

Министерство образования и науки Российской Федерации  
Федеральное государственное автономное образовательное учреждение высшего образования  
«САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО»

**Институт компьютерных наук и технологий**  
**Кафедра Компьютерные интеллектуальные технологии**

«Допустить к защите»  
Зав. каф. КИТ, к.т.н.  
\_\_\_\_\_ А.В. Речинский  
« \_\_\_\_ » \_\_\_\_\_ 20\_\_ г.

## **МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ**

### **ИССЛЕДОВАНИЕ И РЕАЛИЗАЦИЯ МЕТОДИКИ АВТОМАТИЧЕСКОЙ ГЕНЕРАЦИИ ВОПРОСОВ ПО ЗАДАНЫМ НЕСТРУКТУРИРОВАННЫМ ТЕКСТАМ НА РУССКОМ ЯЗЫКЕ**

направление: 09.04.03 «Прикладная информатика»

Выполнил(а):  
Архипова Марина Алексеевна  
Подпись \_\_\_\_\_

Руководитель:  
доцент каф. КИТ ИКНТ, к.т.н.,  
Сабинин Олег Юрьевич  
Подпись \_\_\_\_\_

Консультант:  
ст. преп. каф. КИТ ИКНТ,  
Андреева Наталья Викторовна  
Подпись \_\_\_\_\_

Рецензент:  
ст. преп. каф. ТП СПбГУ,  
Попова Светлана Владимировна  
Подпись \_\_\_\_\_

Санкт-Петербург  
2016

## РЕФЕРАТ

«Исследование и реализация методики автоматической генерации вопросов по заданным неструктурированным текстам на русском языке»

Работа содержит: стр.57, ил. 15, табл. 4, библи.: 24.

Ключевые слова: обработка естественного языка, компьютерная лингвистика, извлечение информации, генерация вопросов.

Автоматическая генерация вопросов на естественном языке является одной из актуальных задач компьютерной лингвистики и относится к направлению text-mining. Системы, обладающие подобной функциональностью, как правило, используются в сфере образования для проверки знаний учащихся, а именно при составлении вопросов по теоретическому материалу. В данной работе предлагается методика автоматической генерации вопросов по неструктурированным текстам на русском языке, а также разработка комплексной программной системы для нее. Проводимое исследование включает в себя обзор имеющихся методов и программных средств для извлечения информации, разработку и реализацию методики автоматической генерации вопросов для текстов на русском языке.

« Research and implementing methods of automatic question generation from unstructured Russian-language text»

This thesis includes: 57 pages, 15 figures, 4 table, 24 references.

Keywords: natural language processing, computational linguistics, information extraction, question generation.

Automatic generation of questions in natural language is one of the topical tasks of computational linguistics and refers to text-mining. Systems that have similar functionality, as a rule, used in the field of education to check the student's knowledge, namely in the preparation of questions for the theoretical material. This research proposes a method of automatic from unstructured Russian-language text and development of an integrated software system for it.. Research includes a review methods and software of

information extraction, development and implementation of a methodology for the automatic generation questions for text in Russian language.

## СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ.....	6
ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ .....	7
ВВЕДЕНИЕ .....	8
1 Аналитический обзор.....	11
1.1 Обзор подходов к извлечению информации из неструктурированного текста	11
1.1.1 Этапы анализа текстовой информации.....	11
1.1.2 Подходы к извлечению информации .....	14
1.2 Существующие подходы к генерации вопросов.....	16
1.3 Обзор систем автоматической обработки текстов на русском языке.....	18
2 Методика автоматической генерации вопросов по неструктурированным текстам на русском языке .....	21
2.1 Классификация вопросов для автоматической генерации.....	21
2.1.1 Виды вопросов для автоматической генерации.....	21
2.1.2 Модификации фактов .....	23
2.2 Этапе автоматической генерации вопросов .....	25
2.2.1 Извлечение информации .....	26
2.2.2 Генерация вопросов .....	30
3 Реализация методики автоматической генерации вопросов .....	31
3.1 Общее описание архитектуры системы .....	31
3.2 Модуль предварительной обработки текста .....	32
3.3 Модуль извлечения фактов .....	34
3.4 Модуль генерации вопросов .....	41
4 Тестирование разработанной системы.....	43
4.1 Данные для экспериментов .....	43
4.2 Результаты экспериментов.....	44
ЗАКЛЮЧЕНИЕ .....	46
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....	48
ПРИЛОЖЕНИЕ 1 .....	51

ПРИЛОЖЕНИЕ 2 .....	54
ПРИЛОЖЕНИЕ 3 .....	55

## ОПРЕДЕЛЕНИЯ

В данной пояснительной записке к выпускной квалификационной работе применяют следующие термины с соответствующими определениями:

- токенизация - это процесс выделения слов, чисел и других токенов, в частности нахождение границ предложений;
- машинное обучение - это раздел искусственного интеллекта, использующий разделы математической статистики, численных методов оптимизации, теории вероятностей, дискретного анализа для извлечения знаний из данных;
- неоднозначная грамматика – это формальная грамматика, которая может породить некоторую строку более чем одним способом;
- словоформа – это слово, представленное в определенной грамматической форме;
- лексема – это отдельное слово во всей системе его значений и форм;
- терминальный символ (терминал) – объект формального языка, имеющий в нём конкретное неизменяемое значение и являющийся элементом построения слов данного языка;
- нетерминальный символ (нетерминал) – объект, обозначающий какую-либо сущность языка и не имеющий конкретного символического значения.

## ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

NLP - Natural Language Processing.

ГЕЯ - генерация на естественном языке.

ФИО - фамилия, имя, отчество.

ФК - фразовые категории.

ML – machine learning.

QFD – quality function deployment.

DC - Definite clause grammar.

GLR - Generalized Left-to-right Rightmost.PDF - Portable Document Format.

XML – Extensible Markup Language.

## ВВЕДЕНИЕ

В настоящее время происходит бурное развитие информационных технологий, которые предоставляют все больше новых возможностей для организации обучения и образования. Объемы информации, в том числе текстовой, увеличиваются, поэтому особую актуальность приобретают вопросы, связанные с ее обработкой и анализом. Обработка текстовой информации является важной задачей во многих сферах человеческой деятельности, таких как образование, наука и др.

Естественный язык является одним из способов представления информации и предназначен, прежде всего, для повседневного общения.

Естественный язык обладает рядом свойств, которые затрудняют автоматическую обработку текстов, написанных на нем:

- неоднозначность – одно и то же выражение, форма, конструкция может означать разное;
- несимметричность – разные языки зачастую имеют различающиеся способы кодирования того или иного смысла;
- избыточность (вариативность) - наличие множества способов выражения некоторого смысла;
- конвенциональность - наличие множества допустимых теоретически способов выражения какого-то смысла, которые не принято использовать;
- эллиптичность – наличие множества умолчаний, необходимость восстановления опущенной информации для понимания;
- непрозрачность - активное использование языком сложных средств референции.

Задача «понимания» произвольных текстов на естественном языке может решаться с использованием различных технологий, в первую очередь на базе методов обработки данных на естественном языке — NLP (Natural Language



Processing). В настоящее время NLP применяется для решения обширного круга задач, таких как [1]:

- текстовый поиск;
- извлечение фактов;
- диалоговые системы и вопросно-ответные системы (Question Answering);
- синтез и распознавание речи;
- оценка тональности отзывов;
- кластеризация и классификация текстов.

Автоматическая генерация вопросов на естественном языке является одной из актуальных задач компьютерной лингвистики [2]. Системы, обладающие подобной функциональностью, как правило, используются в сфере образования для проверки знаний учащихся, а именно при составлении вопросов по теоретическому материалу [3].

В последнее время среди отечественных разработок автоматическая генерация на естественном языке (ГЕЯ) представлена недостаточным количеством работ, поэтому большинство исследований основываются на трудах зарубежных ученых-лингвистов и программистов, работающих в этой области [2]. Среди систем обработки естественного языка, отсутствуют системы, позволяющие комплексно решить проблему генерации вопросов по неструктурированным текстам на русском языке. Многие из них предназначены для решения конкретных задач, возникающих на различных этапах анализа текстов: выделения слов из текста (токенизация), морфологического анализа, построения синтаксической структуры предложений и т. д.

Целью данной работы является исследование и реализация методики автоматической генерации вопросов по заданным неструктурированным текстам на русском языке для повышения эффективности обучения и тестирования знаний на основе text- и data-mining.

Для достижения указанной цели необходимо решить следующие задачи:

- а) провести обзор методов извлечения информации из неструктурированного текста;
- б) рассмотреть существующие подходы к генерации вопросов;
- в) проанализировать существующие системы автоматической генерации текстов на естественном языке;
- г) рассмотреть возможные виды вопросов для автоматической генерации;
- д) разработать методику автоматической генерации вопросов по заданным неструктурированным текстам на русском языке;
- е) реализовать методику автоматической генерации вопросов по заданным неструктурированным текстам на русском языке;
- ж) протестировать реализованную методику на заданных неструктурированных текстах на русском языке.

## **1 Аналитический обзор**

В данной главе будет проведен анализ предметной области и описание современного состояния изучаемой проблемы. Для достижения указанной цели необходимо:

- а) провести обзор методов извлечения информации из неструктурированного текста;
- б) рассмотреть существующие подходы к генерации вопросов;
- в) проанализировать существующие системы автоматической генерации текстов на естественном языке.

### **1.1 Обзор подходов к извлечению информации из неструктурированного текста**

Значительный интерес при изучении возможности обработки текстов на естественном языке представляют методы извлечения информации, позволяющие структурировать содержащуюся в текстах информацию, что делает возможным дальнейшую ее обработку традиционными алгоритмическими методами.

Извлечение информации (information extraction) – это разновидность информационного поиска, связанная с выявлением сущностей и отношений, при которой из неструктурированного текста выделяется структурированная информация [4].

#### **1.1.1 Этапы анализа текстовой информации**

Для того, чтобы извлечь информацию, необходимо провести анализ текста. Согласно уровням представления естественного языка, процесс анализа текстовой информации состоит из следующих этапов (рисунок 1.1):

Графематический анализ является начальным анализом неструктурированного текста, представленного в виде цепочки символов в какой-либо кодировке, вырабатывающий информацию, необходимую для дальнейшей обработки текста.



Рисунок 1.1 - Схема анализа текстовой информации

В задачу графематического анализа входят:

- разделение входного текста на слова, разделители и т.д.;
- сборка слов, написанных в разрядку;
- выделение устойчивых оборотов, не имеющих словоизменительных вариантов;
- выделение ФИО (фамилия, имя, отчество), когда имя и отчество написаны инициалами;
- выделение электронных адресов и имен файлов;
- выделение предложений из входного текста;
- выделение абзацев, заголовков, примечаний.

