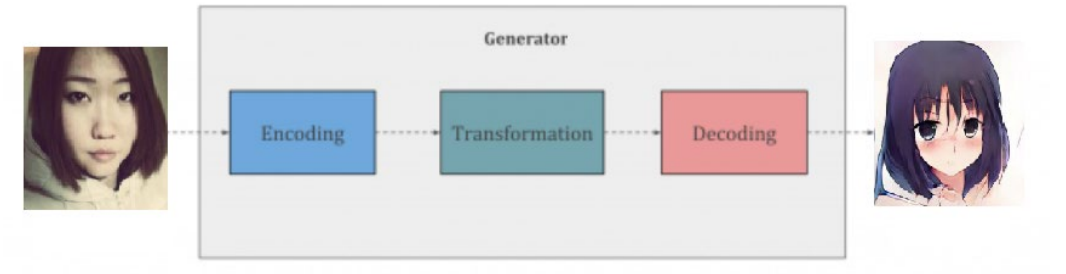


Рис. 4. Схема генератора, изображенного на рисунке 3

Используемая генераторная сеть состоит из трех основных сверточных блоков. Первый находит кодировку изображения лица человека в скрытом слое более низкого измерения. Эта кодировка преобразуется в другую, которая представляет персонажа аниме на том же скрытом слое. Затем декодер создает выходное изображение из преобразованной кодировки, давая нам изображение человеческого

лица, которое выглядит как персонаж аниме [3]. Примеры полученных изображений показаны на рисунке 5.

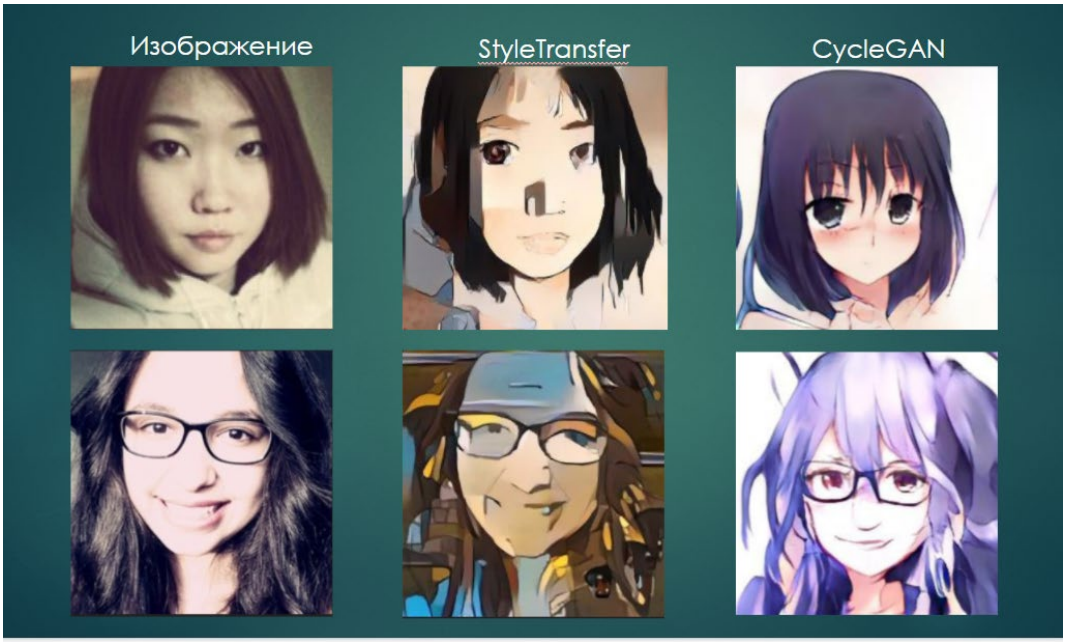


Рис. 5. Примеры работы алгоритмов

Заключение

Сравнили методы переноса стиля изображений, описанные выше. Провели визуальную экспертную оценку полученных результатов, так как такая оценка будет применяться реальными пользователями при использовании приложения. Результаты работы нейронных сетей были продемонстрированы 30 независимым экспертам, для выбора лучшего из алгоритмов. По результатам оценки, 100% экспертов признали результаты работы *CycleGAN* лучшими.

В дальнейшем, планируется разработать сервис, для преобразования изображения человека в аниме персонажа, используя *CycleGAN* нейронную сеть.

Список литературы

1. Leon A. Gatys L.A., Ecker A.S., Bethge M. Image Style Transfer Using Convolutional Neural Networks. – 2019. – [Электронный ресурс]. URL: https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Gatys_Image_Style_Transfer_CVPR_2016_paper.pdf.
2. Набор данных Selfie2Anime. – 2020. – [Электронный ресурс]. URL: https://www.kaggle.com/arnaud58/selfie2anime.
3. Tensorflow CycleGAN. – 2020. – [Электронный ресурс] URL: https://www.tensorflow.org/tutorials/generative/cyclegan.
4. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. – 2019. – [Электронный ресурс]. URL: https://junyanz.github.io/CycleGAN/.

COMPUTING & SOFTWARE ДЛЯ ЭКСПЕРИМЕНТОВ НА УСКОРИТЕЛЬНОМ КОМПЛЕКСЕ NICA

АДАПТАЦИЯ МЕТОДА РЕКОНСТРУКЦИИ ТРЕКОВ L1 ЭКСПЕРИМЕНТА CBM НА FAIR ДЛЯ КООРДИНАТНЫХ ДЕТЕКТОРОВ ЭКСПЕРИМЕНТА BM@N ПРОЕКТА NICA

Ходырев Иван Сергеевич[1], Герценбергер Константин Викторович[2], Пряхина Дарья Игоревна[3]

[1] Студент;
Государственный университет «Дубна»;
Международная школа по информационным технологиям «Аналитика больших данных»;
Направление обучения по основной образовательной программе: Прикладная математика и информатика, группа 4181;
e-mail: hodyrev.i@mail.ru.

[2] к.т.н., начальник группы;
Лаборатория физики высоких энергий им. В.И. Векслера и А.М. Балдина;
Объединенный институт ядерных исследований.
Доцент;
Инженерно-физический институт;
Государственный университет «Дубна».

[3] Научный сотрудник;
Лаборатория информационных технологий им. М.Г. Мещерякова;
Объединенный институт ядерных исследований.
Старший преподаватель;
Кафедра распределенных информационно-вычислительных систем;
Государственный университет «Дубна».

Ключевые слова: реконструкция треков частиц, алгоритм *CBM L*1, клеточные автоматы, эксперимент *BM@N*, комплекс *NICA*.

