«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ
КАФЕДРА ИНФОКОГНИТИВНЫХ ТЕХНОЛОГИЙ

Автоматическое рубрицирование текстов на основе word2vec.

Расчетно-пояснительная записка
курсового проекта по дисциплине
«Методы работы с большими данными»

студент группы
Журавлев Давид Александрович.

Преподаватели:
проф. Ю.Н. Филиппович
асс. Н.Г.Воробьев

Москва, 2024г

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель курсовой работы:

Приобретение навыков обработки и использования естественно-языковых текстовых данных.

Задачи курсовой работы:

- изучение материалов лекционных и практических занятий из курса LMS «Введение в интеллектуальные диалоговые системы» (разделы лекционных курсов — методы сбора данных и анализа запросов, методы формализации и оценки естественно языковых данных, методы технической реализации глубокого обучения и др.; практические занятия — практическое задание "Bag of words", практическое задание "Word2Vec" и др.; специальное информационное программное обеспечение — среда разработки IntelliJ IDEA, PyCharm или Visual Studio Code, текстовые файлы статей на тему дипломной работы; основная и дополнительная литература — см. список программ дисциплин);
- приобретение навыков анализа и изучение методов и приемов исследования ЕЯ описания предметной области (ПО) в процессе выполнения заданий курсовой работы;
- приобретение знаний и навыков интегрированного использования программного обеспечения (текстовых процессоров, электронных таблиц, специального программного обеспечения и других программных средств) для проведения обработки и использования ЕЯ ресурсов, характеризующих ПО;
- приобретение знаний и навыков по оформлению результатов анализа и исследования ПО при оформлении курсовой работы.

Краткое описание предметной области и ее естественно-языкового описания.

В качестве информационных ресурсов для курсовой работы были выбраны 20 научных статей, относящихся к предметной области «Медицина»: «Острые респираторные вирусные инфекции» (Acute Respiratory Viral Infection) и «Обсессивно-компульсивное расстройство» (Obsessive-Compulsive Disorder, OCD). Так же, были проанализированы 20 публикаций, соответствующих тематике бакалаврской выпускной квалификационной работы в подкатегориях «Бессерверная архитектура» и «Оркестрация контейнеров». Эти статьи представляют собой ключевые исследования и разработки, отражающие современные подходы и инструменты, применяемые в указанных областях.

Естественно-языковое описание выбранной предметной области включает большое количество терминов и концепций из сфер информационных технологий, медицины и программирования. Тексты анализируемых материалов написаны как на русском, так и на английском языках, что обусловлено международным характером научных исследований в этих направлениях. Многие англицизмы, такие как «Serverless», «Orchestration», «API», «Cloud Computing», «Obsessive-Compulsive Disorder», являются стандартом профессионального общения и не имеют адекватных русскоязычных аналогов. Их использование необходимо для точного отражения понятий и процессов.

Общий объем анализируемых статей составляет около 600 страниц формата А4. Материалы включают описания ключевых технологий, архитектурных решений и практических кейсов. Например, публикации, связанные с медициной, освещают диагностику, терапию и прогнозирование заболеваний с использованием современных инструментов, включая искусственный интеллект, системы поддержки принятия решений и другие. В свою очередь, статьи по бессерверным архитектурам и оркестрации контейнеров анализируют подходы к созданию масштабируемых и надежных приложений, таких как обработка событий, автоматизация рабочих процессов и управление ресурсами в облачных средах.

1 СБОР ЕСТЕСТВЕННО ЯЗЫКОВЫХ ТЕКСТОВЫХ ОПИСАНИЙ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Сбор естественно языковых описаний по тематике ВКР

В рамках выполнения курсовой работы задания по сбору естественно языковых описаний предметной области была выбрана тема, связанная с будущей ВКР: «Разработка платформы управления облачными функциями». Для углубления в предметную область исследования выделены две подтемы, представляющие основные аспекты разработки:

1. Бессерверная архитектура.
2. Оркестрация контейнеров.

1.1.1 Определение подтем

Подтемы выбраны исходя из ключевых аспектов разработки платформы управления облачными функциями:

– Бессерверная архитектура как основа для реализации гибких и масштабируемых вычислений.

– Оркестрация контейнеров для управления задачами, их распределения и мониторинга в облачной среде.

1.1.2 Поиск научных статей

Для поиска научных публикаций был использован ресурс КиберЛенинка, предоставляющий доступ к широкому спектру статей. По каждой подтеме проведен поиск с использованием ключевых слов, соответствующих тематике:

Для бессерверной архитектуры: “бессерверные вычисления”, “облачная архитектура”, “serverless технологии”.

Для оркестрации контейнеров: “контейнеризация”, “оркестрация”, “облачные платформы”, “управление контейнерами”.

1.1.3 Отбор публикаций

Из выдачи по каждому запросу выбраны наиболее релевантные статьи, учитывая их содержание, актуальность и значимость для изучения темы. Всего для анализа собрано по 10 статей для каждой подтемы.

Список статей по подтеме «Бессерверная архитектура»:

1. Динамические туманные вычисления и бессерверная архитектура: на пути к зеленым ИКТ.
2. DEVOPS в эпоху облачных технологий: современные практики и перспективы развития.
3. Особенности технологий бессерверных вычислений.
4. Разработка бессерверных мобильных приложений.
5. Современная облачная инфраструктура: бессерверные вычисления.
6. Облачные вычисления: современные тенденции, проблемы и перспективы.
7. Высвобождение производительности: глубокое погружение в приложения с высокой нагрузкой.
8. Эффективное использование облачных технологий в смешанном обучении.
9. Стратегии оптимизации для высоконагруженных приложений: повышение общей производительности.
10. Стратегия развертывания микросервисов в облаке.

Список статей по подтеме «Оркестрация контейнеров»:

1. Технологии изоляции приложений и инструментальные средства для управления контейнерами.
2. Управление контейнерами при построении распределенных систем с микросервисной архитектурой.
3. Исследование методов построения облачных платформенных сервисов и реализаций стандарта TOSCA.
4. Использование технологии контейнеризации как компонента обеспечения

информационной безопасности.

5. Построение требований и архитектуры облачного оркестратора платформенных сервисов.

6. Использование методов машинного обучения для предсказания уровня загрузки ресурсов системы.

7. Обзор технологий организации туманных вычислений.

8. Кэширование данных в мультиконтейнерных системах.

9. Автоматизированный инструментарий развертывания облачных сервисов.

10. Оптимизация высоконагруженных веб-проектов с использованием микросервисной архитектуры.

1.1.4 Формирование библиографического списка

Для каждой категории статей составлены отдельные таблицы, содержащие:

- Название статьи
- Ссылка на статью
- Дата обращения к ресурсу
- Ключевые слова
- Рубрика

Полученный библиографический список по рубрике «Бессерверная архитектура» представлен на рисунке 1.

Бессерверная архитекту...