Морфологический анализ обеспечивает определение нормальной формы, от которой была образована данная словоформа, и набора параметров, приписанных данной словоформе.

Синтаксический анализ определяет роли слов и их связи между собой, в результате получая набор деревьев, показывающих такие связи. Поскольку число предложений бесконечно, при синтаксическом разборе имеет смысл ориентироваться на более мелкие единицы – фразовые категории (ФК). ФК – это класс составляющих, грамматические свойства которого определяются частью речи, к которой принадлежит (главная) вершина составляющей [5]. Составляющая представляет собой структурную единицу предложения, целиком составленную из более тесно связанных друг с другом составляющих меньшего размера. Таким образом, алгоритмы автоматического анализа сводятся к вычленению ФК в составе предложения и поиску связей между ними.

Семантический анализ основывается на результатах работы предыдущих фаз обработки текста, которые всегда специфичны для конкретного языка. Следовательно, способы представления их результатов тоже могут сильно варьироваться, оказывая большое влияние на реализацию методов семантического анализа. На семантическом этапе появляется формальное представление смысла текста.

Каждый такой анализ — самостоятельная задача, не имеющая собственного практического применения, но активно используемая для решения более общих задач. Различные лингвистические уровни (рисунок 1.1) участвуют в разных процедурах анализа текста, которые входят в системы извлечения фактов. В общем случае деятельность по распознаванию сущностей можно разделить на следующие этапы:

- распознавание сущностей (Entity Recognition);
- аннотация сущностей (Named Entity Annotation);
- устранение неоднозначности сущностей (Entity Disambiguation).

Распознавание сущностей представляет собой задачу извлечения имен сущностей из текста, т.е. поиск и классификацию элементов текста в заранее определенные категории, такие как списки терминов предметной области, персоналий, организаций, географических названий и т.д. [4]

Существуют различные подходы к распознаванию именованных сущностей: например, с помощью словарей и с помощью регулярных выражений. Словари содержат список имен и фраз определенного класса (например, список стран или компаний), которые будут извлекаться из текста. Если словарь будет не полным, то пропущенные имена просто не будут распознаны в тексте. [4]

Словари не всегда подходят для извлечения сущностей, поскольку зачастую просто невозможно составить полный словарь, например для списка фильмов, книг, имен людей. Но существуют классы сущностей, которые определяются конкретными правилами, например, адреса электронной почты, даты. В этом случае для извлечения сущностей из текста можно использовать регулярные выражения.

Аннотация сущностей – это задача извлечения имен сущностей из текста и их аннотация классом из заданного набора классов (например, классы Person, Organization, Location). Существует два подхода к аннотированию сущностей: на основе правил и на основе статистических моделей [3].

### **1.1.2 Подходы к извлечению информации**

Для того, что получить из неструктурированного текста структурированную информацию, необходимо использовать специальные алгоритмы. Условно, выделяют три основных подхода, применяющихся для извлечения фактов [6]:

- по онтологиям;
- основанный на правилах (Rule-based);
- использующий машинное обучение (ML).

Рассмотрим подробнее каждый из указанных подходов.

Онтология включает в себя понятия и отношения предметной области и описывает данные, которые необходимо извлечь из текста. Онтология - точное,

подробное описание (модель) некоторой области знаний. Формально онтология - это пятерка вида (формула 1.1) [7]:

$$O = (C, A, R_C, T, D), \quad (1.1)$$

где  $C$  – множество классов, описывающих понятия некоторой предметной или проблемной области,

$A$  – множество атрибутов, описывающих свойства понятий и отношений,

$R_C$  – множество отношений, заданных на классах (понятиях),

$T$  – множество стандартных типов значений атрибутов,

$D$  – множество доменов.

Адаптируя данное определение к случаю извлечения фактов, можно сказать, что онтологии представляют собой структуры, в которых описываются некоторые понятия и/или объекты, отношения между ними, а также их характеристики. В качестве примера на рисунке 1.2 представлена онтология методологии QFD в виде концептуальной карты. Онтологический подход позволяет строить гипотезы по отношению к объектам и фактам в тексте, а далее верифицировать или отклонять эти гипотезы.

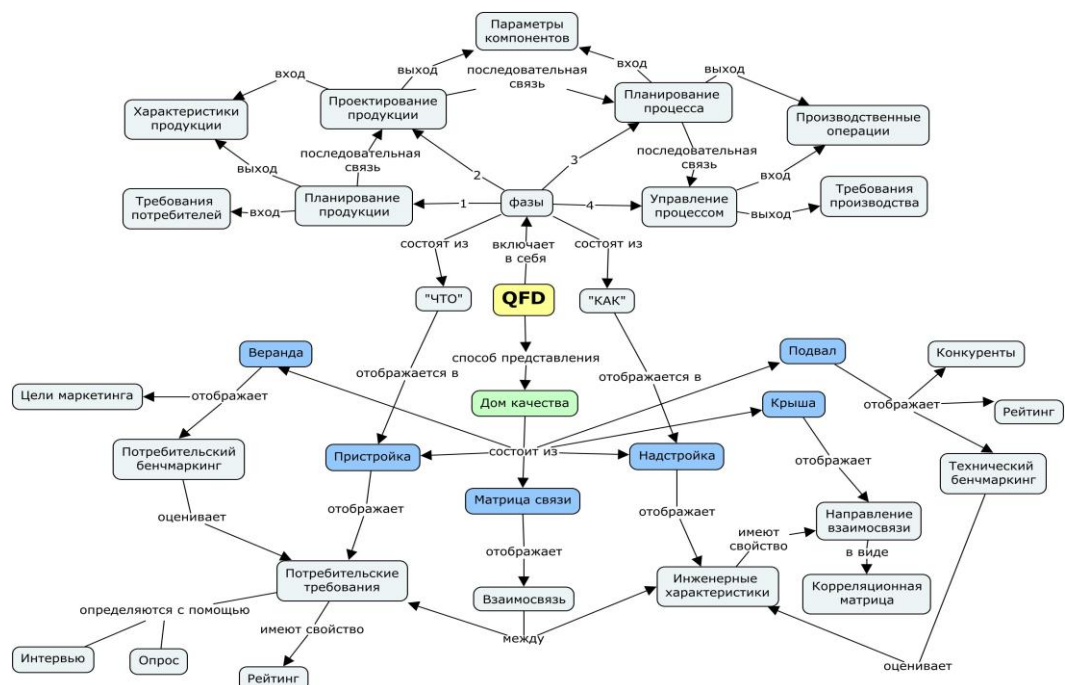


Рисунок 1.2 - Онтология QFD

Следующий рассматриваемый подход - машинное обучение (machine learning). Данный подход требует большого объема входных данных и основывается

на статистике, т.е на основе вероятности появления той или иной лексической единицы в том или ином контексте система подсказывает, куда «определить» ее в конкретном случае. Основная сложность подхода заключается в трудоемкости разметки языковых корпусов (большой объем структурированных текстов, чаще всего размеченных вручную, используемых для обучения алгоритмов обработки естественного языка) и отсутствии развитых инструментов для автоматизации данного процесса.

Третий подход – подход, основанный на правилах. Он представляется собой написание шаблонов вручную на диалекте некоторого формального языка. Эксперт составляет описания типов информации, которые необходимо извлечь. Наиболее простыми примерами объектов, для которых актуален данный подход, являются имена, даты, названия компаний и др.

Анализ методов извлечения фактов из текстов, позволяет сделать вывод о том, что существует возможность разработки систем, как основе лингвистических правил, основанных на грамматиках, так и на основе машинного обучения. Оба направления имеют свои достоинства и недостатки. Основное различие указанных подходов заключается в том, что первый требует привлечения эксперта для ручного составления нетривиальных формальных грамматик, в то время как системы на основе машинного обучения, напротив, практически не требуют участия человека в своей работе. Но подготовка корпусов и обучение по ним может занять время, превосходящее время написания правил [8].

## **1.2 Существующие подходы к генерации вопросов**

Различные исследователи изучали методы автоматической генерации вопросов для оценки фактических знаний. В данном разделе будут освещены некоторые работы в этой области.

Для компьютерного контроля знаний могут использоваться генераторы вопросов - специальные программы, которые, используя формализованное



представление знаний данной предметной области, синтезируют вопросы для компьютерных тестов [9] [10].

В работе Митькова и др. [11] описан подход к автоматической генерации элементов множественного выбора для тестов по электронным документам. Авторы рассматривают систему, которая использует для создания вопросов правила поверхностного разбора конкретных типов предложений (например, правило для создания вопроса: Что / подлежащее / сказуемое?). В дополнение к использованию различных NLP-технологий в работе описывается методика для генерации вопросов с множественным выбором, осуществляя поиск структур для словосочетаний, которые семантически похожи на ответ на фразу.

Kunichika и др. [12] описывают систему для генерации вопросов на основе синтаксического и семантического анализа, которые выводятся с помощью DC-грамматики (грамматика, построенная на определённых предложениях). В работе авторы описывают методы генерации пяти видов вопросов:

- о содержании одного предложения;
- используя антонимы и синонимы;
- используя модификаторы появления объектов во множестве предложений;
- о содержании представленного множества предложений, используя относительные местоимения;
- о времени и пространстве.

Указанные подходы применимы к системам, работающим с английским языком. Незначительная модификация данных приемов не решит поставленную задачу, т.к. вопросы на английском языке имеют четкую структуру и определённый порядок слов. При попытке апробации данных методов на русском языке возникают сложности, связанные с многообразием форм, которые может принимать одно слово и многообразием структуры предложений.

### 1.3 Обзор систем автоматической обработки текстов на русском языке

Автоматическая обработка текста на естественном языке позволяет облегчить поиск и извлечение информации с целью дальнейшей аналитической обработки. Под аналитической обработкой в данном случае подразумевается генерация определенных типов вопросов.

Рассмотрим более подробно те системы, которые могут быть использованы для решения задач данной работы.

**OntosMiner** — это комплексная система, которая дает возможность распознавать в текстах на естественном языке упоминания объектов внимания (именованные сущности) и семантических связей между ними (факты), а также определять общую тональность документа. Ядром технологии является лингвистический процессор **OntosMiner**, который анализирует текст с использованием онтологий и словарей, доступных для редактирования пользователем, а также специальных эвристик. Система построена на принципах машинного обучения. Сейчас активные работы в рамках этого проекта не ведутся и тестовая версия программы, которая должна была быть открыта для вузов и научно-исследовательских учреждений в бесплатном режиме, не функционирует. [13]

**InterSystems iKnow** — встраиваемая технология, которая позволяет разработчикам создавать приложения для извлечения информации из неструктурированных данных. Эта передовая технология обеспечит пользователям доступ к ценной информации, извлекаемой из всех корпоративных источников данных, и поможет принимать обоснованные решения. Уникальная технология **iKnow** позволяет находить важные концепты и связи в неструктурированных данных [14].