Введение

Эксперименты по физике элементарных частиц направлены на изучение процессов, проходящих в экспериментальных комплексах. Одним из комплексов для проведения исследований свойств плотной барионной материи является комплекс *NICA*, создаваемый на базе Объединенного института ядерных исследований.

NICA (*Nuclotron-based Ion Collider facility*, ионный коллайдерный комплекс на базе Нуклотрона) — сверхпроводящий ускорительно-накопительный комплекс тяжелых ионов, строящийся с 2016 года в Лаборатории физики высоких энергий им. В. И. Векслера и А. М. Балдина Объединенного института ядерных исследований (ЛФВЭ ОИЯИ), в городе Дубна Московской области [1, 2].

Эксперимент *BM@N*, сокращенно от *Baryonic Matter at Nuclotron* (барионная материя на Нуклотроне), проводится в ЛФВЭ ОИЯИ в рамках проекта *NICA*. В данном эксперименте изучаются столкновения ионов с фиксированной мишенью при энергиях до 6 ГэВ на нуклон, предоставляя возможность проведения исследований в области сверхплотной ядерной материи [3].

Чтобы получить информацию о частицах, образованных в результате столкновений, по данным, получаемым с детекторов установки в ходе эксперимента, проводится реконструкция событий. Результатом реконструкции являются восстановленные треки частиц, зарегистрированных детекторами, и их характеристики, например, масса частицы и ее импульс, а также первичная вершина столкновения и другие вспомогательные характеристики произошедших событий.

В ходе работы для эксперимента *BM@N* был адаптирован алгоритм реконструкции треков частиц *L*1, применяемый в эксперименте *CBM* [4] на установке *FAIR* [5]. В данном алгоритме, основанном на клеточных автоматах, используется метод фильтра Калмана [6] на этапе фитирования треков частиц и метод наименьших квадратов для оценки качества параметров треков [7, 8].

1. Адаптация алгоритма реконструкции треков CBM L1 в эксперименте BM@N

Для эксперимента *BM@N* разрабатывается программное обеспечение *BmnRoot* [9], основанное на программной среде *CERN ROOT* [10] и объектно-ориентированном фреймворке *FairRoot* [11]. Среда *BmnRoot* предоставляет набор классов и удобных инструментов для изучения производительности установки *BM@N*, разработки алгоритмов реконструкции событий и физического анализа данных.

В программной среде *BmnRoot* одновременно разрабатываются два алгоритма реконструкции треков (трекинга) заряженных частиц: *CellAuto* [9] и алгоритм, основанный на трекинге *CBM L*1.

Оригинальный алгоритм реконструкции треков частиц *L*1, основанный на методе клеточных автоматов, использует зарегистрированные сигналы с кремниевых станций эксперимента *CBM*. Для применения его в эксперименте *BM@N* в программной среде *BmnRoot* внесены изменения в этапы реконструкции треков метода *L*1, такие как: дигитизация, формирование кластеров, поиск и фитирование трек-кандидатов.

Основным отличием двух экспериментов с точки зрения реконструкции треков частиц являются наборы координатных детекторов, регистрирующих сигналы от пролетающих частиц. Данное отличие требует подготовки геометрической конфигурации, описывающей детекторы эксперимента *BM@N*, в структурах, использующихся в алгоритме *L*1. Также необходимо формирование описания откликов детекторов и их хранение в виде объектов, используемых оригинальным алгоритмом реконструкции треков частиц *L*1.

Первым этапом адаптации алгоритма реконструкции треков частиц стало создание структуры, описывающей геометрическую конфигурацию станций координатных детекторов *BM@N* для 6 сеанса эксперимента: кремниевых детекторов и газовых электронных умножителей (GEM). Далее выполняется обработка откликов детекторов средствами программной среды *BmnRoot* и предобработанные оцифрованные сигналы с детектора, полученные при пролете частицы через детектор, называемые дигиты, конвертируются в формат, используемый в алгоритме реконструкции треков частиц *L*1. На последующих этапах работы внедренного алгоритма используются геометрическая конфигурация координатных детекторов и сконвертированные в требуемый формат дигиты.

В программной среде *BmnRoot* реализованы классы, представляющие описание откликов детекторов в виде списка дигитов. Для внутренней трековой системы формируются два списка дигитов: от кремниевого детектора и от *GEM*-детектора. Для каждого дигита понадобилось увеличение максимального значения амплитуды сигнала. Также дигиты из обоих списков конвертируются в формат дигитов алгоритма *L*1 и формируют единый список.

Список конвертированных дигитов с внутренней трековой системы эксперимента *BM@N* далее используется для формирования кластеров. Для каждого кластера вычисляется центр тяжести. Координата пролета частиц через станцию детектора определяется пересечением пары кластеров в считывающих плоскостях станций детекторов. Тестировался данный этап методом попарного сравнения работы двух алгоритмов, *CellAuto* и *L*1, на моделированных данных эксперимента BM@N.

Список восстановленных координат пролета частиц через станции детекторов далее используется алгоритмом поиска трек-кандидатов, который базируется на методе клеточных автоматов. Основным понятием, которым оперирует данный алгоритм, является триплет — три хита на соседних станциях, составляющих часть возможного трека. На первом этапе работы алгоритма происходит поиск всех триплетов для всех троек соседних станций. На втором этапе проводится их фитирование для уточнения модели трека. Для формирования трек-кандидатов триплеты объединяются по двум совпадающим хитам. Далее происходит отбор треков частиц в соответствии со следующими условиями:

1. Отбирается самый длинный трек из пары треков, имеющих общие хиты.
2. При одинаковой длине треков в паре, имеющей общие хиты, выбирается трек с наименьшим значением среднеквадратического отклонения.

В результате работы алгоритма создается список треков частиц. Класс *CbmStsTrack* хранит информацию о найденном треке: список идентификаторов восстановленных хитов, из которых состоит трек, импульс частицы и другие вспомогательные свойства. Для каждого трека частицы рассчитываются параметры модели трека и ковариационной матрицы ошибок для этих параметров.

Список восстановленных треков частиц далее обрабатывается алгоритмом фитирования треков для улучшения модели трека. В алгоритме реконструкции треков *L*1 метод фитирования реализован при помощи фильтра Калмана.

По окончанию работы алгоритма результаты каждого этапа реконструкции треков частиц *L*1 сохраняются в файл с расширением *.root*. Структура данного файла представляет собой иерархическое дерево, ветками которого являются списки объектов, полученные при выполнении каждой из задач макроса реконструкции. Кроме того, была создана задача для оценки работы нового адаптированного алгоритма, результаты работы которой сохраняются в виде гистограмм в отдельный *ROOT*-файл.

