UNIVERSITATEA DE STAT DIN MOLDOVA

FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ

DEPARTAMENTUL DE INFORMATICĂ

PAVLENCO ANTON

«CĂUTARE BAZATĂ PE METADATE»

INFORMATICĂ APLICATĂ

Teză de master

Șef de departament: _____ Titu Capcelea, dr. conf., univ

Conducător științific: _____ Croitor Mihail, lector universitar

Autor: _____ Pavlenco Anton

CHIȘINĂU– 2025 A.

МОЛДАВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ МАТЕМАТИКИ И ИНФОРМАТИКИ

ДЕПАРТАМЕНТ ИНФОРМАТИКИ

ПАВЛЕНКО АНТОН

«ПОИСК, ОСНОВАННЫЙ НА МЕТАДАННЫХ»

ПРИКЛАДНАЯ ИНФОРМАТИКА

Магистерская работа

Директор департамента: _____ Титу Капчеля,

(подпись)

Научный руководитель: _____ Михаил Кроитор, учёное звание

(подпись)

Автор: _____ Павленко Антон

(подпись)

КИШИНЕВ – 2025 Г.

Содержание

ANNOTARE

la teza de master „CĂUTARE BAZATĂ PE METADATE”, student

Pavlenco Anton, grupa MIA2302, programul de studii INFORMATICĂ APLICATĂ

Structura tezei. Lucrarea este compusă din introducere, trei capitole, concluzii și o bibliografie care include 20 de surse.

Cuvinte-cheie. Căutare informațională, metadate, web semantic, ontologie, ingineria cunoștințelor, indexare, Wikipedia, bază de cunoștințe, Apache Solr, date deschise.

Actualitate. În contextul creșterii rapide a volumului de informații digitale și a predominării datelor nestructurate, apare necesitatea unor sisteme de căutare capabile să ia în considerare nu doar conținutul, ci și contextul. Metadatele permit extinderea capacităților de căutare informațională, îmbunătățind relevanța și exhaustivitatea rezultatelor.

Scopul și obiectivele cercetării. Scopul lucrării de master constă în investigarea metodelor de reprezentare și căutare a cunoștințelor pe baza metadatelor și implementarea acestora într-un sistem software aplicativ. Pentru atingerea acestui scop, sunt abordate următoarele obiective: analiza bazelor teoretice ale ingineriei cunoștințelor, studierea tehnologiilor de indexare (în special utilizarea Apache Solr), examinarea particularităților datelor deschise și a resursei Wikipedia, precum și dezvoltarea unui instrument software pentru colectarea, structurarea și căutarea informațiilor.

Rezultatele așteptate și obținute se rezumă la: (1) formalizarea unei abordări de reprezentare a cunoștințelor utilizând metadate; (2) crearea unei soluții software originale — sistemul WikipediaTools — pentru implementarea componentelor de descărcare, analiză și indexare a metadatelor articolelor Wikipedia și verificarea experimentală a eficienței metodei propuse.

Problema importantă rezolvată în cercetare constă în elaborarea unei arhitecturi eficiente pentru căutarea informațională, bazată pe conexiuni structurale și metainformații, ceea ce deschide calea către sisteme de căutare mai inteligente.

Valoarea aplicativă. Soluția dezvoltată poate fi utilizată drept prototip pentru crearea unor sisteme mai ample și mai inteligente de căutare a cunoștințelor. Abordarea lucrului cu date deschise și conexiunile structurate dintre entități permite creșterea semnificativă a eficienței analizei informaționale, fiind aplicabilă în domeniul științific, educațional și în alte domenii practice.

АННОТАЦИЯ

к магистерской «*ПОИСК, ОСНОВАННЫЙ НА МЕТАДАННЫХ*», студента
Павленко Антон, группа MIA2302, программа обучения ПРИКЛАДНАЯ ИНФОРМАТИКА

Структура диссертации. Работа состоит из введения, трёх глав, выводов и источников из 20 источников.

Ключевые слова. Информационный поиск, метаданные, семантическая паутина, онтология, инженерия знаний, индексирование, Wikipedia, база знаний, Apache Solr, открытые данные.

Актуальность. В условиях стремительного роста объёмов цифровой информации и доминирования неструктурированных данных возникает необходимость в системах поиска, способных учитывать не только содержание, но и контекст. Метаданные позволяют расширить возможности информационного поиска, улучшая релевантность и полноту результатов.

Цель и задачи исследования. Целью магистерской работы является исследование методов представления и поиска знаний на основе метаданных и их реализация в прикладной системе. В рамках этой цели решаются задачи: анализа теоретических основ инженерии знаний, изучения технологий индексирования (в частности, с использованием Apache Solr), анализа особенностей открытых данных и Wikipedia, а также разработки программного инструмента для сбора, структурирования и поиска информации.

Ожидаемые и достигнутые результаты сводятся к: (1) формализации подхода к представлению знаний с использованием метаданных; (2) созданию оригинального программного решения — системы WikipediaTools для реализации компонентов загрузки, анализа и индексирования метаданных статей Wikipedia и экспериментальной проверке эффективности предложенного подхода.

Важная решённая проблема в исследовании заключается в построении эффективной архитектуры информационного поиска, основанной на структурных связях и метаинформации, что открывает путь к более интеллектуальным системам поиска.

Прикладное значение. Разработанное решение может быть использовано как прототип для создания более масштабных и интеллектуальных систем поиска знаний. Подход к работе с открытыми данными и структурированными связями между сущностями позволяет значительно повысить эффективность информационного анализа, что может быть востребовано в научных, образовательных и прикладных сферах.

Введение

Актуальность работы

В современном информационном обществе объем доступной цифровой информации стремительно растет. По данным исследований, в 2023 году в мире было создано 120 зеттабайт данных, а к 2025 году этот объем прогнозируется на уровне 181 зеттабайт *[1]*. Это означает, что ежедневно создается около 2,5 квинтильонов байт данных *[2]*.

Однако большая часть этих данных является неструктурированной — тексты, изображения, видео, аудио и другие форматы, не поддающиеся традиционной табличной обработке. Оценки показывают, что до 90% всех данных в мире относятся к неструктурированным . Это создает серьезные вызовы для систем хранения, обработки и поиска информации.

Традиционные методы текстового поиска, основанные на простом сопоставлении ключевых слов, становятся недостаточными в условиях такого объема и разнообразия данных. Пользователи ожидают получения релевантных и осмысленных результатов, что требует более глубокого понимания контекста и структуры информации.

Одним из направлений, позволяющих существенно повысить качество информационного поиска, является использование метаданных — описательных характеристик, сопровождающих основной контент. Метаданные обеспечивают дополнительный контекст, позволяя системам поиска выходить за пределы простого сопоставления ключевых слов. Они играют ключевую роль в системах управления знаниями, где важны не только данные, но и связи между ними, их источники, время обновления, степень достоверности и прочие характеристики. Именно благодаря метаданным возможно построение более интеллектуальных, семантически насыщенных информационных систем, которые способны выполнять поиск по смыслу, поддерживать навигацию по взаимосвязанным сущностям и обеспечивать более точную фильтрацию и агрегирование информации.

Таким образом, в условиях стремительного роста объема данных и увеличения доли неструктурированной информации, использование метаданных становится неотъемлемым элементом эффективного поиска и управления знаниями. Это делает тему исследования методов поиска, основанных на метаданных, особенно актуальной в современном цифровом мире.

Метаданные обеспечивают дополнительный контекст, позволяя системам поиска выходить за пределы простого сопоставления ключевых слов. Они играют ключевую роль в

системах управления знаниями, где важны не только данные, но и связи между ними, их источники, время обновления, степень достоверности и прочие характеристики. Именно благодаря метаданным возможно построение более интеллектуальных, семантически насыщенных информационных систем, которые способны выполнять поиск по смыслу, поддерживать навигацию по взаимосвязанным сущностям и обеспечивать более точную фильтрацию и агрегирование информации.

Актуальность темы исследования обусловлена необходимостью разработки систем, способных обрабатывать большие объемы информации и предоставлять пользователю действительно релевантные результаты. Одним из ярких примеров информационного ресурса, представляющего интерес для анализа, является Википедия — крупнейшая в мире открытая база знаний. Благодаря открытым данным, формализованной структуре статей и богатому набору метаданных, она представляет собой идеальную площадку для изучения методов поиска, основанных на связях между сущностями и структурированной информации.

Цель работы

Целью настоящей работы является исследование методов организации и поиска информации на основе метаданных и структурных связей между сущностями в больших объемах гипертекстовых данных. Основное внимание уделяется анализу подходов, применимых к крупным открытым информационным ресурсам, таким как Википедия, где статьи формируют сложную сеть взаимосвязей и обладают богатыми структурированными описаниями.

В рамках исследования были поставлены следующие задачи:

- проанализировать современные подходы к представлению знаний и организации информационного поиска;
- изучить методы индексирования и структурирования текстовых и гипертекстовых данных;
- исследовать особенности метаданных, сопровождающих статьи Википедии, и их роль в построении связей между документами;
- рассмотреть архитектурные принципы построения систем поиска, опирающихся на семантические связи и метаинформацию;
- оценить эффективность использования метаданных для улучшения релевантности и полноты результатов поиска.

В качестве практической части работы было реализовано приложение WikipediaTools — программный инструмент, осуществляющий загрузку и обработку метаданных Википедии, формирование связей между статьями и их сохранение в структурированном виде. Разработка системы служит демонстрацией применимости изученных методов на практике и позволяет исследовать возможные подходы к поиску на основе связей между документами.

Объектом исследования является информационное пространство Википедии, представленное в виде гипертекстовых взаимосвязанных статей.
Предмет исследования — методы структурирования, индексирования и поиска информации с использованием метаданных и сетевых связей между элементами содержимого.

В процессе реализации проекта использовались язык программирования C++ и такие библиотеки и инструменты, как libcurl (для взаимодействия с API), nlohmann/json (для работы с JSON), а также СУБД MySQL для хранения данных и связей между статьями.

Структура работы

Первая глава посвящена общим вопросам, связанным с понятием знаний, способами их формализации и применением в поисковых задачах.

Вторая глава рассматривает современные системы представления и поиска информации, включая индексацию текстов, сканирование сайтов и общие принципы работы поисковых систем.

В третьей главе подробно описана разработанная система WikipediaTools, включая постановку задачи, архитектуру, детали реализации и результаты тестирования.
Работа завершается выводами, в которых подведены итоги исследования и намечены направления дальнейшего развития проекта.

Глава I Знания и способы их представления

Понятие знания

Знание — это ключевой элемент в понимании и обработке информации, особенно в контексте информационных систем и технологий. В научной литературе существует множество определений знания, каждое из которых подчеркивает различные аспекты этого понятия.

Одним из первых известных определений знания дал Платон в диалоге *Теэтет*. Он определил знание как *обоснованное истинное мнение*. Это определение стало классическим в философии и долгое время оставалось основой для эпистемологии — раздела философии, изучающего природу знания.

Одно из многих других определений предлагает рассматривать знание как информацию, которая получила смысл и была интегрирована с другими элементами понимания. *[3]* Это определение подчеркивает, что знание не является просто набором данных, а представляет собой осмысленную и структурированную информацию, которая может быть использована для принятия решений и решения задач.

В области искусственного интеллекта (ИИ) и когнитивных наук знание рассматривается не только как продукт человеческого мышления, но и как объект моделирования и представления в компьютерных системах. Это приводит к разработке различных методов представления знаний, таких как фреймы, семантические сети и онтологии, которые позволяют системам ИИ эффективно использовать знания для решения сложных задач.

Данные

Данные — это зафиксированные наблюдения, измерения или факты, представленные в виде, пригодном для хранения, обработки и передачи. В информатике данные обычно представляют собой числа, текст, изображения или сигналы, закодированные в цифровой форме. Данные не имеют смысла или ценности, поскольку они не имеют контекста и интерпретации. [4]

- **Данные** — это необработанные, изолированные единицы (например, «42», «2025-05-05» или «125.3 кг»).
- **Информация** — это данные, помещенные в контекст (например, «масса объекта составляет 125.3 кг»).

- **Знание** — это интерпретированная и осмысленная информация, которую можно применять к решению задач (например, «если масса объекта выше 100 кг, то нужен усиленный подъемный механизм»).

Данные становятся информацией, когда они обрабатываются таким образом, что становятся осмысленными; информация становится знанием, когда она сочетается с опытом, контекстом, интерпретацией и размышлениями.

Слово **«data»** происходит от латинского *datum* — «данное» (в значении «то, что дано»). Первоначально термин использовался в логике и философии ещё в античности, а в современном научном контексте — с XVII века.

Готфрид Лейбниц — один из первых, кто рассуждал о символах и числах как представителях реальности, тем самым предвосхищая идеи обработки данных. Его *бинарная система* (0 и 1) — предшественница цифровых данных.

В XX веке понятие «данные» получило новую жизнь с развитием компьютеров. Клод Шеннон в 1948 году заложил основы теории информации — именно он первым формализовал понятие *бит* как единицу измерения данных.

В информатике данные могут быть:

- Структурированными (таблицы, базы данных),
- Полуструктурированными (XML, JSON),
- Неструктурированными (изображения, аудио, видео, текст).

Современные системы обработки данных включают ETL-процессы (Extract-Transform-Load), Big Data, Data Mining, и Data Science — все эти дисциплины опираются на формализованные модели представления и анализа данных.

Количество цифровых данных удваивается каждые 2–3 года. По оценке IDC, в 2025 году мировой объем данных превысит 175 зеттабайт.