**Томи́та-парсер** — это инструмент для извлечения структурированных данных (фактов) из текста на естественном языке. Извлечение фактов происходит при помощи контекстно-свободных грамматик и словарей ключевых слов. На вход отдается анализируемый текст, а также словарь и грамматика. Файл грамматики

состоит из шаблонов, написанных на внутреннем языке/формализме Томита-парсера. Эти шаблоны описывают в обобщенном виде цепочки слов, которые могут встретиться в тексте. Кроме того, грамматики определяют, как именно нужно представлять извлеченные факты в итоговом выводе.

В таблице 1.1 представлены основные системы автоматической обработки текстов на русском языке. Основными критериями для выбора систем стали:

- наличие возможности работы с русским языком;
- область применения должна быть связана с задачей извлечения сущностей или одним из видов анализа текстов.
- наличие документации

Наиболее подходящими среди представленных систем являются Томита-парсер и IKnow поскольку они доступны для конечных пользователей и имеют сопутствующую документацию, поддерживают русский язык и позволяют извлекать структурированную информацию из текстов. Однако, при более глубоком анализе указанных средств, для решения задач данной работы был выбран инструмент Томита-парсер, т.к имеет подробное руководство разработчика на русском языке, что значительно упрощает его изучение, и не требует наличия лицензии в отличие от Intersystems IKnow .

Таблица 1.1- Системы АОТ для русского языка

Название	Применение	Языки	Интерфейс	Целевая аудитория	Доступность
«Томита Парсер» («Яндекс»)	Извлечение структурированной информации из неструктурированного текста	Русский	Командная строка Python	Студенты, исследователи и разработчики области NLP	Бесплатно
IKnow («Intesystems»)	Извлечение информации из неструктурированных данных	Английский, русский,	Командная строка Графический	Студенты, исследователи и разработчики области NLP	Бесплатная демоверсия
OntosMiner	Извлечение знаний из текстовых коллекций и их структурирование в виде онтологий	Английский, русский, немецкий, французский	Графический	Студенты, исследователи и разработчики области NLP	Бесплатная демоверсия
«ЭТАПЗ»	Синтаксический анализ и визуализация дерева разбора	Русский	Графический	Студенты, исследователи и разработчики области NLP	Бесплатно
NLTK	Разработка систем анализа текстов	Английский + поддержка обучения классификаторов для остальных языков	Командная строка Python	Студенты, исследователи и разработчики области NLP	Бесплатно
PyMorphy2	Морфологический словарь для исследовательских и коммерческих проектов	Русский	Командная строка Python	Студенты, исследователи и разработчики области NLP	Бесплатно
2long2read	Выделение ключевых предложений в связном тексте	Русский	Графический	Пользователи Интернета	Бесплатно

## **2 Методика автоматической генерации вопросов по неструктурированным текстам на русском языке**

В данной главе будет рассмотрена методика автоматической генерации вопросов по неструктурированным текстам на русском языке.

Для достижения указанной цели необходимо решить следующие задачи:

- а) рассмотреть возможные виды вопросов для автоматической генерации;
- б) предложить методику автоматической генерации вопросов по заданным неструктурированным текстам на русском языке.

### **2.1 Классификация вопросов для автоматической генерации**

#### **2.1.1 Виды вопросов для автоматической генерации**

Вопросительные предложения выражают вопрос о предмете текста. В некоторых языках вопросительные предложения имеют строго определенный порядок слов. Например, в английском языке в вопросе сначала идёт сказуемое (или его вспомогательная часть), а потом подлежащее. В русском языке строгого порядка слов нет, что усложняет решение проблемы извлечения фактов из неструктурированных текстов.

Вопросительные предложения на знание представляют собой фактические вопросы. Данный тип вопросов предназначен для выяснения фактов и особенностей, которые можно извлечь из текста. Ценность состоит в том, что ответы на фактические вопросы, в отличие от ответов на конвергентные или дивергентные вопросы, можно оценить как правильные или неправильные. Фактические вопросы используются для тестирования и оценки знаний [19].

Для того, чтобы оценить, какие вопросы наиболее удобно генерировать автоматически из текстов на естественном языке, была произведена классификация по сложности их генерации:

- а) извлечение фактов без модификаций;
- ответом на них является исключенное слово из описания факта;
  - основу вопроса составляет описание некоторого факта, представляющее собой предложение/отрывок из текста в исходном виде, помимо исключенного слова, к которому задается вопрос;
  - конструкция вопросительного предложения имеет простую форму, например: «Какое слово пропущено <описание факта>?» или «Чем является <описание факта>?».

- б) извлечение фактов с модификациями;

Модификации можно представить в двух видах:

- 1) искажающие смысл исходного факта;
  - ответами на них являются «да»/«нет», а также один или несколько вариантов ответа (утвердительная или отрицательная фраза, построенная на основе фактов);
  - основу составляет предложение/отрывок из текста в исходном виде, например, с использованием замены утвердительного предложения на отрицательное;
  - конструкция вопросительного предложения имеет простую форму: «Верно ли, что <описание факта> или «Укажите верные утверждения: <часть описания факта1>, <часть описания факта2>...».
- 2) использующие вопросительные слова.
  - ответом является фактическая информация из исходного обрабатываемого текста. Это может быть персона, предмет, место, время и т.д., в зависимости от части предложения, к которой задается вопрос;
  - конструкция генерируемого предложения начинается с вопросительных слов (например, «кто, что, когда?») и требует модификации предложения, извлеченного из исходного текста (порядок слов, словоформа и т.д).

в) извлечение фактов с модификациями и генерацией вариантов ответов.

- вариантом ответа является фактическая информация из исходного обрабатываемого текста, а также похожая на извлеченные факты по структуре и смыслу информация;
- варианты ответов генерируются на основе фактов, извлеченных из исходного текста;
- конструкция генерируемого предложения начинается с вопросительных слов (например, «кто, что, когда?») и требует модификации предложения, извлеченного из исходного текста (порядок слов, словоформа и т.д).

В таблице 2.1 показаны примеры по каждому типу вопросов из классификации, представленной выше.

Таблица 2.1- Классификация вопросов по сложности генерации

	Классификация	Пример
1	Извлечение фактов без модификаций	Какое слово пропущено: «Таблицы в базах данных имеют ... имена»? Чем является совокупность программ, позволяющих осуществить на компьютере автоматизированную обработку информации?
2	Извлечение фактов с модификациями	
2.1	Модификации, искажающие смысл исходного факта	Верно ли, что Таблицы в базах данных не имеют уникальные имена?
2.2.	Модификации, использующие вопросительные слова	Какие имена имеют таблицы в базах данных?
3	Извлечение фактов с модификациями и генерацией вариантов ответов	Какие имена имеют таблицы в базах данных? уникальные произвольные

### 2.1.2 Модификации фактов

В данной работе в первую очередь рассматривается генерация вопросов путем извлечения предложений без их модификаций, поскольку: во-первых данные варианты наиболее просты для исследования на начальном этапе; во-вторых извлечение фактов без модификаций является основой для генерации любого типа вопросов..

Кроме того, предлагается подход к генерации вопросов на основе фактов с модификациями, подразумевающий искажение и преобразование заведомо верных фраз и фактов так, чтобы получались читаемые и несущие определенный смысл предложения.

Можно сформулировать некоторые правила, позволяющие модифицировать исходные предложения, выделенные из текста так, чтобы была возможность проверить знания тех или иных фактов:

- замена утвердительного предложения на отрицательное и наоборот;
- перестановка слов в предложении (перестановка членов предложения, являющих одной частью речи, или слова в словосочетании вида существительное + зависимое существительное, являющимся одним членом предложения; изменение синтаксически главного слова);
- замена слов или части предложения на противоположные по смыслу.

Замена предложения представляет собой средство образования определённых типов предложения, в т.ч. отрицательных. Так, положительную фразу «Столбцы каждой таблицы имеют уникальные имена внутри одной таблицы» можно преобразовать в «Столбцы каждой таблицы *не* имеют уникальные имена внутри одной таблицы».

Перестановка слов в предложении влечёт за собой изменение смысла. Например, предложение «*База данных* – программная система, предназначенная для хранения данных, модификации...» трансформируется в фразу «*Данные базы* – программная система, предназначенная для хранения данных, модификации...», что не является верным.

Замена слов на противоположные по смыслу также может быть эффективной для генерации вопросов. Предложение «Первый ключ должен содержать как можно *меньше* столбцов» модифицируется в «Первый ключ должен содержать как можно *больше* столбцов»

Предполагается, что указанные модификации можно будет формализовать в виде шаблонов.



## 2.2 Этапе автоматической генерации вопросов

В целом, задачу автоматической генерации вопросов можно разделить на два основополагающих этапа:

- автоматическое извлечение определенной информации, фактов, данных из текста на естественном языке;
- генерация конкретных вопросов на основе извлеченной информации на первом этапе.

Схематично данный процесс и его составляющие можно представить в следующем виде (рисунок 2.1):