2. Оценка качества реконструкции треков

При помощи макроса программной среды *BmnRoot* из файлов начального состояния после столкновения частиц, полученных генератором событий *DCM-QGSM* (*Dubna Cascade Model – Quark-Gluon String Model*, Дубненская каскадная модель с моделью кварк-глюонных струн), было смоделировано 10 тысяч Монте-Карло событий столкновения пучка аргона и медной мишени при энергии 3,2 ГэВ/нуклон. Данные события были затем реконструированы адаптированным методом *L*1, представленным выше.

Для оценки эффективности реконструкции хитов построена гистограмма распределения количества восстановленных хитов и Монте-Карло точек по станциям *GEM*-детектора. Данная гистограмма представлена на рисунке 1.

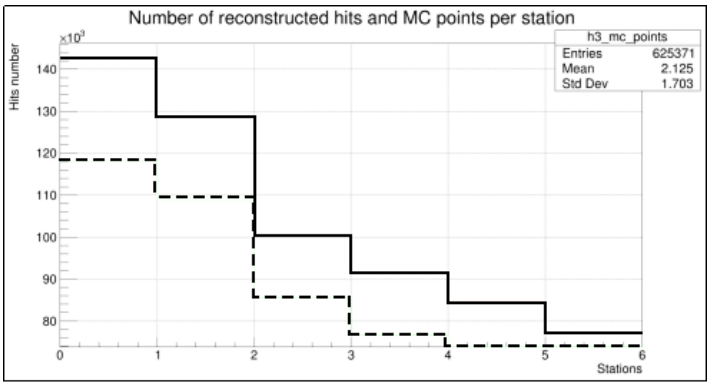


Рис. 1. Число восстановленных хитов (пунктирная линия) и Монте-Карло точек (сплошная линия) по станциям GEM-детектора эксперимента BM@N

Средняя эффективность адаптированного алгоритма реконструкции координат пролет частиц составила около 0.86 (86%). Для оценки качества адаптированного алгоритма реконструкции треков частиц *L*1 построены гистограммы распределения числа треков в событии (см. рис. 2).

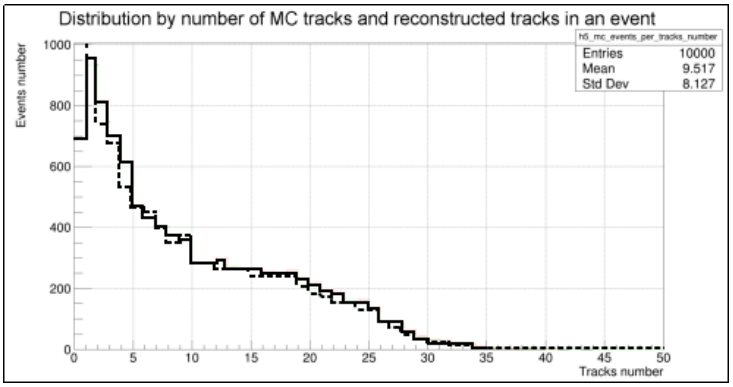


Рис. 2. Распределения чисел реконструированных треков (пунктирная линия) и Монте-Карло треков (сплошная линия) в смоделированных событиях эксперимента BM@N

Таким образом, в данной работе выполнена адаптация алгоритма реконструкции треков частиц *CBM L*1 для эксперимента *BM@N*. Для координаты *X* невязка по модулю на тестовых модельных данных составила 0.0065 см, для координаты *Y* — 0.041. Эффективность данного алгоритма реконструкции треков частиц составила порядка 85%. Оценка качества работы алгоритма на моделированных данных показала, что алгоритм удовлетворяет заданным изначальным критериям.

Заключение

В ходе работы было проведено исследование и описан существующий алгоритм реконструкции треков частиц в эксперименте *BM@N*. Проведен обзор этапов реконструкции треков частиц в экспериментах физики высоких энергий. Изучены методы восстановления координат пролета частиц, выбора модели трека, поиска трек-кандидатов и фитирования треков частиц, а также их реализация в эксперименте *BM@N*. Изучен алгоритм реконструкции треков частиц *L*1 эксперимента CBM. Также разобраны методы оценки качества алгоритмов реконструкции треков частиц.

В ходе работы адаптирован алгоритм реконструкции треков частиц *CBM L*1 для эксперимента *BM@N* и реализован макрос среды *CERN ROOT* для выполнения цепочки реконструкции в соответствии с новым алгоритмом. Оценено качество адаптированного алгоритма на моделированных данных средствами программной среды *BmnRoot*.

Эффективность адаптированного к программной среде *BmnRoot* эксперимента *BM@N* алгоритма реконструкции треков частиц *L*1 для моделированных данных составила около 85%. Данный метод реконструкции может запускаться менеджером выполнения задач вместе с другими алгоритмами программной среды *BmnRoot*. Результаты оценки качества этапов алгоритма на моделированных данных представлены в виде соответствующих гистограмм.

Работа по адаптации алгоритма будет продолжена, также будет проведена коррекция и отладка данного алгоритма на экспериментальных данных эксперимента BM@N.

Список литературы

1. Kekelidze V., Kovalenko A. NICA Complex and JINR — status and plans. – [Электронный ресурс]. URL: https://www.researchgate.net/publication/263052453_NICA_Complex_and_JINR_-_status_and_plans.
2. Отчет о состоянии и ходе выполнения мега-проекта «Комплекс сверхпроводящих колец на встречных пучках тяжелых ионов» (Комплекс NICA). – [Электронный ресурс]. URL: https://nica.jinr.ru/docs/Report_NICA_SB_(12)_short.pdf.
3. BM@N Conceptual Design Report. – [Электронный ресурс]. URL: http://nica.jinr.ru/files/BM@N/BMN_CDR.pdf.
4. The CBM Experiment at FAIR. Volker Friesel Gesellschaft für Schwerionenforschung mbH Planckstraße 1, 64291 Darmstadt, Germany.
5. GSI FAIR. – [Электронный ресурс]. URL: https://www.gsi.de/en/researchaccelerators/fair.html.
6. Jose A. Hernando. The Kalman Filter Technique applied to Track Fitting in GLAST // SCIPP 98/18, 1998, University of California, Santa Cruz.
7. Никитюк Н.М. Методы обработки информации с трековых детекторов заряженных частиц высоких энергий // Физика элементарных частиц и атомного ядра, ОИЯИ, 1995. – Том 26. – Выпуск 3.
8. Киряков А.А. Методы реконструкции координат в кремниевых микростриповых детекторах. // Препринт ИФВЭ 2003-38 ОНФ, Протвино, 2003.
9. Batyuk P., Gertsenberger K., Merts S., Rogachevsky O. The BmnRoot framework for experimental data processing in the BM@N experiment at NICA // EPJ Web of Conferences 214 (05027), 2019.
10. About ROOT. – [Электронный ресурс]. URL: https://root.cern/about/.
11. Al-Turany M., Uhlig F. FairRoot Framework. – [Электронный ресурс]. URL: https://pos.sissa.it/070/048/pdf.