	Название статьи	Ссылка на статью	Дата обращения к ресурсу	Ключевые слова	Рубрика
2	Динамические туманные вычисления и бессерверная архитектура: на пути к зеленым ИКТ	https://cyberleninka.ru/article/n/dinamicheskie-tumannye-vychisleniya-i-besservernaya-arhitektura-na-puti-k-zelenym-ikt	07.12.2024	туманные вычисления, услуги Телеприсутствия, бессерверная архитектура, метаэвристические алгоритмы, IMT-2030, fog computing, telepresence services, serverless architecture, metaheuristic algorithms	Бессерверная архитектура
3	DEVOPS В ЭПОХУ ОБЛАЧНЫХ ТЕХНОЛОГИЙ: СОВРЕМЕННЫЕ ПРАКТИКИ И ПЕРСПЕКТИВЫ РАЗВИТИЯ	https://cyberleninka.ru/article/n/devops-v-epohu-oblachnyh-tehnologiy-sovremennye-praktiki-i-perspektivy-razvitiya	07.12.2024	облачные технологии, инфраструктура как код, контейнеризация, бессерверные архитектуры, тренды	Бессерверная архитектура
4	Особенности технологий бессерверных вычислений	https://cyberleninka.ru/article/n/osobennosti-tehnologiy-besservernyh-vychisleniy	07.12.2024	облачные вычисления, бессерверные вычисления, модель FaaS, функции, автомасштабирование, архитектура приложения, cloud computing, Serverless, FaaS model, functions, autoscaling, application architecture	Бессерверная архитектура
5	Разработка бессерверных мобильных приложений	https://cyberleninka.ru/article/n/razrabotka-besservernyh-mobilnyh-prilozheniy	07.12.2024	мобильные приложения, бессерверная архитектура, облачные сервисы, разработка приложений, mobile applications, serverless architecture, cloud services, application development	Бессерверная архитектура
6	Современная облачная инфраструктура: бессерверные вычисления	https://cyberleninka.ru/article/n/sovremennaya-oblachnaya-infrastruktura-besservernye-vychisleniya	07.12.2024	облачные вычисления, бессерверные вычисления, области использования, преимущества, недостатки, cloud computing, serverless computing, fields of use, advantages, disadvantages	Бессерверная архитектура
7	ОБЛАЧНЫЕ ВЫЧИСЛЕНИЯ: СОВРЕМЕННЫЕ ТЕНДЕНЦИИ, ПРОБЛЕМЫ И ПЕРСПЕКТИВЫ	https://cyberleninka.ru/article/n/oblachnye-vychisleniya-sovremennye-tendentsii-problemy-i-perspektivy	07.12.2024	облачные вычисления, цифровая трансформация, гибкость, масштабируемость, скорость внедрения, снижение затрат, доступ из любой точки мира, безопасность, аналитика, большие данные, совместная работа, кибербезопасность, гибридные облачные решения, крауд-компьютинг, медицина, биотехнологии, устойчивость, экология, интернет вещей, квантовые вычисления, искусственный интеллект., cloud computing, digital transformation, flexibility, scalability, speed of implementation, cost reduction, access from anywhere in the world, security, analytics, big data, collaboration, cybersecurity, hybrid cloud solutions, crowd computing, medicine, biotechnology, sustainability, ecology, internet of things, quantum computing, artificial intelligence.	Бессерверная архитектура
8	ВЫСВОБОЖДЕНИЕ ПРОИЗВОДИТЕЛЬНОСТИ: ГЛУБОКОЕ ПОГРУЖЕНИЕ В ПРИЛОЖЕНИЯ С ВЫСОКОЙ НАГРУЗКОЙ	https://cyberleninka.ru/article/n/vysvobozhdenie-proizvoditelnosti-glubokoe-pogruzhenie-v-prilozheniya-s-vysokoy-nagruzkoy	07.12.2024	параллелизм, архитектура сервера, высокий трафик, оптимизация, обработка на основе событий, дедупликация запросов, пакетная запись, производительность приложений, масштабируемость, отзывчивость, parallelism, server architecture, high-traffic, optimization, event-driven processing, query deduplication, batched writes, application performance, scalability, responsiveness	Бессерверная архитектура
9	ЭФФЕКТИВНОЕ ИСПОЛЬЗОВАНИЕ ОБЛАЧНЫХ ТЕХНОЛОГИЙ В СМЕШАННОМ ОБУЧЕНИИ	https://cyberleninka.ru/article/n/effektivnoe-ispolzovanie-oblachnyh-tehnologiy-v-smeshannom-obuchenii	07.12.2024	облачные технологии, облачные вычисления, смешанное обучение	Бессерверная архитектура
10	СТРАТЕГИИ ОПТИМИЗАЦИИ ДЛЯ ВЫСОКОНАГРУЖЕННЫХ ПРИЛОЖЕНИЙ: ПОВЫШЕНИЕ ОБЩЕЙ ПРОИЗВОДИТЕЛЬНОСТИ	https://cyberleninka.ru/article/n/strategii-optimizatsii-dlya-vysokonagruzhennyh-prilozheniy-povyshenie-obschey-proizvoditelnosti	07.12.2024	высоконагруженные приложения, оптимизация производительности, оптимизация кода, оптимизация базы данных, балансировка нагрузки, оптимизация аппаратного уровня, эффективность алгоритмов, структуры данных, индексация, кэширование, оптимизация запросов, сетевой трафик, использование ресурсов, конфигурация сервера	Бессерверная архитектура
11	СТРАТЕГИЯ РАЗВЕРТЫВАНИЯ МИКРОСЕРВИСОВ В ОБЛАКЕ	https://cyberleninka.ru/article/n/strategiya-razvertyvaniya-mikroservisov-v-oblake	07.12.2024	микросервисная архитектура, развертывание, облачное развертывание, стратегия развертывания., microservice architecture, deployment, cloud deployment, deployment strategy.	Бессерверная архитектура

Рисунок 1. Рубрика Бессерверная архитектура

Полученный библиографический список по рубрике «Оркестрация контейнеров» представлен на рисунке 2.

Оркестрация контейнер...