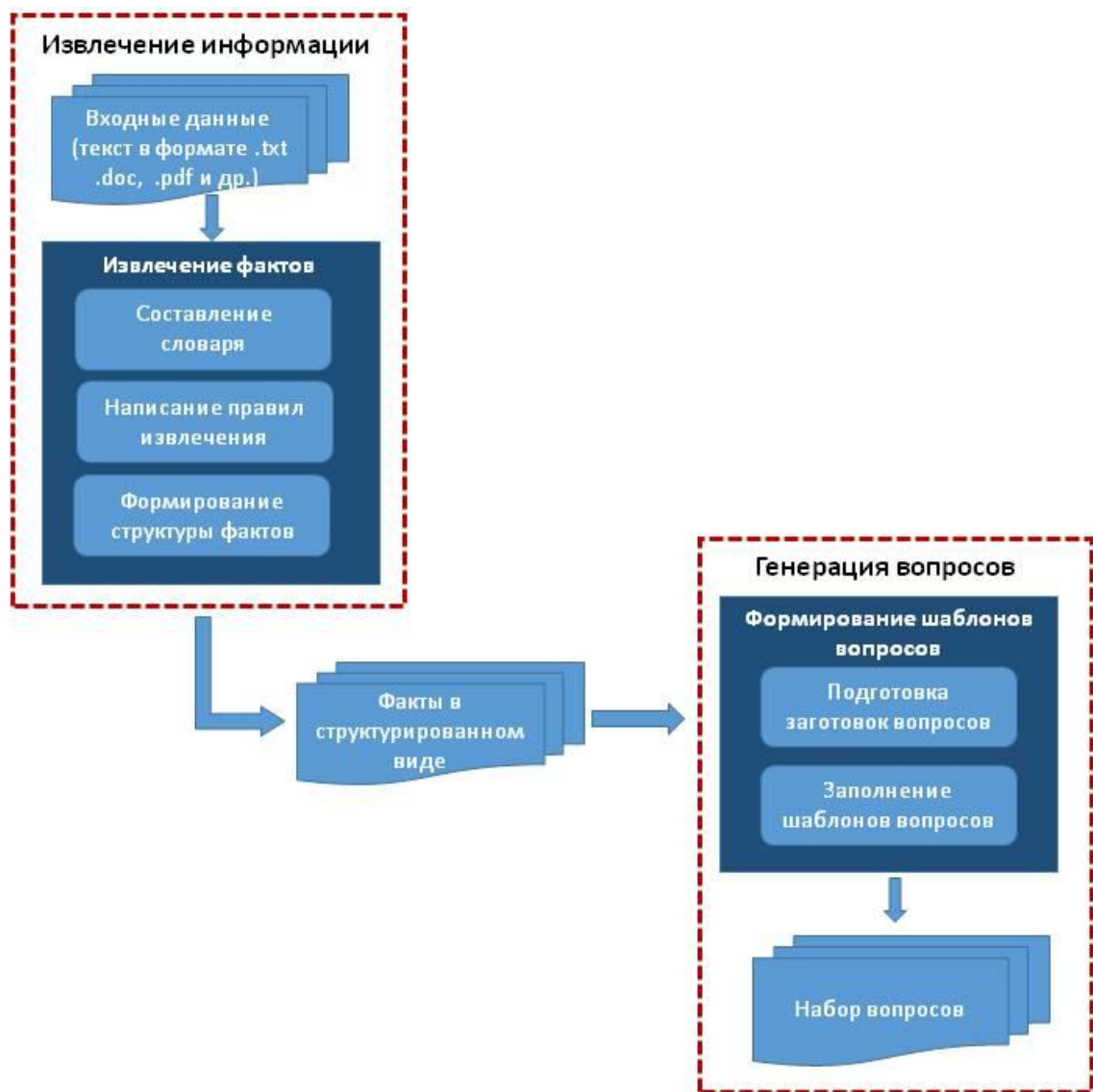


Рисунок 2.1 - Схема автоматической генерации вопросов

### 2.2.1 Извлечение информации

В методике, разрабатываемой в рамках данной работы, будут рассматриваться только методы извлечения информации, основанных на правилах, т.к. они используются в инструменте Томита-парсер.. Методы, использующие машинное обучение или онтологии, не рассматриваются, однако могут быть использованы для развития данной темы.

Для извлечения информации из текста необходимо выполнить определенную последовательность шагов:

- а) Составить словарь ключевых слов;
- б) выделить предложения, в которых встречаются ключевые слова из словаря;
- в) задать правила извлечения фактов из текста;
- г) по заданным правилам выделить факты из текста.

Первым этапом является составление словаря ключевых слов. Существует несколько способов построения словарей конкретной предметной области для извлечения фактов из текстов на естественном языке [15]:

- экспертный (вручную);
- используя онтологии;
- анализируя частотный словарь входного текста с целью выделения наиболее часто встречающихся понятий;
- и др.

Проблема составления словаря по сути является отдельным направлением, использующим NLP технологии и не является предметом исследования в данной работе.

Далее процесс представляет собой выделение предложений, в которых содержатся ключевые слова из словаря. Для этого необходимо проанализировать исходный текст и «пометить» предложения с целью последующей обработки. С точки зрения этапов обработки текста, которые были описаны в разделе 1.1.1,

данная деятельность относится к графематическому анализу. Такой анализ позволяет выделить слова и предложения, имеющие значение для предметной области.

Наибольший интерес в извлечении информации из текстов представляет деятельность по составлению правил, по которым будут выделены факты. Данный этап предполагает использование морфологического и синтаксического анализа. Морфологический анализ позволяет решать проблему согласования слов в словосочетаниях. Для того, чтобы соблюдалось согласование проводится нормализация слов с помощью определенных граммем. Синтаксический анализатор, в основе которого лежат формальные грамматики, обеспечивает построение дерева разбора, которое отражает структуру входных цепочек.

Формальной грамматикой  $G$  называется совокупность четырех объектов (формула 2.1):

$$G = (\Sigma, N, P, S), \quad (2.1)$$

Где  $\Sigma$  — набор (алфавит) терминальных символов,

$N$  — набор (алфавит) нетерминальных символов,

$P$  — набор правил вида: «левая часть»  $\rightarrow$  «правая часть», где:

- «левая часть» — непустая последовательность терминалов и нетерминалов, содержащая хотя бы один нетерминал;
- «правая часть» — любая последовательность терминалов и нетерминалов.

$S$  — стартовый (или начальный) символ грамматики из набора нетерминалов.

Правилами на формальном языке обычно задаются шаблоны. Под шаблоном обычно понимают заготовку текста, в котором некоторые элементы можно изменять в соответствии с заданным алгоритмом. Основным элементом таких шаблонов является элемент-слово, соответствующий отдельному слову текста и описывающий:

- конкретную словоформу;
- множество словоформ конкретной лексемы, обладающих фиксированными морфологическими характеристиками;

- произвольное слово заданной части речи, стоящее в определенной грамматической форме.

Шаблон задает лексический состав языковой конструкции и поверхностно-синтаксические свойства и тем самым может быть использован для распознавания ее в тексте и последующего извлечения. Шаблон строится как последовательность элементов, описывающих соответствующие фрагменты языковой конструкции – в том порядке, в каком они встречаются в этой конструкции. Средства языка позволяют задавать вариативность конструкции, включая набор входящих в нее слов (лексем) и их морфологических характеристик (признаков) [16].

В левой части шаблона указывается его имя, правая часть представляет собой описание нескольких вариантов формализуемой фразы. Типовое описание конструкции на языке описательных лингвистических шаблонов выглядит следующим образом (формула 2.2) [17][18]:

$$AdjNoun = A \ N \ <A=N>, \quad (2.2)$$

где AdjNoun – именная группа из прилагательного и существительного,

A – прилагательное (adjective),

N – существительное (noun),

A=N - условие согласования.

Данный шаблон описывает грамматически согласованную именную группу из прилагательного и существительного: например, «программное обеспечение».

На основе изученных материалов было принято решение составить набор шаблонов для выделения терминов и их определений, как одного из примеров фактов, извлекаемых из текста. Термин – это слово, устойчивое словосочетание или сокращение, которое выражает и классифицирует в данной предметной области определённое понятие или сущность. Для извлечения определений можно составить шаблоны вида (формулы 2.3):

$$\begin{aligned} Definition &= Term + Dash Phrase Punct; \\ Phrase &= Word + Comma Word^*; \\ Term &= Adj Noun / Noun Noun; \\ Punct &= Dot / Semicolon; \end{aligned} \quad (2.3)$$

где *Definition* – начальный нетерминал, описывающий структуру факта который требуется извлечь,

*Phrase* – нетерминал, описывающий описание термина в факте,

*Term* – нетерминал, описывающий термин в факте,

*Punct* - нетерминал, описывающий конец факта;

*Word* – любое слово, состоящее из букв русского или латинского алфавита,

*Dash* – знак пунктуации «тире»,

*Dot* – знак пунктуации «точка»,

*Semicolon* – знак пунктуации «точка с запятой»,

*Adj* – прилагательное,

*Noun* – существительное.

Для написания лексических шаблонов также используются элементы (метасимволы) языка регулярных выражений, например:

а) \* повторение символов,

б) + – непустое повторение,

в) | – альтернатива,

г) и др.

Описанная выше конструкция позволит выделить определения вида:

*Программное обеспечение - это совокупность программ, позволяющих осуществить на компьютере автоматизированную обработку информации.*

*Язык программирования — формальная знаковая система, предназначенная для записи компьютерных программ.*

Более подробно набор шаблонов для определений будет представлен в главе 3.

Выделение фактов из текстов на естественном языке подразумевает использование правил, полученных на предыдущем этапе. В тексте происходит поиск фрагментов, соответствующих заданному шаблону. Процесс выделения фактов представляет собой компоновку фактов в определенную структуру, удобную для дальнейшей генерации по ним вопросов. Достоинство данного подхода заключается в обозримости и ясности правил, а также точности настройки на

конкретную задачу. К недостаткам можно отнести низкую скорость работы системы, большое количество правил, что в случае возникновения необходимости перенастройки системы на другую задачу вызовет проблемы с переписыванием набора правил. На данном этапе проводится семантический анализ, результатом которого является определение структуры, отражающей взаимосвязь между частями выделяемого факта, а также его тип.

### **2.2.2 Генерация вопросов**

Генерация вопросов является вторым важным процессом в задаче автоматической генерации вопросов по заданным неструктурированным текстам на русском языке. Он использует результаты, полученные на предыдущем этапе. Автоматически сгенерировать тестовые вопросы позволит метод генерации вопросов по набору шаблонов [10]. Применение шаблона к определенному фрагменту текста позволит выполнить его перестроение в вопрос. Данный подход будет эффективным для фактов в структурированном виде, например для пар вида: <Термин, Определение>, о чем было упомянуто в разделе 2.2.1.

Рассмотрим формулировки вопросов к терминам, которые могут стать основой шаблона:

- Как называется <Определение>?
- Что означает <Определение>?
- Чем является <Определение>?
- Что такое <Определение>?
- Верно ли, что <Термин, Определение>?

Для реализации возможности генерации вопросов различных типов данный метод можно скомбинировать с методом генерации вопросов на основе текстового корпуса, который предложен в работе [10]. Таким образом это позволит в большей степени разнообразить конечный набор задаваемых к текстам вопросов.

### **3 Реализация методики автоматической генерации вопросов**

#### **3.1 Общее описание архитектуры системы**

Программная система, реализующая методику автоматической генерации вопросов должна иметь легко понятный, простой интерфейс. Для решения данной задачи была выбрана среда разработки Microsoft Visual Studio 2010, включающая в себя интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты обеспечивают возможность разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms.

Windows Form предоставляет обширный набор компонент, таких как кнопки, меню, текстовые поля, диалоговые окна и многие другие элементы управления. Это позволяет создавать простые приложения с удобным пользовательским интерфейсом (UI) Windows. По существу, эти элементы управления являются просто классами из библиотеки .NET Framework. Языком программирования выбран Visual C# – объектно-ориентированный язык программирования.

Общая архитектура системы автоматической генерации вопросов состоит из следующих компонентов:

- модуль предварительной обработки текста – предназначен для преобразования PDF-файлов в TXT-файлы с целью дальнейшей обработки в модуле извлечения фактов;
- модуль извлечения фактов – позволяет извлекать структурированную информацию из TXT-файла, который содержит текст на естественном языке;
- модуль генерации вопросов - формирует список вопросов по данным, полученным из модуля извлечения фактов.

Архитектура системы представлена на рисунке 3.1.