ЧИСЛЕННОЕ МОДЕЛИРОВАНИЕ ДИКВАРКОВ В ПЛОТНОЙ И ГОРЯЧЕЙ ЯДЕРНОЙ МАТЕРИИ

Папоян Георгий Владимирович [1], Калиновский Юрий Леонидович[2]

[1] Студент;
Государственный университет «Дубна»;
Международная школа по информационным технологиям
«Аналитика больших данных»;
Направление обучения по основной образовательной программе:
Прикладная математика и информатика, группа 4181;
e-mail: pgv.17@uni-dubnal.ru.

[2] д.ф.-м.н., ведущий научный сотрудник;
Лаборатория информационных технологий им. М.Г. Мещерякова;
Объединенный институт ядерных исследований.
Профессор;
Кафедра высшей математики;
Государственный университет «Дубна».

Ключевые слова: Намбу-Иона-Лазинио, мезоны, дикварки.

Введение

Актуальной проблемой физики столкновения тяжелых ионов являются описание свойств ядерной материи при больших температурах и плотностях. В настоящее время проводимые эксперименты по столкновению ядер при высоких энергиях на *LHC* и *RHIC*, а также планируемые в ближайшем будущем на *FAIR* и *NICA* предоставят данные о поведении адронов в экстремальных условиях.

Квантовая хромодинамика (КХД) описывает взаимодействие кварков и глюонов на основе обмена цветовыми зарядами, приводящего к феномену конфайнмента кварков. Хорошо известно, что прямые расчеты в КХД невозможны, и для изучения свойств горячей и плотной ядерной материи используются различные эффективные модели, воспроизводящие все свойства КХД при низких энергиях. Проведено исследование поведения массы дикварка при конечной температуре.

Одной из таких моделей является модель Намбу-Иона-Лазинио (НИЛ), возникшая в 60х годах прошлого столетия с целью объяснения природы нарушения киральной симметрии.

Использование модели Намбу-Иона-Лазинио удобно для описания свойств мезонов и изучения свойств адронной материи при конечной температуре. Однако изучение барионов в модели НИЛ является уже нетривиальной задачей. Решение трехчастичного уравнения Фаддеева в модели НИЛ сводится к решению уравнения, схожего с уравнением Бете-Салпитера для масс мезонов, где в качестве одного из кварков выступает кварк-кварковая цветная пара, или дикварк. Дикварки, несмотря на то что переносят цвет, могут являться реальными частицами в малоизученной цветовой сверхпроводящей фазе кварковой материи [1] (в этой фазе цветовая симметрия нарушена), поэтому изучение свойств дикварков также является интересной задачей. Данная работа посвящена исследованию свойств дикварков при конечной температуре и плотности.

1. Мезоны и дикварки в модели НИЛ

Исследование дикварков проводилось в рамках $SU(2)$ модели Намбу-Иона-Лазинио, лагранжиан которой для случая с двумя ароматами кварков имеет вид [2]:

$$\mathcal{L}_{\mathcal{NJL}} = \bar{q}(i\partial - \widehat{m_0} - \gamma_0\mu)q + G_s[(\bar{q}q)^2 + (\bar{q}i\gamma_5\vec{\tau}q)^2] \quad (1)$$

G_s– четырехкварковая константа связи, $\bar{q}$ и q – кварковые поля, $\widehat{m_0} = diag(m_u^0, m_d^0)$, $m_u^0 = m_d^0$ – токовые массы кварков, $\vec{\tau}$ – матрицы Паули в пространстве $SU(2)$.

Масса кварков в модели определяется уравнением щели (уравнение Швингера-Дайсона) (2), которое можно получить, минимизировав по скалярному полю большой термодинамический потенциал, получаемый из лагранжиана модели (1) в приближении среднего поля:

$$m = m_0 + 8GN_cN_f mi I_1, \text{где} \tag{2}$$

$$\mathrm{I_1} = -\mathrm{i}\int_{\Lambda}\frac{\mathrm{d^3p}}{(2\pi)^3}\frac{1-\mathrm{f(E_p\mu)}-\mathrm{f(E_p+\mu)}}{2\mathrm{E_p}}. \tag{3}$$

Мезоны [3] в модели НИЛ вводятся как коллективные моды (кварк - антикварковые связанные состояния). Четырех-кварковое взаимодействие в лагранжиане в приближении случайных фаз приводит к возникновению Т-матрицы:

$$T_M(k^2) = \frac{2iG_s}{1-2G_s\Pi_M(k^2)}. \tag{4}$$

Вся информация о свойствах мезона содержится в функции Π_M, которую называют поляризационным оператором мезонов [4] и который определяется из поляризационной петли, состоящей из кварковой и антикварковой линий (см. рис. 1).

$$\Pi_M(k^2) = i\int\frac{d^4p}{(2\pi)^4}\mathrm{Tr}\left[\Gamma_M S(p+k)\Gamma_M S(p)\right], \tag{5}$$

где $S^{-1}(p) = (\hat{p} + \gamma_0 p_0 - m)$ - кварковый пропагатор, Γ_M - вершинная функция [5], зависящая от типа рассматриваемого взаимодействия. Так, для пиона $\Gamma_\pi = i\gamma_5\tau^5$ а скалярного сигма-мезона $\Gamma_\sigma = 1$.

