МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ВЫСШАЯ ШКОЛА ЭКОНОМИКИ

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ИНСТИТУТ ЭЛЕКТРОНИКИ И МАТЕМАТИКИ

Магистерская диссертация

(230100.68 Информатика и вычислительная техника

Магистерская программа «Информатика и вычислительная техника»)

ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ПРОГРАММНОЙ СИСТЕМЫ ГРАФЕМАТИЧЕСКОГО АНАЛИЗА ТЕКСТОВ НА РУССКОМ ЯЗЫКЕ

Выполнил:
студент 2 курса магистратуры группы АПМ-121 А. А. Первушин

_____
подпись

Научный руководитель:
доцент Э. С. Клышинский
Оценка руководителя:
_____

_____
подпись

Представлена на кафедру
«___»_____2013г.

_____
подпись принявшего работу

Допущена к защите
Зав. Кафедрой С.Р. Тумковский

_____
подпись

Итоговая оценка: _____

_____
подпись

Москва 2013

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время все большее распространение получают информационные системы и базы данных, содержащие в себе огромное количество текстовой информации. Крупные образовательные центры организуют в Интернете для студентов и сотрудников базы научных статей, авторефератов, многие организации предоставляют доступ к ресурсам электронных библиотек, оргкомитеты конференций публикуют тысячи полных текстов докладов и т. п. Количество электронной информации возросло настолько, что человеку просто не по силам проанализировать её самостоятельно, хотя необходимость проведения такого анализа вполне очевидна, ведь в этих данных заключены знания, которые могут быть использованы при принятии решений. Для автоматической обработки таких данных широко используются методы компьютерной лингвистики.

Компьютерная лингвистика (КЛ) – одно из направлений прикладной лингвистики, которое ориентировано на использование различных компьютерных инструментов (программы, компьютерные технологии обработки данных и их организации) для моделирования функций языка в тех или иных условиях, ситуациях, сферах и т. д., а также вся сфера применения компьютерных моделей языка в лингвистике и смежных дисциплинах. На практике же к термину компьютерная лингвистика относят всё, что связано с использованием компьютеров в языкознание. Компьютерную лингвистику так же часто называю автоматической обработкой текста или машинной лингвистикой [1].

Компьютерная лингвистика возникла на стыке таких наук, как лингвистика, информатика, математика и искусственный интеллект. Компьютерная лингвистика решает различные задачи автоматической обработки текстов на естественных языках такие как:

– машинный перевод;

- распознавание речи;
- информационный поиск;
- автоматическая классификация и реферирование документов;
- автоматическая лингвистическая обработка и составление машинных словарей;
- построение лингвистических процессоров, обеспечивающих общение пользователей с автоматизированными интеллектуальными системами;
- извлечение фактографической информации из неформализованных текстов;
- и многие другие [2, 3].

Естественный язык представляет собой сложную многоуровневую структуру символов, используемую для общения между людьми, которая постоянно меняется в процессе практической деятельности человека. Это является главной сложностью задач, решаемых компьютерной лингвистикой. Еще одной важной сложностью является многообразие естественных языков и существенные отличия между ними: в лексике, в морфологии, в синтаксисе, в различных способах выражения одного и того же смысла [4].

В наши дни сложно представить существование таких поисковых гигантов как Google и Яндекс без использования методов компьютерной лингвистики. Крупные агентства СМИ, имеющие свои интернет порталы, широко использую компьютерную лингвистику для решения большого количества задач, например, для поиска похожих статей.

Практически все продукты автоматической обработки текста обладают функциональностью первичного анализа текста - графематического. В некоторых программных продуктах данная функциональность сводится к разбиению текста на слова, другие в дополнении к этому осуществляют поиск текстовых конструкций по определённым шаблонам, разбиение текста на предложения и т. д.

В данной диссертации будут подробно рассмотрены задачи графематического анализа текста, предложены методы их решения, а так же представлена программная реализация, осуществляющая решение поставленных задач.

1 Автоматическая обработка текстов на естественном языке

Автоматическая обработка текстов на естественном языке – сложный многоуровневый процесс. Наилучшее качество результатов анализа можно получить только проведя полный анализ текста. Однако в процессе создание такого анализа возникает большое количество сложностей, главная из которых – создание полноценной экспертной системы.

Для проведения полноценного анализа текста на естественном языке система должна иметь возможность проанализировать исходный текст с точки зрения синтаксиса (структуры предложений), семантики (понятий, применяемых в тексте) и прагматики (правильности употребления понятий и целей их употребления). После этого система должна сгенерировать свой отклик во внутреннем представлении, пригодном для логического вывода, и просинтезировать свой отклик на естественном языке [2].

Система, проводящая полный анализ текста на естественном языке должна состоять из нескольких модулей, осуществляющих поэтапный анализ исходного текста:

- графематический анализ;
- морфологический анализ;
- предсинтаксический анализ;
- синтаксический анализ;
- постсинтаксический анализ;
- семантический анализ.

Графематический анализ – первичный этап в процессе автоматической обработки текстов на естественном языке. Основной задачей графематического анализа является выделение структурных единиц из входного текста, а именно предложений абзацев, слов, знаков препинания и т. д. Входной текст может иметь как линейную структуру, содержащую единый фрагмент текста, так и

нелинейную. В этом случае текст содержит различные структурные единицы: основной текст, заголовки, примечания, комментарии и т. д. Выходными данными этапа графематического анализа является графематическая таблица, которая в свою очередь служит входными данными для следующего этапа автоматической обработки текста – морфологического анализа.