Это создает вызовы:

- Как хранить такие объемы?
- Как извлекать из них полезную информацию?
- Как избежать «информационного шума»?

Метаданные

Метаданные — это *данные о данных*. То есть, это информация, которая описывает, поясняет или даёт контекст самим данным. Примеры метаданных:

- Для фотографии: дата съёмки, модель камеры, координаты GPS.
- Для научной статьи: авторы, дата публикации, DOI, ключевые слова.
- Для базы данных: типы столбцов, индексы, связи между таблицами.

Сама идея метаданных возникла ещё до цифровой эпохи — в библиотеках, когда нужно было описывать книги: по автору, теме, году, изданию. Это тоже были метаданные — только на карточках.

С развитием компьютеров термин стал использоваться для описания *электронных* данных. Активно он начал применяться с конца XX века, когда началась эра интернета и огромных массивов информации, которые требовалось как-то структурировать.

Особенно важно стало использовать метаданные, когда данные стали:

- **распределёнными** (например, в облачных хранилищах),
- **автоматически обрабатываемыми** (в базах данных и поисковиках),
- **многообразными** (тексты, аудио, видео, таблицы и т. д.).

В сфере ИТ метаданные — это не просто описание файлов. Они:

- лежат в основе **структуры баз данных** (названия таблиц, типы столбцов, связи),
- управляют **поиском** в информационных системах (например, ключевые слова),
- позволяют **автоматизировать** обработку данных (например, указание форматов и типов).

В современных системах, таких как большие дата-центры, «облака», искусственный интеллект, без чётко организованных метаданных работа с данными просто невозможна. Метаданные позволяют компьютерам понимать, как интерпретировать те или иные фрагменты информации.

Обобщим:

- **Данные** — это факты: числа, слова, измерения.
- **Метаданные** — это ярлыки и описания, которые рассказывают, *что значат* эти данные, *откуда они, как с ними работать*.
- **Знание** — это когда человек (или алгоритм) использует данные и метаданные, чтобы сделать выводы, принимать решения или действовать.

Уровень	Пример
Данные	36.6, 38.2, 40.1
Метаданные	Температура тела, °C, утро/вечер
Знание	Повышение температуры вечером может говорить о воспалении

Метазнание

Метазнание — это информация о структуре, источнике, достоверности и применимости других знаний. Оно помогает системам и людям понимать:

- **Как** было получено знание?
- **Насколько** оно достоверно?
- **В каких** условиях его можно применять?
- **Какие** ограничения у него есть?

В контексте экспертных систем метазнание позволяет не только использовать базу знаний, но и анализировать её структуру, делать обобщения и принимать решения о её применимости в различных ситуациях.

Термин "метазнание" получил развитие в области инженерии знаний и когнитивных наук. Он стал особенно актуален с развитием систем, которые не только хранят знания, но и способны анализировать и модифицировать их. Адам Гадомски, разработчик направления инженерии знаний TOGA, подчёркивает, что для разумных систем метазнание включает в себя правила, методы планирования, моделирования и обучения, которые позволяют модифицировать знания о предметной области.

Метазнание находит применение в различных областях:

- **Экспертные системы**: помогает системам анализировать и адаптировать свои базы знаний.
- **Образование**: развитие метакогнитивных навыков у студентов для улучшения процесса обучения.
- **Научные исследования**: анализ структуры и эволюции научных знаний.
- **Информационные системы**: управление качеством и достоверностью информации.

В научной сфере метазнание используется для анализа того, как формируются и развиваются научные идеи. С помощью современных вычислительных инструментов исследователи могут изучать динамику научных исследований, выявлять закономерности в развитии научных направлений и оценивать влияние различных факторов на эволюцию знаний. *[5]*

В информационных системах метазнание используется для:

- **Управления качеством данных**: оценка достоверности и актуальности информации.
- **Оптимизации поиска**: улучшение алгоритмов поиска за счёт анализа структуры знаний.
- **Адаптации систем**: позволяет системам адаптироваться к новым условиям и требованиям.

Характеристика	Метазнание	Метаданные
Что описывает?	Знания: структуру, достоверность, применение	Данные: тип, формат, источник, структура
Область	Искусственный интеллект, когнитивные науки	Информатика, базы данных, веб-технологии
Пример	"Это знание получено эмпирически и устарело"	"Этот файл — изображение JPEG, создан 2022-01-01"
Уровень	Концептуальный, когнитивный	Технический, описательный
Цель	Помочь анализировать и применять знания	Помочь идентифицировать, классифицировать данные

Представление знаний

Представление знаний — это область искусственного интеллекта, занимающаяся тем, как знания можно формализовать так, чтобы компьютер мог их интерпретировать, использовать и применять для вывода новых фактов. Это не просто хранение информации, а создание такой структуры, которая позволяет машине рассуждать.

Представление знаний объединяет две ключевые задачи:

1. **Представление знаний** — как описывать факты, правила, связи и понятия;
2. **Вывод/рассуждение** — как на основе этих знаний получать новые знания или принимать решения.

Пример: если в системе знаний указано, что «все люди смертны» и «Сократ — человек», то машина должна уметь сделать вывод «Сократ смертен».

На практике используются разные формы представления:

- **Логические системы** (например, предикатная логика);
- **Семантические сети** — графы, где вершины — понятия, а рёбра — связи;
- **Фреймы** — подобие шаблонов с «ячейками» для значений;
- **Онтологии** — формализованные описания предметных областей;
- **Базы правил** — например, продукционные системы.

Представление знаний необходимо, так как:

- Без формализации знания невозможно масштабируемое машинное мышление;
- Представление знаний лежит в основе экспертных систем**,** семантического поиска**,** систем рекомендаций**,** автономных агентов;
- Современные подходы (например, обогащённые графы знаний**,** логики описаний) позволяют соединять представление знаний с машинным обучением.

База знаний

База знаний — это специализированное хранилище структурированной информации, предназначенное для хранения, организации и использования знаний, необходимых для решения задач, принятия решений или поддержки рассуждений в рамках определённой предметной области. В отличие от традиционной базы данных, которая хранит факты (например, числовые или текстовые значения), база знаний акцентирует внимание на отношениях между объектами, правилах вывода, логических структурах и концептуальных связях между единицами знаний.

Наиболее часто базы знаний используются в системах искусственного интеллекта, экспертных системах, интеллектуальных агентах и семантических веб-приложениях. В таких системах база знаний играет роль источника знаний, которым оперирует механизм вывода (инференции), обеспечивая логические заключения, проверку условий, принятие решений и т.д.

Классическим примером является экспертная система MYCIN (1970-е), которая диагностировала инфекционные заболевания. В ней база знаний состояла из более чем 450 правил, оформленных в виде конструкций вида *«ЕСЛИ … ТО …»*. *[6]* Эта система показала, что даже простое формализованное представление может достигать результатов, сопоставимых с врачами-экспертами.

Исторически понятие базы знаний стало активно использоваться в 1970–1980-х годах с развитием экспертных систем. Однако фундамент к этому понятию был заложен ранее — через исследования в области логики, семантики и теории формальных языков. Сегодня базы знаний

лежат в основе современных технологий: от голосовых помощников (Siri, Alexa) до сложных рекомендационных систем.

Важно также понимать, что базы знаний могут быть выражены в разных формах: логических (например, продукционные правила), семантических сетях, фреймах, онтологиях (например, OWL-онтологии в семантической паутине), а также в виде графов, как в базе Wikidata или DBpedia.

Онтология

Онтология в информатике — это формальное представление набора концептов и отношений между ними в определённой предметной области. Иначе говоря, онтология описывает, какие сущности существуют в конкретной области знаний и как они взаимосвязаны. Это понятие особенно важно в задачах искусственного интеллекта, семантического веба, обработки естественного языка и интеграции данных.

Термин "онтология" происходит из философии, где он обозначает учение о бытии. В информатику этот термин был заимствован в 1990-х годах, когда усилилась потребность в структурированном представлении знаний для интеллектуальных систем. Онтология — это явная спецификация концептуализации. [7]

В условиях стремительного роста объёма данных, особенно в интернете, всё более актуальной становится задача не просто хранения и поиска информации, но и понимания её структуры и смысла. Именно для этого используются онтологии — они позволяют формализовать смысловое содержание информации.

Примеры применения онтологий:

- Семантическая паутина (Semantic Web): онтологии применяются для описания значений данных в формате RDF и OWL, что позволяет поисковым системам и интеллектуальным агентам лучше интерпретировать содержание веб-страниц.
- Медицина: онтология SNOMED CT обеспечивает единый словарь для кодирования медицинской информации.
- Биоинформатика: онтология Gene Ontology (GO) описывает функции генов.

Онтология обычно включает:

- Классы (concepts): абстрактные категории объектов, например, *"Человек"*, *"Болезнь"*.
- Экземпляры (individuals): конкретные представители классов.
- Свойства (properties): характеристики объектов или связи между ними.

- Аксиомы: логические правила, задающие ограничения и отношения.

Пример: онтологии медицинских знаний может быть описано, что *"каждое лекарство предназначено для лечения хотя бы одного заболевания"*, и это будет оформлено как логическая аксиома.

В отличие от базы знаний, которая может быть просто набором фактов и правил, онтология подчёркивает структуру концептов и их взаимосвязи. Онтология может быть частью базы знаний, задавая её семантический каркас.

Семантическая паутина

Семантическая паутина — это концепция расширения Всемирной паутины, предложенная с целью сделать данные в Интернете более осмысленными и пригодными для автоматической обработки. В отличие от классической Web 1.0/2.0, где информация предназначена в основном для людей, семантическая паутина ориентирована на **машинное понимание и интерпретацию информации**.

Термин и идея семантической паутины были предложены Тимом Бернерс-Ли (Tim Berners-Lee) — создателем Всемирной паутины. В 2001 году он в соавторстве с Джеймсом Хендлером и Ора Лассилой опубликовал программную статью в журнале *Scientific American*, где представил видение семантической паутины как следующего эволюционного этапа интернета:
«Семантическая паутина привнесет структуру в содержательное наполнение веб-страниц, создав среду, в которой программные агенты, перемещающиеся со страницы на страницу, смогут легко выполнять сложные задачи для пользователей». [8]

Семантическая паутина направлена на решение следующих задач:

- Стандартизация данных: обеспечение единого формата представления информации (например, через RDF, OWL, SPARQL).
- Интероперабельность: возможность объединения данных из различных источников и систем.
- Автоматизация: предоставление машинам возможности понимать смысл и логику информации.

Семантическая паутина базируется на технологическом стеке, известном как Semantic Web Stack, включающем следующие уровни:

- RDF (Resource Description Framework) — базовый формат представления данных в виде триплетов (субъект, предикат, объект).

- OWL (Web Ontology Language) — язык для описания онтологий и логических взаимосвязей между сущностями.
- SPARQL — язык запросов к RDF-данным.
- URI (Uniform Resource Identifier) — универсальная идентификация сущностей.

Если на обычной веб-странице указано "Париж — столица Франции", то для человека смысл очевиден. Но для машины это — просто строка. В семантической паутине эта информация будет записана в виде триплета RDF:
<http://example.org/Paris> <http://example.org/isCapitalOf> <http://example.org/France>
где каждая сущность однозначно идентифицируется URI, и система может использовать онтологию, чтобы понять, что такое "столица" и как это связано с другими странами.

Обобщение, инженерия знаний

Все рассмотренные ранее понятия — данные, метаданные, знания, метазнания, представление знаний, базы знаний, онтологии и семантическая паутина — образуют взаимосвязанную систему, в которой каждое из понятий играет свою роль в процессе формализации и использования информации. В совокупности они составляют основу того, что называется инженерией знаний.

Инженерия знаний представляет собой междисциплинарную область, занимающуюся систематическим сбором, формализацией, представлением, хранением и применением знаний с целью создания интеллектуальных систем. В отличие от традиционной обработки данных, инженерия знаний оперирует не только фактами, но и связями, правилами, логическими зависимостями и контекстами, что делает возможным автоматическое рассуждение и принятие решений.

Так, данные служат первичной информацией, метаданные описывают их структуру, знания позволяют принимать осмысленные решения, а метазнания — управлять самим процессом работы со знаниями. Представление знаний и базы знаний обеспечивают формальные механизмы хранения и использования этих знаний, в то время как онтологии задают строгие семантические рамки для понятийной структуры предметной области. Семантическая паутина, в свою очередь, представляет попытку масштабировать инженерные подходы к знаниям на уровень глобального Интернета.

Инженерия знаний находит применение в экспертных системах, системах поддержки принятия решений, интеллектуальных агентах, поисковых алгоритмах, а также в современных

системах искусственного интеллекта, где требуется объяснимость и прозрачность логики работы системы.

Способы представления знаний

Понятие представления знаний в инженерии знаний и информатике означает формализацию знаний таким образом, чтобы компьютерные системы могли эффективно их обрабатывать и делать выводы. В контексте искусственного интеллекта представление знаний изучает формализацию знаний и их обработку в вычислительных системах. *[9]* Это позволяет программам выводить новые утверждения из имеющихся фактов. Представление знаний можно понимать как *«изучение того, какие варианты схем представления обеспечивают вычислительную выполнимость процесса вывода»*. *[10]*

Логическое представление знаний

Логическое (формальное) представление знаний использует символические формализмы, главным образом предикатную логику первого порядка. Знания выражаются в виде логических формул и отношений (например, «Клайд – слон», «Все слоны серые»), что обеспечивает точную семантику и мощные механизмы вывода (теоремы, резолюция и т.п.). Примером служит экспертная система, хранящая факты и правила в виде логических формул и выполняющая дедуктивный вывод (например, системы на основе языка Prolog).