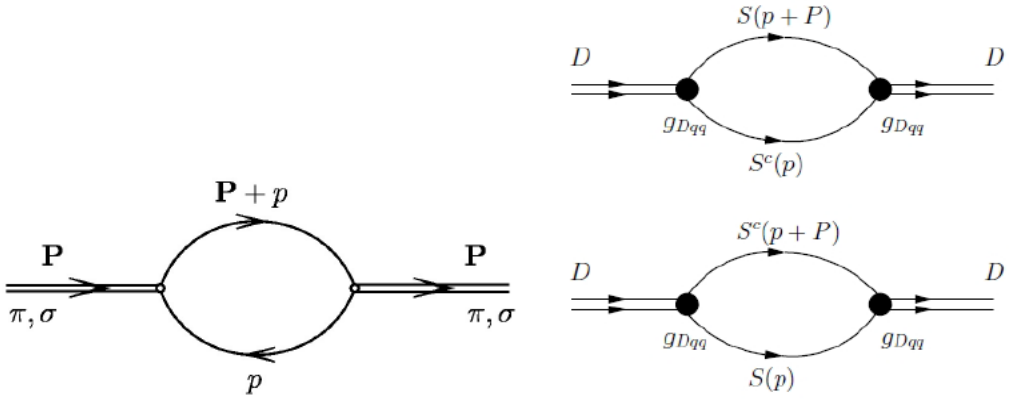


Рис. 1. Мезонная (слева) и дикварковая (справа) петли модели НИЛ

Для изучения дикварковых состояний потребуется построить матрицу рассеяния, аналогичную (2), однако вместо одной из кварковых линий нужно подразумевать зарядово-сопряженный кварк:

$$\Pi_D(k^2) = i\int\frac{d^4p}{(2\pi)^4}\mathrm{Tr}[\Gamma_M S(p+k)\Gamma_M S^C(p)]. \tag{6}$$

Уравнение Бете-Салпитера [6] и поляризационные операторы для вычисления масс мезонов, а также дикварков сводится в модели НИЛ к вычислению однотипных интегралов I_1, I_2, которые можно обобщить на случай конечных температур, воспользовавшись техникой, предложенной Матсубарой [7]. Покажем на примере мезонов:

$$\Pi_{\mathrm{ps}}(\mathrm{k^2}) = \mathrm{N_cN_fI_1} - 2\mathrm{N_cN_fk^2I_2(k^2)} \tag{7}$$

$$\Pi_{\mathrm{s}}(\mathrm{k^2}) = 4\mathrm{N_cN_fI_1} - 2\mathrm{N_cN_f(k^2-4m^2)I_2(k^2)}, \tag{8}$$

где интеграл I_1 показан в (3), и I_2 имеет вид:

$$I_2(k^2) = i\int \frac{dp}{(2\pi)^4}\frac{1}{(p^2-m^2)((p-k)^2-m^2)}. \quad (9)$$

2. Расчеты

Модель Намбу-Иона-Лазинио является неперенормируемой и нуждается в процедуре регуляризации расходящихся интегралов [8]. Наиболее часто используемым является регуляризация обрезанием интегралов по трехмерному импульсу. Свободные параметры модели фиксируются (см. табл. 1) из значений физических величин: константы распада пиона, плотности кваркового конденсата и значения массы пиона [9]. Самосогласованное решение уравнений щели, уравнения на массу связанного состояния и константы слабого распада пиона при нулевых значениях температуры и химического потенциала [10] позволяет зафиксировать параметры модели НИЛ.

Табл. 1. Параметры модели НИЛ

Рег.	m_0 [МэВ]	Λ [ГэВ]	G [ГэВ]$^{-2}$	F_π [ГэВ]	M_π [ГэВ]	m [ГэВ]
3D	5.5	0.639	5.227	0.092	0.139	0.31

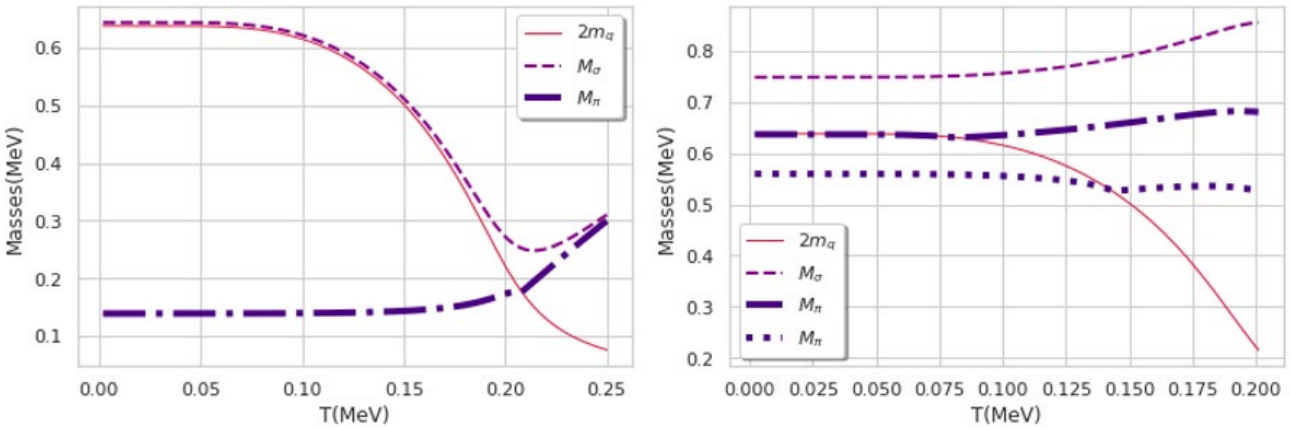


Рис. 2. Массы скалярного и псевдоскалярного мезонов (слева) и скалярного дикварка (справа) как функции температуры

На рисунке 2 показаны температурные зависимости массы кварков и мезонов в модели НИЛ. С ростом температуры масса кварка падает, а масса пиона возрастает и при некотором значении T совпадает с массой двух масс кварков. Такая температура называется температурой Мотта и выше температуры Мотта пион становится квазичастицей. На фазовой диаграмме этот переход соответствует переходу адронной фазы вещества в состояние кварк - глюонной плазмы. На рис. 2 показано, что при увеличении температуры масса скалярного мезона падает и при больших температурах происходит вырождение скалярного и псевдоскалярного мезонов (восстановление киральной симметрии).

Масса дикварка падает с ростом температуры и при некотором значении T также совпадает с суммарной массой составляющих его кварков. Из рисунка видно, что масса дикварка сильно зависит от константы связи.

Заключение

Для вычисления массы дикварка, как функции температуры, создан код, написанный на языке *FORTRAN* для самосогласованного решения системы нелинейных интегральных уравнений. Созданный код протестирован на вычислении физических характеристик мезонов, как функции температуры в рамках *SU*(2) модели НИЛ. Проведено исследование поведения массы скалярного дикварка при конечной температуре. Показано, что при увеличении константы связи дикварки ведут себя как ква-

зичастичные состояния. Написанный код позволяет исследовать температурное поведение и других типов дикварков.

