МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**
Кафедра математической кибернетики и компьютерных наук

ИНТЕЛЛЕКТУАЛЬНАЯ СИСТЕМА СБОРА И ФОРМАТИРОВАНИЯ ДАННЫХ О НАУЧНЫХ ПУБЛИКАЦИЯХ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 451 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Козорезова Максима Алексеевича

Научный руководитель
профессор, д. т. н., доцент _____ А. С. Богомолов

Заведующий кафедрой
к. ф.-м. н., доцент _____ С. В. Миронов

Саратов 2026

ВВЕДЕНИЕ

Рост объёма научных публикаций и разнообразие требований к оформлению списков литературы (ГОСТ Р 7.0.5-2008, правила журналов и вузов, цифровые идентификаторы DOI) увеличивают трудоёмкость ручного переноса библиографических реквизитов с сайтов издателей и каталогов. Типовые менеджеры библиографии (Zotero, Mendeley) ориентированы на ручное ведение коллекций и не всегда обеспечивают единый серверный конвейер извлечения метаданных по DOI, URL и текстовому запросу с прозрачным указанием источника каждого поля, оценкой полноты записи и контролируемой обработкой произвольных веб-адресов. На страницах издателей набор полей и вёрстка непредсказуемы: эвристики по DOM, мета-теги `citation_*` и машинное обучение для классификации блоков HTML дополняют обращение к открытым каталогам Crossref, DataCite и OpenAlex. Это определяет актуальность разработки доступной интеллектуальной системы, снижающей время оформления и вычитки списка литературы при подготовке статей и квалификационных работ.

Цель работы - разработать интеллектуальную систему BiblioParser, объединяющую эвристический и каталожный сбор данных о публикации, оценку достоверности результата и набор клиентских интерфейсов: веб-сайт, REST API, Telegram-бот и автономную локальную сборку под Windows (единый исполняемый файл без установщика). Для достижения поставленной цели сформулированы следующие основные задачи:

1. систематизировать библиографические поля и правила оформления (ГОСТ и распространённые стили);
2. рассмотреть идентификацию публикаций по DOI и каталожные сервисы (Crossref, OpenAlex и др.);
3. описать извлечение метаданных из HTML и классификацию блоков страницы;
4. сформулировать угрозы при обработке URL и требования к архитектуре клиент–сервер;
5. обосновать разработку системы на основе обзора аналогов;
6. спроектировать, реализовать и оценить BiblioParser (разбор цитаты, проверка списка литературы, индексация журнала).

В качестве материалов исследования использованы открытые API

каталогов научных публикаций, HTML-страницы сайтов издателей и журналов, размеченный корпус текстовых блоков для обучения классификатора (3637 фрагментов, шесть классов, разбиение с группировкой по домену издателя), контрольный набор запросов для оценки извлечения метаданных и журнал нагрузочного прогона (16 запросов, май 2026).

Структура выпускной квалификационной работы определена поставленными задачами и состоит из введения, двух глав, заключения, списка литературы и приложения. Первая глава «Библиографические данные и автоматизация их обработки» содержит систематизацию библиографических элементов и требований к оформлению, обзор DOI и сервисов метаданных, подходы к извлечению из HTML, анализ угроз при обработке URL, сравнение существующих решений и постановку задачи. Во второй главе «Реализация системы BiblioParser» рассматриваются архитектура и конвейер обработки, разбор библиографической строки и валидация черновика, обучение классификатора блоков HTML, стек технологий и развёртывание, экспериментальная оценка по метрикам полноты, корректности DOI, доли ручной правки и времени ответа. В приложении приведены фрагменты программного кода ключевых функций конвейера.

Научная новизна работы (на уровне инженерной разработки) заключается в конкретизации унифицированного конвейера обработки для широкого класса издательских HTML-шаблонов; в разработке явной модели уверенности и пояснений при расхождении источников; в реализации разбора свободной библиографической строки с сопоставлением записи с открытыми каталогами и отечественным агрегатором научных публикаций «КиберЛенинка»; в построении процедуры автоматической валидации черновика списка литературы по строкам с оценкой полноты реквизитов.

