УДК 004

ПРОЕКТИРОВАНИЕ СЕРВИСА ИНЛАЙНОВОГО ПОИСКА НА БАЗЕ TELEGRAM BOT API ДЛЯ АВТОМАТИЗИРОВАННОГО ETL-ПАРСИНГА СЛОЖНОСТРУКТУРИРОВАННЫХ УЧЕБНЫХ ПЛАНОВ ВУЗА

Пикин К.Р.,

студент,

Калужский государственный университет им. К.Э. Циолковского,

Калуга, Россия

Борисова Т.В.,

студент,

Калужский государственный университет им. К.Э. Циолковского,

Калуга, Россия

Емец Е.А.,

студент,

Калужский государственный университет им. К.Э. Циолковского,

Калуга, Россия

Соколов Н.В.,

старший преподаватель кафедры информатики и информационных технологий

Калужский государственный университет им. К.Э. Циолковского,

Калуга, Россия

Аннотация

В статье рассматривается проблема цифровой трансформации доступа к учебно-методической документации в высших учебных заведениях (на примере КГУ им. К.Э. Циолковского). Обоснована неэффективность использования PDF-файлов и статических веб-страниц для оперативной навигации студентов по учебным планам. Предложена и детально описана архитектура программного комплекса на базе Telegram Bot API, реализующего концепцию «данные как сервис» (Data-as-a-Service). Ключевым элементом системы является модуль прямого ETL-

парсинга файлов формата XLSX, использующий гибридный алгоритм на основе библиотек *openpyxl* и *pandas*. Описаны такие методы преодоления структурной неопределенности исходных данных как обработка объединенных ячеек (merged cells), идентификация семантических блоков через стилевое оформление и использование якорных заголовков для динамического определения границ таблиц. Реализация инлайнового поиска позволяет интегрировать доступ к данным непосредственно в коммуникационную среду студентов.
Ключевые слова: цифровая образовательная среда, Telegram Bot API, ETL-процессы, парсинг данных, openpyxl, pandas, инлайновый поиск.

DEVELOPMENT OF AN INTEGRATED SEARCH SERVICE BASED ON THE TELEGRAM BOT API FOR AUTOMATED ETL ANALYSIS OF COMPLEX STRUCTURED UNIVERSITY CURRICULA

Pikin K.R.,
student,
Kaluga State University named after K.E. Tsiolkovsky,
Kaluga, Russia
Borisova T.V.,
student,
Kaluga State University named after K.E. Tsiolkovsky,
Kaluga, Russia
Yemets E.A.,
student,
Kaluga State University named after K.E. Tsiolkovsky,
Kaluga, Russia
Sokolov N.V.,
Senior Lecturer at the Department of Computer Science and Information Technology
Kaluga State University named after K.E. Tsiolkovsky,
Kaluga, Russia

Abstract

The article discusses the problem of digital transformation of access to educational and methodological documentation in higher education institutions (using the example of K.E. Tsiolkovsky KSU). The inefficiency of using PDF files and static web pages for students' rapid navigation through curricula is substantiated. The architecture of a software package based on the Telegram Bot API, which implements the concept of "Data as a service" (Data-as-a-Service), is proposed and described in detail. The key element of the system is the direct ETL parsing module for XLSX files, which uses a hybrid algorithm based on the openpyxl and pandas libraries. Such methods of overcoming the structural uncertainty of the source data as the processing of merged cells, the identification of semantic blocks through stylistic design, and the use of anchor headers to dynamically define table boundaries are described. The implementation of inline search allows you to integrate data access directly into the students' communication environment.

Keywords: digital educational environment, Telegram Bot API, ETL processes, data parsing, openpyxl, pandas, inline search.