Список литературы

1. Blanquier E. Journal of Physics G: Nuclear and Particle Physics, 2011.
2. Nambu Y., Jona-Lasinio G. Dynamical model of elementary particles based on an analogy with superconductivity I, II. // Physical Review Journals, 1961. – Vol. 122, 124.
3. Klevansky S. P. The Nambu-Jona-Lasinio model of quantum chromodynamics // Reviews of Modern Physics, 1992. – Vol. 64.
4. Kalinovsky Yu. L., Friesen A. V. Properties of mesons and critical points in the Nambu–Jona-Lasinio model with different regularizations // Physics of Particles and Nuclei Letters, 2015. – Vol. 12. – № 6.
5. Parker L., Christensen S. M. MathTensor: a system for doing tensor analysis by computer // Addison-Wesley, 1994.
6. Jones W. T., Fogelin R. J. The Twentieth Century to Quine and Derrida, A History of Western Philosophy // Harcourt Brace College Publishers, 1997.
7. Sheldrick G. M. A Short History of SHELXL // International Union of Crystallography and Oxford University Press, 2006.
8. Arduengo A. J. III, Harlow R. L., Kline M. A stable crystalline carbine // Journal of the American Chemical Society, 1991. – № 113.
9. Booth J. Chatt. The reactions of carbon monoxide and nitric oxide withtertiary phosphine complexes of iron({II}), cobalt({II}), and nickel({II}) // Journal of the Chemical Society, 1962.
10. Hope E., Bennett J., Stuart A. Fluorous zirconium phosphonates: novel inorganic supports for catalysis, in: Pacifichem // International Chemical Congress of Pacific Basin Societies, 2020. – № 961.

РАЗРАБОТКА «ОБЛАЧНОГО МЕТА-ПЛАНИРОВЩИКА»

Цегельник Никита Сергеевич[1], Балашов Никита Александрович[2]

[1] Студент;
Государственный университет «Дубна»;
Международная школа по информационным технологиям
«Аналитика больших данных»;
Направление обучения по основной образовательной программе:
Физика, группа 6161;
e-mail: tsegelnik@jinr.ru.

[2] Инженер-программист;
Лаборатория информационных технологий им. М.Г. Мещерякова;
Объединенный институт ядерных исследований.

Ключевые слова: облачные технологии, распределенные вычисления, системы планирования, разделение ресурсов, микросервисная архитектура.

Введение

Объединенный институт ядерных исследований (ОИЯИ) участвует в научных проектах международного уровня, предполагающих использование облачных и распределенных вычислительных ресурсов. В связи с этим Лабораторией информационных технологий им. М.Г. Мещерякова (ЛИТ) была реализована облачная инфраструктура, активно развивающаяся по сей день [1].

Данная инфраструктура предоставляет возможность свободно пользоваться сервисами и ресурсами ОИЯИ как конкретным пользователям, так и различным исследовательским группам. Каждый предоставляемый сервис имеет свои собственные системы и алгоритмы планирования, которые эффективно управляют ресурсами, выделенными для определенной группы пользователей. Однако стоит вопрос рационального распределения ресурсов, выделяемых для всех сервисов и всех групп пользователей, в зависимости от предъявляемых требований конкретной группы к конкретной системе, а также от реальной нагрузки.

В частном случае такую задачу может решить администратор, закрепленный за определенным типом пользователей и используемыми им сервисами. Однако разнообразие типов пользователей и сервисов делает слишком трудоемким, а в перспективе и невозможным реализацию такого подхода. В связи с вышесказанным возникла необходимость в разработке программного обеспечения, динамически распределяющего облачные ресурсы между сервисами ОИЯИ, в зависимости от типов пользователей и предъявляемых ими требований к необходимым системам. В данной работе мы представляем наше видение такого программного продукта, реализованный прототип, а также перспективы его дальнейшего развития.

1. Облачные ресурсы ОИЯИ

Платформой управления виртуальной инфраструктурой ОИЯИ выступает *OpenNebula*. Далее, упоминая облако или управление облачными ресурсами, мы будем иметь в виду именно платформу *OpenNebula*. Все сервисы, которые мы будем рассматривать, являются распределенными и представляют собой некоторым образом заранее преднастроенные шаблоны, из которых впоследствии создаются виртуальные машины (ВМ), выступающие в роли сервиса, с которым прямо или косвенно взаимодействует пользователь. Следовательно, такие системы являются как горизонтально (количество ВМ), так и вертикально (характеристики ВМ) масштабируемыми. Интересующими нас сервисами являются *HTCondor*, *Jupyter*, *Gitlab* (*CI Runner*).

HTCondor – это программная среда для высокопроизводительных вычислений, предназначенная для крупномасштабного распределенного распараллеливания ресурсоемких вычислительных задач [2]. Обычные пользователи могут отправлять свои задачи в очередь *HTCondor* на общих ресурсах ОИЯИ, но для каждой экспериментальной группы существует своя очередь на зарезервированных

для них ресурсах. В период обработки экспериментальных данных эти ресурсы активно используются, однако нередко в период между обработкой и следующим запуском соответствующего эксперимента эти ресурсы полностью простаивают, то есть не используются. Следовательно, в последнем случае представляется рациональным на некоторое время предоставить свободные ресурсы для других пользователей и/или систем.

Проект *Jupyter* включает в себя такие программные продукты, как *Jupyter Notebook*, *JupyterHub* и *JupyterLab*. Мы планируем реализовать свой сервис на базе *JupyterHub*, который позволяет автоматически создавать виртуальные сервера в облаке ОИЯИ для обработки пользовательских ноутбуков *Jupyter Notebook*.

Наконец, *Gitlab CI Runner* – это обработчик *CI*/*CD* задач. ОИЯИ предоставляет свой сервер *Gitlab*, обработчик *CI*/*CD* задач которого входит в рамки нашей деятельности.

Все вышеперечисленные облачные сервисы отличаются возможностью динамического изменения конфигураций, а также, как уже было упомянуто в начале этого раздела, горизонтальной и вертикальной масштабируемостью (особенно горизонтальной). Ввиду независимости таких систем друг от друга, управление масштабированием и перераспределением ресурсов между ними необходимо осуществлять централизованно с помощью стороннего программного обеспечения, которое смогло бы учитывать потребности всех облачных сервисов одновременно. Таким решением стал Облачный Мета-Планировщик, прототип которого мы представляем в следующем разделе.