	Название статьи	Ссылка на статью	Дата обращения к ресурсу	Ключевые слова	Рубрика
2	ТЕХНОЛОГИИ ИЗОЛЯЦИИ ПРИЛОЖЕНИЙ И ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА ДЛЯ УПРАВЛЕНИЯ КОНТЕЙНЕРАМИ	https://cyberleninka.ru/article/n/tehnologii-izolyatsii-prilozheniy-i-instrumentalnye-sredstva-dlya-upravleniya-konteynerami	07.12.2024	контейнеризация, виртуализация, докер, кубернетес, мезосфер марафон	Оркестрация контейнеров
3	УПРАВЛЕНИЕ КОНТЕЙНЕРАМИ ПРИ ПОСТРОЕНИИ РАСПРЕДЕЛЕННЫХ СИСТЕМ С МИКРОСЕРВИСНОЙ АРХИТЕКТУРОЙ	https://cyberleninka.ru/article/n/upravlenie-konteynerami-pri-postroenii-raspredelennyh-sistem-s-mikroservisnoy-arhitekturoy	07.12.2024	микросервис, контейнер, микросервисная архитектура, облачные вычисления, программное обеспечение	Оркестрация контейнеров
4	ИССЛЕДОВАНИЕ МЕТОДОВ ПОСТРОЕНИЯ ОБЛАЧНЫХ ПЛАТФОРМЕННЫХ СЕРВИСОВ И РЕАЛИЗАЦИЙ СТАНДАРТА TOSCA	https://cyberleninka.ru/article/n/issledovanie-metodov-postroeniya-oblachnyh-platformennyh-servisov-i-realizatsiy-standarta-tosca	07.12.2024	оркестрация, развертывание по	Оркестрация контейнеров
5	ИСПОЛЬЗОВАНИЕ ТЕХНОЛОГИИ КОНТЕЙНЕРИЗАЦИИ КАК КОМПОНЕНТА ОБЕСПЕЧЕНИЯ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ	https://cyberleninka.ru/article/n/ispolzovanie-tehnologii-konteynerizatsii-kak-komponenta-obespecheniya-informatsionnoy-bezopasnosti	07.12.2024	информационная безопасность, контейнер, оркестратор, information security, containerization, orchestrator	Оркестрация контейнеров
6	ПОСТРОЕНИЕ ТРЕБОВАНИЙ И АРХИТЕКТУРЫ ОБЛАЧНОГО ОРКЕСТРАТОРА ПЛАТФОРМЕННЫХ СЕРВИСОВ	https://cyberleninka.ru/article/n/postroenie-trebovaniy-i-arhitektury-oblachnogo-orkestratora-platformennyh-servisov	07.12.2024	облачные вычисления, оркестрация, распределенные системы	Оркестрация контейнеров
7	Использование методов машинного обучения для предсказания уровня загрузки ресурсов системы	https://cyberleninka.ru/article/n/ispolzovanie-metodov-mashinnogo-obucheniya-dlya-predskazaniya-urovnya-zagruzki-resursov-sistemy	07.12.2024	машинное обучение, микросервисы, виртуализация, высокопроизводительные вычисления, machine learning, microservices, virtualization, high performance computing.	Оркестрация контейнеров
8	ОБЗОР ТЕХНОЛОГИЙ ОРГАНИЗАЦИИ ТУМАННЫХ ВЫЧИСЛЕНИЙ	https://cyberleninka.ru/article/n/obzor-tehnologiy-organizatsii-tumannyh-vychisleniy	07.12.2024	облачные вычисления, туманные вычисления, краевые вычисления, интернет вещей, cloud computing, fog computing, edge computing, internet of things	Оркестрация контейнеров
9	Кэширование данных в мультиконтейнерных системах	https://cyberleninka.ru/article/n/keshirovanie-dannyh-v-multikonteynernyh-sistemah	07.12.2024	контейнеры, кеширование, облачные вычисления, containers, caching, cloud computing	Оркестрация контейнеров
10	АВТОМАТИЗИРОВАННЫЙ ИНСТРУМЕНТАРИЙ РАЗВЕРТЫВАНИЯ ОБЛАЧНЫХ СЕРВИСОВ	https://cyberleninka.ru/article/n/avtomatizirovannyy-instrumentariy-razvertyvaniya-oblachnyh-servisov	07.12.2024	облачные сервисы, развертывание сервисов, Kubernetes, Docker, Docker Swarm, Mesos, Rancher, cloud services, service deployment, kubernetes, Docker, Docker Swarm, Mesos, Rancher	Оркестрация контейнеров
11	ОПТИМИЗАЦИЯ ВЫСОКОНАГРУЖЕННЫХ ВЕБ-ПРОЕКТОВ С ИСПОЛЬЗОВАНИЕМ МИКРОСЕРВИСНОЙ АРХИТЕКТУРЫ	https://cyberleninka.ru/article/n/optimizatsiya-vysokonagruzhennyh-veb-proektov-s-ispolzovaniem-mikroservisnoy-arhitektury	07.12.2024	микросервисная архитектура, оптимизация, высоконагруженные системы, масштабируемость, отказоустойчи-вость, Docker, Kubernetes, microservice architecture, optimization, high-load systems, scalability, fault tolerance, Docker, Kubernetes	Оркестрация контейнеров

Рисунок 2. Рубрика Оркестрация контейнеров

1.2 Сбор естественно языковых описаний по медицинской тематике

В рамках выполнения задания по сбору естественно языковых описаний по медицинской тематике использовались материалы, указанные в таблице распределения вариантов работы на вкладке «Темы».

1.2.1 Получение базовых статей

Для поиска статей использовался Telegram-бот @SciArticleBot. Запросы по DOI позволили быстро найти и скачать указанные публикации. В случае затруднений с использованием бота, поиск выполнялся в браузере через сервисы Google Scholar и ResearchGate.

1.2.2 Сбор статей, цитируемых в базовых публикациях

Для каждой базовой статьи был проведен анализ списка литературы. Из ссылок, приведенных в библиографическом списке, было выбрано по 10 статей.

Статьи были получены при последовательном выполнении следующих операций:

– Просмотр полного списка источников базовой статьи.

– Проверка доступности статей по названиям через Telegram-бот или браузер.

– При недостатке ссылок на первом уровне (менее 10 подходящих), выполнен переход на второй уровень ссылок (источники, цитируемые в найденных статьях).

1.2.3 Разделение статей на категории

Собранные статьи были разделены на две подкатегории, соответствующие подтематикам:

– Acute Respiratory Viral Infection (ARVI);

– Obsessive-Compulsive Disorder (OCD).

Для каждой категории было собрано по 10 релевантных статей.

Статьи для категории Acute Respiratory Viral Infection:

– Respiratory Viral Infection-Induced Microbiome Alterations and Secondary Bacterial Pneumonia

– Evaluating the Value of Defensins for Diagnosing Secondary Bacterial Infections in Influenza-Infected Patients

– Secondary Bacterial Infections in Patients With Viral Pneumonia

- Respiratory Syncytial Virus's Non-structural Proteins: Masters of Interference
- A cooperativity between virus and bacteria during respiratory infections
- Metagenomic next-generation sequencing of bronchoalveolar lavage fluid from children with severe pneumonia in pediatric intensive care unit
- Innate Immune Cell Suppression and the Link With Secondary Lung Bacterial Pneumonia
- Peptidoglycan from Immunobiotic Lactobacillus rhamnosus Improves Resistance of Infant Mice to Respiratory Syncytial Viral Infection and Secondary Pneumococcal Pneumonia
- Enhanced mucosal antibody production and protection against respiratory infections following an orally administered bacterial extract
- Respiratory Dysbiosis in Canine Bacterial Pneumonia: Standard Culture vs. Microbiome Sequencing
- Streptococcus pneumoniae and Influenza A Virus Co-Infection Induces Altered Polyubiquitination in A549 Cells

Статьи для категории Obsessive-Compulsive Disorder:

- Pharmacological Treatment of Obsessive-Compulsive Disorder
- Protocol for a Pragmatic Trial of Pharmacotherapy Options Following Unsatisfactory Initial Treatment in OCD
- The Effectiveness of Selective Serotonin Reuptake Inhibitors for Treatment of Obsessive-Compulsive Disorder in Adolescents and Children: A Systematic Review and Meta-Analysis
- Mindfulness-Based Cognitive Therapy for Unmedicated Obsessive-Compulsive Disorder: A Randomized Controlled Trial With 6-Month Follow-Up
- Three-Week Inpatient Treatment of Obsessive-Compulsive Disorder: A 6-Month Follow-Up Study
- The Neuronal Glutamate Transporter EAAT3 in Obsessive-Compulsive Disorder
- The Impact of COVID-19 Pandemic on Individuals With Pre-existing Obsessive-Compulsive Disorder in the State of Qatar: An Exploratory Cross-Sectional Study
- Neurotransmitter system gene variants as biomarkers for the therapeutic efficacy of rTMS and SSRIs in obsessive-compulsive disorder
- A comprehensive review for machine learning on neuroimaging in obsessive-compulsive disorder
- Beneficial Effects of GLP-1 Agonist in a Male With Compulsive Food-Related Behavior Associated With Autism

Predictors of Intensive Treatment in Patients With Obsessive-Compulsive Disorder

1.2.4 Скачивание статей для создания датасета

Все собранные статьи были скачаны в формате PDF для дальнейшего анализа и хранения.