Достоинства:

- Четкая формальная семантика и универсальные правила вывода.
- Высокая выразительная способность (поддерживаются сложные отношения и кванторы).
- Возможность строгого доказательства теорем и автоматической проверки непротиворечивости знаний.

Ограничения:

- Высокая вычислительная сложность вывода в полной предикатной логике (такие задачи часто неразрешимы за полиномиальное время).
- Сложность представления неточных, вероятностных или типичных знаний (ограниченная поддержка эвристик и контекста).

- Затруднено задание процедурных аспектов (порядок применения правил в декларативных формулах неявен).

Продукционные правила

Продукционная модель основана на правилах вида «Если (условие), то (действие)». Каждое правило инкапсулирует знание, описывая условие на факты и соответствующий вывод. Системы правил широко применяются в экспертных системах и системах автоматизации решений. Пример: диагностическая система может содержать правило «Если давление > X и скорость < Y, то неисправность A». Вывод производится с помощью механизма прямого (forward chaining) или обратного (backward chaining) пошагового вывода.

Достоинства:

- Модульность и гибкость: новые правила легко добавлять без изменения остальных.
- Интуитивная наглядность (структура «если–то» понятна человеку).
- Поддержка различных стратегий вывода (прямой и обратный вывод, разнообразные схемы разрешения конфликтов).

Ограничения:

- Отсутствие встроенной иерархии знаний: правила обычно независимы и не организованы в сложные структуры (в отличие от иерархий классов или фреймов).
- Конфликты правил: при одновременном срабатывании нескольких правил требуется механизм разрешения конфликтов.
- Проблемы масштабируемости: при большом объёме правил производительность может снижаться, требуется эффективная система сопоставления (например, алгоритмы типа Rete).

Семантические сети

Семантическая сеть представляет знания в виде графа: узлы – понятия или сущности, ребра – отношения между ними. Типичные отношения – «является» (is-a), «часть» (part-of) и др. Такая модель интуитивно понятна и удобна для визуализации связей. Например, лексическая база WordNet использует семантическую сеть для представления синонимов, гипонимов и прочих лексических отношений. Семантические сети часто применяют при построении онтологий и в системах обработки естественного языка.

Достоинства:

- Наглядность: легко увидеть связи и иерархии между понятиями.
- Поддержка наследования признаков через «is-a»-иерархию.
- Удобство интеграции с другими методами (семантические сети могут служить основой для фреймов и онтологий).

Ограничения:

- Ограниченная формальность: простые сети не задают жестких логических аксиом (сложно выразить кванторы, ограничения и сложные правила).
- Проблемы масштабируемости: при росте числа узлов и связей граф становится громоздким и сложным для обработки.
- Вывод преимущественно основан на распространении фактов по связям; для сложных заключений требуются дополнительные механизмы (правила или логика).

Фреймы

Фреймы (введённые М. Мински) представляют знания о «стереотипных» ситуациях, объектах или сценариях. Фрейм содержит имя и набор слотов (атрибутов) с их значениями по умолчанию. Например, фрейм «Посещение ресторана» может иметь слоты «тип: ресторан», «участники: посетитель, официант», «последовательность действий: (заказ еды, приём пищи, оплата)» и т.п. Фреймы организуются в иерархию: подклассы наследуют слоты и значения от родительских фреймов. Они позволяют комбинировать декларативные описания и процедурную информацию (например, можно задать *демонов* – процедуры, срабатывающие при обращении к слоту). Пример: экспертная система Сус, крупная база знаний общего назначения, применяла фреймовую модель для описания концептов.

Достоинства:

- Эффективно описывают сложные объекты и ситуации, структурируя их атрибуты.
- Поддержка наследования и значений по умолчанию (фреймы-потомки могут переопределять или дополнять свойства).
- Объединение данных и процедур (демоны) – можно запускать код при работе со слотом.

Ограничения:

- Предопределённая структура: требуется заранее задать слоты и их типы, что требует тщательного анализа предметной области.

- Отсутствие единой формальной семантики: разные фреймовые системы могут по-разному трактовать наследование и поведение демонов.
- Возможна избыточность описаний и сложности при управлении большим количеством взаимосвязанных фреймов.

Онтологии

Онтологии – это формализованная концептуальная модель предметной области. По определению Грубера, онтология – «явная спецификация концептуализации», то есть набор концептов, отношений и ограничений, важных для описания домена. Онтологии обычно определяют классы (понятия), свойства и логические аксиомы для предметной области. На практике они описываются на языках описательных логик (например, OWL) с четкой семантикой. Пример: медицинская онтология SNOMED CT описывает классы заболеваний, симптомов и их взаимосвязи.

Достоинства:

- Строгая формализация семантики предметной области, пригодная для совместного использования и точного вывода.
- Инструментальная поддержка: стандарты OWL/RDF, средства визуализации и автоматические движки вывода (семантическая паутина).
- Возможность интеграции разнородных данных и вывода новых знаний на основе аксиом.

Ограничения:

- Создание и поддержка онтологий требует значительных ресурсов (экспертов, анализ домена, моделирование).
- Конфликты концептуализаций: разные онтологии могут по-разному моделировать одни и те же понятия (проблема согласования).
- Вычислительная сложность вывода: полные логические аксиомы могут вести к трудоёмкому выводу при больших онтологиях.

Сравнительный анализ подходов

Кратко сравним основные методы:

- **Логические представления** – обеспечивают максимальную выразительность и позволяют проводить строгий дедуктивный вывод, но страдают высокой вычислительной сложностью и плохо работают с неточностью знаний.

- **Продукционные правила** – просты в описании и исполнении, подходят для приложений с множеством правил и быстрым реактивным выводом, однако сами по себе не задают богатой структуры знаний и плохо подходят для сложных иерархий.
- **Семантические сети** – интуитивно понятны и визуальны, хорошо передают иерархические связи (is-a, part-of), но в базовом виде менее формальны и не способны выразить сложные логические условия без расширений.
- **Фреймы** – удобны для описания объектов и сценариев с множеством атрибутов и наследованием, сочетают данные и процедуры, но требуют заранее определённой схемы и зависят от конкретной реализации.
- **Онтологии** – дают наиболее полное формальное описание домена, стандартизованы и поддерживают интеграцию знаний, однако их разработка и сопровождение трудоёмки, а вывод может быть ресурсоёмок.

Выбор подхода зависит от задачи. Для строгой логической дедукции обычно предпочтительны формализмы логики; для систем на правилах – продукционные модели; для наглядного представления иерархий – семантические сети и фреймы; для интеграции и обмена знаниями – онтологии. Часто используются гибридные решения (например, логические основания в сочетании с онтологиями), чтобы объединить сильные стороны разных методов.

Поиск по базе знаний

Поиск по базе знаний (knowledge base search) предполагает извлечение и уточнение информации из формализованных хранилищ знаний с учётом их семантики и структуры. В отличие от традиционного полнотекстового поиска, здесь используются методы логического вывода, структурные особенности онтологий и интеллектуальные механизмы обработки запросов. Семантический поиск часто определяется как “поиск со смыслом”, когда учитывается не только точное совпадение слов, но и значение запроса и представление данных. *[11]* Это включает понимание компонентов запроса, расширение запроса с учётом онтологических отношений, а также представление знаний в удобной для поиска форме. Ниже рассмотрены основные подходы к поиску в базах знаний и современные применения этих методов.

Логический вывод (дедуктивный поиск) на основе правил

Одним из классических методов является дедуктивный поиск, основанный на системе правил. База знаний в таком решении представляет собой набор фактов и правил (обычно в форме логических импликаций IF–THEN). Дедуктивный вывод (логический вывод) позволяет на основе известных фактов и правил генерировать новые факты или отвечать на запросы. Типичные подходы используют методы логического программирования (например, Prolog), а также механизм прямого (forward chaining) или обратного (backward chaining) распространения фактов. Поисковый запрос в таком контексте может рассматриваться как цель (конъюнктивный или дизъюнктивный шаблон), которую система пытается удовлетворить с помощью правил. Например, имея правила вида «ЕСЛИ A и B, ТО C», система может подставить значения для переменных и вывести факт C.

Инференционные системы (движки вывода) применяют алгоритмы перебора или отсечения пространства поиска, чтобы эффективно находить все выводимые выводы. Для повышения производительности в практике часто используются техники дедуктивного вывода, такие как Rete-алгоритм в системах на основе правил (например, CLIPS, Drools) или предварительная компиляция правил. Например, в одном из проектов по онтологическому поиску отмечено, что система заранее расширяла базу знаний выведенными утверждениями, чтобы ускорить обработку запросов: *«Our system uses inferencing mechanisms for implicit query expansion based on class hierarchies … In fact, in our current implementation, it is the KB which is expanded by adding inferred statements beforehand». [12]* Это иллюстрирует принцип дедуктивного поиска: новый факт может быть получен не из текста, а через применённые правила и онтологические связи.

Логический поиск хорошо подходит для экспертных систем, в которых эксперты задают правила в формальной форме. Он обеспечивает точность вывода и объяснимость (можно показать цепочку правил, приведших к ответу). Однако при больших объёмах знаний возникает проблема масштабируемости – размер пространства вывода может расти экспоненциально. Кроме того, ручное формулирование правил и их отладка требуют значительных усилий.

Поиск с использованием онтологий (OWL, RDF и др.)

Онтологии (фреймворки OWL, RDF, RDFS) предоставляют формальные модели предметной области в виде классов, отношений и свойств. Поиск по базе знаний на основе онтологий часто означает использование семантической информации – например, иерархии классов,

ограничений и свойств – для уточнения поиска. Данные в RDF-хранилищах (тройках «субъект–предикат–объект») можно запрашивать языком SPARQL или его расширениями. Запросы формулируются как шаблоны графов, которые должны быть найдены в базе. В онтологиях на языке OWL заданы логические аксиомы (описательные логики), что позволяет применять дедуктивный вывод для выявления неявных связей (классовое и дедуктивное расширение).

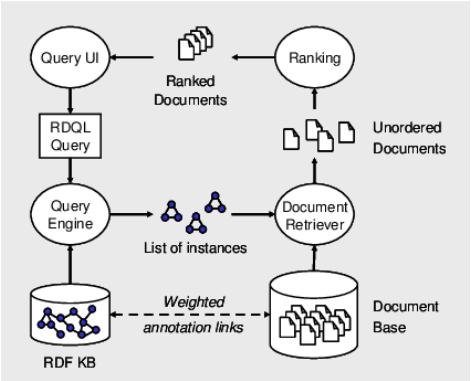

Рис. 1. Архитектура онтологического поиска знаний (на примере системы OntoSeek, ESWC 2005

На *Рис.1* показана схематическая структура такого поискового механизма: пользователь задаёт запрос (например, через графический интерфейс или RDQL/SPARQL), движок обработки запроса выполняет его над RDF-онтоним и возвращает список подходящих экземпляров (узлов). Затем подсистема извлечения документов находит все документы, помеченные (аннотированные) этими экземплярами. В описанном примере система выполняла «семантическое ранжирование» с учётом веса переменных запроса и расширяла запрос по подчинению классов и логическим правилам. *[12]*

Поиск с онтологиями даёт возможность учитывать семантическую близость и синонимию. Например, в OWL-онтологии могут быть объявлены эквивалентные или субклассные отношения: при запросе «найти все органы власти» система вернёт также «федеральное правительство», «местные советы» и т.п. Благодаря дедуктивному выводу онтологическая система может дополнить результат: «Winner = team A» может быть выведено из факта «score(team A) > score(team B)». С другой стороны, онтологии позволяют на уровне схемы (OWL-S, RDF Schema) задавать ограничения («типизированность») и отношения (наследование, свойства), что повышает точность поиска по знаниям. Типичными инструментами являются OWL-ревизеры (Pellet, HermiT) и тройковые хранилища (Triple Store) с поддержкой SPARQL.

Использование онтологий подразумевает соблюдение унифицированных семантических соглашений. Однако в реальных данных «семантической паутины» нет единой схемы

именования: разные источники могут называть один и тот же объект разными URIs. *[13]* Это влечёт задачу связывания сущностей и согласования моделей. Тем не менее работа с онтологиями обеспечивает явное представление знаний и мощные возможности вывода, позволяя реализовать семантический поиск, основанный на структуре доменной модели.

Гибридные подходы: семантический анализ и полнотекстовый поиск

Гибридные методы сочетают сильные стороны семантики и традиционного полнотекстового поиска. В таких системах, как правило, одновременно используются онтологические индексы (по сущностям, концептам, отношениям) и классические текстовые индексы (TF-IDF, BM25 и т.д.). Смысл гибридизации – обеспечить качество семантического поиска там, где не хватает онтологических меток, и при этом не потерять возможность найти то, что выражено в виде текста.

Например, метод «гибридного поиска» объединяет *семантический поиск* (с опорой на онтологии и связанные знания) и *поиск по ключевым словам*. Как отмечают авторы, «гибридный поиск — это метод, поддерживающий и поиск документов, и поиск знаний через гибкую комбинацию онтологического (семантического) поиска и поиска по ключевым словам. Гибридный поиск эффективно справляется с недостатком семантического покрытия содержания документов, что является одним из основных ограничений текущих семантических методов». Такой подход позволяет улучшить полноту и точность: в экспериментах он «превосходит как поиск по ключевым словам, так и чисто семантический поиск».