2. Реализация прототипа

В связи с вышесказанным, можно выделить основные черты конечной системы, которые будет наследовать и отражать наш прототип: надежность, отказоустойчивость, рациональность в распределении ресурсов, высокая производительность, масштабируемость, журналирование. Также для прототипа было решено ограничиться взаимодействием с *OpenNebula* и *HTCondor*, исключая разделение по типу пользователей.

Учитывая вышеперечисленные требования, а также масштабность и различие в системах взаимодействия, было принято решение использовать микросервисную архитектуру [3, 4] и язык программирования *Python*. Теперь, говоря о сервисе, мы будем подразумевать именно некоторый реализованный нами микросервис в рамках Облачного Мета-Планировщика (ОМП). Во избежание путаницы мы решили называть такие сервисы одноименно с системами, для которых они будут выступать в качестве интерфейса. Для интеграции микросервисов между собой использовался программный пакет *Pyro* [5], который удовлетворяет требованиям отказоустойчивости и высокой производительности.

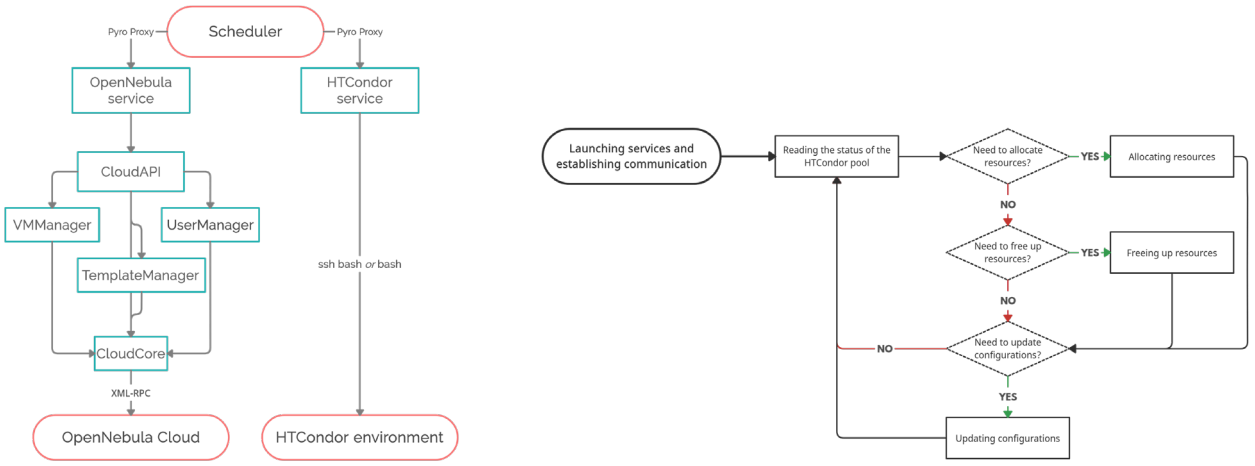
Итак, в рамках прототипа ОМП мы реализовали следующие три микросервиса (см. рис. 1а):

1. Сервис *OpenNebula*, взаимодействующий с облачной платформой *OpenNebula* через *XML-RPC API*, для работы с которым был реализован *CloudAPI*; является *Pyro*-сервером.
2. Сервис *HTCondor*, взаимодействующий с программной средой *HTCondor* посредством *CLI* или *bash*-скриптов; в случае, когда интерактивный узел (*Submit Node*) среды *HTCondor* и машина, на которой запущен данный микросервис, различны, используется *ssh*-соединение; является *Pyro*-сервером.
3. Сервис *Scheduler*, реализующий алгоритм планирования и взаимодействие со всеми остальными микросервисами через *Pyro*-прокси.

Для каждого сервиса предусмотрен набор параметров, который задается конфигурационным файлом. Таких файлов может быть несколько (соответственно каждому микросервису) или же один с общей конфигурацией.

Обобщенный цикл работы всей системы ОМП приведен на рисунке 1б, на котором процесс или действие выделены прямоугольниками, а принятие решения ромбами. На начальном этапе до самого цикла происходит задание конфигураций, запуск всех сервисов, затем *Scheduler* устанавливает связь с *OpenNebula* и *HTCondor*, которые, в свою очередь, настраивают связь (на этапе своего запуска) с облаком *OpenNebula* и интерактивным узлом *HTCondor* соответственно. При успешном установлении связи происходит считывание статуса очереди *HTCondor*, после которого принимается решение по

выделению/освобождению ресурсов. Затем, после выполнения соответствующего действия (включая состояние без изменений), принимается решение о необходимости обновления конфигурации. Наконец, цикл заканчивается обновлением конфигурации (или пропуском этого этапа) и возвращается к считыванию статуса очереди *HTCondor*. Следует отметить, что при обрыве связи между *Scheduler* и остальными сервисами, происходит автоматическое переподключение, параметры которого можно также указать в конфигурационном файле.



а) *б)*

Рис. 1. Блок-схемы реализованного прототипа Облачного Мета-Планировщика: архитектура (а) и обобщенный цикл работы системы (б)

На рис. 2 приведена подробная схема принятия решения необходимости выделения облачных ресурсов. Кратко поясним данную схему. Проверяются и сравниваются с переменными следующие параметры: минимальное/максимальное количество рабочих узлов *HTCondor* с текущим количеством узлов, число свободных задач и свободных рабочих узлов, облачная квота и количество задействованных ВМ. По итогу работы алгоритма принимается решение: выделять ресурсы или не выделять. Если выделять облачные ресурсы не требуется, цикл переходит к проверке на их освобождение. В противном случае, после выделения ресурсов цикл переходит к этапу обновления конфигураций.