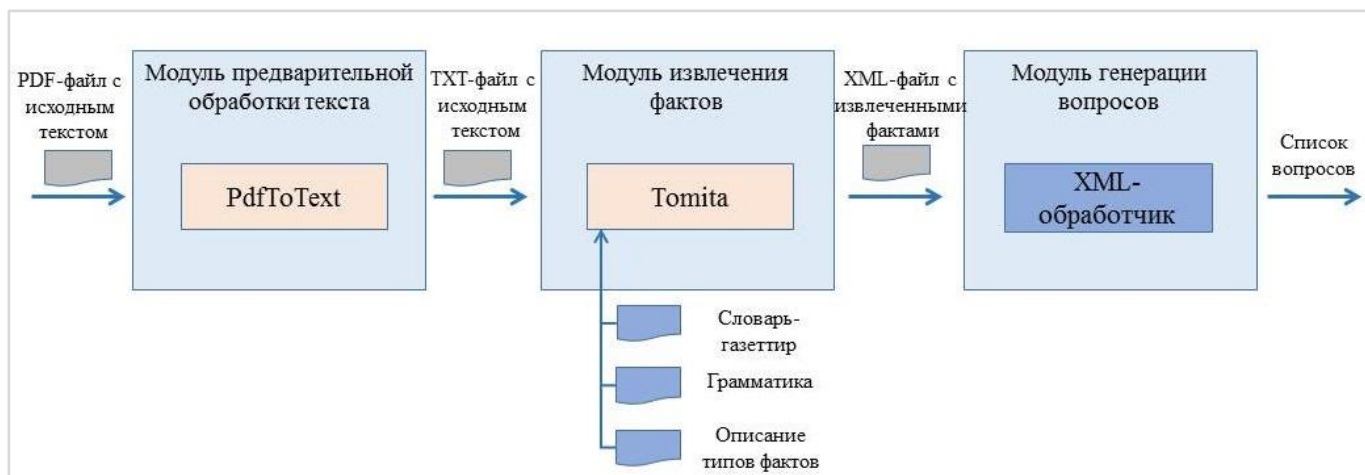


Рисунок 3.1 – Архитектура системы генерации вопросов

Стоит отметить, что модуль предварительной обработки текста использует для своей работы внешнюю программу-конвертер PdfToText. Аналогично этому, модуль извлечения фактов эксплуатирует внешний инструмент Томита-парсер.

Главное окно системы – форма, содержащая вкладки для каждого компонента системы, перечисленных выше. По умолчанию открытой вкладкой является Модуль предварительной обработки текста.

Далее более подробно будет рассмотрен каждый модуль.

### 3.2 Модуль предварительной обработки текста

Чтобы обеспечить возможность работы системы с различными форматами (кроме .txt) необходимо разработать модуль предварительной обработки текста.

Модуль представляет собой интегрированную в разрабатываемую систему программу-конвертер, которая позволяет преобразовывать данные из одного формата в другой. Одним из наиболее распространенных является формат PDF (Portable Document Format) – кроссплатформенный формат для электронных документов. По этой причине для системы был выбран просмотрщик PDF файлов, с открытым исходным кодом Xpdf [20].

В состав пакета Xpdf входит несколько компонент для работы с указанным форматом, однако для данной работы интерес представляет инструмент PdfToText -



утилита командной строки, преобразующая файлы формата PDF в простой текст формата TXT. К основным преимуществам программы является то, что требует небольшого количества ресурсов, отличается стабильностью и удобен для внедрения в систему.

Запуск PdfToText осуществляется из командной строки в следующем виде (формула 3.1):

$$pdftotext.exe [options] [PDF-file [text-file]], \quad (3.1)$$

где *options* – параметры запуска (например, кодировка),

*PDF-file* – исходный файл,

*text-file* – конвертированный файл.

Пример работы представлен на рисунке 3.2.

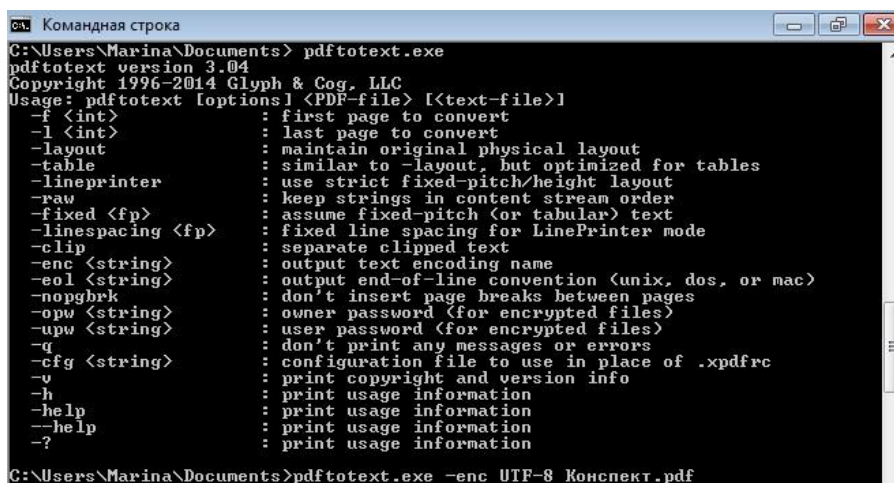


Рисунок 3.2 – Вызов PdfToText из командной строки

В общем интерфейсе системы за данный модуль отвечает вкладка «Предварительная обработка текста». При выборе этой вкладки, для пользователя открывается форма, показанная на рисунке 3.3.

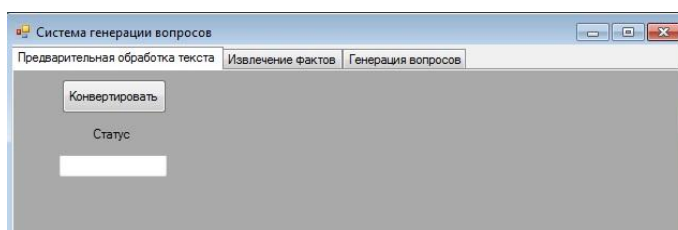


Рисунок 3.3 – Интерфейс модуля предварительной обработки текста

Кнопка «Конвертировать» запускает программу PdfToText следующим образом:

- а) открывается диалоговое окно с помощью формы OpenFileDialog, где пользователь может выбрать исходный файл в формате PDF.
- б) создается экземпляр класса System.Diagnostics.Process proc;
- в) метод Process.Start запускает ресурс процесса, передавая в качестве аргумента путь файла полученный в п.2. и связывает его с компонентом Process.
- г) при корректной отработке программы-конвертера, на экране выводится сообщение «Ок», при некорректной - «Ошибка».

Это означает, что пользователь может приступить к следующему этапу – извлечение фактов.

### **3.3 Модуль извлечения фактов**

Для решения задачи извлечения фактов из неструктурированного текста на русском языке было решено использовать Томита-парсер. Томита-парсер — это инструмент для извлечения структурированных данных (фактов) из текста на естественном языке. Извлечение фактов происходит при помощи контекстно-свободных грамматик и словарей ключевых слов.

В основу инструмента заложен алгоритм GLR-парсинга (Generalized left-to-right algorithm), автором которого является японский ученый Масару Томита (1984). GLR-парсер является расширенным алгоритмом LR-парсера, предназначенным для разбора по недетерминированным и неоднозначным грамматикам.

Набор файлов необходимых для типового сценария использования парсера включает в себя файлы, представленные в таблице 3.1.

Таблица 3.1 – Набор файлов для работы Томита-парсер

Название файла	Описание
<i>config.proto</i>	конфигурационный файл парсера
<i>mydic.gzt</i>	корневой словарь - газеттир
<i>mygram.cxx</i>	грамматика
<i>facttypes.proto</i>	описание типов фактов

В конфигурационном файле указывается:

- где находятся тексты, которые нужно обработать;
- путь к корневому словарю;
- какие статьи корневого словаря запускать;
- какие факты нужно сохранить и в каком формате;
- куда сохранять факты;
- несколько вспомогательных параметров.

**Газеттир** — словарь ключевых слов, которые используются в процессе анализа КС-грамматиками. Каждая статья такого словаря задает множество слов и словосочетаний, объединенных общим свойством. Слова или словосочетания можно задавать явно списком, а можно «функционально», указав грамматику, которая описывает нужные цепочки.

**Грамматика** — множество правил на языке КС-грамматик, описывающих синтаксическую структуру выделяемых цепочек. Грамматический парсер запускается всегда на одном предложении. Перед запуском терминалы грамматики отображаются на слова (или словосочетания, об этом будет сказано ниже) предложения. Одному слову может соответствовать много терминальных символов. Файл грамматики состоит из шаблонов, написанных на внутреннем языке/формализме Томита-парсера. Эти шаблоны описывают в обобщенном виде цепочки слов, которые могут встретиться в тексте. Кроме того, грамматики определяют, как именно нужно представлять извлеченные факты в итоговом выводе.

**Факты** — таблицы с колонками, которые называются полями фактов. Факты заполняются во время анализа парсером предложения. В процессе интерпретации происходит сопоставлении множества строящихся нетерминалов грамматики множеству выделяемых для определенного типа факта объектов, т.е. в распределении подцепочек слов по заранее заданным полям в таблице. [21]

Формально, алгоритм работы Томита-парсер представляется следующим образом – рисунок 3.4.



Рисунок 3.4 – Алгоритм работы Томита-парсер

На первом этапе, если ключ состоит из нескольких слов создается мультиворд.

Отбор ключей, упоминаемых в грамматике происходит среди всех найденных ключей словаря на предыдущем шаге.

Если среди отобранных ключей встречаются мультиворды, пересекающиеся друг с другом или включающие в себя одиночные ключевые слова, то парсер пытается покрыть предложение непересекающимися ключевыми словами так, чтобы как можно большие куски предложения были охвачены ими.

На следующем этапе на входные слова и мультиторды отображаются терминалы грамматики. Далее происходит построение GLR-парсером всех возможных вариантов на последовательности множеств терминалов.

Интерпретация подразумевает под собой отбор специально помеченных подузлов и запись слов, которые им соответствуют, порождаемые грамматикой поля фактов.

Рассмотрим пример извлечения фактов из текста, посвященного базам данных. Непосредственно текст представлен в Приложении 1. Основной акцент сделан на извлечение определений из текста (формулировок, раскрывающих содержание, смысл чего-либо).

Для начала необходимо было определить грамматику, т.е набор правил, описывающих текстовые цепочки. Грамматика пишется на языке Томита. Структура правила состоит из левой и правой части, разделенных  $\rightarrow$ . Левая часть представляет собой один нетерминал, правая часть состоит из нетерминалов и терминалов.