1.2.5 Выделение ключевых слов

На основе текста статей авторские ключевые слова были выделены для каждого материала. Например:

- ARVI: viral infection, bacterial infection, diagnosis, defensin, gene expression.
- OCD: obsessive-compulsive disorder, treatment-naïve, pharmacotherapy, alternatives, remission.

1.2.6 Формирование библиографического списка

Для каждой категории статей составлены отдельные таблицы, содержащие:

- Название статьи
- Ссылка на статью

– Дата обращения к ресурсу
– Ключевые слова
– Рубрика

Полученный библиографический список по рубрике «Acute Respiratory Viral Infection» представлен на рисунке 3.

Acute Respiratory Viral Inf...

Название статьи	Ссылка на статью	Дата обращения к ресурсу	Ключевые слова	Рубрика
Respiratory Viral Infection-Induced Microbiome Alterations and Secondary Bacterial Pneumonia	https://doi.org/10.3389/fimmu.2018.02640	30.11.2024	gut microbiome, respiratory viral infection, bacterial pneumonia, viral-bacterial interaction, influenza, host-microbe interaction	Acute Respiratory Viral Infection
Evaluating the Value of Defensins for Diagnosing Secondary Bacterial Infections in Influenza-Infected Patients	https://www.frontiersin.org/journals/microbiology/articles/10.3389/fmicb.2018.02762/full	30.11.2024	viral infection, bacterial infection, diagnosis, defensin, gene expression	Acute Respiratory Viral Infection
Secondary Bacterial Infections in Patients With Viral Pneumonia	https://www.frontiersin.org/journals/medicine/articles/10.3389/fmed.2020.00420/full	30.11.2024	secondary bacterial infection, pulmonary viruses, viral pneumonia, SARS-CoV-2, COVID-19, influenza, SARS, antibiotic resistance	Acute Respiratory Viral Infection
Respiratory Syncytial Virus's Non-structural Proteins: Masters of Interference	https://www.frontiersin.org/journals/cellular-and-infection-microbiology/articles/10.3389/fcimb.2020.00225/full	30.11.2024	nonstructural (NS) proteins, respiratory syncitial virus, antiviral pathway, innate immunity, mitochondria	Acute Respiratory Viral Infection
A cooperativity between virus and bacteria during respiratory infections	https://www.frontiersin.org/journals/microbiology/articles/10.3389/fmicb.2023.1279159/full	30.11.2024	viral infection, respiratory diseases, COVID-19, bacterial contamination, cooperative relation	Acute Respiratory Viral Infection
Metagenomic next-generation sequencing of bronchoalveolar lavage fluid from children with severe pneumonia in pediatric intensive care unit	https://www.frontiersin.org/journals/cellular-and-infection-microbiology/articles/10.3389/fcimb.2023.1082925/full	30.11.2024	metagenomic next-generation sequencing, severe pneumonia, bronchoalveolar lavage fluid, Epstein–Barr virus, Pneumocystis jirovecii	Acute Respiratory Viral Infection
Innate Immune Cell Suppression and the Link With Secondary Lung Bacterial Pneumonia	https://www.frontiersin.org/journals/immunology/articles/10.3389/fimmu.2018.02943/full	30.11.2024	lung, macrophage, innate immunity, bacteria, virus, matrix, apoptotic cells, training	Acute Respiratory Viral Infection
Peptidoglycan from Immunobiotic Lactobacillus rhamnosus Improves Resistance of Infant Mice to Respiratory Syncytial Viral Infection and Secondary Pneumococcal Pneumonia	https://www.frontiersin.org/journals/immunology/articles/10.3389/fimmu.2017.00948/full	30.11.2024	immunobiotics, peptidoglycan, toll-like receptor 3, viral immunity, Streptococcus pneumoniae, respiratory syncytial virus	Acute Respiratory Viral Infection
Enhanced mucosal antibody production and protection against respiratory infections following an orally administered bacterial extract	https://www.frontiersin.org/journals/medicine/articles/10.3389/fmed.2014.00041/full	30.11.2024	influenza, lung, super-infection	Acute Respiratory Viral Infection
Respiratory Dysbiosis in Canine Bacterial Pneumonia: Standard Culture vs. Microbiome Sequencing	https://www.frontiersin.org/journals/veterinary-science/articles/10.3389/fvets.2019.00354/full	30.11.2024	microbiota, pneumonia, respiratory, dog, bronchoalveolar lavage, culture, sequencing, dysbiosis	Acute Respiratory Viral Infection
Streptococcus pneumoniae and Influenza A Virus Co-Infection Induces Altered Polyubiquitination in A549 Cells	https://www.frontiersin.org/journals/cellular-and-infection-microbiology/articles/10.3389/fcimb.2022.817532/full	30.11.2024	co-infection, ubiquitin, influenza A virus, Streptococcus pneumoniae D39, A549	Acute Respiratory Viral Infection

Рисунок 3. Рубрика Acute Respiratory Viral Infection

Полученный библиографический список по рубрике «Obsessive-Compulsive Disorder» представлен на рисунке 4.