Объектом исследования являются процессы получения, структурирования и представления библиографических сведений о научных статьях в электронной среде; предметом - методы и программная реализация автоматического извлечения метаданных по DOI, URL и текстовому запросу с последующим формированием цитаты и экспортом в машиночитаемые форматы (BibTeX, RIS). Практическая значимость состоит в существенном снижении трудоёмкости оформления и проверки списков литературы для исследователей и обучающихся.

ОСНОВНОЕ СОДЕРЖАНИЕ РАБОТЫ

Первая глава посвящена теоретико-методологическим основам автоматизации библиографических данных. В первом разделе систематизированы состав библиографического описания научной статьи и требования ГОСТ Р 7.0.5-2008, ГОСТ Р 7.0.100-2018 и распространённых международных стилей (APA, IEEE). Для автоматизации целесообразно хранить каноническую структуру полей (авторы, заголовок, журнал, год, том, выпуск, страницы, DOI, URL) и применять шаблон стиля на этапе вывода, не смешивая данные и представление: логические поля одни и те же, но меняются порядок авторов, пунктуация и формат DOI (`DOI:...`, `https://doi.org/...`, `doi:...`). Поля делятся на обязательные и условно-обязательные (том, выпуск, страницы или электронный номер статьи при online-first публикации), что используется при расчёте метрики полноты. Рассмотрены неоднозначности нормализации: отсутствие тома и выпуска при ранней онлайн-публикации, замена страниц на `article-number`, наличие заголовков на нескольких языках с хранением основного и альтернативного варианта в канонической модели.

Во втором разделе рассмотрены цифровые идентификаторы DOI и роль сервисов Crossref, DataCite, OpenAlex. Каталожные API обеспечивают воспроизводимое обогащение записи при известном DOI; OpenAlex дополняет поиск по названию и уточнение неоднозначных вариантов. Приведено сопоставление элементов Dublin Core с полями журнальной статьи; для институциональных контуров отмечена применимость протокола OAI-PMH. Рассмотрены смежные отечественные работы по автоматическому пополнению метаданных цифровых коллекций и извлечению сведений с порталов научных публикаций. В третьем разделе описаны подходы к извлечению метаданных из HTML. Конвейер фиксируется как последовательность: очистка HTML, извлечение структурированных полей, fallback-эвристики по DOM и регулярным выражениям, оценка уверенности. Приоритет отдаётся машиночитаемым тегам (`citation_*`, JSON-LD, Open Graph); для «шумных» вёрсток документ рассматривается как набор блоков. Признаковое пространство и правило классификации заданы формулами (1)–(2) (TF-IDF, глубина DOM, плотность ссылок).

$$\mathbf{x} = \left[\mathbf{x}_{\text{text}},\, \mathbf{x}_{\text{dom}}\right], \tag{1}$$

$$\hat{y} = \arg\max_{c} P(c \mid \mathbf{x}), \tag{2}$$

Классификация направлена на то, чтобы не принять элемент меню за заголовок статьи. Рассмотрены типовые ошибки: захват навигационного заголовка, принятие блока «related articles» за список авторов; для их снижения используются контекстные проверки и сверка с каталогом.

Четвёртый раздел формулирует угрозы при обработке произвольных URL (Server-Side Request Forgery, перегрузка oversized response) и требования к архитектуре клиент–сервер: все проверки URL и политика редиректов сосредоточены на сервере. Описаны принципы FAIR-метаданных и сопоставление с Dublin Core для интероперабельности экспорта.

Пятый раздел содержит обзор аналогов (Zotero, Mendeley, веб-сервисы по DOI) и обоснование разработки собственной системы по критериям автоматизации, воспроизводимости, API-доступа и прозрачности происхождения полей. В таблице 1 сопоставлены ограничения типовых менеджеров библиографии и возможности BiblioParser.

Таблица 1 – Сравнение типовых решений и системы BiblioParser

Типовые менеджеры (Zotero, Mendeley)	BiblioParser
Ручное ведение коллекций, слабый серверный конвейер	Единый конвейер DOI/URL/название/строка/черновик
Непрозрачное происхождение полей	Источник и уверенность для каждого реквизита
Нет встроенной SSRF-политики для URL	Блокировка частных адресов, лимиты редиректов и размера HTML
Ограниченный API для интеграций	REST API, Telegram-бот, portable-сборка

Шестой раздел задаёт постановку: гибридный каскад «каталог + HTML + ML», формализация качества через функционал (3), где C_{fields} - полнота обязательных полей, C_{doi} - корректность извлечения DOI, R_{manual} - доля случаев ручной правки, T - латентность ответа.