Современный этап цифровизации высшего образования характеризуется переходом от простой автоматизации административных процессов к созданию человеко-ориентированных сервисов, которые экономят время и снижают когнитивную нагрузку для обучающихся. Если на ранних этапах цифровизация в вузах чаще сводилась к автоматизации документооборота и административных процедур, то сегодня приоритетом становится встраивание цифровых инструментов в повседневные сценарии студента: быстро найти нужную информацию, не переключаясь между системами и не тратя усилия на интерпретацию «сложных» документов. Независимо от наличия электронных образовательных сред многие ключевые артефакты учебного процесса остаются труднодоступными в прикладном смысле. К таким документам относится учебный план, который задаёт траекторию обучения, фиксирует перечень

дисциплин, распределение трудоёмкости по семестрам (в том числе в зачётных единицах (ЗЕТ)), виды промежуточной аттестации и, соответственно, требования к ресурсам кафедр и расписанию.

Противоречие заключается в том, что учебный план - один из самых часто запрашиваемых документов (студенту регулярно нужно уточнить семестр, форму контроля, объём часов, наличие курсового проекта), но при этом он нередко публикуется в виде набора PDF-файлов или таблиц, ориентированных на печать, а не на поиск. В результате для получения ответа на простой вопрос (например, «В каком семестре экзамен по информатике?») студент вынужден воспроизводить последовательность действий, присущих эпохе стационарных персональных компьютеров (ПК), а именно найти файл на сайте, скачать его, открыть в приложении на смартфоне, пролистать многостраничную таблицу, визуально отсканировать строки и столбцы, сопоставить обозначения и сноски. На практике это превращается в так называемый «порог доступа» (информация формально открыта, но функционально труднодоступна). В образовательных исследованиях такой разрыв между доступностью данных и удобством их использования часто связывают с проблемой пользовательского опыта и информационной архитектуры сервисов [9].

Дополнительным фактором является то, что учебные планы по своей природе создаются как регламентный документ, где приоритетом выступает юридическая и методическая корректность, а не «машиночитаемость». Визуальная структура таблицы (объединённые ячейки, блоки циклов, шапки с многоуровневой иерархией, подписи с сокращениями) делает документ удобным для печати и согласования, но затрудняет автоматическую обработку и поиск. На фоне роста цифровых коммуникаций в студенческой среде это приводит к типичной практике, где студент спрашивает у одногруппников скриншоты нужных фрагментов, вместо того чтобы получать ответ из первоисточника. Такая практика увеличивает риск устаревшей информации и ошибок, в связи с

тем, что студенты работают не с версией учебного отдела, а пользовательскими копиями.

Таким образом целью данного исследования является проектирование архитектуры программного агента (чат-бота), который автоматически извлекает данные из первичных файлов формата «XLSX», преобразует их в структурированный нормализованный вид и предоставляет пользователю через интерфейс мессенджера Telegram. Под «программным агентом» [11] в данном контексте понимается автономный сервис, выполняющий цепочку действий по запросу пользователя: принять текстовый запрос, интерпретировать его как поисковую задачу, обратиться к заранее подготовленному индексу и вернуть релевантный ответ в удобном формате. Важно, что такой агент не подменяет собой учебный план как нормативный документ, а выполняет роль «интерфейсного слоя» - облегчает доступ к утверждённым данным без изменения источника.

Выбор Telegram как канала взаимодействия обоснован распространённостью мессенджеров в академических сообществах и тем, что пользователь уже постоянно находится в этой среде коммуникации. Здесь принципиально значима технология Inline Mode: она позволяет использовать функциональность бота «внутри любого чата», не переключаясь в отдельный диалог с ботом и не воспроизводя командную модель вида /start и /search. В терминах пользовательского опыта (User Experience - UX) это снижает количество контекстных переключений и уменьшает время до ответа, что особенно критично для микросценариев - быстрых уточнений перед занятием или при составлении индивидуального плана. Официальная документация Telegram Bot API описывает механику инлайн-запросов и формат ответов, что делает канал технологически предсказуемым для интеграции [7]. Одновременно использование мессенджера не отменяет требований к защите данных и корректному обращению с документами вуза; в данном проекте это учитывается

через ограничение функционала областью открытой учебной информации и отказ от хранения персональных данных без необходимости.