Onychomycosis

Название статьи	Ссылка на статью	Дата обращения к ресурсу	Ключевые слова	Рубрика
Pharmacological Treatment of Obsessive-Compulsive Disorder	https://www.sciencedirect.com/science/article/abs/pii/S0193953X14000549?via%3Dihub	30.11.2024	Obsessive-compulsive, disorder, OCD, Pharmacotherapy, SSRI, Antidepressant	Obsessive-Compulsive Disorder (OCD)
Protocol for a Pragmatic Trial of Pharmacotherapy Options Following Unsatisfactory Initial Treatment in OCD	https://www.frontiersin.org/journals/psychiatry/articles/10.3389/fpsyt.2022.822976/full	30.11.2024	obsessive-compulsive disorder, treatment-naïve, pharmacotherapy, alternatives, remission	Obsessive-Compulsive Disorder (OCD)
The Effectiveness of Selective Serotonin Reuptake Inhibitors for Treatment of Obsessive-Compulsive Disorder in Adolescents and Children: A Systematic Review and Meta-Analysis	https://www.frontiersin.org/journals/psychiatry/articles/10.3389/fpsyt.2019.00523/full	30.11.2024	selective serotonin reuptake inhibitors, obsessive-compulsive disorder, adolescents, children, cognitive behavioral therapy	Obsessive-Compulsive Disorder (OCD)
Mindfulness-Based Cognitive Therapy for Unmedicated Obsessive-Compulsive Disorder: A Randomized Controlled Trial With 6-Month Follow-Up	https://www.frontiersin.org/journals/psychiatry/articles/10.3389/fpsyt.2021.661807/full	30.11.2024	MBCT, mindfulness, obsessive-compulsive disorder, randomized controlled trial, psychotherapy	Obsessive-Compulsive Disorder (OCD)
Three-Week Inpatient Treatment of Obsessive-Compulsive Disorder: A 6-Month Follow-Up Study	https://www.frontiersin.org/journals/psychology/articles/10.3389/fpsyg.2018.00620/full	30.11.2024	obsessive-compulsive disorder (OCD), inpatient treatment, follow-up, effectiveness, exposure with response prevention (ERP), cognitive-behavioral therapy (CBT)	Obsessive-Compulsive Disorder (OCD)
The Neuronal Glutamate Transporter EAAT3 in Obsessive-Compulsive Disorder	https://www.frontiersin.org/journals/pharmacology/articles/10.3389/fphar.2019.01362/full	30.11.2024	EAAT3, OCD, obsessive-compulsive disorder, glutamate transporter, synaptic function, NMDAR, animal model, SLC1A1	Obsessive-Compulsive Disorder (OCD)
The Impact of COVID-19 Pandemic on Individuals With Pre-existing Obsessive-Compulsive Disorder in the State of Qatar: An Exploratory Cross-Sectional Study	https://www.frontiersin.org/journals/psychiatry/articles/10.3389/fpsyt.2022.833394/full	30.11.2024	Qatar, Obsessive-Compulsive Disorder (OCD), COVID-19 pandemic, Yale-Brown Obsessive-Compulsive Scale (YBOCS), OCD subtypes	Obsessive-Compulsive Disorder (OCD)
Neurotransmitter system gene variants as biomarkers for the therapeutic efficacy of rTMS and SSRIs in obsessive-compulsive disorder	https://www.frontiersin.org/journals/psychiatry/articles/10.3389/fpsyt.2024.1350978/full	30.11.2024	OCD, COMT, SLC6A4, GRIN2B, repetitive transcranial magnetic stimulation	Obsessive-Compulsive Disorder (OCD)
A comprehensive review for machine learning on neuroimaging in obsessive-compulsive disorder	https://www.frontiersin.org/journals/human-neuroscience/articles/10.3389/fnhum.2023.1280512/full	30.11.2024	neuroimaging, MRI, machine learning, obsessive-compulsive disorder, AI	Obsessive-Compulsive Disorder (OCD)
Beneficial Effects of GLP-1 Agonist in a Male With Compulsive Food-Related Behavior Associated With Autism	https://www.frontiersin.org/journals/psychiatry/articles/10.3389/fpsyt.2019.00097/full	30.11.2024	obsessive-compulsive disorder, autism, glucagon-like peptide-1 agonist, intellectual disability, weight, overeating	Obsessive-Compulsive Disorder (OCD)
Predictors of Intensive Treatment in Patients With Obsessive-Compulsive Disorder	https://www.frontiersin.org/journals/psychiatry/articles/10.3389/fpsyt.2021.659401/full	30.11.2024	obsessive-compulsive disorder, OCD, intensive treatment, longitudinal, quality of life, psychotropic medication, comorbid depression	Obsessive-Compulsive Disorder (OCD)

Рисунок 4. Рубрика Obsessive-Compulsive Disorder (OCD)

2 СОСТАВЛЕНИЕ ДАТАСЕТА ДЛЯ ДООБУЧЕНИЯ ВЕКТОРНОЙ МОДЕЛИ

В рамках выполнения курсовой работы была поставлена задача преобразования статей из формата PDF в текстовый, очистки текстов от ненужных данных и формирования датасетов для дальнейшего анализа. Задание выполнялось по следующему плану:

Для организации работы были созданы следующие папки:

- resources/clean — для исходных PDF-файлов.
- resources/clean_result — для хранения очищенных текстовых файлов.
- resources/clean_result / merged— для итоговых датасетов.

PDF-файлы, относящиеся к медицинской тематике и теме ВКР, были предварительно собраны и размещены в папке resources/clean.

2.1 Разработка и настройка инструмента для обработки файлов

Для автоматизации обработки файлов использовался скрипт cleaner.py, написанный на Python. Скрипт выполнял следующие задачи:

- Извлечение текста из PDF-файлов с помощью библиотеки PyMyPDF.
- Удаление списков источников и переносов строк для улучшения качества текстов.
- Сохранение каждого текста в отдельный .txt файл с аналогичным названием.

Настройка путей к папкам была выполнена следующим образом:

- Путь к исходным файлам: resources/clean.
- Путь для сохранения результатов: resources/clean_result.

Алгоритм работы скрипта состоит из нескольких последовательных шагов, направленных на извлечение текста из PDF-документов, их очистку от ненужных данных и сохранение в текстовый формат.

1. Чтение PDF-документов: Скрипт начинается с обработки PDF-файлов, расположенных в указанной папке. Для каждого файла вызывается функция convert_pdf_to_txt, которая открывает PDF с помощью библиотеки fitz (часть библиотеки PyMuPDF[1]). Эта функция извлекает текст с каждой страницы документа и объединяет его в одну строку. Текст с каждой страницы добавляется в переменную text.
2. Очистка текста: После того как текст извлечен из документа, скрипт выполняет очистку данных. В частности, удаляются разделы, начинающиеся с фраз вроде “References”, “Список литературы”, “Библиография” и так далее. Это достигается с помощью регулярных выражений[2], которые ищут эти заголовки и удаляют все, что идет после них (включая сами заголовки). Также удаляются ненужные символы, такие как переносы строк, которые могут появляться в процессе извлечения текста.
3. Запись очищенного текста в файл[3]: После того как текст очищен, он сохраняется в текстовый файл с тем же именем, что и исходный PDF, но с расширением .txt. Результирующий файл сохраняется в указанной папке. Для этого используется функция process_folder, которая перебирает все файлы в папке и вызывает функцию преобразования и очистки для каждого PDF-файла.

Листинг кода скрипта cleaner.py представлен в приложении 1.

Скрипт был запущен в консоли, в результате чего для каждого PDF-файла был создан текстовый файл с очищенным содержимым.

2.2 Очистка текстов

Скрипт автоматически удалял разделы с источниками («References», «Список литературы» и т.д.) и исправлял переносы строк. После выполнения обработки консоль выводила уведомления об успешной обработке каждого файла.

2.3 Разделение текстов на группы

Очищенные тексты были разделены на две категории:

- Медицинская тематика.
- Тема ВКР: бессерверная архитектура и оркестрация контейнеров.

Для этого е текстовым файлам были добавлены префиксы, определяющие принадлежность файла к той или иной подгруппе.

2.4 Формирование датасетов

Для создания датасетов использовался скрипт create_dataset.py, который автоматически объединяет текстовые файлы, предварительно проименованные с использованием тематических префиксов, в единые датасеты.

Алгоритм работы скрипта:

1. Организация файлов:

Все очищенные текстовые файлы из папки resources/clean_result были проименованы с использованием тематических префиксов:

- Для медицинской тематики использовались префиксы arvi и ocd.
- Для темы ВКР использовались префиксы co (Container Orchestration) и sla (Serverless Architecture).

2. Настройка путей:

- Путь к исходной папке с текстовыми файлами (resources/clean_result) был указан в переменной directory_path.
- Итоговые датасеты сохраняются в подпапку resources/clean_result/merged, которая создается автоматически, если она отсутствует.

3. Сопоставление префиксов и файлов:

В скрипте file_mapping определяет, какие файлы объединять в какой итоговый датасет:

```
file_mapping = {
   "medicine.txt": ["arvi", "ocd"],
   "cloud_infra.txt": ["co", "sla"],
}
```

Все файлы, начинающиеся с arvi и ocd, объединяются в файл medicine.txt.

4. Объединение файлов:

Скрипт перебирает все файлы в папке resources/clean_result. Для каждого файла он проверяет, начинается ли его имя с одного из префиксов, указанных в file_mapping. Если совпадение найдено:

- Файл открывается и его содержимое добавляется в соответствующий итоговый файл.
- Между текстами добавляется разделитель в виде пустой строки.