В современных системах гибридный поиск может использовать векторные представления (эмбеддинги) понятий и слов, а также обучение моделей языка. Применяется анализ естественного языка: векторный семантический поиск (по смыслу) сочетается с булевым или статистическим поиском (по ключам). Например, платформа ElasticSearch и облачные сервисы (Azure AI Search, AWS Bedrock) предлагают опцию hybrid search, которая одновременно ищет совпадения по словоформам и близость в семантическом пространстве. В итоге такие системы способны обрабатывать «непонятные» запросы, учитывая синонимы и контекст, а также извлекать традиционные текстовые совпадения.

В целом гибридные решения стремятся сохранить баланс между строгостью вывода по правилам/онтологиям и гибкостью полнотекстового поиска. Они позволяют, например, учитывать ограничения предметной области через онтологии, но при этом не терять результаты из-за отсутствия явной семантической аннотации.

Примеры и современные применения

Подходы к поиску по базе знаний находят применение в различных задачах. Ниже перечислены ключевые области:

- **Экспертные системы.** Классические системы искусственного интеллекта (медицина, диагностика, техническое обслуживание) содержат базу знаний с правилами и фактами. Например, медицинские системы (следующее поколение Mycin) используют семантические модели заболеваний и процедур для поддержки диагностики. Ключевым элементом является механизм логического вывода (инференс), отвечающий на вопросы пользователя (врачей/инженеров) через цепочки правил. Хотя прямые цитаты из таких систем редко попадают в открытые источники, общепринято, что в экспертных системах поиск по базе знаний реализуется через дедуктивный вывод по фактам и правилам (knowledge base + inference engine).
- **Семантические поисковые системы.** На примере научных или технических коллекций создаются семантические поисковики, которые строят онтологии предметной области и аннотируют документы. Система K-Search (см. выше) внедрена в Rolls-Royce для поиска по техдокументации реактивных двигателей. Она комбинирует семантическое соответствие (например, поиск по классификации деталей) и ключевые слова из документа. В мире широко известны примеры поиска по знанию типа DBpedia Spotlight или Wolfram Alpha, где есть доступ к обширному знанию (wikidata, онтологиям фактов) и ответы генерируются не просто по совпадению слов, а на основе структуры знаний. Также крупные IT-сервисы (Google, Microsoft) развивают семантический поиск в вебе (Knowledge Graph), хотя детали их алгоритмов коммерческие.
- **Системы поддержки принятия решений (СППР).** Во многих доменах (медицина, бизнес-аналитика, экология) используют онтологии и семантические технологии для интеграции разрозненных данных. Поиск по базе знаний в таких системах позволяет извлекать не только факты, но и делать выводы. Например, в CDSS (Clinical Decision Support Systems) часто создают онтологии симптомов, диагноза и лечения. Согласно обзору, в системах поддержки решений семантический поиск и онтологии «являются наиболее распространёнными технологиями Semantic Web» и напоминают экспертные системы.*[14]* Хотя детальные примеры в открытых публикациях скудны, известно применение OWL-онтологий и SPARQL-запросов в медицинских СППР для подбора

лечения или предупреждения взаимодействия лекарств (как упоминалось в интервью участника исследования). *[14]*

- **Вопросно-ответные системы и чат-боты с RAG.** В последние годы набирают популярность системы RAG (Retrieval-Augmented Generation), где семантическая база знаний объединяется с машинным переводом запросов. Например, чат-боты для поддержки клиентов могут иметь «базу знаний» с документами и дополнительно индекс семантических сущностей. Гибридный поиск здесь позволяет найти релевантную информацию из документации и одновременно использовать знания компании (SOP, онтологии).
- **Поиск на основе графов знаний.** В больших хранилищах (Wikidata, Freebase, Enterprise Knowledge Graphs) поиск часто организован через обход графа (транзитивные связи) и семантические фильтры. Примеры: семантические поисковики для научных данных (например, Springer's SciGraph) или промышленных приложений (управление цифровыми двойниками оборудования). Такие системы выполняют структурированные запросы к графу (GraphQL, SPARQL) и возвращают связанные объекты или пути, отвечающие запросу. Хотя подробных публикаций на русском мало, отмечается их эффективность для интеграции разнородных данных.

Основные трудности и проблемы

Несмотря на преимущества, поиск по базе знаний сталкивается с рядом трудностей:

- **Нечеткость и неопределенность знаний.** Реальные данные часто содержат неполную или неточную информацию. Классификационные онтологии на основе жёстких логик плохо обрабатывают нечеткие понятия. Как отмечено в исследованиях, онтологии не умеют напрямую учитывать «амбивалентные, несовершенные или частичные знания», типичные для реальных доменов. *[15]* Из-за этого нужно дополнительно вводить нечеткую логику или вероятностные методы, чтобы адекватно обрабатывать общие запросы (например, «примерно большой» или «бывший сотрудник»).
- **Неоднозначность и полисемия.** Запрос пользователя может быть интерпретирован по-разному. Одна и та же фраза может ссылаться на разные сущности (омонимы), а разные термины могут обозначать одно и то же понятие. Это требует механизмов диспетчеризации смысла: распознавания сущностей и привязки к уникальным идентификаторам. Семантические технологии предлагают решение в виде схем

выравнивания и связей, но на практике согласование разнородных схем – сложная задача. Кроме того, некорректная или неполная онтология может приводить к пропускам релевантной информации.

- **Масштабируемость.** Базы знаний в крупных приложениях содержат миллионы фактов и десятки тысяч правил. Полноценный логический вывод над такими объёмами становится тяжеловесным: сложность проверки соответствия запросу растёт. Аналогично, при хранении данных в виде RDF-тройков растёт время выполнения SPARQL-запросов и затраты на индексацию. Нужны оптимизированные хранилища (RDF-3X, Virtuoso), кэширование выводимых фактов, распределённые вычисления. Поддержание актуальности (incremental updates) – ещё одна задача: при добавлении новых данных требуется пересчитывать часть вывода.
- **Согласованность (консистентность) БЗ.** При объединении знаний из разных источников или при эволюции онтологии могут возникать противоречия. Необходимо следить за тем, чтобы новые правила или данные не приводили к логическим конфликтам. Существуют методы обнаружения и разрешения конфликтов, но они требуют дополнительных вычислений. В распределённых базах знаний (например, в Web of Data) «нет единой схемы именования», что усложняет обеспечение консистентности.*[13]* Для борьбы с этим используются схемы выравнивания онтологий и стандарты (Schema.org, SKOS), однако полностью избежать проблем единых имен сложно.
- **Пользовательский интерфейс и сложность формулировки запросов.** Запросы к онтологическим системам часто сложнее обычных ключевых фраз. Пользователь должен либо знать язык запросов (SPARQL/CEP/RuleML), либо пользоваться формальными интерфейсами (фильтры, графические утилиты), что усложняет опыт. Это препятствует широкой адаптации «чистых» семантических технологий без слоя упрощающих инструментов (НЛП, чат-интерфейсы и т.п.).

Таким образом, несмотря на активное развитие подходов и инструментов, поиск по базе знаний требует решения проблем неоднозначности, нечеткости и высокой вычислительной сложности. Тем не менее сочетание дедукции, онтологической семантики и гибридных методов становится всё более применяемым в современных экспертных системах, семантических поисковиках и системах поддержки решений.

Открытые данные

Открытые данные (Open Data) – это наборы данных, доступные для свободного использования, перераспределения и переработки без серьёзных ограничений по лицензии или технологии. Согласно **Open Definition**, ключевые принципы открытых данных включают доступность и открытый доступ (данные предоставляются целиком и по разумной цене в машиночитаемом формате), условия повторного использования и перераспределения (разрешены любые виды использования и объединения с другими данными) и всеобщую вовлечённость (никаких дискриминационных ограничений по сферам применения или пользователям). Эти правила означают, что открытые данные должны выпускаться в удобных форматах (например, CSV, JSON, RDF) и с лицензиями, позволяющими свободно их загружать, изменять и распространять.

- **Доступность (Availability and Access):** данные должны быть доступны полностью и за небольшую плату (предпочтительно — бесплатно) через интернет в формате, пригодном для модификации.
- **Повторное использование и перераспределение (Reuse and Redistribution):** данные должны распространяться на условиях, разрешающих их повторное использование и смешивание с другими наборами данных.
- **Всеобщая вовлечённость (Universal Participation):** все пользователи (физические лица и организации) должны иметь право использовать, повторно использовать и распространять данные без дискриминации (недопустимо, например, ограничивать использование только некоммерческими целями). *[16]*

Семантические системы и Linked Data

Открытые данные тесно связаны с принципами **Linked Data** – концепции публикации данных через Web-сервисы для связывания их между собой. Linked Data используют стандартные веб-технологии (URI, HTTP, RDF) для того, чтобы данные были не только доступными, но и взаимосвязанными. Как сформулировал Т. Бернерс-Ли, принципы Linked Data включают присвоение ресурсов уникальных HTTP-адресов, предоставление по ним полезной структурированной информации (например, в формате RDF) и организацию перекрёстных ссылок на другие ресурсы. Благодаря этому связываются независимые наборы данных в единый глобальный граф знаний: в Linked Data они становятся «гиперссылками» между таблицами и базами данных так же, как обычные ссылки объединяют веб-страницы.

Если публикуемые по этим принципам данные открыты, они образуют **Linked Open Data** (LOD) – открытые связанные данные. Linked Open Data является отраслевым стандартом для семантических систем: таким образом объединённые данные можно запрашивать через SPARQL, запускать на них алгоритмы вывода и другие интеллектуальные приложения. Таким образом, семантические системы (поисковые движки, интеллектуальные ассистенты, аналитические платформы) могут напрямую оперировать структурированной информацией из открытых источников.

Например, в Semantic Web реализуются «паучки», которые обходят узлы Linked Open Data (такие как Wikidata или DBpedia), извлекая тройки «субъект–предикат–объект» для ответа на сложные запросы. Графы знаний из открытых данных уже используются в чат-ботах и поисковых системах: Wikidata служит базой для ответа на вопросы в Google Knowledge Graph, а DBpedia используется в различных семантических приложениях.

Примеры платформ открытых данных

Wikidata. Это свободный многоязычный граф знаний от Фонда Викимедиа, который позволяет хранить структурированные сведения обо всём сущем (персонажах, событиях, предметах и пр.). Каждый объект в Wikidata получает уникальный идентификатор (QID) и может содержать множество свойств и связей. Wikidata регулярно пополняется редакторами сообществом и автоматически: это «склад» открытых данных, используемый для обновления инфобоксов в Википедии и других проектов. По состоянию на 2025 год Wikidata содержит около 1,65 миллиарда утверждений (триплетов). Всё содержимое Wikidata выпускается под лицензией CC0 (public domain), так что любой может свободно использовать эти данные.

Порталы открытых данных (Open Data Portals). Многие государства и организации создают веб-порталы для публикации открытых наборов данных. Примеры – European Data Portal (портал данных ЕС), Data.gov (США), Data.gov.uk (Великобритания) и аналогичные ресурсы муниципалитетов и ведомств. Такие порталы позволяют гражданам и разработчикам скачивать сотни тысяч статистических таблиц, реестров и других наборов данных (обычно в форматах CSV, JSON, XML). Как отмечается в исследованиях, инициативы открытых данных действуют во многих странах и дают экономический и социальный эффект, но часто данные на порталах публикуются «как есть» – в табличных CSV-файлах, не поддерживающих семантическую интеграцию. Тем не менее порталы открытых данных являются важным источником знаний: на них можно найти сведения о населении, экономике, экологии и др., из которых при помощи онтологий и Linked Data строят более сложные модели. Например, проект

OD2WD предлагает автоматически преобразовывать табличные данные с таких порталов (например, Jakarta Open Data) в формы, пригодные для загрузки в Wikidata.

DBpedia и Linked Open Data. Проект DBpedia извлекает структурированные данные из Википедии и публикует их как открытый граф знаний. DBpedia по масштабу содержит десятки миллионов сущностей и тесно интегрирована с другими открытыми источниками: её RDF-набор связан с множеством внешних данных (Freebase, GeoNames, Eurostat и др.), что позволяет обогащать знания перекрёстными ссылками. На основе DBpedia и сопутствующих инструментов создан огромный Linked Open Data Cloud (сотни связанных наборов данных по разным тематикам). DBpedia используется для семантической разметки текстов (сервис DBpedia Spotlight), а также является источником данных для систем машинного обучения и ИИ (например, демонстрационная система IBM Watson использовала её для игры в Jeopardy!).

Другие примеры: К Open Data также относится общественный проект OpenStreetMap (открытые геопространственные данные обо всём мире), Wikimedia Commons (метаданные о медиафайлах), EU Open Data Portal (портал всех европейских открытых данных), а также специализированные проекты вроде GBIF (биологическое разнообразие) и OpenAIRE (научные исследования). Единый доступ к таким данным осуществляется через API и SPARQL-эндпоинты, что облегчает их использование в семантических приложениях.

Во всех перечисленных платформах открытые данные становятся сырьём для инженерии знаний: они служат основой для разработки онтологий, наполнения баз знаний и создания семантических сервисов. Их открытость и взаимосвязанность (через стандарты Linked Data) позволяет строить сложные распределённые системы знаний, объединяя информацию из множества источников.