Анализ предметной области показал, что основной источник сложности лежит не в «поиске как таковом», а в извлечении и нормализации данных из исходных файлов учебных планов. На массиве планов КГУ им. К.Э. Циолковского выявляются типовые особенности, препятствующие прямому применению стандартных инструментов чтения Excel. Во-первых, таблицы имеют сложную топологию - активно используются объединения ячеек (merge cells) для группировки дисциплин по блокам, циклам и кафедрам. Для человека такая вёрстка интуитивна, а для машины создаёт разрывы данных (значение, записанное в верхней левой ячейке объединённого диапазона, «логически» относится ко всем ячейкам диапазона, но физически хранится в файле как значение только одной ячейки). Во-вторых, планы разных лет и направлений подготовки нередко используют вариативные шаблоны где нет фиксированного набора столбцов, поэтому индексы колонок, где находятся ЗЕТ, часы, семестр или формы контроля, могут смещаться. В-третьих, часть смысла кодируется визуально, то есть не текстом, а форматированием (полужирным начертанием, курсивом, заливкой, рамками), например, принадлежность дисциплины к вариативной части или выделение определённого блока иногда задаётся только стилем. В-четвёртых, в данных встречаются неразрывные пробелы, скрытые переносы строк, непоследовательные сокращения, опечатки в заголовках. Эти дефекты критичны для поиска, потому что поисковый движок опирается на совпадение токенов и нормализованных строк.

В этой ситуации существует несколько подходов к предоставлению учебных данных пользователю. Классический подход заключается в использовании веб-портала или личного кабинета, где учебный план встроен в LMS и доступен как часть официальной цифровой инфраструктуры. Преимущество такого решения заключается в высоком уровне интеграции и официальном статусе LMS. Однако портальные решения часто требуют

авторизации, не оптимизированы под быстрый точечный поиск и в ряде вузов имеют слабую мобильную версию (даже если план доступен, его поиск и просмотр всё равно остаётся проблемой). Альтернативный подход заключается в парсинге PDF (включая OCR и извлечение текста) [14]. Он удобен тем, что работает с финальными утверждёнными документами и формально не зависит от Excel-шаблонов. Но PDF-извлечение плохо сохраняет табличную структуру, затрудняет точное сопоставление ячеек и часто даёт высокий процент ошибок, особенно на многоуровневых таблицах и документах, ориентированных на печать. На практике для учебного плана ключевой риск PDF-подхода заключается в потере связей между полями «дисциплина», «семестр», «вид контроля» и «трудоёмкость», то есть происходит нарушение структуры документа. Поэтому в качестве базового подхода при проектировании нами выбран метод прямого парсинга XLSX, который позволяет работать с первичной структурой, метаданными и стилями файла. Такой подход повышает достоверность извлечения и поддерживает принцип целостности данных (data integrity), когда значения часов и зачётных единиц переносится без округлений и интерпретаций.

Предлагаемая нами система спроектирована как модульный монолит на Python. Под модульным монолитом понимается архитектурный стиль, при котором приложение разворачивается как единый сервис, но внутренне разделено на изолированные модули с чёткими интерфейсами. Это даёт предсказуемость развертывания и снижает операционные затраты на старте (по сравнению с микросервисами), сохраняя возможность эволюции (при росте нагрузки отдельные компоненты можно выделять в самостоятельные сервисы без полной переработки) [10]. В практической реализации система разделяется на три логических слоя. Слой данных отвечает за чтение файлов, доступ к файловой системе и преобразование XLSX в Python-объекты и табличные структуры. Слой бизнес-логики содержит доменную модель, поисковый движок и менеджер состояний (например, для выбора образовательной программы или

группы, если это требуется). Слой представления реализует взаимодействие с Telegram Bot API, обрабатывает инлайн-запросы и формирует сообщения-результаты. Таким образом для нашей системы диаграмма потоков данных (DFD) в соответствие с этими слоями будет выглядеть следующим образом (рисунок 1).