$$J = \big(C_{\text{fields}},\, C_{\text{doi}},\, 1 - R_{\text{manual}},\, T\big), \tag{3}$$

Сравниваются три класса подходов: чисто эвристический парсинг HTML, извлечение только из каталогов и гибридный вариант, выбранный как компромисс между полнотой и воспроизводимостью. Зафиксированы

формализованные требования к системе: доля успешных ответов на тестовом наборе, корректность BibTeX/RIS, наличие REST-эндпоинтов, ограничение частоты запросов, SSRF-фильтрация и лимит размера HTML-ответа.

Во второй главе представлены проектирование, реализация и оценка системы BiblioParser. В первом разделе описан подход к решению задачи и архитектура вокруг модуля `inference_pipeline`, вызываемого из веб-приложения, REST API, Telegram-бота и локальной сборки. Сценарии включают ввод по DOI, URL, названию, разбор одной библиографической строки, проверку многострочного черновика списка, пакетное оформление, экспорт BibTeX/RIS и отображение индексации журнала по локальному справочнику. Для поиска по названию формируется ранжированный список вариантов; пользователь выбирает запись, после чего конвейер повторяет нормализацию DOI и обогащение. Граница доверия архитектуры проходит между клиентом, сервером извлечения и внешними источниками (Crossref, OpenAlex, сайты издателей, «КиберЛенинка»), что позволяет централизованно применять SSRF-фильтрацию, кэширование и журналирование.

Во втором разделе детализирован конвейер из пятнадцати этапов. На шаге приёма определяется тип ввода (DOI, URL, название, строка цитирования, черновик); для URL извлекается домен для последующей группировки в ML-экспериментах. Нормализация DOI удаляет префиксы издательских ссылок и проверяет допустимый формат. Запрос к Crossref при 404 инициирует ветку OpenAlex или HTML; при 429 применяется экспоненциальная задержка. Слияние полей каталога не перезаписывает ненулевые значения пустыми. Загрузка HTML сопровождается SSRF-проверкой, таймаутом и ограничением размера. Извлекаются `citation_*`, Open Graph, JSON-LD; при неполноте выполняется сегментация DOM и классификация блоков. DOI ищется в тексте и гиперссылках; Unpaywall уточняет доступность OA-версии. Формируются цитата, BibTeX/RIS, взвешенная оценка уверенности по формуле (4) и запись в журнал.

$$S = \sum_{k \in K} w_k q_k, \qquad \sum_{k \in K} w_k = 1, \tag{4}$$

где q_k - локальная оценка согласованности поля, w_k - вес важности (для DOI, авторов и заголовка веса выше). Политика слияния задаёт приоритет каталожного DOI и предупреждения при конфликте заголовков или авторов

между каталогом и HTML. Клиентская форма дублирует проверку URL до отправки (схема, хост, учётные данные, цепочка редиректов).

Третий раздел посвящён разбору свободной библиографической строки и валидации черновика. Алгоритм извлекает DOI регулярным выражением либо выделяет авторов, заголовок и источник по шаблонам, сопоставляет запись с Crossref (метрика `SequenceMatcher`) и при кириллице в заголовке - с «КиберЛенинкой». Режим `/validate_draft` обрабатывает строки черновика независимо, формирует статус полноты и перечень отсутствующих реквизитов. На главной странице веб-клиента объединены основная форма запроса и блок «Оформить список литературы» с переключателями режимов; предусмотрено локальное сохранение последнего результата. На странице результата для неоднозначного поиска по названию выводится ранжированный список вариантов; карточка записи содержит цитату, индикатор полноты, метки Scopus/WoS/ВАК/РИНЦ при совпадении в `journal_indices.json` и пояснение происхождения каждого поля.

Четвёртый раздел описывает обучение классификатора блоков HTML в scikit-learn (TF-IDF, признаки DOM). Исходные данные - размеченный корпус текстовых блоков, извлечённых из HTML-страниц научных журналов разных доменов; признаки включают комбинацию символьного и словного TF-IDF, признаки DOM-структуры, плотность ссылок и простые символьные статистики. Для контроля обобщающей способности применяется разбиение с группировкой по домену издателя. Сравнивались логистическая регрессия, случайный лес и LinearSVC; для конвейера выбран `random_forest_balanced`. Сводные метрики на валидации приведены в таблице 2. Анализ ошибок по доменам показал наибольшую долю путаницы на страницах, где заголовок статьи визуально близок к навигационному; для таких случаев полезны признаки глубины DOM и доля ссылок внутри блока.