Задача морфологического анализа – определение нормальной формы, от которой была образована данная словоформа, а так же получение набора её морфологических характеристик. Это делается для того, чтобы в последующих этапах анализа использовать только нормальную форму слова, а не все его словоформы и использовать морфологические характеристики для проверки согласованности слов. Нормальной формой слова называется форма, принятая для обозначения понятия, связанного с данным словом. Нормальной формой считается слово в именительном падеже и единственном числе. Морфологические характеристики – это набор пар «ключ, значение». В роле ключа выступает, например, род, число, падеж, склонение, время и другие признаки слов, используемые в русском языке. Значением является какое-либо конкретное значение, которое может принимать данный признак (ключ). Например, ключ «число» может принимать значения «единственное», «множественное» или «двойственное», ключ «род» может принимать значения «мужской», «женский», «средний» и т. д.

Предсинтаксический анализ выполняет следующие функции: объединение отдельных лексических единиц в одну синтаксическую и, наоборот, разделение одной лексической единицы на несколько синтаксических. В одну синтаксическую единицу объединяются неразрывные словосочетания. Например, «золотые руки», «тянуть за язык» и т. д. Еще до проведения синтаксического анализа есть возможность провести поверхностный анализ структуры разбираемого предложения. Синтаксический анализ в свою очередь сможет, опираясь на результаты такого анализа, отбросить часть вариантов синтаксического разбора предложения. Анализ

структуры разбираемого предложения называется синтаксической сегментацией. Первая задача синтаксической сегментации – используя определенные правила языка, уменьшить количество омонимов, соответствующих каждой словоформе. Второй задачей является выделение различных синтаксических конструкций. Например, выделение простых предложений в составе сложных для проведения в дальнейшем над ними независимого синтаксического анализа, выделение синтаксических групп (именных, глагольных, и т. д.) [2].

Задачей синтаксического анализа является определение роли слов и их связи между собой. Результатом данного этапа является набор синтаксических деревьев, отражающих такие связи. Синтаксический анализ является самым сложным этапом анализа текста на естественном языке. В связи с неоднозначностью как входных данных (одна словоформа может быть получена от нескольких различных нормальных форм), так и самих правил разбора, как правило, существует большое количество различных вариантов разбора [2].

Постсинтаксический анализ решает две следующие задачи. Первая заключается в необходимости уточнить смысл слов, который выражается с помощью различных средств языка: предлогов, префиксов и т. д. Проблематика второй задачи заключается в том, что одну и ту же мысль можно выразить разными конструкциями языка. Например, в многоязычно диалоговой системе одна и та же мысль может быть выражена различными синтаксическими конструкциями. В связи с этим появляется необходимость в нормализации дерева, чтобы свести конструкции, которые выражают одно действие различным образом для разных ситуаций, к одному нормализованному дереву. Так же на этапе постсинтаксического анализа может проводиться поиск изменяемых словосочетаний, составные части (слова) которых могут быть разделены другими словами [2].

Семантический анализ – завершающий этап автоматической обработки текста. На данном этапе производится смысловой анализ текста – уточняются связи, которые не смог уточнить постсинтаксический анализ, т. к. большое количество ролей выражается не только при помощи средств языка, но и с помощью значения слова. Вторая задача семантического анализа – исключить некоторые значения слов, а иногда даже целые варианты разбора как семантически несвязные [2].

Описанные выше этапы отражают всю сложность процесса автоматической обработки текста. Все они тесно связаны между собой – выходные данные одного этапа являются входными данными для другого. Чем выше качество и полнота входных данных, тем выше качество полученных результатов. В наши дни существует большое количество систем и модулей морфологического анализа, дающих хороший и качественный результат. Но, как будет показано в обзоре ниже, на текущий момент нет качественной системы графематического анализа.

2 Постановка задачи

Целью данной работы является проектирование и разработка программной системы графематического анализа текстов на русском языке.

2.1 Функциональные требования

Входными данными для системы является текстовый файл в кодировке cp1251, содержащий текст на русском языке. Входной текст не имеет строго заданного формата. Размер входного файла ограничен несколькими гигабайтами, весь файл может не помещаться в оперативную память компьютера. В связи с этим необходимо реализовать чтение и анализ файла частями.

Система должна соответствовать следующим требованиям:

- выделение в исходном тексте предложений;
- токенизация исходного текста (выделение в тексте слов, чисел, и иных токенов, в том числе, например, нахождение границ предложений) [5];
- поиск в тексте сокращений, аббревиатур;
- определение расшифровок для найденных сокращений и аббревиатур;
- возможность поиска текстовых конструкций с использованием шаблонов;
- возможность добавления новых шаблонов для поиска;
- простейшие средства визуализации;
- возможность экспорта результата в систему управления базами данных (СУБД) для проведения дальнейшего анализа;
- система должна функционировать без предварительного машинного обучения на корректно обработанной экспертами коллекции текстов;
- программная реализация должна быть оптимальной, т. е. затрачивать разумное количество времени, памяти и прочих ресурсов в процессе своей работы;
- должна быть предусмотрена возможность последующей интеграции с системой морфологического анализа «Кросслейтор».