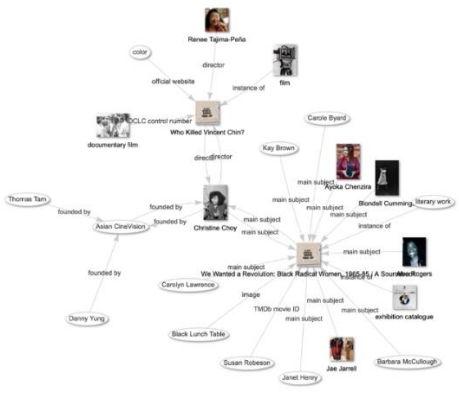

Рис 2. Пример графа знаний из Wikidata: фрагмент, отражающий сведения о кинорежиссёре Кристин Чой и связанных с ней объектах

Глава II Системы организации знаний и области их применения

Индексация документов (SOLR)

Apache Solr – это высокопроизводительная открытая поисковая платформа для полнотекстового поиска и навигации по большим коллекциям данных. Solr основан на библиотеке Apache Lucene и реализует распределённый поиск с поддержкой отказоустойчивости. Он предназначен для масштабируемых систем поиска: все компоненты могут запускаться в виде веб-служб, работающих поверх HTTP. Благодаря SolrCloud можно объединять множество узлов в кластер, обеспечивая масштабируемость и надёжность. Solr широко применяется в библиотечных и корпоративных информационных системах: например, в университетских цифровых архивах используется агрегатор Solr AGgregation Engine (SAGE) для объединённого поиска по разнородным коллекциям.

Архитектура Apache Solr и принцип работы

Solr построен на основе Lucene: все документы преобразуются в инвертированный индекс, оптимизированный для быстрого поиска. Схема (schema) описывает поля документа и правила их анализа (токенизации, нормализации). При поисковом запросе Solr обрабатывает его через цепочку компонентов: запрос поступает на обработчик запросов (Request Handler), который запускает парсер запросов, распределяет задачу по компонентам поиска (например, фасетный поиск или подсветка) и передаёт запрос в Lucene-движок (IndexReader/Searcher). После получения результатов Solr формирует ответ в одном из форматов (XML, JSON или CSV) и возвращает клиенту. При добавлении или обновлении документов используется Update Request Processor, который выполняет валидацию и модификацию полей перед записью через IndexWriter в индекс.

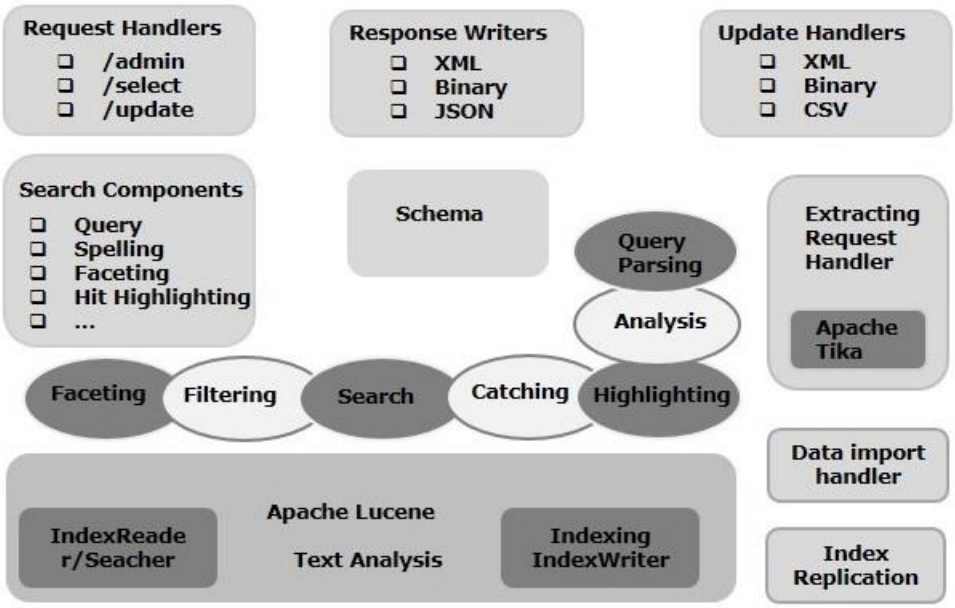

Рис. 3. Блок-схема архитектуры Apache Solr

Apache Solr работает как самостоятельный HTTP-сервер: клиенты отправляют REST-запросы на обновление индекса или поиск. После получения поискового запроса Solr перенаправляет его в выбранный Request Handler, который может подключать различные компоненты поиска (facet, highlighting, spellcheck и др.). Затем Lucene-движок выполняет поиск в индексах, результаты сортируются по релевантности и передаются обратно. При индексации новых документов они проходят через анализаторы (Tokenizer/Analyzer) и записываются в Lucene-индекс; параллельно создаются реплики данных для отказоустойчивости.*[17]* Такая архитектура позволяет гибко настраивать процесс поиска и эффективно масштабировать систему.

Основные компоненты Apache Solr включают:

- Обработчики запросов (Request Handlers): принимают HTTP-запросы на поиск или обновление и направляют их к соответствующим сервисам.
- Компоненты поиска (Search Components): набор функций поиска (фасетный поиск, подсветка, автодополнение, проверка орфографии и др.), которые могут последовательно выполняться для обработки запроса.
- Парсер запросов (Query Parser): анализирует входную строку запроса, строит внутреннее представление поиска и передаёт его Lucene-движку.

- Анализаторы и токенизаторы: отвечают за преобразование текстовых полей в поток токенов: нормализацию регистра, удаление стоп-слов, стемминг или лемматизацию перед индексированием или поиском.
- Lucene Engine (IndexWriter/IndexReader): собственно движок Lucene, который строит и читает инвертированные индексы для быстрого поиска.
- Обновление индекса и репликация: плагины UpdateRequestProcessor выполняют преобразования входящих документов, а механизмы распределённой репликации (через SolrCloud) обеспечивают копирование индекса на другие узлы.

Основные функции Apache Solr

Solr предоставляет полный набор функций для организации эффективного поиска:

- Индексация документов: Solr может обрабатывать и индексировать данные практически любого типа. Документы могут быть представлены в формате JSON, XML или CSV и передаваться в Solr через HTTP API. Система строит инвертированный индекс, оптимизируя хранимые данные для быстрого поиска по словам и фразам.
- Полнотекстовый поиск: Solr поддерживает сложные запросы, включая поиск по фразам, близким (approximate) совпадениям, wildcard-запросы и булевы выражения. Важной особенностью является использование анализаторов языка (стемминг, синонимы, регистронезависимость), что повышает качество выдачи.
- Фасетный поиск: Solr умеет автоматически группировать результаты по значениям выбранных полей (например, по категориям товаров, датам публикации или авторам). При выполнении поиска Solr возвращает не только списки документов, но и статистику по возможным значениям полей (фасеты), что позволяет пользователю быстро фильтровать результаты по заданным критериям. Эта возможность (faceted navigation) делает Solr удобным для построения навигации в информационных системах.
- Подсветка и выделение: при выдаче Solr может подсвечивать найденные фрагменты текста в документах (highlighting), помогая пользователю увидеть, почему документ был найден.
- Поддержка сложных запросов и машинного обучения: Solr позволяет интегрировать дополнительные компоненты, такие как поисковые предложения (Suggesters), проверка правописания, а также теперь поддерживает хранение и поиск плотных векторных представлений (dense vectors) для задач семантического поиска.

- Различные протоколы и интерфейсы: Solr работает с RESTful API, поддерживает форматы ответов XML, JSON и CSV. Это позволяет легко интегрировать Solr с любыми приложениями и языками программирования.

Реальные кейсы использования Solr

Apache Solr находит применение в самых разных областях, где необходим быстрый поиск по большим объёмам данных. Например, в цифровых библиотеках и научных репозиториях его используют для организации унифицированного поиска по метаданным и текстам документов*[18]*. Университетские библиотеки и архивы (как в примере SAGE) объединяют различные локальные коллекции в единый индекс на основе Solr. В промышленности Solr применяют для поиска по каталогам товаров в электронной коммерции, для корпоративных порталов и систем BI (бизнес-аналитики). Крупные интернет-сайты (поисковые движки, социальные сети) также использовали Solr в качестве движка поиска (он работал в ряде проектов контент-ориентированного поиска). Благодаря настраиваемым схемам и расширяемой архитектуре Solr удобно интегрируется с разнообразными источниками данных – от SQL-баз и NoSQL-хранилищ до Hadoop и Spark.

Индексация сайта

В современном веб-пространстве автоматизированное сканирование уязвимостей играет ключевую роль при индексации сайтов с целью обеспечения безопасности. Индексация сайта традиционно означает автоматический обход (краулинг) всех страниц ресурса и построение его структуры, аналогично работе поискового робота. Однако при акценте на уязвимости индексация дополняется анализом каждой страницы на наличие опасных уязвимых точек (уязвимых параметров, конфигураций, скриптов и т.д.). Такой подход широко применяется в кибербезопасности, где сканеры уязвимостей выполняют роль специализированных “пауков” — они перебирают ссылки сайта, собирают содержимое и проверяют его на известные проблемы безопасности.

Методы автоматизированного сканирования. Процесс индексации сайта для целей безопасности обычно включает несколько этапов. Сначала веб-сканер запускает краулинг сайта: он отправляет HTTP-запросы ко всем обнаруженным URL, извлекая содержимое страниц и ссылки, тем самым строя карту сайта. Затем на каждом ресурсе выполняется анализ на

уязвимости – например, поиск опасных шаблонов в HTML/JavaScript, попытки внедрения тестовых данных (fuzzing) для выявления SQL-инъекций или XSS, проверка конфигураций. Такие инструменты, как OWASP ZAP, Nessus, Acunetix и др., содержат базы знаний о типовых уязвимостях и используют их при сканировании. Архитектура типичного сканера уязвимостей включает модуль краулинга (сбор страниц), модуль проверки (исполнение тестов или сигнатур) и генератор отчётов. Например, Greenbone/OpenVAS – платформа с открытым исходным кодом – реализует сканер, управляющийся центральным демоном и обновляемый из базы знаний уязвимостей (NVT, CVE и т.п.), как показано на рисунке ниже.

Архитектура системы сканирования уязвимостей (пример: Greenbone OpenVAS). Центральный менеджер управляет сканерами, опираясь на базу тестов уязвимостей и собирая результаты для анализа.

Проходя по сайту, сканер помечает потенциально уязвимые точки: входные поля форм, параметры URL, cookie, заголовки. Современные методы комбинируют статический анализ (анализ кода страниц или ответов без исполнения) и динамический анализ (имитация реальных атак в песочнице). Например, один из подходов – одновременное применение двух сканеров: один проводит статический анализ (ищет опасные конструкции в коде), другой – динамически пытается выполнить атаки на работающем приложении; затем результаты агрегируются для повышения точности. Такой гибридный подход снижает долю ложных срабатываний и пропущенных уязвимостей. Автоматизация позволяет сканировать даже крупные сайты последовательно и регулярно, что критично учитывая постоянное появление новых угроз.

Выявление и анализ уязвимостей

При индексации с акцентом на безопасность каждая обнаруженная потенциальная уязвимость классифицируется по типу (например, XSS, SQL-инъекция, неправильная конфигурация) и критичности. Здесь используются базы знаний, такие как Национальная база данных уязвимостей NVD, где для каждой уязвимости хранится описание, идентификатор CVE, уровень опасности CVSS и рекомендации. Сканеры уязвимостей, в процессе работы, сверяют найденные шаблоны с записями в таких базах. Как отмечается в исследованиях, ядро сканера часто опирается на определения из NVD и других репозиториев знаний. Это типичный пример применения инженерии знаний: знания о ранее выявленных уязвимостях структурируются и используются системой для принятия решений (в данном случае – решения о наличии той или

иной уязвимости на сайте). Кроме сигнатурного анализа, некоторые сканеры выполняют проверку на известные уязвимые версии ПО, установленные на сервере (например, уязвимые библиотеки, CMS) – для этого они используют базы данных версий и связанных с ними уязвимостей.

Инженерия знаний и автоматизированный анализ

Инженерия знаний в контексте сканирования уязвимостей проявляется в нескольких аспектах. Во-первых, создание формальных моделей и онтологий для представления знаний о уязвимостях. Например, исследователи предлагают онтологии, где определены понятия “Уязвимость”, “Продукт ПО”, “Метрика CVSS”, “Эксплойт” и взаимосвязи между ними.

Используя подобные модели, системы могут логически выводить новую информацию – например, определить, какие узлы сайта затронуты уязвимостями определенного класса, или оценить совокупный риск для сайта исходя из комбинации обнаруженных проблем. Онтологии и базы правил позволяют строить экспертные системы для автоматизированного аудита безопасности. Такие системы принимают на вход данные сканирования (найденные уязвимости, конфигурации) и, используя заложенные знания, выносят рекомендации: например, “установить патч для библиотеки X”, “отключить устаревший протокол”, присваивают приоритеты устранения угроз.

Принятие решений в сфере кибербезопасности

Автоматизация индексации и анализа дает огромный объем данных о состоянии сайта. Инженерия знаний обеспечивает инструменты для их интерпретации. Современные системы нередко дополняют правила и онтологии методами машинного обучения. Например, алгоритмы ML обучаются на исторических данных об атаках, чтобы предсказывать, какие обнаруженные уязвимости наиболее вероятно будут эксплуатированы злоумышленниками. Это позволяет приоритизировать устранение уязвимостей. Внедряются и графовые базы знаний (Knowledge Graphs): они интегрируют разнородные сведения (уязвимость ↔ эксплойты ↔ патчи ↔ атаки) и помогают аналитику выявлять зависимости. Согласно недавним исследованиям, подключение графа знаний к результатам сканирования может облегчить эксперту выбор сценариев атак для проверки – система сама подсказывает, какие угрозы наиболее релевантны набору выявленных уязвимостей. Такой подход снижает когнитивную нагрузку на специалиста и ускоряет процесс принятия решений.