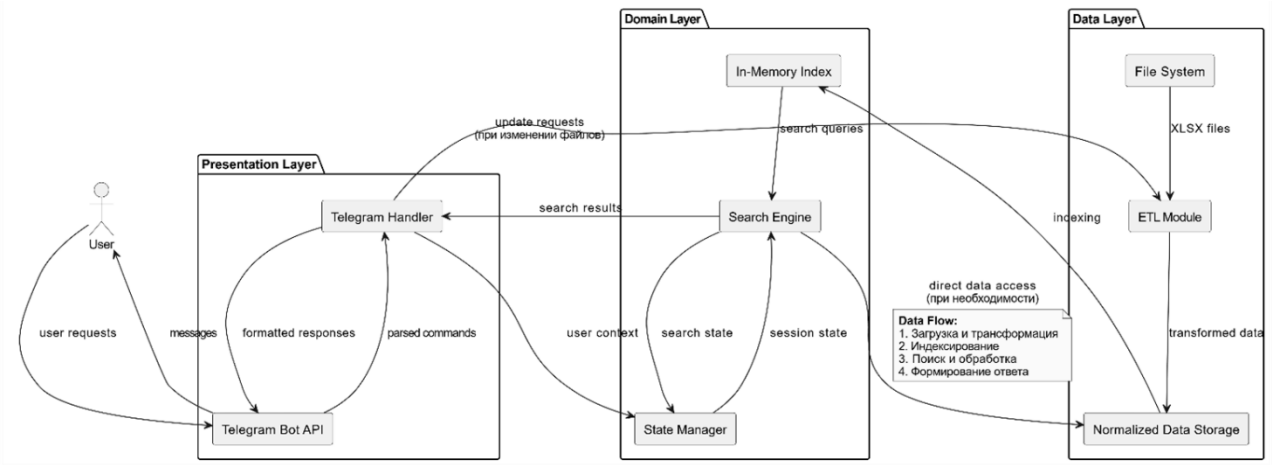

Рисунок 1 Data Flow Diagram с учётом слоёв размещения чат-бота авторская разработка

Технологический стек для разработки выбран с учётом потребности извлекать не только значения ячеек, но и метаданные разметки. В качестве низкоуровневого инструмента используется openpyxl, поскольку библиотека позволяет читать Excel-файлы как структуру рабочих листов, включая объединённые диапазоны и стили [3] Для высокоуровневой табличной обработки применяется pandas, обеспечивающий удобную очистку, трансформации и векторные операции, которые существенно ускоряют обработку по сравнению с построчными циклами [4]. Telegram-интерфейс реализуется на базе python-telegram-bot версии «20.0+» [6] с поддержкой асинхронной модели, что важно для обслуживания множества одновременных инлайн-запросов [1]. На этапе тестовой версии системы для организации хранения данных достаточно использовать in-memory структуры (например, словаря списков или заранее собранных индексных таблиц), потому что объём учебных планов ограничен и редко изменяется. Для промышленной

эксплуатации, аудита изменений и удобства администрирования оправдано использование PostgreSQL как устойчивого хранилища [5], особенно если планируется подключать несколько факультетов, версионность и контроль доступа.

Рассмотрим более подробно алгоритм гибридного парсинга. Ключевым элементом является ETL-модуль. Термин ETL (Extract, Transform, Load) описывает стандартный конвейер обработки данных (извлечение из источника, преобразование в целевой формат и загрузка в хранилище или индекс) [8]. В данном исследовании извлечение предполагает чтение XLSX с учётом метаданных, преобразование включает восстановление иерархии таблицы, очистку и нормализацию, а «загрузка» - формирование поисковых структур. Важное практическое замечание состоит в том, что стандартные методы чтения Excel, такие как pandas.read_excel, не подходят для учебных планов с активным использованием объединений и визуальных маркеров. Они корректно читают значения, но игнорируют значимую часть структуры и оставляют пустыми ячейки внутри объединённых диапазонов (заполненной остаётся только верхняя левая ячейка), что приводит к тому, что, например, название блока дисциплин или кафедры теряется для последующих строк.