2.2 Методы

Программный модуль должен предоставлять следующие методы для возможности дальнейшей интеграции с другими программными продуктами:

- Загрузка необходимых словарей. Входным параметром для методов, осуществляющих загрузку словарей, является путь до файла, в котором содержится конкретный словарь. При необходимости могут быть заданы дополнительные параметры, например, формат словаря.
- Загрузка шаблонов для поиска. На вход методу подаётся список шаблонов или путь до входного файла, содержащего шаблоны для поиска. Список и файл с шаблонами не имеют строго заданного формата.
- Анализ входного текста. На вход подаются абсолютный или относительный путь до текстового файла, над которым необходимо провести анализ и размер анализируемой порции данных;
- Выдача размеченного текста. Возвращает строку с исходным текстом, содержащую выделенные в тексте предложения, сокращения, текстовые конструкции, найденные с использованием шаблонов. Входные параметры отсутствуют.
- Экспорт результатов в базу данных. Входными параметрами являются параметры подключения к базе данных, такие как ip-адрес, порт, имя пользователя, пароль и другие. Перечень параметров может быть изменён в зависимости от выбранной СУБД.
- Получение списка выделенных в тексте предложений. Метод возвращает список предложений, содержащий позицию начала предложения и количество символов в нём. Список отсортирован по позиции начала предложения, предложения не пересекаются между собой.
- Получение списка выделенных в графематических дескрипторов (сокращений, текстовых конструкций, соответствующим заданным шаблонам и т. д.). Выходными данными является список, содержащий позицию начала дескриптора в тексте, длина дескриптора (количество

символов), тип дескриптора. Для найденных в тексте сокращений дополнительно должны быть возвращены их расшифровки.

2.3 Требования к реализации

Программная система должна быть представлена в виде подключаемого модуля (библиотека или набор файлов исходного кода). Разработка должна вестись с использованием свободного или бесплатного для некоммерческого использования программного обеспечения. Программа должна взаимодействовать с СУБД, которая так же является бесплатной для некоммерческого использования.

Программа должна быть кроссплатформенной и работать в таких операционных системах как Windows и Linux.

3 Обзор существующих решений

За время развития компьютерной лингвистики появилось большое количество продуктов и технологий, решающих различные задачи этой науки. Это и различные системы поиска, машинного перевода, индексирования текстов, извлечения из текстов фактографической информации и т. д.

Отдельным классом среди этих программных продуктов являются системы, осуществляющие полную автоматическую обработку текстов на естественном языке. Кроме того, существуют программы, представляющие собой модули и библиотеки, решающие одну или несколько задач автоматической обработки тексты. Например, к ним можно отнести модули морфологического и синтаксического анализа.

В обзоре будут представлены программные продукты, осуществляющие автоматическую обработку текстов на естественном языке и содержащие в себе модуль графематического анализа, или продукты, которые являются таким модулем. Минимальными требования для включения обзор являются:

- выделение предложений из входного текста;
- токенизация;
- поддержка русского языка.

Минимальные требования обусловлены тем, что существует большое количество продуктов, в функциональности которых заявлена составляющая графематического анализа, которая представляет собой разбиение текста на слова. Примерами таких продуктов являются некоторые системы классификации документов, системы извлечения из текстов полезной информации и т. д. В ряде случаев простой токенизации текстов достаточно для реализации функциональности таких систем.

При проведении сравнения были изучены следующие программные продукты:

- Продукты компании Lingsoft.

– Greeb;
– FreeLing;
– AOT (сокр. автоматическая обработка текста);
– Solarix [6, 7];

Компания Lingsoft занимается производством различных продуктов, решающих задачи компьютерной лингвистики. Среди продуктов присутствую компоненты грамматического разбора, морфологического анализа и лемматизации для английского, немецкого, финского, датского, норвежского, шведского, эстонского и русского языков. Это продукты, которые могут быть использованы при разработке других систем. Основные недостатки:

– продукты являются коммерческими;
– отсутствует описание правил и алгоритмов их работы.

В списке демонстрационных продуктов нет продукта с поддержкой русского языка [6, 8].

Следующий изученный продукт – Greeb. Данный продукт представляет собой графематический анализатор, разработанный на языке Ruby. Принцип его функционирования основан на регулярных выражениях. Greeb выполняет две следующие задачи:

– токенизация текста;
– сегментация текста (разбивка на предложения).

Среди поддерживаемых естественных языков заявлены русский и английский. Продукт распространяется под лицензией MIT, его исходные коды доступны на GitHub. Недостатками данного продукта являются:

– отсутствие возможности добавления новых регулярных выражений без редактирования исходного кода программы;
– нет возможности поиска сокращений и аббревиатур с использованием словаря;

– отсутствие функциональности экспорта результатов анализа, что существенно усложняет интеграцию с другими продуктами [6].

FreeLing – проект по созданию свободного набора инструментов обработки текстов на естественном языке, созданный в Политехническом университете в Каталонии. Заявлена поддержка следующих языков: русский, английский, итальянский, испанский, португальский, астурийский, валийский, галисийский, каталанский. Библиотеки проекта и их исходные коды распространяются под лицензией GPL и бесплатны для некоммерческого использования [6, 9].

Модуль графематического анализа проекта FreeLing обладает следующей функциональностью:

– автоматическое определение языка входного текста;
– выделение в тексте предложений;
– токенизация.

Среди достоинств продукта стоит отметить гибкость настройки. Например, пользователь сам обозначает символы, характеризующие конец предложения для каждого языка, перечень скобок и кавычек, которые должны иметь закрывающую пару и т. д. Основным недостатком продукта является отсутствие поиска сокращений и аббревиатур с использованием словаря [9].

Проект АОТ – совокупность инструментов обработки текста на естественном языке, которые были разработаны во время создания системы автоматического перевода ДИАЛИНГ. Программный пакет состоит из нескольких компонентов – лингвистических процессоров, которые последовательно обрабатывают входной текста. Авторы проекта выделяют такие компоненты, как графематический, морфологический, синтаксический и семантический анализаторы. Начиная с 2010 года продукт распространяется на условиях лицензии LGPL, а его исходные коды доступны в SVN-репозитории [6].