5. Сохранение итогов:

Созданные датасеты записываются в папку resources/clean_result/merged с названиями medicine.txt и cloud_infra.txt.

Листинг скрипта create_dataset.py представлен в приложении 2.

Полученные датасеты были приложены к отчету и будут использоваться для последующего анализа и обучения модели.

3 ОБУЧЕНИЕ МОДЕЛИ WORD2VEC

Для выполнения задания была проведена процедура обучения моделей Word2Vec на основе созданных тематических датасетов. Обучение проводилось с использованием библиотеки Gensim, а сами датасеты находились в папке resources/clean_result/merged. Процесс обучения модели подробно описан ниже.

Этапы выполнения задания:

1. Импорт шаблона

Файл model_create.py, предоставленный в шаблоне курса, был добавлен в проект. В скрипте указаны параметры обучения моделей и процедуры их проверки. Для работы скрипта была выбрана среда разработки PyCharm.

2. Подготовка датасетов

В папку resources/clean_result/merged были добавлены итоговые текстовые файлы датасетов:

- medicine.txt — датасет для медицинской тематики.
- cloud_infra.txt — датасет для темы ВКР.

Пути к этим файлам были добавлены в константы CLOUD_INFRA_DATASET, MEDICINE_DATASET.

3. Процесс обучения моделей

Для каждой темы была обучена своя модель, используя следующие параметры обучения в классе Word2Vec[4]:

- sentences — коллекция предложений из датасета.
- min_count=10 — минимальное количество вхождений слова для включения его в представление.
- window=2 — размер окна для учета соседних слов.
- vector_size=16 — размерность вектора слов.
- alpha=0.03 — начальная скорость обучения.
- negative=15 — количество “отрицательных” примеров для обучения.
- min_alpha=0.0007 — минимальная скорость обучения.
- sample=6e-5 — частота выборки слов.

Обучение проходило в три шага:

- Предобработка текста (удаление спецсимволов, токенизация).
- Построение словарного запаса (build_vocab).
- Тренировка модели (train).

4. Сохранение моделей

После обучения модели сохранялись в папку resources с помощью метода model.save(). Названия файлов для сохранения:

- CLOUD_INFRA_MODEL.model для датасета cloud_infra.txt.
- MEDICINE_MODEL.model для датасета medicine.txt.

5. Проверка качества модели

Для проверки качества модели были выбраны:

- Слова для проверки вхождения в представление:
- Для темы ВКР: слово контейнер.
- Для медицинской темы: слово pneumonia.
- Сравнение близости векторов:
- Для темы ВКР: контейнер, [“приложение”, “сервер”].
- Для медицинской темы: pneumonia, [“disease”, “illness”].

В скрипте использованы функции model.wv.has_index_for,

model.wv.similar_by_vector, model.wv.most_similar_to_given для проверки включения слов в представление и анализа близости векторов.

6. Настройка параметров

В ходе тестирования было установлено, что:

– Слова, не вошедшие в представление, добавлялись при снижении параметра min_count.

– Для устранения избыточного сходства между словами в представлении увеличивался параметр negative.

Результаты выполнения:

– Обученные модели:

– Модель для медицинской тематики: MEDICINE_MODEL.model.

– Модель для темы ВКР: CLOUD_INFRA_MODEL.model.

– Оценка качества показала, что ключевые слова из текстов успешно вошли в представление, а их векторные представления соответствуют тематике.

– Полученные модели были включены в состав проекта и приложены к пояснительной записке в виде файлов.

Листинг скрипта model_create.py представлен в приложении 3.

Результат работы скрипта представлен в приложении 4.

Этот процесс обеспечивает корректное создание векторного представления для текста, что является необходимым шагом для дальнейшего использования моделей в системе анализа данных или обработки текста.

4 АВТОМАТИЧЕСКОЕ РУБРИЦИРОВАНИЕ

Для реализации автоматического рубрицирования текста была разработана программа, которая основывается на использовании модели Word2Vec. Программа обеспечивает выполнение алгоритма, описанного в задании, и предоставляет функционал для работы с готовой моделью.

В рамках выполнения задания был разработан алгоритм автоматического рубрицирования текстов с использованием моделей Word2Vec (Приложение 5). Для работы использовались две заранее обученные модели, полученные в результате выполнения предыдущих заданий.

Исходные данные представляли собой текстовые файлы, очищенные от лишних символов и форматирования. Очищение выполнялось с использованием регулярных выражений, удаляющих все небуквенные символы, за исключением пробелов. Такой подход обеспечил корректное разбиение текста на слова и исключение нежелательных элементов, таких как знаки препинания или специальные символы.

Для каждого файла вычислялось его числовое представление, которое характеризует текст с точки зрения семантической близости слов.

Для каждого слова из текста выполняются следующие действия:

1. Проверяется, содержится ли слово в модели Word2Vec.
2. Если слово присутствует в модели, извлекается его векторное представление.
3. Если слово отсутствует, ему присваивается векторное значение, равное нулю.
4. Все извлечённые векторы суммируются, и одновременно фиксируется количество слов, которые присутствуют в модели.

Это значение представляло текст в виде одного числового вектора, который учитывал семантическую информацию о словах, содержащихся в тексте.

Полученные числовые представления всех текстов сортировались в порядке возрастания для дальнейшего распределения на категории. Сортировка позволяла определить логические группы текстов на основе их векторных характеристик.

В каждой категории вычислялось среднее арифметическое числовых значений текстов, что позволяло представить категорию в виде одного усреднённого вектора. Этот вектор служил основой для извлечения ключевых слов, описывающих категорию.

Подбор ключевых слов выполнялся путём поиска ближайших слов к усреднённому вектору категории в пространстве Word2Vec. Ближайшие слова определялись на основании их косинусной близости с вектором категории.

Результаты работы алгоритма представлены в приложении 6.

Итоговый алгоритм обеспечил автоматическое распределение текстов на категории с учётом их семантического содержания, а также выделение ключевых слов, которые могли быть использованы для описания категорий.

5 ТЕХНОЛОГИЯ ПРОВЕДЕНИЯ ИССЛЕДОВАНИЯ

Проект включает обработку статей по двум темам: медицина и облачная инфраструктура. Для этого были использованы различные инструменты и подходы в области обработки текста и машинного обучения.

5.1 Программное обеспечение

Для реализации проекта использовались:

- PyCharm для разработки и отладки скриптов на языке Python.
- Python 3.10 — основной язык программирования.
- Gensim[5] для создания моделей Word2Vec.
- re для работы с регулярными выражениями.
- PyMyPDF (fitz) для извлечения текста из PDF.

5.2 Исходные данные

Для создания датасетов были использованы научные статьи, скачанные через телеграм-бот @SciArticleBot и поиск по DOI. Статьи были сохранены в формате PDF и использованы для дальнейшей обработки.

5.3 Этапы обработки данных

1. Извлечение текста из PDF:

Для преобразования PDF-документов в текст был написан скрипт cleaner.py, который использует библиотеку PyMuPDF для извлечения текста и удаления ненужных элементов, таких как списки литературы и специальные символы. Этот скрипт сохраняет очищенные тексты в текстовых файлах (.txt).

2. Создание тематических датасетов:

Очищенные текстовые файлы были переименованы с добавлением префиксов, отражающих тематику статей (например, arvi_1.txt для медицинской темы). Для объединения файлов по каждой теме использовался скрипт create_dataset.py, который собирал статьи, относящиеся к одной теме, в один файл, например, medicine.txt для медицины и cloud_infra.txt для облачной инфраструктуры.