Как раз для решения данной проблемы нами и будет использоваться двухэтапный гибридный алгоритм обработки, который сочетает низкоуровневое сканирование метаданных и высокоуровневую векторную обработку табличных данных. На первом этапе выполняется итерация по структуре листа с помощью openpyxl. Сначала строится карта объединённых диапазонов, т.е., фактически создаётся отображение, позволяющее по координатам ячейки определить, входит ли она в merged-range, и, если входит, то найти «якорную» ячейку (top-left), где хранится значение. Это даёт возможность логически заполнить все ячейки диапазона, не меняя исходный файл, а формируя корректный набор значений для дальнейшей обработки. Затем выполняется поиск так называемых «якорей» (markers) в первых строках листа. Вместо жёстко заданных индексов

используется словарь токенов, включающий основные варианты заголовков («дисциплина», «наименование», «зет», «часы», «контроль», «вид отчётности», «экзамен», «зачёт», «кр/кп») и допускающий нечеткое сопоставление. Нечеткое сопоставление (fuzzy matching) - это класс методов, которые позволяют считать строки достаточно похожими, даже если они различаются опечатками, порядком слов или формой написания [2]. На практике для русскоязычных заголовков полезны метрики расстояния (например, Левенштейна [12]), токенизация и нормализация пробелов/символов. Ценность данного подхода заключается в том, что система становится устойчивой к эволюции шаблонов (если в одном плане написано «Зачётные единицы», а в другом «ЗЕТ», модуль всё равно найдёт столбец трудоёмкости).

После того как определены границы таблицы и смысловые столбцы, данные загружаются в pandas DataFrame для второго этапа, а именно, векторной обработки и обогащения. Здесь решается задача восстановления контекста иерархической таблицы. Учебный план часто устроен так, что заголовок блока (например, «Блок 1. Дисциплины (модули)») записан один раз, а ниже идут строки дисциплин, в которых соответствующая колонка пуста. Для человека пустота читается как продолжение того же блока, а для машины - как пропуск, поэтому применяется стратегия распространения значений вперёд, а именно, каждое пустое значение в контекстных колонках заменяется последним встреченным непустым значением сверху. В pandas это типовой приём для восстановления иерархии в денормализованных таблицах, и он хорошо работает при условии, что корректно выделены колонки, несущие «родительский» контекст (например, цикл, кафедра, блок). Далее выполняется очистка данных, при которой удаляются строки без названий дисциплин и нормализуются строки (приведение к нижнему регистру, замена неразрывных пробелов на обычные, удаление лишней пунктуации, унификация дефисов и кавычек). Такая нормализация напрямую влияет на качество поиска (чем меньше различий

между запросом пользователя и строкой в данных, тем выше вероятность релевантного совпадения).

После нормализации формируется индекс для быстрого поиска. Наиболее понятной и производительной структурой для текстового поиска является инвертированный индекс. Инвертированный индекс - это отображение терминов (список документов или строк), где документом выступает, например, запись дисциплины [13]. Когда пользователь вводит запрос, он разбивается на термы (слова или их нормализованные формы), и по каждому терму быстро извлекается список кандидатов, а пересечение и ранжирование этих списков дают результат за миллисекунды даже на сравнительно большом количестве записей. В отличие от полного сканирования таблицы при каждом запросе, индексирование переносит вычислительную нагрузку на этап подготовки данных. Для небольшого вуза объём дисциплин по всем планам обычно достаточно мал, чтобы держать индекс в памяти, но при масштабировании (несколько институтов, много годов набора, несколько языков) целесообразно хранить индекс в базе данных или использовать специализированный поисковый движок. В рамках выбранной нами архитектуры заложена возможность перехода от in-memory индекса к PostgreSQL, не меняя протокол взаимодействия между слоями.