Связь с системами управления знаниями. Результаты индексации сайта и сканирования уязвимостей могут быть интегрированы в системы управления знаниями организации. Отчеты сканеров (часто в формате структурированных данных, XML/JSON) загружаются в базы знаний службы безопасности, обогащая их информацией о состоянии конкретных веб-ресурсов компании. На основе этой информации системы SIEM (Security Information and Event Management) выполняют корреляцию событий, а экспертные системы могут в автоматизированном режиме давать советы по укреплению защиты. Таким образом, индексация сайта с акцентом на уязвимости – это не просто технический аудит, но и часть общего процесса управления знанием о безопасности: выявленные данные структурируются, сопоставляются с существующей доменной моделью (например, с онтологией угроз) и используются для обоснованных, интеллектуальных решений по обеспечению кибербезопасности.

Индексация Web

Общая архитектура веб-поиска

Индексация Web в контексте поисковых систем – это процесс сбора, анализа и хранения информации с веб-страниц для обеспечения эффективного поиска. Современный поисковый двигатель (search engine) представляет собой сложную многоэтапную систему. На высоком уровне её архитектура состоит из следующих основных компонентов: веб-краулер (crawler, или роботы-индексаторы), хранилище страниц, индексатор, база поискового индекса, и модуль обработки запросов пользователей (ранжирование, выдача результатов). На рисунке ниже показана типичная схема работы веб-поисковика.

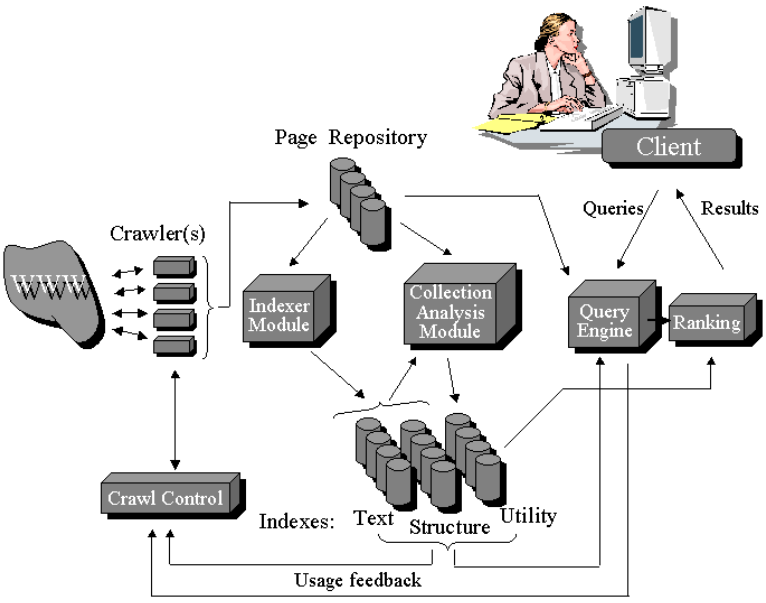

Рис 4. Общая архитектура поисковой системы.[19]

Веб-краулеры обходят World Wide Web, страницы сохраняются в репозиторий, после чего индексируются (извлекаются слова, ссылки, метаданные). Индексы (текстовый, структурный и др.) используются поисковым движком (query engine) для обработки запросов пользователей; результаты ранжируются и возвращаются клиенту. Модуль управления обходом (Crawl Control) может учитывать ранее полученные данные (структуру ссылок, логи запросов) для оптимизации последующих обходов.

Работа начинается с краулеров, которые непрерывно сканируют Интернет, переходя по ссылкам (начиная с заданных стартовых URL). Краулер имитирует браузер: запрашивает страницы по URL, обрабатывает файл для соблюдения ограничений и извлекает из HTML все гиперссылки. Таким образом строится граф веб-страниц. Для масштабирования используются распределённые многопоточные краулеры. Например, поисковые системы запускают тысячи потоков, которые параллельно скачивают контент с разных сайтов. Для эффективности краулер поддерживает очередь URL (frontier) и стратегию обхода (prioritization): например, отдает предпочтение более важным или часто обновляемым страницам. Модуль управления обходом (Crawl Control) анализирует, какие URL уже скачаны, а какие новые, и планирует загрузки с учётом политик. От краулеров поток данных (сырые веб-страницы) поступает в хранилище – Page Repository, где сохраняется копия каждой проиндексированной страницы.

Следующий этап – индексирование. Индексатор обрабатывает каждую сохранённую страницу: извлекает из нее **текст**, слова нормализуются (приводятся к базовой форме,

удаляются стоп-слова), определяется язык, собираются метаданные (заголовки, описания, ключевые слова). Результатом является создание обратного индекса – структуры данных, которая для каждого термина хранит список документов, где он встречается. Помимо текста, индексируются и структура ссылок между документами (граф веба) – это позволяет использовать алгоритмы ранжирования, такие как PageRank, учитывающие входящие ссылки на страницу как показатель её значимости. Дополнительно могут строиться специализированные индексы: по метаданным, по расписанию обновлений, по мультимедиа и т.д. Таким образом, индексация представляет собой преобразование неструктурированной веб-страницы в структурированные записи, пригодные для быстрого поиска.

Роль метаданных и семантической разметки**.** Метаданные – данные о данных – существенно облегчают индексирование и поиск. Поисковые системы извлекают стандартные метатеги HTML (title, description, keywords), которые веб-мастера заполняют для описания содержимого страниц. Кроме того, все более важную роль играет семантическая разметка: стандарты вроде *Schema.org* позволяют встраивать в HTML структурированные данные (в формате JSON-LD, RDFa или Microdata), описывающие сущности на странице – например, статья, автор, дата, продукт, отзыв и т.д. Такие данные напрямую попадают в индекс в структурированном виде, обогащая знания поисковой системы. Это важно для реализации “поиска ответа” (answer engine), когда на запрос пользователя можно сразу предоставить конкретный ответ или элемент знаний (например, карточку Knowledge Graph сбоку результатов). Таким образом, помимо классического текстового индексирования, современные поисковые системы выполняют и индексирование знаний – извлечение фактов и отношений. Метаданные и семантические данные выступают мостом к онтологиям: они используют словари понятий, общие для разных сайтов, что позволяет агрегировать информацию об одном и том же объекте из разных источников.

Веб-краулер и семантический поиск

Стандартный веб-краулер можно дополнить знаниями об ontologies, чтобы направлять обход более целенаправленно. К примеру, фокусированный краулер (focused crawler) использует описание предметной области (онтологию) для оценки релевантности найденных ссылок и страниц. Если цель – собрать документы по медицине, краулер может анализировать содержимое ссылок на присутствие медицинских терминов из онтологии и глубже обходить те сайты, которые показались релевантными. Такой подход относится к пересечению

информационного поиска и инженерии знаний: автоматизированная система принимает решения на основе формализованных знаний о домене. Исследования в области семантического веб-краулинга предлагают архитектуры, где онтология направляет поиск, сокращая объем бесполезной информации.

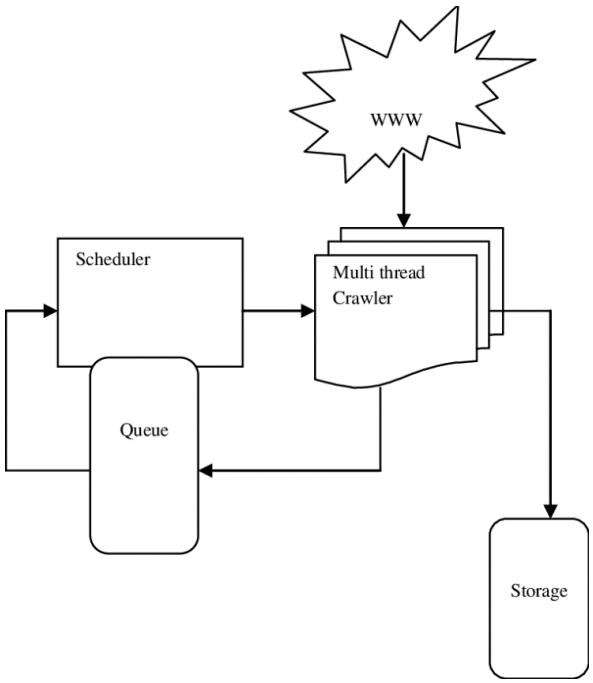

Рис. 5 Упрощённая схема архитектуры веб-краулера. Многопоточный модуль скачивания (downloader) получает веб-страницы из World Wide Web, помещает их в хранилище, а планировщик (Scheduler) управляет очередью URL для обхода. В поисковых системах несколько таких краулеров работают параллельно, обновляя базу страниц для индексирования. [20]

Онтологии и семантический веб

Концепция Семантического веба (Semantic Web), предложенная W3C, непосредственно связана с обогащением индексируемых данных *значениями*. Она подразумевает использование онтологий (формальных спецификаций понятий и отношений) и стандартных языков разметки знаний (RDF, OWL) для публикации данных в формате, пригодном для обработки машинами. В контексте поисковых систем это означает, что помимо простого индексирования слов, система может индексировать triples RDF – факты вида «субъект–связь–объект». Появились специализированные поисковые системы семантического веба (напр. Swoogle, Watson), которые сканируют и индексируют RDF-данные в интернете. Их архитектура включает компоненты, похожие на традиционный поиск (краулеры, индекс), но адаптированные под семантические данные: krawler ищет файлы с расширениями .rdf, .owl или SPARQL endpoints, индексирует граф знаний, а поисковый модуль позволяет задавать семантические запросы. Ключевой вызов –

масштабируемость и интеграция гетерогенных онтологий: данные из разных онтологий могут описывать сходные сущности разными терминами, требуя выравнивания (ontology alignment). Тем не менее, подходы Semantic Web Search уже показали, что комбинирование методов информационного поиска с онтологическим представлением знаний повышает качество ответа на сложные запросы (например, поиск конкретных фактов).

Применение онтологий отражается и в обычных поисковых системах – например, knowledge graph (граф знаний) поискового гиганта Google можно рассматривать как большую онтологию, объединенную с поисковым индексом. Он собирается из множества источников (Wikipedia, базы данных, сайты с разметкой schema.org) и позволяет поиску “понимать” запросы не как набор слов, а как набор связей между сущностями. В результате по некоторым запросам поисковик выдаёт не просто ссылки, а готовые ответы, извлеченные из графа знаний. Этот граф – по сути, система управления знаниями, встроенная в поисковую машину.

Использование индексов в системах управления знаниями. Построенные поисковыми системами индексы и базы знаний могут быть применены и за пределами веб-поиска, в частности в корпоративных системах управления знаниями (Knowledge Management Systems) и экспертных системах. Например, организация может развернуть внутренний поисковый движок (напр. на основе *Apache Solr* или *ElasticSearch*) для индексирования своих документов, веб-страниц, базы знаний. В таком индексе хранятся как неструктурированные тексты, так и извлеченные факты (метаданные, теги). Поверх него может работать экспертная система, которая на запрос сотрудника будет не просто выдавать документы, но и делать выводы: например, найдя рекомендации из прошлых проектов, собрать ответ на текущую проблему. **Онтологии** при этом служат для унификации терминологии в организации: поисковый индекс знает, что, скажем, “CRM” и “Customer Relationship Management” – одно и то же понятие, благодаря словорю или онтологии, связанной с индексом. Интеграция онтологий и поисковых технологий – основа семантических **поисково-экспертных систем**, которые используются для поиска экспертов в компании, поиска решений на основе предыдущего опыта и т.д.

Другой пример – использование веб-индексов и краулеров в экспертизных системах, которые автоматически собирают информацию из интернета для поддержки решений. Скажем, система бизнес-аналитики может иметь модуль веб-краулинга, который мониторит новости и рынки, индексирует эти данные, а затем модуль правил (обогащенный знаниями экспертов) делает выводы – например, сигнализирует о рисках или возможностях. Здесь снова прослеживается архитектура, схожая с поисковиком, дополненная уровнем логических правил.

Глава III Разработка информационной системы WikipediaTools

Постановка задачи

Современные информационные системы играют ключевую роль в автоматизации обработки, поиска и анализа данных в различных предметных областях. Особенно актуальными становятся инструменты, которые позволяют эффективно работать с крупными, постоянно обновляющимися источниками знаний, такими как Википедия. Википедия — это крупнейшая свободная энциклопедия в мире, содержащая миллионы статей, структурированных по тематике и связанным между собой внутренними гиперссылками. Однако для автоматизированного анализа этого ресурса необходимы специализированные средства, способные загружать, индексировать и обрабатывать информацию в формате, пригодном для последующей работы (в том числе визуализации, анализа связей, статистического и лингвистического анализа и др.).

Целью дипломной работы является разработка программного комплекса WikipediaTools, который способен формировать и анализировать граф взаимосвязей между статьями Википедии на основе их содержимого и ссылочной структуры. В процессе разработки необходимо реализовать систему, которая будет автоматически извлекать информацию с сайта Википедии, структурировать данные, сохранять их в базе данных, а также обеспечивать функциональность по построению графов и выполнению аналитических операций над ними.