Модуль графематического анализа (ГрафАн) в проекте АОТ – это программа начального анализа текста на естественном языке, который представляет собой цепочку ASCII символов. ГрафАн вырабатывает информацию, которая используется для дальнейшей обработки морфологическим и синтаксическим процессорами. Модуль графематического анализа обладает следующей функциональностью:

- разделение входного текста на слова, разделители и т.д.;
- сборка слов, написанных в разрядку;
- выделение устойчивых оборотов, не имеющих словоизменительных вариантов;
- выделение ФИО, когда имя и отчество написаны инициалами;
- выделение электронных адресов и имен файлов;
- выделение предложений из входного текста;
- выделение абзацев, заголовков, примечаний [10, 11, 12].

Недостатками ГрафАн являются:

- отсутствие механизма добавления новых шаблонов поиска;
- отсутствие механизма поиска сокращений по словарю.

Solarix – проект, целью которого является разработка кросс-платформенных и легко портируемых программных инструментов для работы с естественными языками. Графематический модуль обладает следующей функциональностью:

- токенизация;
- разбиение текста на предложения.

За разбиение текста на предложения отвечает сегментатор предложений. Данный модуль содержит несколько интересных функций, выделяющих его на фоне остальных продуктов:

- параметр, задающий максимальную длину предложения (на случай наличия опечатки).

– Использование другого модуля под названием «Лексикон» для определения полных словоформ. Точка после полной словоформы является безусловным разделителем предложений.
– Имеется возможность определить список специальных токенов. Если после такого токена идет полная словоформа, начинается она с заглавной буквы и в лексиконе она отмечена, как не начинающаяся с заглавной буквы, то специальный токен является границей предложения [13].

Недостатками графематической составляющей проекта Solarix является отсутствие реализации полноценного словаря сокращений, а так же отсутствие возможности добавления новых шаблонов для поиска текстовых конструкций.

Кроме рассмотренных выше продуктов стоит так же перечислить проекты, посвященные автоматической обработки текстов на естественном языке и имеющие в своём составе модули графематического анализа, которые на текущий момент не имеют поддержки русского языка:

– Apache OpenNLP [14];
– Stanford CoreNLP [15];
– Twitter NLP and Part-of-Speech Tagger [16];
– Natural Language Toolkit (NLTK) [17];

Как показало проведенное исследование и сравнение функциональности систем графематического анализа, у каждого продукта есть свои отличительные плюсы и минусы. Наиболее успешными и многофункциональными из них являются AOT, FreeLing и Solarix. Ни в одном из изученных продуктов нет такой функциональности, как поиск сокращений с использованием словаря и лишь в некоторых реализован поиск текстовых конструкций с использованием шаблонов.

4 Описание модуля графематического анализа

Как писалось ранее, графематический анализ – это начальный этап автоматической обработки текстов на естественном языке. На вход графематическому анализу подается текстовый файл.

Модуль графематического анализа должен выполнять следующие задачи:

- разбиение исходного текста на слова, разделители и т. д.;
- выделение в тексте сокращений и аббревиатур;
- поиск в тексте ФИО в тех случаях, когда имя и отчество представлены в виде инициалов;
- выделение электронных адресов, URL – адресов и т. д.;
- определение в исходном тексте границ предложений.

Выходные данные должны быть представлены в формате, пригодном для передачи на следующий этап – морфологический анализ, и содержать следующую информацию:

- список выделенных в тексте предложений;
- список найденных графематических дескрипторов.

В списке предложений должны содержаться:

- порядковый номер предложения в тексте;
- позиция его начала в тексте;
- его длина (количество символов);
- текст предложения.

Список графематических дескрипторов должен содержать в себе следующие данные:

- тип дескриптора;
- позиция начала дескриптора в тексте;
- его длина (количество символов);
- текст дескриптора;

– информацию о том, может ли данный дескриптор стоять в конце предложения;
– семантическую информацию о дескрипторе (для сокращений и аббревиатур).

Минимальный поддерживаемый набор типов дескрипторов представлен в табл. 1.

Таблица 1 – поддерживаемые типы дескрипторов.

Тип дескриптора	Обозначение	Пример
последовательность кириллических символов	RUSSIAN_LEX	«мама»
последовательность символов из латиницы	FOREIGN_LEX	«mother»
последовательность числовых символов	NUM_SEQ	«1941»
символ открывающейся скобки	OPEN_BRACKET	«(», «[», «{»
символ закрывающейся скобки	CLOSE_BRACKET	«)», «]», «}»
символ открывающейся кавычки	OPEN_QUOTE	«“», ««»
символ закрывающейся кавычки	CLOSE_QUOTE	«”», «»»
символ пунктуации	PUNCTUATION_CHAR	«;», «;», «:», «!», «.», «?», «…»
дескрипторы, не удовлетворяющие	OTHER_LEX	«4-метилендиоксиме

предыдущим 8 правилам		мфетамин», «2000-м»
сокращение или аббревиатура	SHORT_WORD	«т. д.», «ЦНС», «др.»
последовательности символов, определенная как конец предложения	SENTENCE_END	«…», «!», «?», «.»
прямая речь	DIRECT_SPEACH	

Для решения поставленных задач необходимо провести анализ текста в несколько этапов:

1) поиск в тексте сокращений и аббревиатур;
2) поиск текстовых конструкций с использованием шаблонов;
3) выделение границ предложений и прямой речи, окончательная разметка входного текста.