Важным проектным решением выступает организация пользовательского сценария через Inline Mode [1]. В этом режиме пользователь набирает в любом чате конструкцию вида @bot_name и далее текст запроса, Telegram отправляет на сервер инлайн-запрос, включающий строку пользователя и контекст. Сервер выполняет поиск по индексу и возвращает массив результатов, каждый из которых является готовой карточкой, которую можно вставить в чат. Такой формат делает бота не просто справочником, а инструментом совместной работы (студент может мгновенно поделиться найденной информацией с одногруппниками или преподавателем). На стороне сервера реализуется комбинация поиска по подстроке и нечеткого сопоставления. Подстрочный

поиск обеспечивает хорошую точность на простых запросах (когда пользователь вводит часть названия дисциплины), а нечеткое сравнение (fuzzy matching) повышает устойчивость к опечаткам и различиям в написании. Ранжирование результатов можно строить как взвешенную сумму метрик (доля совпавших токенов, расстояние редактирования, наличие точного совпадения по началу слова, совпадение по коду/индексу дисциплины). Возвращаемая карточка обычно включает название дисциплины, семестр или семестры, виды контроля, а также трудоёмкость с разбиением на лекции, практики и самостоятельную работу, если это присутствует в исходном документе.

Отдельного внимания заслуживает модель обновления данных. В вузовской практике важно соблюдать принцип «единого источника истины» (Single Source of Truth) где данные считаются корректными только в том виде, в котором они утверждены и опубликованы ответственным подразделением [15]. Если бот получает отдельную панель редактирования, возникает риск расхождений между версией бота и версией учебного отдела. Поэтому выбран подход, при котором бот не редактирует учебные планы и не хранит ручные исправления, а отслеживает выделенную сетевую директорию, где размещаются исходные XLSX. При появлении нового файла или изменении существующего запускается переиндексация. Для того чтобы не выполнять лишнюю работу, используется проверка хэш-суммы. Хэширование позволяет определить, изменилось ли содержимое файла, даже если имя осталось прежним. В результате студент видит ту же версию плана, что и сотрудники, а жизненный цикл данных остаётся управляемым и прозрачным.

Эмпирическая проверка на массиве учебных планов показала, что предложенная архитектура и гибридный парсинг устойчивы к структурным изменениям входных файлов. Использование гибридного парсинга позволило корректно обработать 98% учебных планов университета без ручной адаптации. Отказ от тяжеловесных решений (OCR) в пользу работы с XML-структурой

файла Excel обеспечил 100% точность извлечения числовых данных (часы, ЗЕТ), что критически важно для учебного процесса.

В заключение можно отметить, что предложенная архитектура программного агента и гибридный подход к парсингу демонстрируют прикладную ценность. Они переводят учебный план из документа для чтения глазами в источник данных для мгновенного ответа без потери юридической и числовой точности. Система легко масштабируется и может быть развернута в любом вузе РФ, использующем стандартные макеты учебных планов, без значительных доработок.

Библиографический список:

1. Inline mode (Telegram). [Электронный ресурс]. — Режим доступа — URL: https://core.telegram.org/bots/inline (Дата обращения 20.01.2026)

2. Mo, L. Fuzzy matching algorithm of network information retrieval based on discrete mathematics / L. Mo // Applied Nanoscience. – 2023. – Vol. 13, No. 4. – P. 2865-2873. – DOI 10.1007/s13204-021-02190-y. [Электронный ресурс]. — Режим доступа — URL: https://elibrary.ru/item.asp?id=59687163 (Дата обращения 23.01.2026)

3. Openpyxl Documentation. [Электронный ресурс]. — Режим доступа — URL: https://openpyxl.readthedocs.io/ (Дата обращения 20.01.2026).

4. Pandas User Guide. [Электронный ресурс]. — Режим доступа — URL: https://pandas.pydata.org/docs/ (Дата обращения 20.01.2026)

5. PostgreSQL Documentation: pg_trgm (trigram matching). [Электронный ресурс]. — Режим доступа — URL: https://www.postgresql.org/docs/current/pgtrgm.html (Дата обращения 20.01.2026)

6. Python-telegram-bot Documentation (v20+ / v21). [Электронный ресурс]. — Режим доступа — URL: https://docs.python-telegram-bot.org/ (Дата обращения 20.01.2026)

7. Telegram Bot API Documentation. [Электронный ресурс]. — Режим доступа — URL: https://core.telegram.org/bots/api (Дата обращения 20.01.2026).