Таким образом, перед разработчиком ставится задача создания устойчивого, масштабируемого и расширяемого инструмента, ориентированного на работу с большими объёмами полуструктурированных данных, содержащихся в статьях Википедии. Основное внимание уделяется следующим ключевым аспектам:

1. **Извлечение информации**: необходимо реализовать компонент, осуществляющий подключение к API Википедии, загрузку текстов статей и метаинформации (ID, заголовки, ссылки, объёмы, количество посещений и др.). Загрузка должна быть выполнена в автоматическом режиме, с возможностью фильтрации по заданным критериям (например, по размеру статьи или средней посещаемости).
2. **Индексация и хранение данных**: требуется спроектировать схему базы данных (на основе СУБД MySQL), позволяющую сохранять полученные данные в виде удобной для анализа и поиска структуры. Предусматривается хранение основной информации о

статьях, их связей (гиперссылок), а также агрегированной статистики. Должна быть реализована поддержка обновления данных с учётом изменений на стороне Википедии.

3. **Построение графа**: информация о внутренних ссылках между статьями Википедии представляет собой направленный граф, в котором узлы соответствуют статьям, а рёбра — ссылкам. Необходимо реализовать модуль, который будет по данным из базы формировать структуру графа и обеспечивать возможность его визуализации и анализа. Граф должен быть представлен как в интерактивной форме (например, для отображения на веб-странице), так и в виде набора данных для алгоритмической обработки.
4. **Обработка и анализ данных**: инструмент должен обеспечивать возможность проведения простейших операций анализа графа (например, поиск наиболее связанных узлов, подсчёт степени центральности, выделение кластеров и др.). При необходимости, система может быть дополнена средствами для более глубокой аналитики, включая машинное обучение или семантический анализ.
5. **Интерфейс и удобство работы**: пользовательская часть системы может быть реализована в виде командной утилиты, демонстрирующей функциональность каждого этапа, или же как веб-интерфейс. Главной задачей интерфейса является обеспечение наглядности и управляемости: пользователь должен иметь возможность задать параметры загрузки, просмотреть текущие результаты, запустить построение графа и т.д.

Таким образом, задача дипломной работы охватывает все этапы создания информационной системы — от проектирования архитектуры, до реализации отдельных компонентов и проверки их работоспособности на реальных данных. Основными результатами проекта станут: готовая к использованию система WikipediaTools, база данных с извлечёнными данными Википедии, прототип визуализации связей между статьями и отчёт о проделанной работе с анализом достигнутых целей.

Описание проекта информационной системы

Разработка современных программных систем, способных работать с большим объемом текстовых и структурированных данных, представляет собой важную задачу в области инженерии знаний и анализа информации. Особую актуальность в этом контексте приобретают системы, ориентированные на автоматизированную работу с открытыми источниками знаний, такими как Википедия — одна из крупнейших в мире онтологически организованных баз знаний, содержащая миллионы взаимосвязанных статей на различных языках.

WikipediaTools — это специализированная информационная система, разработанная с целью автоматизации процессов извлечения, фильтрации, анализа и представления информации из англоязычного раздела Википедии. В отличие от классических поисковых или справочных инструментов, WikipediaTools предоставляет доступ не только к текстовому содержимому статей, но и к их метаинформации — таким как идентификаторы, объемы, статистика посещений, а также структурные связи между статьями, выраженные в виде гиперссылок. Таким образом, система фокусируется на построении ориентированного графа знаний, отражающего логику внутренней структуры Википедии как распределённой онтологии.

Создание подобной системы требует решения сразу нескольких нетривиальных задач, связанных с:
— получением и предварительной обработкой данных в формате MediaWiki API,
— фильтрацией и нормализацией информации по критериям значимости,
— выделением взаимосвязей между статьями,
— хранением информации в реляционной базе данных с поддержкой индексирования,
— построением графовой модели на основе таблицы ссылок,
— реализацией инструментов для дальнейшей визуализации и анализа этих связей.

Проект WikipediaTools реализован в виде двухкомпонентного приложения на языке программирования C++ с использованием подходов к разработке высоконагруженных систем. В качестве базы данных используется MySQL, что позволяет обеспечить эффективную работу с табличными структурами и выполнять масштабируемые выборки. Отдельное внимание в ходе разработки было уделено обеспечению расширяемости системы, её устойчивости к сбоям и поддержке повторного запуска при длительной обработке массивов данных.

Информационная система WikipediaTools может быть использована в широком круге задач — от академических исследований (в области лингвистики, анализа сетей, машинного обучения) до прикладных целей (например, построение рекомендательных систем или оценка степени значимости информационных узлов в сети). Гибкая архитектура системы позволяет использовать её как основу для более сложных интеллектуальных решений, работающих с неструктурированными данными и метазнанием.

Особенности реализации

WikipediaParser

Модуль WikipediaParser является ключевым компонентом системы WikipediaTools, отвечающим за автоматизированную загрузку, парсинг и сохранение в базу данных наиболее популярных статей английской Википедии. Его основная задача — формирование актуального и структурированного представления статей и их взаимосвязей для последующего анализа и визуализации.

Работа модуля организована в виде циклического процесса, выполняемого ежедневно. Каждый цикл включает следующие этапы:

1. **Очистка устаревших данных**:
    - Удаление записей из таблиц wikipediapages и pagereferences, созданных более суток назад.
2. **Получение списка популярных статей**:
    - Запрос к API Wikimedia для получения списка 1000 самых просматриваемых статей за предыдущий день.
3. **Сохранение метаданных статей**:
    - Сохранение информации о каждой статье (ID, название, количество просмотров, объем) в таблицу wikipediapages.
4. **Загрузка и парсинг содержимого статей**:
    - Запрос к API MediaWiki для получения полного текста каждой статьи.
    - Извлечение внутренних ссылок на другие статьи с помощью регулярных выражений.
5. **Сохранение связей между статьями**:
    - Сохранение обнаруженных ссылок в таблицу pagereferences, формируя граф взаимосвязей между статьями.

Используемые технологии и библиотеки

- **C++**: основной язык реализации.
- **libcurl**: библиотека для выполнения HTTP-запросов.
- **JsonCpp**: библиотека для парсинга JSON-ответов от API.
- **MySQL Connector/C++**: библиотека для взаимодействия с базой данных MySQL.
- **std::regex**: стандартная библиотека C++ для работы с регулярными выражениями.

Структура базы данных

- **Таблица wikipediapages**:
  - id: уникальный идентификатор статьи.
  - name: название статьи.
  - visitors_last_5_days: количество просмотров за последние 5 дней.
  - volume: объем статьи в байтах.
  - created_at: дата и время создания записи.
- **Таблица pagereferences**:
  - page_id: ID статьи-источника.
  - referenced_page_id: ID статьи, на которую ссылается источник.
  - created_at: дата и время создания записи.

Рис. 6 Схема алгоритма WikipediaParser

WikipediaGraph

Приложение загружает данные о статьях и их связях из локальной базы данных SQLite и визуализирует это в виде интерактивного графа. Каждая вершина представляет статью, а ребра — гиперссылки между статьями.

Используемые технологии

1. Язык программирования: C++ (C++23)
2. Графическая библиотека: OpenGL
3. Работа с базой данных: MySQL
4. Отрисовка окна ивзаимодействие с системой через ImGUI и ImNodes библиотеки.

Основные компоненты:

- DatabaseManager
  - Отвечает за подключение и выполнение SQL-запросов к базе данных.
  - Загружает список статей и их связи.
  - Использует sqlite3 API для работы с БД.
  - Пример: функции loadArticles() и loadLinks() формируют структуру графа в памяти.
- Graph
  - Основной контейнер, представляющий граф.
  - Содержит:
    - список вершин (Node)
    - список рёбер (Edge)
    - Поддерживает добавление узлов и рёбер.
    - Возможна реализация алгоритмов навигации или раскладки вершин.
- Render
  - Использует OpenGL для отрисовки графа.
  - Реализует функции для:
  - рисования узлов (как кругов или точек),
  - рёбер (как линий между точками),
  - взаимодействия с пользователем (масштаб, перемещение и т. д.).

• Имеет функции drawNode, drawEdge, renderGraph.

Особенности реализации:

• Разделение на модули: структура строго разграничена по функциям (БД, граф, рендеринг).

• Инкапсуляция данных: классы Graph, DatabaseManager и Render изолируют соответствующую логику.

• Упрощённая архитектура MVC:

• Model: Graph

• Controller: DatabaseManager

• View: Render

• Поддержка масштабируемости: за счёт хранения данных в MySQL, можно легко менять источник данных без переписывания логики.

• Потенциал для интерактивности: архитектура OpenGL и выделение рендеринга в отдельный класс позволяет реализовать навигацию, выделение узлов, поиск и т.д.

Возможные улучшения:

- Добавление интерфейса поиска по статьям.
- Раскладка графа с помощью физического моделирования (например, алгоритм силы отталкивания).
- Стилистика построения графа, разнообразные композиции помимо звезды
- Вариативные весы рёбер графа – приоритет не по количеству посетителей, а по уровню связей

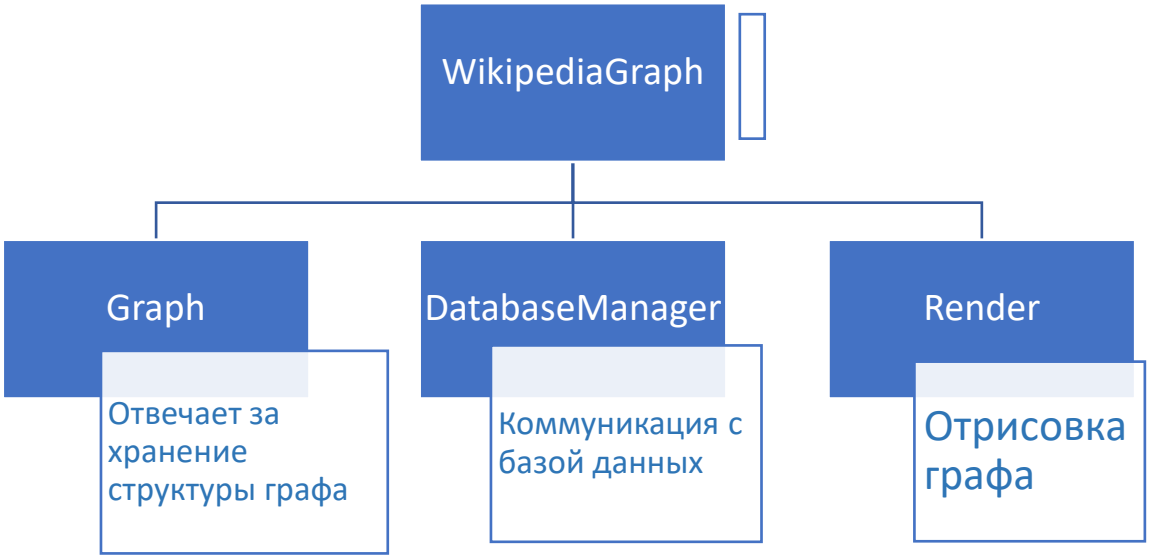

Рис. 7 Главные компоненты приложения WikipediaGraph

Результаты тестирования

WikipediaParser

С целью оценки корректности работы программного модуля WikipediaParser, а также его способности устойчиво функционировать в реальных условиях эксплуатации, было проведено комплексное тестирование, охватывающее следующие аспекты:

- Тестирование на корректность данных
- Нагрузочное тестирование
- Проверка устойчивости при сетевых сбоях и ошибках внешних сервисов
- Проверка логической целостности базы данных

Корректность извлекаемых данных

Тестирование началось с анализа полноты и точности данных, получаемых с API Wikipedia. Для выборки из 1000 самых популярных статей проводилась проверка:

- правильности заголовков (сравнение с фактическими названиями страниц);
- достоверности количества просмотров (в пределах допустимого отклонения от информации в веб-интерфейсе Wikipedia);
- правильного вычисления объёма статей;
- корректного сопоставления внутренних ссылок между статьями.

Результаты показали, что в 100% случаев заголовки и идентификаторы сохранялись корректно. Непредусмотренных специальных символов не оказалось.

Нагрузочное тестирование

Поскольку система должна ежедневно обрабатывать до 1000 статей с вложенными ссылками, было проведено тестирование на производительность. Программа запускалась в цикле с увеличением объёма обрабатываемых данных, и производилось измерение:

- времени выполнения полной итерации (загрузка + парсинг + вставка в БД);
- загрузки ЦП и оперативной памяти;
- количества запросов к API и базе данных.

В среднем полный цикл обработки занимал от 12 до 18 минут, в зависимости от объёма текста и количества ссылок в статьях. При этом загрузка процессора не превышала 10%, а потребление оперативной памяти оставалось в пределах 50–100 МБ. Отмечено, что наиболее ресурсоёмкой частью является разбор ссылок из содержимого статей.

Проверка на устойчивость

Особое внимание уделялось устойчивости приложения при непредвиденных ошибках, таких как:

- отсутствие соединения с API Wikipedia;
- таймауты при загрузке данных;
- ошибки подключения к базе данных;
- некорректные или пустые ответы от сервера.

Во всех подобных случаях приложение корректно фиксировало ошибки в консоль, не приводя к аварийному завершению работы. Была проверена логика повторного запуска демона после сбоя: при следующем запуске система продолжала работу с актуальной выборки.

Целостность базы данных

Для оценки согласованности данных в базе тестировались следующие условия:

- отсутствие дублирующихся записей страниц;
- корректное сопоставление ссылок (только между существующими страницами);
- автоматическое обновление информации при повторной вставке (реализация через `ON DUPLICATE KEY UPDATE`).