4.1 Поиск сокращений, аббревиатур

Первый этап анализа – поиск в тексте сокращений, аббревиатур. На данном этапе важно выявить наиболее полный список сокращений, т.к. они содержат в себе символ «.», являющийся признаком конца предложения. Чем больше сокращений будет выявлено, тем выше будет точность определения границ предложений. Выявленные сокращения и аббревиатуры, а так же их расшифровки будут использоваться на последующих этапах анализа текста – синтаксическом и семантическом.

Ввиду того, что при поиске перечисленных выше текстовых конструкций необходимо использовать словарь важным является вопрос, связанный с производительностью системы. Ведь чем больше записей содержит в себе словарь, тем большее количество сравнений необходимо провести. Поэтому

необходимо производить поиск по словарю с наименьшими временными затратами.

4.2 Поиск по шаблонам

Следующий этап – поиск в тексте графематических дескрипторов с использованием шаблонов. На данном этапе в тексте выделяются следующие дескрипторы:

- ФИО, когда имя и отчество представлены в виде инициалов;
- email и URL – адреса;
- даты в числовых форматах;
- номера телефонов;
- и т. д.

Предполагается, что каждый выделенный шаблон будет полностью принадлежать одному из предложений в тексте, т.е. не будет существовать шаблонов, которые принадлежат сразу нескольким предложениям. Это является важным фактором для такой задачи, как определение границ предложений.

4.3 Завершающий этап анализа текста.

На завершающем этапе графематического анализа текста решаются следующие задачи:

- выделение прямой речи;
- определение границ предложений;
- разметка текста – деление на слова, знакам препинания и т.д.

Оформление прямой речи в русском языке можно разделить на четыре основных группы:

- прямая речь стоит после авторских слов;
- прямая речь расположена перед словами автора;
- авторские слова находятся внутри прямой речи;

– авторские слова разделяются прямой речью [18].

Кроме того, для этих групп существуют дополняющие правила, в зависимости от которых прямая речь оформляется тем или иным способом. Некоторые из этих правил основываются на семантике текста и не могут быть обработаны на этапе графематического анализа. В качестве примера рассмотрим одно из правил оформления прямой речи, стоящей после авторских слов [18].

«Двоеточие ставится, помимо обычных случаев, когда авторские слова содержат в себе глагол со значением речи или мысли (сказать, спросить, ответить, подтвердить, начать, продолжать, прервать, подумать, вспомнить и др.) либо же близкое по значению имя существительное (вопрос, ответ, слова, восклицание, голос, звук, шепот, крик, мысль и т.п.), также в тех случаях, когда в функции слов, вводящих прямую речь, используются глаголы, обозначающие чувства говорящего, его внутреннее состояние (обрадоваться, огорчиться, обидеться, ужаснуться и др.), а также глаголы, обозначающие мимику, жесты, движения (улыбнуться, усмехнуться, рассмеяться, нахмуриться, вздохнуть, вскочить, подойти, подбежать и т.п.). Подобные глаголы допускают возможность добавить к ним глагол речи (например: обрадовался и сказал, удивился и спросил, улыбнулся и ответил, подбежал и воскликнул), поэтому они воспринимаются как вводящие прямую речь».

Для обработки данного правила необходимо не только определять морфологические характеристики слов, но понимать их значение.

Список шаблонов предложений с прямой речью, которые можно выявить на этапе графематического анализа, приведён в табл. 2.

Таблица 2 – шаблоны предложений с прямой речью.

Группа	Шаблон	Пример

Прямая речь после слов автора	А: «П».	Он сказал: «Я зайду завтра».
	А: «П?(! или …)»	Он спросил: «Я зайду завтра?»
Прямая речь перед словами автора	«П», - а.	«Я зайду завтра», - сказал он.
	«П?(! или …)» - а.	«Я зайду завтра?» - спросил он.
Прямая речь прерывается словами автора	«П, - а, - п».	"Беги вперед, - сказал он, - я за тобой".
	«П, - а. - П».	"Беги вперед, - сказал он. - Я догоню".
	«П?(! или …) - а. - П».	"Какой чудесный день, не правда ли? – воскликнул он. – Я никогда не видел такого яркого солнца".
Прямая речь внутри авторских слов	А: «П», - а.	Он сказал: «Я зайду завтра», - и ушёл.
	А: «П?(! или …)» - а.	Он просил: "К вам можно?" - и вошёл в комнату.

Определение в тексте границ предложений на первый взгляд кажется достаточно простой задачей. Обычно предложение заканчивается точкой, вопросительным или восклицательным знаком, иногда – многоточием. Однако на практике эти же знаки препинания применяются и для других целей. Например, точка часто используется в сокращениях. При этом если сокращение находится в конце предложения, то ставится только одна точка, относящаяся и к сокращению и к концу предложения. Знаки вопроса и восклицания часто используются в вопросительных и восклицательных конструкциях. Еще одну проблему представляет собой прямая речь, в состав которой могут входить как несколько слов, так и несколько предложений. Такие предложения целесообразно анализировать отдельно.

По перечисленным выше причинам определение границ предложений – один из последних этапов графематического анализа, идущий после поиска сокращений по словарю, текстовых конструкций по шаблону и прямой речи.

Процесс поиска конца предложения представляет собой посимвольный анализ текста и основывается на следующих основных правилах:

1) начало первого предложения всегда совпадает с началом текста;
2) конец последнего предложения всегда совпадает с концом текста;
3) предложение всегда начинается с заглавной буквы;
4) предложение не может состоять только из знаков препинания.

Обозначим несколько вспомогательных предикатов:

– SentenceStartPos – метка, характеризующая начало предложения. Присваивается первому символу в предложении.
– SentenceEndPos – метка, обозначающая конец предложения. Присваивается последнему символу в предложении.
– SentenceEndSeq – последовательность символов, характеризующих конец предложения. Под такой последовательностью понимается один или более следующие друг за другом символы «.», «!», «?» или «…».