Таблица 2 – Метрики классификации блоков HTML на размеченном корпусе

Модель	Accuracy	F1 macro	F1 weighted
Logistic Regression	0,048	0,015	0,004
Random Forest (balanced)	0,560	0,265	0,476
Linear SVC	0,532	0,214	0,434

Абляционный анализ подтвердил вклад JSON-LD, DOM-эвристик и ML-этапа: отключение JSON-LD увеличивает долю неполных записей на

сайтах современных издателей; отключение эвристик снижает покрытие нестандартных шаблонов; отключение ML увеличивает частоту ошибок «навигация вместо заголовка». На доменах с полным набором `citation_*` вклад модели менее критичен, однако она выполняет страховочную роль при противоречии мета-тегов и видимого заголовка.

Пятый раздел раскрывает программную реализацию и развёртывание. Описана политика слияния полей каталога и HTML в виде правил приоритета (таблица политик в работе): DOI из каталога, согласованные поля объединяются с выбором лучшего качества, конфликты сопровождаются предупреждением. Приведено сопоставление ключей JSON Crossref (`message.title`, `message.author`, `message.container-title`, `message.volume`, `message.DOI` и др.) с полями внутренней модели BiblioParser. Серверная часть построена на Python/Flask; Flask-Limiter и Redis кэшируют ответы Crossref и DataCite; SQLite журналирует запросы. Внутренняя модель публикации отделена от строки цитирования; экспорт BibTeX/RIS сопоставляет поля тегам `@article`, `TY`, `AU`, `TI`, `JO`, `PY`, `DO`. Docker-образ фиксирует зависимости; `docker-compose` поднимает веб-сервис, Redis и бота (профиль `bot`). Публичное развёртывание на Render использует Gunicorn и `/health`. Windows-сборка PyInstaller запускает встроенный сервер на `127.0.0.1:8765` и окно pywebview без установщика. REST-контракты включают `/api/work`, `/parse_citation`, `/validate_draft`; коды 400 (SSRF) и 429 (лимит) документированы. Telegram-бот вызывает тот же `inference_pipeline`, что исключает расхождение логики.

Шестой раздел посвящён экспериментальной оценке и интерфейсам. Кроме таблицы 3, зафиксированы показатели ML на корпусе (таблица 2) и сравнение с аналогами (таблица 1). Качество оценивалось по полноте обязательных полей, корректности DOI, доле ручной правки и времени ответа. Контрольный прогон включал смешанные сценарии; для автоматики при успешном завершении принята полнота $F{=}1{,}00$. По журналу прогона (таблица 3) среднее время при вводе DOI составило около 7 с, при URL - около 20 с; усреднённо по смешанному набору после минимальной правки - 13,49 с. Для сопоставления с ручным оформлением использована оценка по формуле (5) при $|F|{=}8$ обязательных реквизитов, $\bar{t}{=}18$ с и $t{=}40$ с, что даёт

$t \approx 184$ с при полноте около 0,93.

$$t \approx |F| \cdot \bar{t} + t, \tag{5}$$

Отношение $t/t_{\mathrm{DOI}} \approx 26$ согласуется с качественной оценкой выигрыша по времени на порядок.

Седьмой раздел описывает обработку ошибок и воспроизводимость интеграций. Зафиксированы типовые ответы API: успешное получение записи по DOI (`200`), разбор строки с полями `metadata` и `parse_info`, валидация черновика со сводкой `summary`, отказ по SSRF (`400`), превышение лимита (`429`). При недоступности каталога выполняется деградация к частичному результату из HTML с понижением уверенности. В приложении к работе приведены листинги функций `extract_metadata_from_url`, SSRF-проверки, `parse_citation_string`, `validate_draft_text` и обработчиков REST API.