8. Баева В. Р., Дроздов А. Ю. ETL: актуальность и применение. преимущества и недостатки ETL инструментов // Вестник науки. 2019. №5 (14). [Электронный ресурс]. — Режим доступа — URL: https://cyberleninka.ru/article/n/etl-aktualnost-i-primenenie-preimuschestva-i-nedostatki-etl-instrumentov (Дата обращения 24.01.2026).

9. Бурцев В. А. Проблемы и перспективы пользовательского опыта в цифровых платформах: теоретические подходы и значение для бизнес-приложений // Инновации и инвестиции. 2025. №4. [Электронный ресурс]. — Режим доступа — URL: https://cyberleninka.ru/article/n/problemy-i-perspektivy-polzovatelskogo-opyta-v-tsifrovyh-platformah-teoreticheskie-podhody-i-znachenie-dlya-biznes-prilozheniy (Дата обращения 24.01.2026).

10. Егоркин, Е. С. Сравнение монолитной и микросервисной архитектуры в создании современных и удобных веб - приложений / Е. С. Егоркин // Синтез науки и общества в решении глобальных проблем современности: Сборник статей по итогам Международной научно-практической конференции, Тюмень., 30 мая 2024 года. - Стерлитамак: ООО "Агентство международных исследований", 2024. - С. 183-186. [Электронный ресурс]. — Режим доступа — URL: https://elibrary.ru/item.asp?id=67334736 (Дата обращения 15.01.2025)

11. Мальцева, Е. Н. Локальные вычислительные сети: программная среда, значение для пользователей сети. Ценность любой информационной сети / Е. Н. Мальцева, О. В. Дударева // Информатизация и виртуализация экономической и социальной жизни: Материалы IX Международной студенческой научно-практической конференции, Иркутск, 28 марта 2022 года. – Иркутск: Иркутский национальный исследова, 2022. – С. 199-203. [Электронный ресурс]. — Режим доступа — URL: https://elibrary.ru/item.asp?id=48354682 (Дата обращения 15.01.2025)

12. Траулько М. В. Программная реализация нечеткого поиска текстовой информации в словаре с помощью расстояния Левенштейна // Форум

молодых ученых. 2017. №12 (16). [Электронный ресурс]. — Режим доступа — URL: https://cyberleninka.ru/article/n/programmnaya-realizatsiya-nechetkogo-poiska-tekstovoy-informatsii-v-slovare-s-pomoschyu-rasstoyaniya-levenshteyna (Дата обращения 25.01.2026)

13. Трифонов, А. А. Алгоритмы построения инвертированного индекса для коллекции текстовых данных / А. А. Трифонов // Известия высших учебных заведений. Поволжский регион. Технические науки. – 2013. – № 3(27). – С. 52-61. [Электронный ресурс]. — Режим доступа — URL: https://elibrary.ru/item.asp?id=21469330 (Дата обращения 19.01.2026).

14. Черноусов, В. О. Использование нейросетевых моделей и технологии OCR для автоматизированной обработки и анализа PDF-документов / В. О. Черноусов // Нанотехнологии: наука и производство. – 2025. – № 5. – С. 69-75. [Электронный ресурс]. — Режим доступа — URL: https://elibrary.ru/item.asp?id=83210359 (Дата обращения 22.01.2026).

15. Шкоков, И. О. Архитектурный принцип единого источника данных для построения надежной бизнес-аналитики в крупных финансовых учреждениях / И. О. Шкоков // Финансовые рынки и банки. – 2025. – № 10. – С. 232-235. [Электронный ресурс]. — Режим доступа — URL: https://elibrary.ru/item.asp?id=83217194 (Дата обращения 20.01.2026).