Алгоритм поиска концов предложений состоит из следующих шагов:

Шаг 1. Присваиваем метку SentenceStartPos первому не пробельному символу в тексте.

Шаг 2. Находим в тексте первую последовательность SentenceEndSeq, следующую за SentenceStartPos и за SentenceEndSeq, если SentenceEndSeq определена. Если такая последовательность найдена, запоминаем её позицию и переходим к следующему шагу. Если такая последовательность не найдена, и в процессе поиска достигнут конец файла, то присваиваем метку конца предложения последнему символу в тексте и переходим к шагу 5

Шаг 3. После нахождения последовательности SentenceEndSeq ищем первый не пробельный символ, следующий за SentenceEndSeq. Пробельными

символами считаются: символ пробела, символ табуляции, символ перевода строки.

Если следующий за SentenceEndSeq не пробельный символ является заглавной буквой, то найден предположительный конец предложения, переходим к следующему шагу, иначе переходим к шагу 2. Если в процессе поиска достигнут конец файла, то присваиваем метку SentenceEndPos последнему символу из SentenceEndSeq и переходим к шагу 5.

Шаг 4. После нахождения предположительного конца предложения необходимо осуществить несколько проверок.

Первая из них – проверка на наличие незакрытых скобок и кавычек. Предложение не может быть закончено, если в нём есть незакрытые скобки или кавычки (за исключением описанных выше случаев с прямой речью). Для этого ведётся учет всех открывающихся и закрывающихся скобок и кавычек: в случае если текущий символ не является буквенным или цифровым, то осуществляется его проверка на то, является ли он скобкой или кавычкой. Для каждой открывающейся скобки (или кавычки) запоминается её позиция, и в случае если для данной скобки не была найдена закрывающая её пара на расстоянии более 150 символов после неё, то данный случай считается опечаткой, а открытая скобка забывается. Данный параметр является настраиваемым, его значение было выбрано исходя из средней длины предложения в русском языке, которое составляет 54 символа.

Если не все скобки закрыты, то переходим к шагу 2, в противном случае осуществляется следующая проверка – на пересечение последовательности SentenceEndSeq с найденными на предыдущих этапах графематическими дескрипторами: ФИО, когда имя и отчество прописаны в виде инициалов, сокращений, аббревиатур и т. д. Текущая последовательность SentenceEndSeq не является концом предложения в двух случаях:

1) если SentenceEndSeq разбивает хотя бы один графематический дескриптор на две части (т. е. не является его концом);

2) Если SentenceEndSeq находится в конце хотябы одного графематического дескриптора, который не может стоять в конце предложения.

В случае если SentenceEndSeq не является концом предложения, переходим к шагу 2. Иначе осуществляется следующая проверка – на пересечение символа, предшествующего SentenceEndSeq, с графематическими дескрипторами, найденными на предыдущих этапах. Если символ, предшествующий SentenceEndSeq, пересекается хотя бы с одним дескриптором, который не может быть концом предложения, то найденная последовательность SentenceEndSeq не является концом предложения, переходим к шагу 2.

Если все три проверки пройдены, то текущая SentenceEndSeq является концом предложения. Метка SentenceEndPos присваивается последнему символу в SentenceEndSeq. Необходимо сохранить границы найденного предложения SentenceStartSeq и SentenceEndSeq, а затем перейти к шагу 2.

Шаг 5. Конец.

Помимо выделения предложений с прямой речью и поиска концов предложений на данном этапе так же производится окончательная разметка текста на слова и знаки препинания. В процессе такой разметки выделяются следующие типы дескрипторов, значения которых описаны в табл. 1:

- RUSSIAN_LEX,
- FOREIGN_LEX,
- NUM_SEQ,
- OPEN_BRACKET,
- CLOSE_BRACKET,
- OPEN_QUOTE,
- CLOSE_QUOTE,
- PUNCTUATION_CHAR,
- OTHER_LEX.

5 Формат входных данных и выходных данных

5.1 Входной текстовый файл

Входными данными для модуля графематического анализа является текстовый файл в кодировке cp1251. Размер входного файла может достигать нескольких гигабайт и не помещаться в оперативную память компьютера.

Текстовый файл может содержать слабо структурируемый текст без четкого формата. Такие тексты, как правило, получаются средствами автоматической конвертации из таких форматов, как pdf, doc и т. д., а так же системами распознавания текстов из изображений. В этом случае текст содержит фиксированную длину строки, следовательно, символы перевода строки могут и, как правило, не являются признаками нового предложения или абзаца. Пример такого текста приведён на рис. 1.

```
Княжна Элен улыбалась; она поднялась с тою же неизменяющеюся улыбкой
вполне красивой женщины, с которою она вошла в гостиную. Слегка шумя своею
белою бальною робой, убранною плющем и мохом, и блестя белизною плеч,
глянцем волос и брильянтов, она прошла между расступившимися мужчинами и
прямо, не глядя ни на кого, но всем улыбаясь и как бы любезно предоставляя
каждому право любоваться красотою своего стана, полных плеч, очень открытой,
по тогдашней моде, груди и спины, и как будто внося с собою блеск бала,
подошла к Анне Павловне. Элен была так хороша, что не только не было в ней
заметно и тени кокетства, но, напротив, ей как будто совестно было за свою
несомненную и слишком сильно и победительно-действующую красоту. Она как
будто желала и не могла умалить действие своей красоты. Quelle belle
personne! 50 говорил каждый, кто ее видел.
```

Рисунок 1 – Слабо структурированный текст

5.2 Словарь сокращений и аббревиатур

Для поиска в тексте сокращений и аббревиатур был использован словарь. Словарь содержит в себе следующую информацию:

- текст сокращения или аббревиатуры с разбивкой по словам;
- информацию о регистре первой буквы в каждом слове сокращения;
- информацию о регистре последующих букв в каждом слове сокращения;
- количество символов в каждом слове сокращения;
- текст полного слова или словосочетания, характерный для данного сокращения;

– часть речи полного слова;
– морфологические характеристики, соответствующие полному слову или словосочетанию.