Терминал – объект, имеющий конкретное, неизменяемое значение (алфавит языка Томита – Noun, Verb, Comma и др.). Нетерминал – объект, обозначающий какую-либо сущность языка, строится из терминалов. Перечень используемых терминалов представлен в Приложении 2. На рисунке 3.5 показана грамматика, для извлечения фактов, являющихся определениями:

```
1  #encoding "utf-8"
2  #GRAMMAR_ROOT S
3
4  S -> Def | Def1 | Def2 | Def3 | Def4;
5  Def -> Part interp(Definition.Term) Hyphen WordCom+ interp(Definition.Description) Punct;
6  Def1 -> Part interp(Definition.Term) Hyphen "это" WordCom+ interp(Definition.Description) Punct;
7  Def2 -> WordCom+ interp(Definition.Description) Verb<kwtype=call> Word<gram="PART">* Part interp(Definition.Term)
  LBracket* Word<lat>* RBracket* Punct;
8  Def3 -> Part interp(Definition.Term) Verb<kwtype=is> Word+ interp(Definition.Description) Punct;
9  Def4 -> WordCom+ interp(Definition.Description) Verb<kwtype=is> Part interp(Definition.Term) Punct;
10 WordCom -> Word<gnc-agr[1]>+ Comma* Word<gnc-agr[1]>*;
11 Part -> Noun* Noun | Adj<gnc-agr[1]> Noun<gnc-agr[1]> | Noun | Noun Adj Noun | Adj Noun Noun;
```

Рисунок 3.5 – Грамматика, для извлечения фактов

В данном случае представлено несколько вариантов формирования определений:

- определением является предложение содержащее некоторое количество слов (термин) перед знаком пунктуации «-», а после – описание термина;

- определением является предложение, содержащее глагол «называть/называться» (и его различные формы). Часть предложения до указанного глагола можно задать как описание термина, а после – сам термин;
- определением является предложение, содержащее глагол «является». Часть предложения до указанного глагола можно задать как описание термина, а после – сам термин;
- определением является предложение, содержащее глагол «является». Часть предложения до указанного глагола можно задать как термин, а после – описание термина.

Грамматика взаимодействует с парсером не напрямую, а через корневой словарь. Корневой словарь является агрегатором файлов, используемых в проекте (рисунок 3.6).

```

3 import "base.proto";
4 import "articles_base.proto";
5 import "kwtypes_my.proto";
6 import "facttypes.proto";
7
8 TAuxDicArticle "ключевые_слова"
9 {
10     key = { "keyword.txt" type=FILE }
11 }
12
13 TAuxDicArticle "моя_грамматика"
14 {
15     key = { "tomita:first.cxx" type=CUSTOM }
16 }
17
18 call "называться"
19 {
20     key = { "называют" | "называются" | "называться" | "называется" }
21 }
22
23 is "являться"
24 {
25     key = { "является" }
26 }

```

Рисунок 3.6 – Корневой словарь

Корневой словарь состоит из статей, каждая из которых задает множество слов и словосочетаний, объединенных общим свойством. Ключом указанной газеттирной статьи («моя\_грамматика») являются все цепочки, выделяемые созданной грамматикой из файла. Согласно подходу в разделе 2, для извлечения фактов предварительно необходимо составить словарь ключевых слов. В данном

случае словарь формируется вручную экспертом предметной области либо как газеттирная статья, либо в файле формата TXT.

Статья call описывает возможные варианты глагола, который позволяет выделять определения из текста. Данную статью можно дополнить и другими ключевыми словами, например, «имеет» / «содержит» / «это» и др.

Интерпретация является последним этапом обработки текста в Томита-парсере. Процесс интерпретации состоит в распределении подцепочек из выделенной грамматикой цепочки по полям факта. Каждый факт можно представить как таблицу из одной строки и одной или нескольких колонок: в каждой ячейке этой таблицы содержится объект.

Описание типов фактов показано на рисунке 3.7.

Как было сказано ранее, в данном примере рассматривается тип факта - определение. Структуру такого факта можно представить как Term – термин, и Description – описание термина.

```
1 import "base.proto";
2 import "facttypes_base.proto";
3
4 message Definition: NFactType.TFact
5 {
6     required string Term = 1;
7     required string Description = 2;
8 }
9 }
```

Рисунок 3.7 – Описание типов фактов

Для записи цепочки в поле факта интерпретации в Томита-парсер используется оператор interp. Цепочка, распознанная символом за которым следует interp, записывается в поле факта, которое обозначается в скобках после interp. На рисунке 3.5 показано, как и в какие поля файлов должны интерпретироваться выделяемые цепочки из текста.

В результате запуска проекта и обработки текста будут сгенерирован файл в следующем виде (файл с отладочным выводом) – рисунок 3.8:

В случае, когда для анализа используются тексты большего объема результат не всегда отвечает ожиданиям. Например, если термин, состоит из нескольких слов, может обнаружиться ситуация, что не все слова будут согласованы между собой и



вопрос получится некорректным. Решить данную проблему можно путем написания множества шаблонов, покрывающих большинство употребляемых в русском языке конструкций, с совместным использованием помет Томита-парсер, которые позволяют задать согласование терминалов выделяемой цепочки.

Definition	
Term	Description
<a href="#">база данных</a>	программная система, предназначенная для хранения данных, модификации и извлечения необходимой информации для пользователя информации
<a href="#">система управления базами данных</a>	программная система, предназначенная для создания баз данных, модификации, поддержания баз данных в работоспособном состоянии и обеспечения безопасности баз данных
<a href="#">суперключ</a>	один или несколько столбцов, значения в которых позволяет однозначно отличить одну запись в таблице от другой.
<a href="#">потенциальный ключ</a>	один или несколько столбцов, которые позволяют однозначно отличить одну строку от другой и среди которых нет тех столбцов, которые обладают таким же свойством.

Рисунок 3.8 – Результат работы Томита-парсер (отладочный вывод)

Для того, чтобы файл с выделенными фактами подходил для дальнейшего использования его в качестве основы для генерации вопросов, он должен быть сформирован в формате XML. XML – это расширяемый язык разметки, позволяющий структурировать информацию разного типа, используя для этого произвольный набор инструкций. Томита-парсер поддерживает такую возможность.

В системе данный модуль представляет вкладка «Извлечение фактов» (рисунок 3.9). Аналогично модулю предварительной обработки текста, при нажатии кнопки «Извлечь» запускается программа `tomitaparser.exe`. В результате работы данного модуля генерируется XML-файл с извлеченными фактами. При корректной работе Томита-парсера, выводится статус-сообщение «Ок», при некорректной - «Ошибка».

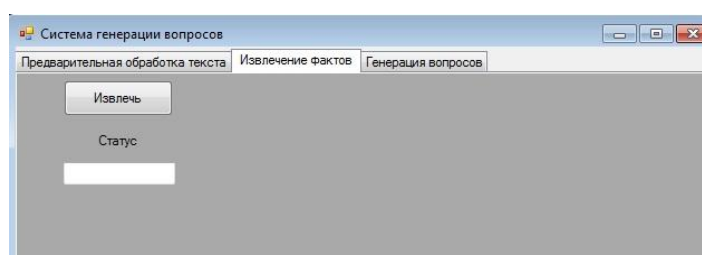


Рисунок 3.9 – Интерфейс модуля извлечения фактов



### 3.4 Модуль генерации вопросов

Модуль генерации вопросов отвечает за формирование списка вопросов по извлеченным из текста фактам, в данном случае – терминам. Полученный в модуле извлечения фактов XML-файл имеет определенную структуру информации, которую необходимо обработать.

Для реализации разбора XML-файла в C# предусмотрено пространство имен System.Xml, предоставляющее основанную на стандартах поддержку обработки XML.

Как было упомянуто в разделе «Генерация вопросов» Главы 2 для порождения вопросов будут использованы шаблоны. Заготовки формулировок хранятся в массиве строк `string[] forQuestions = {"Чем является ", "Что такое ", "Как называется "};`

Структура XML-файла с фактами представляется в следующем виде (рисунок 3.10.):

```
<?xml version="1.0" encoding="utf-8" ?>
<fdo_objects>
  <document url="" di="0" bi="-1" date="2016-05-19">
    <facts>
      <Definition FactID="0" LeadID="0" FieldsInfo="n0;n1;" pos="1418" len="128" sn="24" fw="0" lw="16">
        <Term val="ЭКОНОМИЧЕСКАЯ ТЕОРИЯ"/>
        <Description val="НАУКА О НАИБОЛЕЕ ПОЛНОМ УДОВЛЕТВОРЕНИИ ПОТРЕБНОСТЕЙ ЛЮДЕЙ , ПУТЕМ РАЦИОНАЛЬНОГО ИСПОЛЬЗОВАНИЯ РЕСУРСОВ"/>
      </Definition>
    </facts>
  </document>
</fdo_objects>
```

Рисунок 3.10 – Листинг XML-файла с извлеченными фактами

Из рисунка 3.10 видно, что информация о фактах хранится в тегах `<facts>`. Получение доступа к ним обеспечивает метод `XmlNode.SelectSingleNode("/fdo_objects/document/facts")`, выделяющий первый объект `XmlNode`, соответствующий выражению в скобках. Далее необходимо «запаковать» термин и его определение, хранящиеся соответственно в тегах `<Term>` и `<Description>`, в определенную структуру. Для сбора всех дочерних узлов использует алгоритм, включающий в себя два вложенных цикла `foreach`.

Непосредственно сам вопрос формируется конкатенацией заготовки формулировки вопроса и значения атрибута тега `<Description>`, например:

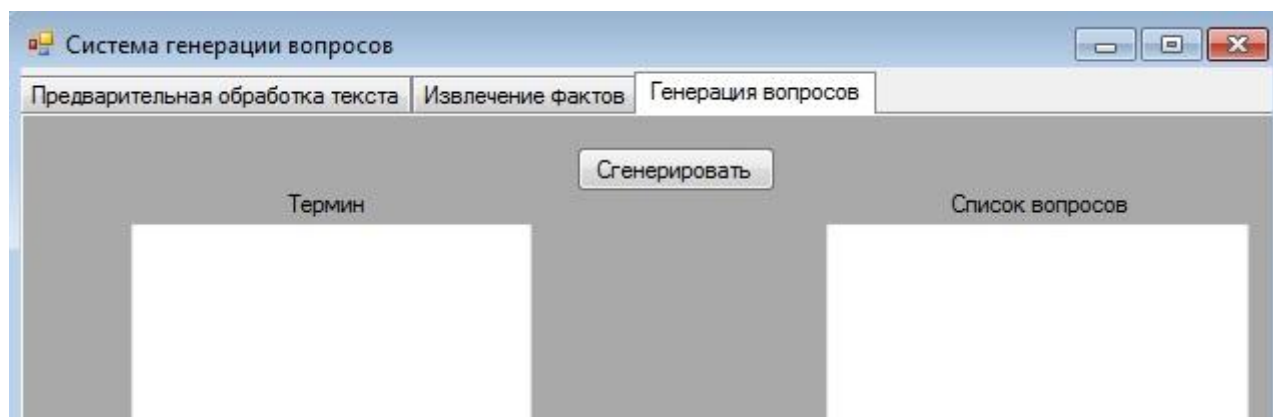
`forQuestions[0] + attr.Value + "?",`

где `forQuestions[0]` – один из элементов массива строк

`attr.Value` - значения атрибута тега `<Description>`;

Исходя из такой конструкции, будет сгенерирован вопрос: «Чем является наука о наиболее полном удовлетворении потребностей людей, путем рационального использования ресурсов?». Элементы массива `forQuestions` пронумерованы, что позволяет либо циклически использовать заготовки вопросов для конкатенации, либо случайно.