3. Обучение моделей:

Модели для каждой темы были обучены с использованием библиотеки Gensim и алгоритма Word2Vec. В скрипте model_create.py был реализован процесс предобработки текста, обучение модели и проверка качества модели с помощью проверки векторных представлений слов. Для обучения модели использовались параметры, такие как vector_size=16, min_count=10, window=2, а также проверка близости слов, таких как контейнер и приложение для облачной инфраструктуры или pneumonia и disease для медицины.

4. Проверка качества моделей:

Для оценки качества работы моделей были проверены следующие параметры:

- Наличие ключевых слов в словарном запасе модели.
- Близость векторных представлений выбранных слов (например, для слова контейнер проверялась близость к слову приложение).

5.4 Результаты

После выполнения всех этапов были получены:

- Два тематических датасета: один для медицины, другой для облачной инфраструктуры.
- Обученные модели Word2Vec для каждой темы, которые успешно включили ключевые слова в свои векторные представления.

Таким образом, в рамках курсового проекта был использован комплекс методов обработки текста, включая извлечение текста, очистку данных, создание датасетов и обучение моделей. Полученные результаты могут быть использованы для дальнейших исследований и практических применений в области обработки текста и анализа данных.

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсового проекта была проведена серия исследований, направленных на создание и обработку тематических датасетов, а также на обучение моделей для анализа текстов научных статей. Работа включала следующие ключевые этапы:

1. Сбор и очистка данных: Были собраны научные статьи по двум темам — медицина и облачная инфраструктура. Для обработки текстов использовались специальные скрипты, которые позволили извлечь содержимое из PDF-документов и очистить его от ненужных элементов, таких как списки литературы и специальные символы.
2. Создание тематических датасетов: Очищенные тексты были аккуратно организованы и распределены по тематическим категориям с использованием префиксов, что позволило создать два отдельных датасета для дальнейшего анализа.
3. Обучение моделей: На основе подготовленных датасетов были обучены модели Word2Vec, что позволило создать векторные представления для ключевых терминов в каждой теме. Модели успешно включили в себя важные термины и продемонстрировали способность находить близкие по смыслу слова.
4. Оценка качества моделей: Качество работы моделей было проверено через анализ векторных представлений и проверку близости ключевых слов. Результаты показали, что выбранные параметры обучения модели удовлетворяют поставленным целям.

Таким образом, в ходе исследования были успешно решены поставленные задачи: получены два тематических датасета и обучены модели для анализа текстов. Эти результаты могут быть использованы в дальнейшем для разработки более сложных моделей, анализа больших объемов научных данных и в других областях, связанных с обработкой естественного языка и машинным обучением.

Проект продемонстрировал эффективность применения методов машинного обучения и обработки текста для извлечения значимой информации из научных публикаций, что открывает возможности для дальнейших исследований и применения полученных моделей в реальных проектах.

ЛИТЕРАТУРА

1. Работа с PDF-файлами в Python // Мартын-Назаров Кирилл Надеюсь не отчислят URL: https://is42-2018.susu.ru/nazarovka/2021/05/24/rabota-s-pdf-fajlami-v-python-chast-1. (дата обращения: 14.12.2024).
2. Регулярные выражения в Python от простого к сложному. Подробности, примеры, картинки, упражнения // Хабр URL: https://habr.com/ru/articles/349860/ (дата обращения: 14.12.2024).
3. Файлы. Работа с файлами. // Python World URL: https://pythonworld.ru/tipy-dannyx-v-python/fajly-rabota-s-fajlami.html (дата обращения: 14.12.2024).
4. Обучаем Word2vec: практикум по созданию векторных моделей языка // Системный Блокъ URL: https://sysblok.ru/knowhow/obuchaem-word2vec-praktikum-po-sozdaniju-vektornyh-modelej-jazyka/ (дата обращения: 14.12.2024).
5. Gensim Docuentation // Gensim URL: https://radimrehurek.com/gensim/auto_examples/index.html (дата обращения: 14.12.2024).

ПРИЛОЖЕНИЯ

Приложение 1.

```
import os
import re
import fitz


def convert_pdf_to_txt(pdf_path):
    # Открываем PDF и извлекаем текст
    doc = fitz.open(pdf_path)
    text = ""
    for page_num in range(doc.page_count):
        page = doc[page_num]
        text += page.get_text("text")
    doc.close()

    # Удаляем раздел с источниками, если он есть
    # Предполагаем, что список начинается с "References" или "Список
литературы"
    text = re.sub(r"(References|Список
литературы|Библиография|REFERENCES)(.*)", "", text, flags=re.DOTALL)
    text = re.sub(r"-\n", "", text, flags=re.DOTALL)
    return text


def process_folder(folder_path):
    for filename in os.listdir(folder_path):
        if filename.endswith(".pdf"):
            pdf_path = os.path.join(folder_path, filename)
            text = convert_pdf_to_txt(pdf_path)

            # Сохраняем текст в .txt файл
            txt_filename = filename.replace(".pdf", ".txt")
            txt_path = os.path.join("resources/clean_result", txt_filename) #
Укажите куда сохранять результаты работы
            with open(txt_path, "w", encoding="utf-8") as txt_file:
                txt_file.write(text)
            print(f"Файл {txt_filename} успешно создан.")

# Пример использования
folder_path = "resources/clean"  # Укажите путь к папке с PDF
process_folder(folder_path)
```

Приложение 2.

```
import os
from pathlib import Path


def merge_files_with_prefixes(directory: str, file_map: dict):
    output_dir = Path(directory) / "merged"
    output_dir.mkdir(parents=True, exist_ok=True)

    for dest_file, prefixes in file_map.items():
        dest_path = output_dir / dest_file

        with open(dest_path, "w", encoding="utf-8") as out_file:
            for file in Path(directory).iterdir():
                if file.is_file() and any(file.name.startswith(prefix) for
prefix in prefixes):
                    with open(file, "r", encoding="utf-8") as in_file:
                        out_file.write(in_file.read())
                        out_file.write("\n")


if __name__ == "__main__":
    directory_path = "./resources/clean_result"
    file_mapping = {
        "medicine.txt": ["arvi", "ocd"],
        "cloud_infra.txt": ["co", "sla"],
    }

    merge_files_with_prefixes(directory_path, file_mapping)
```