В случае если одному сокращению или аббревиатуре соответствует несколько полных слов или словосочетаний, то в словаре будет содержаться несколько записей. Приведём пример: аббревиатуре «ПВО» соответствуют следующие значения:

– полное внутреннее отражение;
– противовоздушная оборона;
– противовыбросовое оборудование.

Использованный словарь представляет собой плоский файл, каждая строка в котором содержит запись об одном сокращении и одном соответствующем ему полном слове. Примеры таких записей:

– <искл;0;0;4><.;0;0;1>=<исключительно:adv>;
– <исл;0;0;3><.;0;0;1>=<исландский:adj;comp=pos;short=n;case=i;number=sg>;
– <ИСЛ;1;1;3>=<искусственный спутник Луны;number=sg;gender=m>;
– <ДК;1;1;2>=<дом культуры;number=sg;gender=m>;
– и т.д.

Символ «=» является разделителем между двумя основными частями записи: сокращении или аббревиатуре и несокращенном варианте.

Слева от разделителя «=» находятся один или несколько блоков, заключённых в «< >». Каждый такой блок содержит в себе одно слово сокращения, информацию о регистре первой и последующих букв в этом слове, а так же количество символов в слове. Флаг информации о регистре может принимать следующие значения:

– 0 – не имеет значения;
– 1 – символ должен быть заглавной буквой;

– 2 – символ должен быть прописной буквой.

Справа от разделителя «=» находятся один или несколько блоков, содержащих полное слово или словосочетание, которое соответствует сокращению, расположенному в левой части, его часть речи и другие морфологические характеристики, перечисленные после следующего за полным словом знака «:». Морфологические характеристики разделены между собой знаком «;».

Первой после «:» идёт часть речи, которая может принимать одно из следующих значений:

– adj – прилагательное;
– adv – наречие;
– card_num – количественное числительное;
– compar – сравнительная степень;
– conj – союз;
– deepr – деепричастие;
– dem_pron – указательное местоимение;
– idiom – идиома;
– interj – междометье;
– noun – существительное;
– ord_num – порядковое числительное;
– parenthesis – вводное слово;
– participle – причастие;
– particle – частица;
– pers_pron – личное местоимение;
– poss_pron – притяжательное местоимение;
– postpos – послелог;
– predic – предикатив;
– predic_pron – личный предикатив;

– prep – предлог;
– refl_pron – возвратное местоимение;
– sign – знак препинания;
– verb – глагол.

За частью речи следует одна или несколько морфологических характеристик в виде связки «название характеристики» + «значение характеристики», в которой название и значения разделены символом «=». При этом часть речи и один или несколько морфологических параметров могут отсутствовать, их наличие не является обязательным. Полный список морфологических характеристик и их значений представлен в табл. 3.

Таблица 3 – список морфологических характеристик в словаре сокращений.

Название параметра	Обозначение	Значение	Расшифровка значения
одушевленность	animate	0	не выражена
		an	одушевленный
		inan	неодушевленный
совершенность	aspect	0	не выражена
		sov	совершенная форма
		nesov	несовершенная форма
падеж	case	0	не выражен
		i	именительный
		r	родительный
		v	винительный
		d	дательный
		t	творительный
		p	предложный

степень сравнения	comp	0	не выражена
		com	сравнительная степень
		pos	позитивная степень
род	gender	0	не выражен
		m	мужской
		f	женский
		n	средний
нормальная форма	inf	0	не выражена
		y	нормальная форма
		n	не нормальная форма
наклонение	mood	0	не выражено
		imp	повелительное
		ind	изъявительное
число	number	0	не выражено
		sg	единственное
		pl	множественное
лицо	person	0	не выражено
		1	первое
		2	второе
		3	третье
кратность	short	0	не выражена
		y	краткая форма
		n	полная форма

время	tense	0	не выражено
		past	прошедшее
		pres	настоящее
		fut	будущее
переходность	transit	0	не выражена
		tr	переходный
		intr	не переходный
залог	voice	0	не выражен
		a	активный
		p	пассивный

Объём словаря составил порядка 20000 записей.

5.3 Входные данные для поиска по шаблонам

Регулярные выражения, используемые для поиска графематических дескрипторов по шаблону, хранятся в отдельном текстовом файле. Запись об одном шаблоне содержится в четырёх строках, разделенных символом перевода строки, и содержит следующую информацию:

- название графематического дескриптора (шаблона);
- регулярное выражение, по которому необходимо произвести поиск;
- номер группы в регулярном выражении, которой соответствует искомый дескриптор;
- флаг того, может ли данный дескриптор находиться в конце предложения (0 – не имеет значения, 1 – не может, 2 – обязательно является концом предложения).

Рассмотрим пример шаблона для поиска ФИО, когда имя и отчество прописаны в виде инициалов. Название шаблона – «FIO». Регулярное

выражение для поиска – *«[A-ZA-Я]{1}[a-za-я]{1,}\s{1}[A-ZA-Я]{1}[.]{1}\s?[A-ZA-Я]{1}[.]{1}|[A-ZA-Я]{1}[.]»*. Номер группы в регулярном выражении, соответствующий данному дескриптору – 0, т.е. всё регулярное выражение. Флаг, информирующий о том, может ли данный графематический дескриптор являться концом предложения – 0.