Модуль генерации вопросов соответствует вкладке «Генерация вопросов». Пользовательский интерфейс показан на рисунке 3.11. Кнопка «Сгенерировать» запускает процесс разбора XML-файла и последовательность действий, описанных выше. В результате, в текстовой форме «Термин» выводится ключевые понятия обработанного текста, а в поле «Список вопросов» - набор вопросов, составленных к терминам.



Полный программный код всех описанных модулей представлен в Приложении 3.

## **4 Тестирование разработанной системы**

### **4.1 Данные для экспериментов**

В качестве текстов для экспериментов предлагается рассматривать информационно-значимые тексты, например, научно-популярные, которые содержат в себе фактическую информацию. К такому типу текстов можно отнести:

- научные статьи;
- учебные пособия;
- диссертации;
- конспекты лекций;
- доклады;
- справочники
- и др.

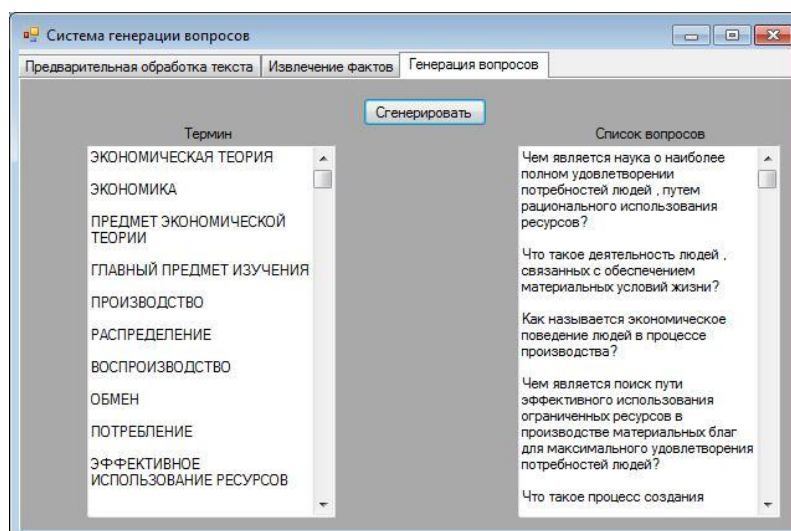
Научный стиль имеет свои особые характеристики, например интересен тот факт, почти каждая лексическая единица в научном стиле обозначает понятие или абстрактный предмет. Для специальных понятий научной сферы существуют особые лексические единицы, раскрывающие их содержание – термины. Термин — это слово или словосочетание, обозначающее понятие специальной области знания или деятельности [22]. О них упоминалось в главах 2 и 3.

Входными данными в задаче генерации вопросов выступает неструктурированный текст на русском языке на заданную тему. В ходе обработки и анализа текста формируется список возможных вопросов, позволяющих определить знание фактов из исходного текста.

Для тестирования было выбрано три научно-популярных текста из разных предметных областей: базы данных (Приложение 1), экономика [23], физика [24].

## 4.2 Результаты экспериментов

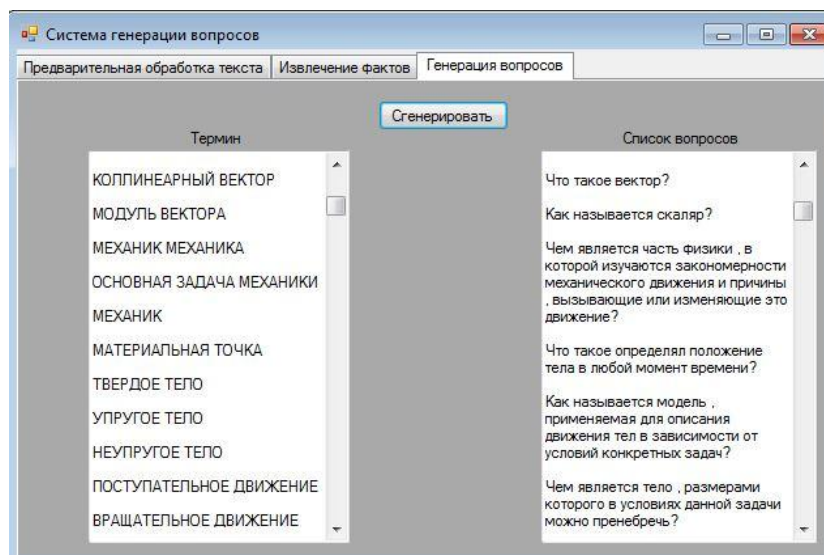
Для конспекта лекций по экономике в формате TXT, состоящего из 56 страниц общее число сгенерированных вопросов составило 173. Корректными из них, можно считать ~ 112. Процентное соотношение корректных вопросов к некорректным, составляет 66%. Под корректными в данном случае подразумеваются вопросы, имеющие правильный синтаксический порядок слов, согласованные части речи в предложении, ответами на которые являются выделенные термины. Было выделено два критерия оценки сгенерированных вопросов: грамматическая согласованность предложений; покрытие всех возможных фактов определенного типа из исходного текста. Пример работы системы для указанного текста представлен на рисунке 4.1.



Для научных материалов по физике в формате PDF, состоящих из 206 страниц общее число сгенерированных вопросов составило 259. Корректными из них, можно считать ~150. Процентное соотношение корректных вопросов к некорректным составляет 58%. Результат работы системы для данного текста показан на рисунке 4.2.

По базам данных был использован краткий конспект лекций в формате TXT, что повлекло за собой увеличение % корректных вопросов от общего числа сгенерированных вопросов по исходному тексту.

Поскольку программа-конвертер не может гарантировать на 100% корректное и точное преобразование PDF-файла в TXT-файл, то количество ложных срабатываний в данном случае больше, чем при изначальном использовании TXT-файла. Несмотря на это, результат обработки текста на естественном языке можно считать успешным.



## ЗАКЛЮЧЕНИЕ

В данной работе была описана методика автоматической генерации вопросов заданным неструктурированным текстам на русском языке. Для решения задач, поставленных в работе, были использованы системный подход, а также методы логического, системного анализа и синтеза.

В ходе данной квалификационной работы были выполнены следующие задачи:

- изучена предметная область, проведен обзор существующих систем извлечения информации;
- проведена классификация вопросов по сложности их генерации;
- предложена методика генерации вопросов по неструктурированным текстам на русском языке;
- разработана архитектура программной системы для генерации вопросов на русском языке;
- реализована методика автоматической генерации вопросов генерации вопросов по неструктурированным текстам на русском языке.

Методика была реализована как комплексная программная система и опробована на реальных научно-популярных текстах. Испытание реализованной методики дает возможность заключить, что предложенный подход позволяет генерировать определенные типы вопросов с целью их дальнейшего использования в качестве материалов для тестирования знаний фактов из разнообразных текстов. Разработанная система может оказать помощь преподавателям для разработки тестовых вопросов, а также позволит сэкономить время и затраты на данную деятельность.

Потенциал данной работы заключается в рассмотрении и применении иных подходов к извлечению фактов из текстов на русском языке, а также повышении качества шаблонов разработанной системы для генерации вопросов. В качестве

дальнейшего развития изучаемой темы, могут быть выбраны следующие направления:

- доработка модуля предварительной обработки текста для обеспечения возможности использования большего количества форматов текстовой информации (.doc(x), .rtf и др.);
- применение как методов машинного обучения, так и гибридных методов извлечения фактов;
- генерация вопросов с использованием фактов с модификациями;
- улучшение быстродействия системы за счет использования иных программных средств для реализации;
- интеграция с e-Learning системами

В настоящей работе были рассмотрены подходы, позволяющие решить актуальные проблемы современной компьютерной лингвистики.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Ильвовский, Д. А., Черняк Е. Л. Системы автоматической обработки текстов // Открытые системы. – 2014. – № 1. – С. 51-53.
2. Куртасов А.М., Швецов А.Н. Программа генерации учебных тестов на основе семантического подхода // Труды Международной научно-методической конференции «Информатизация инженерного образования» - ИНФОРИНО-2012. (10—11 апреля 2012 г. Москва). М.: Издательский дом МЭИ. С. 71-74.
3. Братчиков И.Л. Генерация тестовых заданий в экспертно-обучающих системах // Вестник Российского университета дружбы народов. Серия: Информатизация образования. 2012. № 2. С. 47-60.
4. Брюхов Д.О., Скворцов Н.А. Извлечение информации из больших коллекций русскоязычных текстовых документов в среде Hadoop // Труды 16-й Всероссийской научной конференции «Электронные библиотеки: перспективные методы и технологии, электронные коллекции» RCDL'2016. Дубна: ОИЯИ, 2014
5. Кагиров И.А., Леонтьева Ан. Б. Модуль синтаксического анализа для литературного русского языка // Труды СПИИРАН. Вып. 6. — СПб.: Наука, 2008
6. Блог компании Яндекс. Извлечение объектов и фактов из текстов в Яндексе. Лекция для Малого ШАДа: [Электронный документ]. — (<https://habrahabr.ru/company/yandex/blog/205198/>). Проверено 26.05.2016
7. Боровикова О.И. Организация порталов знаний на основе онтологий / Боровикова О.И., Загорулько Ю.А. // Компьютерная лингвистика и интеллектуальные технологии: Труды международного семинара «Диалог 2002» (Протвино, 6-11 июня 2002 г.). – Москва: Наука, – 2002. –Т.2, –С.76–82
8. Зубарев В.С. Исследование и разработка алгоритма семантической информационно-поисковой системы: [Электронный документ]. — (<http://storage.vas3k.ru/files/Main.pdf>). Проверено 26.05.2016