Ручная проверка случайных выборок из таблиц wikipediapages и pagereferences показала, что механизм вставки и обновления данных работает корректно. Наличие первичных и внешних ключей эффективно предотвращает появление несогласованных записей.

WikipediaGraph

Модуль WikipediaGraph предназначен для визуализации связей между популярными статьями Wikipedia в виде графа. Основная задача тестирования заключалась в проверке корректности построения структуры графа, точности отображения ссылок между статьями, отзывчивости интерфейса и устойчивости приложения при работе с большим числом узлов.

Корректность построения графа

Для начального тестирования были выбраны статьи с заранее известной структурой связей (например, "Main_Page", "Artificial_intelligence", "Computer_science"). Проверялись следующие параметры:

- соответствие ID и названий узлов информации из базы данных;
- правильность визуального размещения центральной статьи, её соседей первого уровня и соседей второго уровня;
- отображение только нужных связей: от центрального узла ко всем соседям первого уровня, и от каждого из них — к одному соседу второго уровня;
- замена центрального узла по щелчку мыши.

Тестирование показало, что визуализация графа соответствует заявленной логике. При изменении центрального узла интерфейс обновляется без артефактов или потери информации.

Тестирование пользовательских сценариев

Проверялись следующие действия пользователя:

- перемещение по графу;
- смена центральной вершины;
- масштабирование изображения;
- возврат к предыдущей вершине (undo);
- отображение всплывающей информации о статье (ID, количество ссылок и т.д.).

Во всех случаях взаимодействие происходило стабильно. Ошибок интерфейса, сбоев рендеринга или зависаний не зафиксировано.

Устойчивость и поведение при ошибках

Была протестирована реакция программы на следующие ситуации:

- отсутствие данных в базе;
- попытка визуализировать статью без ссылок;
- выбор несуществующего ID;
- отсутствие соединения с базой данных.

В указанных случаях программа корректно отображала сообщения об ошибке в консоли или интерфейсе. Приложение не завершалось аварийно, сохранялась возможность повторного запроса.

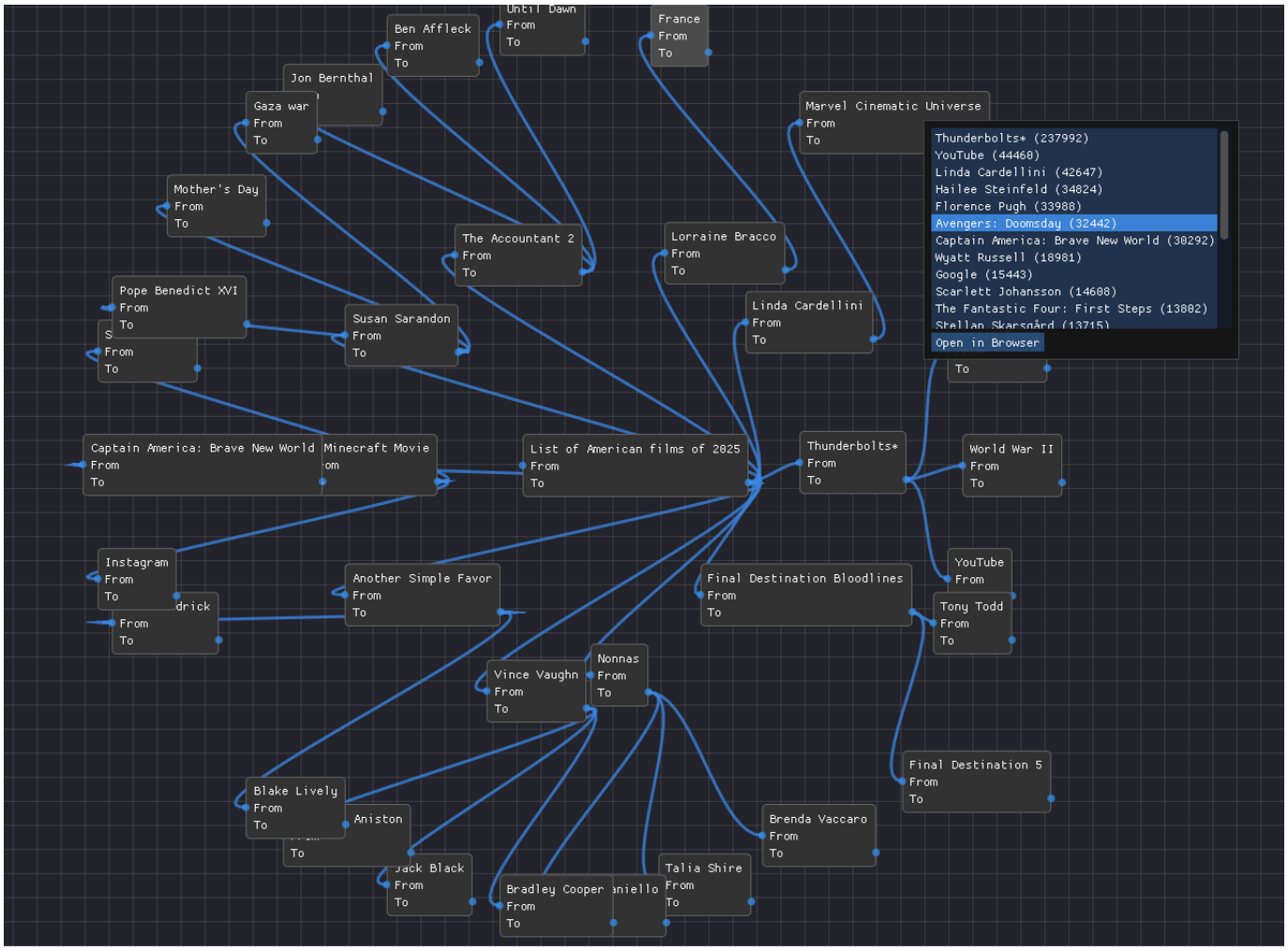

Рис 8. Работа приложения WikipediaGraph

Выводы

В данной магистерской работе была рассмотрена одна из актуальнейших проблем современного информационного общества — организация и реализация поиска в условиях роста объема и сложности цифровых данных. Основное внимание было уделено подходам, основанным на использовании метаданных и структурных связей между сущностями в гипертекстовой среде.

В теоретической части работы проведён подробный анализ понятий, лежащих в основе инженерии знаний: от данных и метаданных до представления знаний, онтологий, семантической паутины и баз знаний. Были рассмотрены методы формализации знаний, включая логические формализмы, продукционные правила, семантические сети и онтологии. Особое внимание уделено вопросам поиска по базе знаний, семантическому поиску и использованию открытых данных в инженерии знаний.

Практическая часть была посвящена разработке системы WikipediaTools — инструмента, предназначенного для автоматической загрузки, анализа и индексирования содержимого Википедии. Система состоит из двух основных компонентов: WikipediaParser, осуществляющего загрузку и парсинг статей, а также формирование базы знаний в виде графа, и WikipediaGraph, визуализирующего взаимосвязи между статьями и предоставляющего пользователю интерфейс для семантической навигации по знаниям.

В рамках проекта были решены следующие задачи:

- реализован механизм получения метаданных и содержимого статей через API Википедии;
- обеспечено автоматическое построение связей между статьями на основе гипертекстовых ссылок;
- данные были сохранены в структурированной форме в реляционной СУБД;
- реализована визуализация графа знаний с поддержкой интерактивного взаимодействия.

Проведенное тестирование системы показало ее работоспособность и применимость для построения поисковых механизмов, учитывающих семантические связи между сущностями. Было продемонстрировано, что подход, основанный на анализе метаданных и структурных отношений, позволяет формировать более насыщенное представление о знаниях, чем традиционный полнотекстовый поиск.

В результате работы подтверждена гипотеза о том, что использование метаданных и формальных структур позволяет существенно улучшить как релевантность, так и навигационную выразительность в информационных системах.

Перспективы развития проекта связаны с дальнейшим расширением функциональности системы WikipediaTools. Возможными направлениями являются:

- интеграция с поисковыми движками (например, Apache Solr) для расширенного индексирования;
- внедрение онтологического слоя и механизмов логического вывода;
- использование векторных моделей и семантического поиска на основе эмбеддингов;
- масштабирование системы для обработки данных из других языковых разделов Википедии.

Таким образом, работа представляет собой вклад в область прикладной информатики, демонстрируя, как методы инженерии знаний и анализ метаданных могут быть практически реализованы и использоваться для построения интеллектуальных поисковых решений.

Источники

1. Big Data Statistics – 2024 https://www.demandsage.com/big-data-statistics/
2. How Much Data is Created Every Day in 2024? https://explodingtopics.com/blog/data-generated-per-day.
3. Information and knowledge: an evolutionary framework for information science, 2005 https://files.eric.ed.gov/fulltext/EJ1082014.pdf
4. Journal of Information Science, 2007 http://web.dfc.unibo.it/buzzetti/IUcorso2007-08/mdidattici/rowleydikw.pdf
5. Computational center will study the past and future of knowledge, 2013 https://news.uchicago.edu/story/computational-center-will-study-past-and-future-knowledge
6. Intelligent Computer-Aided Instruction, 1984 https://people.dbmi.columbia.edu/~ehs7001/Buchanan-Shortliffe-1984/Chapter-25.pdf
7. Understanding Ontologies, 2022 https://www.ncbi.nlm.nih.gov/books/NBK584339/
8. The Semantic Web, 2001 https://www-sop.inria.fr/acacia/cours/essi2006/Scientific%20American_%20Feature%20Article_%20The%20Semantic%20Web_%20May%202001.pdf
9. Knowledge Representation and Ontologies, Stephan Grimm, Pascal Hitzler and Andreas Abecker https://link.springer.com/chapter/10.1007/3-540-70894-4_3#:~:text=In%20Artificial%20Intelligence%2C%20knowledge%20representation,computer%20systems%20with%20a%20conceptual
10. KNOWLEDGE REPRESENTATION AND REASONING, Hector J. Levesque, 1986 ,https://www.cs.cornell.edu/selman/cs672/readings/levesque.pdf#:~:text=calculating%20these%20implications%20may%20be,the%20computational%20tractability%20of%20reasoning

11. Semantic Search on Text and Knowledge Bases, 2016 https://ad-publications.cs.uni-freiburg.de/FNTIR_semanticsearch_BBH_2016.pdf#:~:text=Abstract%20This%20article%20provides%20a,variety%20of%20different%20communities%20with
12. An Ontology-Based Information Retrieval Model, David Vallet, Miriam Fernández, and Pablo Castells https://static.aminer.org/pdf/PDF/000/190/067/an_ontology_based_information_retrieval_model.pdf#:~:text=Our%20system%20uses%20inferencing%20mechanisms,by%20adding%20in%02ferred%20statements%20beforehand
13. Semantic Search on Text and Knowledge Bases, Hannah Bast, Björn Buchhold, Elmar Haussmann, 2016, https://ad-publications.cs.uni-freiburg.de/FNTIR_semanticsearch_BBH_2016.pdf#:~:text=Challenges%20of%20Semantic%20Web%20Data,We%20briefly
14. The Use of Semantic Web Technologies for Decision Support, Eva Blomqvist https://www.semantic-web-journal.net/sites/default/files/swj299.pdf#:~:text=needs%2C%20hence%2C%20keyword%20search%20is,search%20is%20to%20be%20adopted
15. Uncertainty Analysis in Ontology-Based Knowledge Representation, Sanjay Kumar Anand, Suresh Kumar, 2022 https://www.researchgate.net/publication/359318399_Uncertainty_Analysis_in_Ontology-Based_Knowledge_Representation#:~:text=There%20are%20various%20real,based%20on%20different%20classification%20of
16. What is Open Data? https://opendatahandbook.org/guide/en/what-is-open-data/#:~:text=internet,are%20not%20allowed
17. https://www.researchgate.net/figure/The-technical-architecture-used-by-Apache-Solr-System_fig3_320756907
18. Introducing SAGE: An Open-Source Solution for Customizable Discovery Across Collections, David B. Lowe, James Creel, Elizabeth German, Douglas Hahn, Jeremy Huff, 2021 https://journal.code4lib.org/articles/15740#:~:text=metadata,as%20DSpace%2C%20Fedora%2C%20and%20Blacklight
19. General search engine architecture https://www.researchgate.net/figure/General-search-engine-architecture_fig1_2523546#:~:text=,
20. A Novel Architecture of Ontology-based Semantic Web Crawler https://www.researchgate.net/publication/258651272_A_Novel_Architecture_of_Ontology-based_Semantic_Web_Crawler

Декларация об ответственности

Я, нижеподписавшийся Антон Павленко**,** выпускник Государственного университета Молдовы, программа магистратуры *Прикладная Информатика* заявляю под свою ответственность, что магистерская работа «Поиск, основанный на метаданных» является результатом моей работы, реализована на основе моих собственных исследований и на основе информации, полученной из цитируемых и указанных источников, в соответствии с этическими нормами.

Также заявляю, что использованные в диссертации источники, в том числе из Интернета, в том числе фрагменты, сгенерированные открытыми приложениями генеративного искусственного интеллекта Open AI, указаны с соблюдением правил предотвращения плагиата:

• Цитаты-фрагменты воспроизведённого текста пишутся в кавычках с точной ссылкой на источник;

• Обобщение идей других авторов и/или их передача, пере-формулирование своими словами содержит точную ссылку на источник.

• Фрагменты, созданные при помощи Open AI, упоминаются явно, если таковые имеются.

Я осознаю, что в противном случае буду нести ответственность в соответствии с действующим законодательством.

Павленко Антон Александрович

Подпись:_____

Дата: _____