Приложение 3.

```
import gensim
from gensim.models import Word2Vec
import pandas as pd
import re

patterns = "[!#$%&'()*+,./:;<=>?@[\]^_`{|}~—\"\-]+"
CLOUD_INFRA_DATASET = ("./resources/clean_result/merged/cloud_infra.txt",
"CLOUD_INFRA_MODEL")
MEDICINE_DATASET = ("./resources/clean_result/merged/medicine.txt",
"MEDICINE_MODEL")

def train_model(dataset:  tuple[str, str]) -> Word2Vec:
    response = []
    with open(dataset[0], encoding='utf-8') as f:
        lines = f.readlines()
        for line in lines[1:]:
            temp = line.split('\t')
            response.append(re.sub(patterns, ' ', temp[0]))

    data = pd.DataFrame(list(zip(response)))
    data.columns = ['response']
    response_base = data.response.apply(gensim.utils.simple_preprocess)
    model = Word2Vec(
        sentences=response_base, min_count=10,
        window=2, vector_size=16, alpha=0.03,
        negative=15, min_alpha=0.0007,
        sample=6e-5
    )

    model.build_vocab(response_base, update=True)
    model.train(response_base, total_examples=model.corpus_count,
epochs=model.epochs)
    model.save(f"resources/{dataset[1]}.model")

    return model

def print_model_stats_by_word(
        model_name: str,
        model: Word2Vec,
        word: str,
        most_similar: list[str]
):
    print(f"""
    Model: {model_name}
    model.corpus_count: {model.corpus_count},
    model.wv.has_index_for {word}: {model.wv.has_index_for(word)},
    model.wv.similar_by_vector {word}:
{model.wv.similar_by_vector(model.wv[word])}
    model.wv.most_similar_to_given {word} {most_similar}:
{model.wv.most_similar_to_given(word, most_similar)}
    """)

if __name__ == "__main__":
    cloud_infra_model = train_model(CLOUD_INFRA_DATASET)
    print_model_stats_by_word('cloud_infra_model', cloud_infra_model,
'контейнер', ['приложение', 'сервер'])

    medicine_model = train_model(MEDICINE_DATASET)
    print_model_stats_by_word('medicine_model', medicine_model, 'pneumonia',
['disease', 'illness'])
```

Приложение 4.

```
    Model: cloud_infra_model
    model.corpus_count: 11190,
    model.wv.has_index_for контейнер: True,
    model.wv.similar_by_vector контейнер: [('контейнер', 1.0), ('данная',
0.9975045919418335), ('кроме', 0.9973993897438049), ('части',
0.997326672077179), ('load', 0.9973011016845703), ('стоит',
0.9972853660583496), ('приложения', 0.9972342252731323), ('создание',
0.9972232580184937), ('облачной', 0.9970948696136475), ('такие',
0.9969913959503174)]
    model.wv.most_similar_to_given контейнер ['приложение', 'сервер']:
приложение


    Model: medicine_model
    model.corpus_count: 19438,
    model.wv.has_index_for pneumonia: True,
    model.wv.similar_by_vector pneumonia: [('pneumonia', 1.0), ('mood',
0.9976933598518372), ('via', 0.9972329139709473), ('acinetobacter',
0.9971780776977539), ('cap', 0.9971160292625427), ('responsible',
0.9968923926353455), ('estimate', 0.996885359287262), ('gut',
0.996840238571167), ('functions', 0.9966867566108704), ('discharge',
0.9966611266136169)]
    model.wv.most_similar_to_given pneumonia ['disease', 'illness']: illness
```

Приложение 5.

```
import re
import numpy as np
from gensim.models import Word2Vec

MODELS = [
    "resources/CLOUD_INFRA_MODEL.model",
    "resources/MEDICINE_MODEL.model"
]

SAMPLE_TEXT_DIRECTORY = "resources/texts"

TEXTS = [
    "resources/clean_result/arvi_fcimb-10-00225.txt",
...
hdenie-proizvoditelnosti-glubokoe-pogruzhenie-v-prilozheniya-s-vysokoy-
nagruzkoy.txt",
    ]

PATTERN = r'[^\w\s]'

def get_words_from_file(file_path):
    word_set = set()
    pattern = r'[^\w\s]'

    with open(file_path, 'r', encoding='utf-8') as file:
        for line in file:
            cleaned_line = re.sub(pattern, '', line)
            words = cleaned_line.lower().split()
            word_set.update(words)

    return set(filter(lambda word: len(word) > 3, word_set))

def count_word_vector(model, file_name) -> float:
    result = 0
    words = 0

    text_words = get_words_from_file(file_name)
    for word in text_words:
        if model.wv.has_index_for(word):
            result += model.wv.get_vector(word).sum()
            words += 1

    return result / words

def split_dict_into_chunks(input_dict, chunks = 5):
    items = list(input_dict.items())
    submaps = []
    n = len(items)
    chunk_size = n // chunks
    remainder = n % chunks

    start = 0
    for i in range(chunks):
        size = chunk_size + (1 if i < remainder else 0)
        submaps.append(dict(items[start:start + size]))
        start += size

    return submaps

def calculate_arithmetic_mean(input_dict):
    return sum(input_dict.values()) / len(input_dict)

def process_files_with_model(model_file):
```

```
    print(model_file)
    model = Word2Vec.load(model_file)
    files_avg_vectors = { file_name:count_word_vector(model, file_name) for
file_name in TEXTS }
    sorted_files_avg_vectors = dict(sorted(files_avg_vectors.items(),
key=lambda item: item[1]))
    chunked_files_avg_vectors =
split_dict_into_chunks(sorted_files_avg_vectors)
    i = 0
    for chunk in chunked_files_avg_vectors:

        i += 1
        avg = calculate_arithmetic_mean(chunk)
        sim_words = model.wv.similar_by_vector(vector=np.array(avg), topn=5)

        print(f"""
        Group: {i}
        Average_vector: {avg}
        Keywords: {sim_words}
        """)


for model_name in MODELS:
    process_files_with_model(model_name)
```

Приложение 6.

resources/CLOUD_INFRA_MODEL.model

```
            Group: 1
            Average_vector: 2.1882773921258907
            Keywords:       [('контейнеризации',       0.39451637864112854),
('различных',    0.391132116317749),    ('оркестрацию',    0.388195276260376),
('ошибок', 0.38803353905677795), ('объектов', 0.3852406442165375)]


            Group: 2
            Average_vector: 2.2315099456606826
            Keywords:       [('контейнеризации',       0.39451637864112854),
('различных',    0.391132116317749),    ('оркестрацию',    0.388195276260376),
('ошибок', 0.38803353905677795), ('объектов', 0.3852406442165375)]


            Group: 3
            Average_vector: 2.256962587341373
            Keywords:       [('контейнеризации',       0.39451637864112854),
('различных',    0.391132116317749),    ('оркестрацию',    0.388195276260376),
('ошибок', 0.38803353905677795), ('объектов', 0.3852406442165375)]


            Group: 4
            Average_vector: 2.2832635925958393
            Keywords:       [('контейнеризации',       0.39451637864112854),
('различных',    0.391132116317749),    ('оркестрацию',    0.388195276260376),
('ошибок', 0.38803353905677795), ('объектов', 0.3852406442165375)]
```

resources/MEDICINE_MODEL.model

```
            Group: 1
            Average_vector: 2.132985140626544
            Keywords:      [('hsa',      0.4060875475406647),      ('based',
0.4016115069389343),      ('co',      0.3912436366081238),      ('community',
0.3847077786922455), ('balance', 0.37818023562431335)]


            Group: 2
            Average_vector: 2.1545271150101097
            Keywords:      [('hsa',      0.4060875475406647),      ('based',
0.4016115069389343),      ('co',      0.3912436366081238),      ('community',
0.3847077786922455), ('balance', 0.37818023562431335)]


            Group: 3
            Average_vector: 2.1627514109855395
            Keywords:      [('hsa',      0.4060875475406647),      ('based',
0.4016115069389343),      ('co',      0.3912436366081238),      ('community',
0.3847077786922455), ('balance', 0.37818023562431335)]


            Group: 4
            Average_vector: 2.1964812589022134
            Keywords:      [('hsa',      0.4060875475406647),      ('based',
0.4016115069389343),      ('co',      0.3912436366081238),      ('community',
0.3847077786922455), ('balance', 0.37818023562431335)]
```