9. Кручинин В. В., Морозова Ю. В. Модели и алгоритмы генерации задач в компьютерном тестировании // Известия ТПУ. 2004. №5.
10. Тарасенко С. В., Рязанова Н. Ю. Анализ методов автоматической генерации вопросов на естественном языке: [Электронный документ] // Инженерный вестник. – 2015. – № 12. – (<http://engbul.bmstu.ru/doc/829687.html>). Проверено 26.05.2016
11. Mitkov, R., Ha, L.A., Karamanis, N: A computer-aided environment for generating multi-choice test items. Nat. Lang. Eng. 12, 177-194, 2006
12. Kunichika, H, Katayama, T, Hirashima, T, and Takeuchi, A. Automated question generation methods for intelligent English learning systems and its evaluation. Proceeding of ICCE2004, 2003
13. Ontosminer: [Электронный документ]. — (<http://my-eventos.com/solution/ontosminer>). Проверено 26.05.2016
14. IKnow: [Электронный документ]. — (<http://www.intersystems.com/ru/our-products/embedded-technologies/iknow>). Проверено 26.05.2016
15. Астраханцев Н. А. Автоматическое извлечение терминов из коллекции текстов предметной области с помощью Википедии // Труды ИСП РАН. 2014. №4.
16. Описание языка LSPL (1.0.1): [Электронный документ]. — ([www.lspl.ru/articles/LSPL\\_Refguide\\_13.pdf](http://www.lspl.ru/articles/LSPL_Refguide_13.pdf)). Проверено 26.05.2016
17. Власов, Д.Ю. Автоматизация извлечения отношений между понятиями из текстов естественного языка. / Д.Ю. Власов, Д.Е. Пальчунов, П. А. Степанов // Вестник НГУ. Серия: Информационные технологии. – 2010 – Т. 8. – Выпуск 3. – С. 272–274
18. Большакова, Е. И. Методы построения систем автоматического анализа текста на базе лингвистических шаблонов: [Электронный документ]. — (<http://nlpseminar.ru/lecture58/>). Проверено 26.05.2016
19. Переговорщики.РУ: [Электронный документ]. — (<http://peregovorshiki.ru/tipy-voprosov/iskusstvo-zadavat-voprosy-mini-trening-myshleniya.html>). Проверено 26.05.2016

- 20.Xpdf: A PDF Viewer for X: [Электронный документ]. – (<http://www.foolabs.com/xpdf/home.html>). Проверено 26.05.2016
21. Томита-парсер. Руководство разработчика. Версия 1.0: [Электронный документ]. – (<https://tech.yandex.ru/tomita/doc/dg/concept/about-docpage>). Проверено 26.05.2016
- 22.Лингвистический энциклопедический словарь / гл. ред. В.Н. Ярцева. – 2-е изд., доп. – М. : Большая рос. энцикл., 2002. – 709 с.
- 23.Конспект лекций по экономике / А. А. Нечитайло, А.А. Гнутова. – Самара: Изд-во СГАУ, 2012. - 56 с.
- 24.Механика, молекулярная физика и термодинамика: конспекты лекций / Е.В.Полицинский. – Юргинский технологический институт Национального исследовательского Томского политехнического университета, 2010 – 206 с.

## ПРИЛОЖЕНИЕ 1

### Пример исходного текста

База данных – программная система, предназначенная для хранения данных, модификации и извлечения необходимой информации для пользователя информации. Физические данные хранятся на компьютере в файле.

Система управления базами данных - программная система, предназначенная для создания баз данных, модификации, поддержания баз данных в работоспособном состоянии и обеспечения безопасности баз данных.

Примеры СУБД:

- Access – фирма MicroSoft
- Sql Server – фирма MicroSoft
- ORACLE – фирма ORACLE

Информация в большинстве баз данных для пользователей представляется в виде таблиц, которые обладают рядом свойств.

Таблицы в базах данных имеют уникальные имена (2-ух одинаковых быть не может).

Столбцы каждой таблицы имеют уникальные имена внутри одной таблицы.

Порядок расположения строк и столбцов в таблице произвольный.

Данные в столбце должны быть однотипными.

Типы данных

- Символьные (~2000-4000 символов)
- Текстовые (несколько ГБ)
- Числовые
- Дата/Время
- Логические
- Двоичные (звук, картинки)
- Денежные
- Интервальные.

Таблицы не могут хранить множественную информацию. Данные в ячейках должны быть атомарными.

Не рекомендуется дублировать данные в таблице. При дублировании информации в таблицах может быть нарушена целостность данных.

В таблицах базы данных строки не могут повторяться (не может быть двух одинаковых строк).

Каждая таблица должна иметь первичный ключ Primary key.

Один или несколько столбцов, значения в которых позволяет однозначно отличить одну запись в таблице от другой, называют суперключом (superkey).

Один или несколько столбцов, которые позволяют однозначно отличить одну строчку от другой и среди которых нет тех столбцов, которые обладают таким же свойством, называют потенциальным ключом (Candidate key).

Один из потенциальных ключей выбирается для однозначного определения записи в таблице. Этот выбранный ключ будет называться первичным ключом (Primary Key).

Оставшиеся потенциальные ключи – альтернативные ключи (Alternate Key).

Рекомендации по выбору первичного ключа:

Первый ключ должен содержать как можно меньше столбцов.

Желательно, чтобы в столбце первичного ключа информация не изменялась.

Желательно, чтобы объем информации в столбцах первичного ключа был бы минимален.

Искусственно введенный ключ называется суррогатным ключом (в таблице Сотрудник – это столбец Номер).

Для каждой связи таблица, в которой связь подходит к первичному ключу называется главной, вторая подчинённой.

Тот столбец (или столбцы), к которым подходит связь в подчиненной таблице, называют вторичным ключом (внешним ключом, Foreign Key).

Базы данных, которые построены на таблицах, обладающими перечисленными выше свойствами, называют реляционными базами данных (Relational Database Management System).

Теорию реляционных баз данных предложил Эдвард Кодд в 1970 г. В конце 70-х годов на рынке появились первые реляционные СУБД.

Помимо реляционных СУБД существуют и другие виды СУБД, например, иерархические, сетевые, многомерные, объектно-ориентированные и пр.

## ПРИЛОЖЕНИЕ 2

### Список терминалов

Таблица П2.1 – Список используемых терминалов

Терминал	Значение
Word	Любое слово, состоящее из букв русского или латинского алфавита. Также разрешаются слова записанные через дефис. Под это определение не попадают цепочки, которые содержат знаки пунктуации (кроме дефиса), специальные ASCII символы и цепочки цифр
Noun	Существительное (слово с граммемой «S»). Сюда не входят имена, фамилии и отчества
Verb	Глагол
Adj	Прилагательное
Punct	Точка
Comma	Запятая
Hyphen	Тире

## ПРИЛОЖЕНИЕ 3

### Исходный код разработанной программы

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Xml;
using System.IO;

namespace GenInterface
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            string[] forQuestions = { "Чем является ",
                                      "Что такое ",
                                      "Как называется " };

            XmlDocument xDoc = new XmlDocument();
            xDoc.Load("facts.xml");
            //XmlElement xRoot = xDoc.DocumentElement;
            XmlNode xn =
xDoc.DocumentElement.SelectSingleNode("/fdo_objects/document/facts");
            XmlNodeList xnList = xn.ChildNodes;
            int count = 0;
            List<string> term = new List<string> { };
            List<string> questions = new List<string> { };
            foreach (XmlNode node in xnList)
            {
                foreach (XmlNode child in node.ChildNodes)
                {
                    if (child.Name == "Term")
                    {
                        XmlNode attr = child.Attributes.GetNamedItem("val");
                        if (attr != null)
                        {
                            term.Add(attr.Value);
                        }
                    }
                    if (child.Name == "Description")
                    {
                        if (count >= forQuestions.Length)
                            count = 0;
                        XmlNode attr = child.Attributes.GetNamedItem("val");
```

```

        String question = forQuestions[count] +
attr.Value.ToLower().Trim(new Char[] { ',', ' ' }) + "?";
        if (attr != null)
        {
            questions.Add(question);
        }
        count = count + 1;
    }
}

for (int i = 0; i < term.Count; i++)
{
    Term.Text += term[i].ToString();
    Term.Text += Environment.NewLine;
    Term.Text += Environment.NewLine;
}

for (int i = 0; i < questions.Count; i++)
{
    Description.Text += questions[i].ToString();
    Description.Text += Environment.NewLine;
    Description.Text += Environment.NewLine;
}

}

void inputFile()
{
    OpenFileDialog openFileDialog1 = new OpenFileDialog();

    openFileDialog1.InitialDirectory = "c:\\";
    openFileDialog1.Filter = "pdf files (*.pdf)|*.pdf|All files (*.*)|*.*";
    openFileDialog1.RestoreDirectory = true;

    string inputForExtract = "";

    if (openFileDialog1.ShowDialog() == DialogResult.OK)
    {
        try
        {
            inputForExtract = openFileDialog1.FileName;
        }

        catch (Exception ex)
        {
            MessageBox.Show("Error: Could not read file from disk. Original
error: " + ex.Message);
        }
    }

}

private void button2_Click(object sender, EventArgs e)
{
    OpenFileDialog openFileDialog1 = new OpenFileDialog();

    openFileDialog1.InitialDirectory = "c:\\";
    openFileDialog1.Filter = "pdf files (*.pdf)|*.pdf|All files (*.*)|*.*";
    openFileDialog1.RestoreDirectory = true;

    string inputForConvert = "";

```



```

        if (openFileDialog1.ShowDialog() == DialogResult.OK)
        {
            try
            {
                inputForConvert = openFileDialog1.FileName;
                //File.Text = inputForConvert;
            }

            catch (Exception ex)
            {
                MessageBox.Show("Error: Could not read file from disk. Original
error: " + ex.Message);
            }
        }

        System.Diagnostics.Process proc = new System.Diagnostics.Process();
        proc.StartInfo.FileName = "pdftotext.exe";
        proc.StartInfo.Arguments = "-enc UTF-8 " + inputForConvert;

        proc.Start();
        proc.WaitForExit();

        Status1.Text = "Ok";
    }

    private void button3_Click(object sender, EventArgs e)
    {
        System.Diagnostics.Process proc = new System.Diagnostics.Process();
        proc.StartInfo.FileName = "Tomita\\tomitaparser.exe";
        proc.StartInfo.Arguments = "Tomita\\config.proto";

        proc.Start();
        proc.WaitForExit();
        Status2.Text = "Ok";
    }
}

```

## ЗАКЛЮЧИТЕЛЬНЫЙ ЛИСТ РАБОТЫ

Магистерская диссертация выполнена мною самостоятельно.  
Использованные в работе материалы и концепции из опубликованной научной литературы и других источников имеют ссылки на них.

Список использованных источников: 24 наименований.

Работа выполнена на 57 листах,  
включая приложения на 7 листах.

Один экземпляр сдан на кафедру.

Подпись \_\_\_\_\_ / \_\_\_\_\_ /  
(фамилия, инициалы)

Дата «\_\_\_\_\_» \_\_\_\_\_ 20\_\_\_\_ г.