Таблица 3 – Среднее время оформления одной записи (нагрузочный прогон, май 2026)

Сценарий	N	Время, с	Полнота F
Ручное оформление (оценка)	16	184	0,93
BiblioParser, DOI (до правки)	4	6,98	1,00
BiblioParser, URL (до правки)	3	20,49	1,00
BiblioParser (после мин. правки)	16	13,49	1,00
Разбор строки (`/parse_citation`)	2	6,87	1,00
Проверка черновика (`/validate_draft`)	6	4,29	1,00

Для пакетного режима «Список: быстро» одновременно обрабатываются до десяти непустых строк с выгрузкой TXT, BibTeX, RIS, LaTeX и CSL JSON. При последовательной обработке DOI-записей накопленное время ориентировочно 35 с на пять ссылок, 70 с на десять и 350 с на пятьдесят; пакетный режим снижает накладные расходы. Сценарий `title_multi` (неоднозначный поиск по названию) показал среднее время 33,54 с при $N{=}3$. Метрики и журнал прогона сохранены в `reports/` репозитория.

Реализованы веб-интерфейс с режимами «одна строка», «список с диагностикой» и «список: быстро», мобильная вёрстка, Telegram-бот (команды `/cite`, `/draft`) и единый модуль конвейера для всех каналов. Исходный код организован вокруг `inference_pipeline`, `parse_citation_string`, `validate_draft_text` и отдельного SSRF-фильтра; листинги ключевых

фрагментов приведены в приложении выпускной работы. Практическая значимость - снижение трудоёмкости оформления списка литературы при прозрачности источников данных и контролируемой безопасности обработки URL; система различает получение метаданных и оформление ссылки с учётом языка публикации и согласованности записей в списке.

ЗАКЛЮЧЕНИЕ

В рамках работы спроектирована и реализована интеллектуальная система BiblioParser для автоматического сбора, структурирования и форматирования библиографических сведений о научных публикациях. Анализ существующих менеджеров библиографии и веб-сервисов позволил выделить ограничения: ориентация на ручное ведение коллекций, непрозрачное происхождение полей в готовой записи, отсутствие единой SSRF-политики при загрузке страниц по URL пользователя. Обоснована необходимость серверного конвейера с каталожным обогащением, HTML-разбором, оценкой уверенности и программными интерфейсами для интеграций.

В теоретической части систематизированы требования к библиографическому описанию, рассмотрены DOI и открытые каталоги, подходы к извлечению из HTML и риски обработки URL; выполнен обзор аналогов и формализована постановка задачи с измеримыми критериями качества.

Разработан гибридный конвейер, объединяющий каталожное обогащение (Crossref, OpenAlex, DataCite, Unpaywall), разбор HTML с эвристиками и классификатором блоков (лучший результат на корпусе 3637 блоков - случайный лес с accuracy 0,56, F1-macro 0,27), оценку уверенности при расхождении источников, разбор готовой библиографической строки с сопоставлением в «КиберЛенинке», проверку черновика списка литературы, пакетное оформление и отображение индексации журнала (ВАК, Scopus, WoS, РИНЦ). Веб-интерфейс поддерживает предварительную проверку URL, режимы оформления списка, карточки вариантов с экспортом BibTeX/RIS и диагностику некорректного DOI или отсутствия полнотекста. Реализованы веб-интерфейс, REST API, Telegram-бот, Docker-развёртывание и локальная Windows-сборка.

Экспериментальная оценка показала сокращение времени получения записи по сравнению с ручным оформлением (порядка 7 с против 184 с на одну ссылку при вводе DOI) при полноте обязательных полей $F{=}1{,}00$ в успешных автоматических ответах контрольного прогона. Реализованные каналы доступа (веб, API, Telegram, локальное приложение) используют единый серверный конвейер, что исключает расхождение бизнес-логики. Развёртывание на Render и Docker-конфигурация обеспечивают воспроизводимость демонстрационного стенда; метрики ML и журнал прогона сохранены в репозитории.

Цель работы достигнута, поставленные задачи решены. Полученные результаты могут использоваться при подготовке списков литературы к публикациям и квалификационным работам, интеграции в учебные и исследовательские процессы через REST API, а также как основа для дальнейшего расширения корпуса разметки HTML, пополнения справочника индексации журналов и повышения устойчивости классификатора к новым шаблонам издательских сайтов. Перспектива развития — расширение регрессионного набора, калибровка показателя уверенности (4), переход на PostgreSQL и фоновые воркеры при росте нагрузки, углубление семантического слоя (ORCID/ROR, дедупликация версий публикаций).