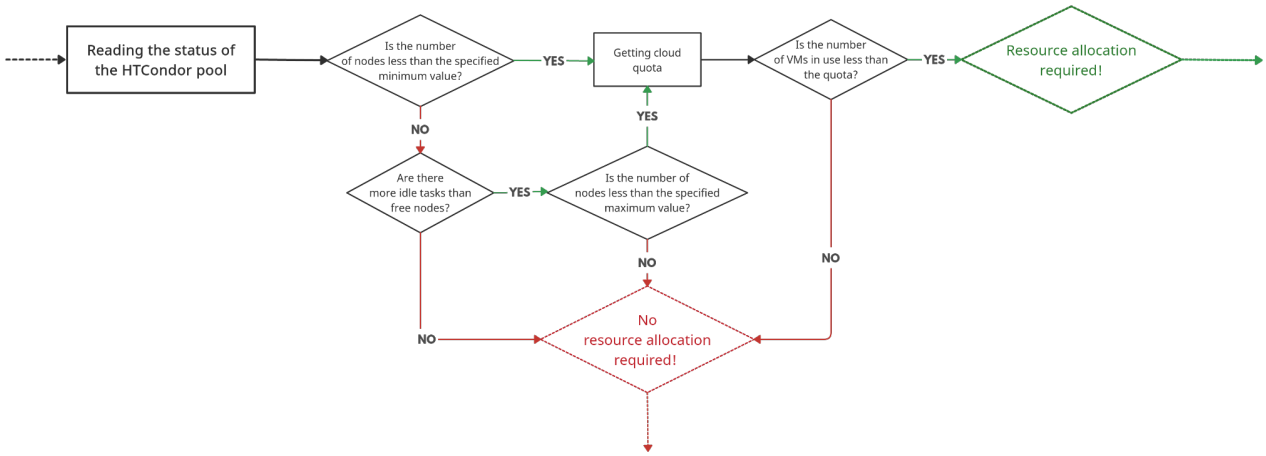


Рис. 2. Схема принятия решения необходимости выделения облачных ресурсов

Подробная схема принятия решений о необходимости освобождения облачных ресурсов и обновления конфигураций приведена на рисунке 3. В первом случае проверяется отсутствие свободных задач, наличие свободных узлов, а также меньше ли количество рабочих узлов заданного минимального значения. Если условие удовлетворяется, то облачные ресурсы освобождаются, после чего цикл переходит к принятию решения обновления конфигураций. Если условие не выполнено, то этап освобождения пропускается. На этапе принятия решения обновления конфигураций проверяются на

изменения все конфигурационные файлы сервисов ОМП. Если таковые имеются, то обновляется конфигурация лишь того сервиса, файл которой был изменен за время одной итерации цикла. После этого этапа цикл переходит на следующую итерацию.

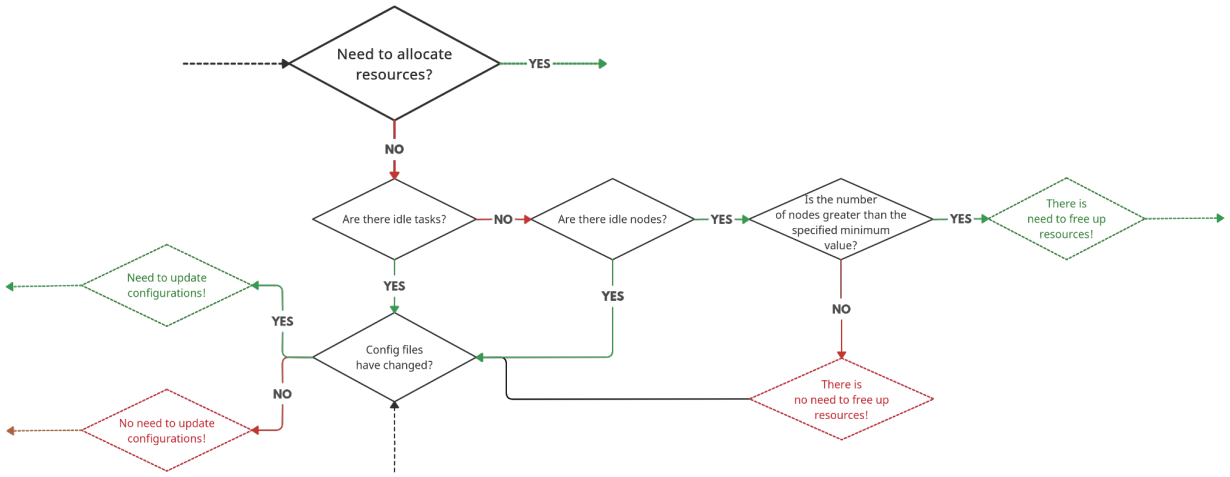


Рис. 3. Схема принятия решений необходимости освобождения облачных ресурсов и обновления конфигураций сервисов

Необходимо отметить, что каждое действие какой-либо части ОМП полностью журналируется. Излишние сведения вынесены в *debug*-режим, который включается соответствующим конфигурационным параметром.

Мы перечислили не все параметры конфигурации, например, различные задержки в операциях цикла или же количество повторов этих операций при неудаче. Полностью ознакомиться со всеми конфигурационными параметрами можно на *web*-странице документации проекта [6].

Заключение

На данный момент реализован описанный в прошлом разделе прототип Облачного Мета-Планировщика, реализована автоматическая сборка *Docker*-образа и его автоматическое обновление в реестре ОИЯИ. Как уже было сказано, проект задокументирован [6]. Процесс разработки ведется на платформе *Gitlab* ОИЯИ. Прототип введен в тестовую эксплуатацию, и на текущем этапе наблюдается и анализируется его поведение «в реальных условиях».

В наших дальнейших планах модернизировать прототип по результатам тестовой эксплуатации, улучшить имеющиеся сервисы, реализовать более сложный алгоритм планирования, реализовать автоматическое тестирование системы, внедрить новые сервисы, реализовать разделение ресурсов между пользователями/экспериментальными группами.

Список литературы

4. Baranov A.V., Balashov N.A., Kutovskiy N.A., Semenov R.N. JINR cloud infrastructure evolution // Phys. Part. Nuclei Lett, 2016. – Vol. 13. – № 5(203).
5. Litzkow M., Livny M., Mutka M. Condor – A Hunter of Idle Workstations // Proceedings of the 8th International Conference of Distributed Computing Systems, 1988.
6. Chen R., Li S., Li Z. From Monolith to Microservices: A Dataflow-Driven Approach // 24th Asia-Pacific Software Engineering Conference (APSEC), 2017.
7. Dragoni N. et al. Microservices: yesterday, today, and tomorrow // ArXiv160604036 Cs, 2016.
8. Pyro - Python Remote Objects. – [Электронный ресурс]. URL: https://pyro5.readthedocs.io.
9. Страница документации Облачного Мета-Планировщика. – [Электронный ресурс]. URL: http://cloud-meta-scheduler.pages.jinr.ru/cmscheduler.

Научное издание

Сборник отчетов о научно-проектной деятельности выпускников Международной школы по информационным технологиям «Аналитика больших данных»

Выпуск 2

В авторской редакции

Подписано в печать 11.08.2021.
Формат 60x84/8. Усл. печ. л. 6.
Тираж 60 экз. Заказ № 60220.

Издательский отдел Объединенного института ядерных исследований
141980, г. Дубна, Московская обл., ул. Жолио-Кюри, 6
E-mail: publish@jinr.ru
www.jinr.ru/publish/