Важным замечанием является то, что любой графематический дескриптор должен полностью принадлежать одному из предложений, т. е. предполагается, что первый и последний символы любого из графематических дескрипторов принадлежат одному предложения.

5.4 Формат выходных данных

Выходными данными графематического анализа являются:

- список выделенных в тексте предложений;
- список найденных графематических дескрипторов.

Список выделенных в тексте предложений содержит следующую информацию:

- порядковый номер предложения в тексте;
- позиция начала предложения в тексте (позиция первого символа предложения);
- количество символов в предложении;
- текст предложения.

Информация, содержащаяся в списке графематических дескрипторов:

- тип графематического дескриптора;
- позиция начала графематического дескриптора в исходном тексте (позиция его первого символа в тексте);
- количество символов в дескрипторе;
- текст дескриптора;

– полный текст и его морфологические характеристики (для сокращений и аббревиатур).

Для проведения морфологического и последующих этапов анализа были разработаны база данных и механизм экспорта в неё результатов графематики. Схема таблиц базы данных отображена на рис. 2.

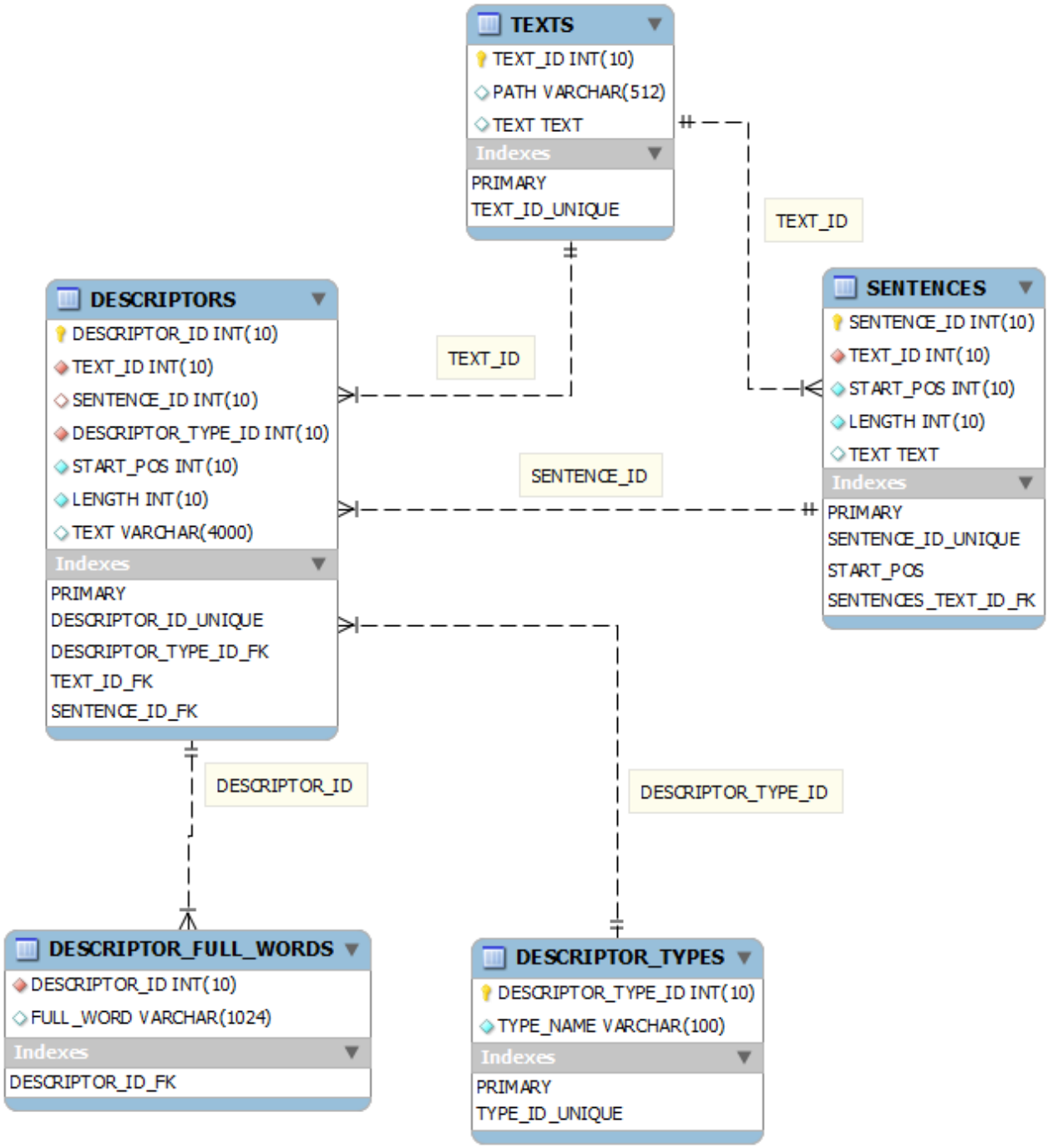

Рисунок 2 – Структура базы данных

В таблице «TEXTS» содержатся проанализированные тексты. Поля таблицы:

- «TEXT_ID» – уникальный идентификатор текста;
- «PATH» – путь до исходного текстового файла;
- «TEXT» – содержимое исходного текстового файла.

Таблица «SENTENCES» содержит список предложений, которые были выявлены в процессе графематического анализа, и следующую информацию о них:

- «SENTENCE_ID» – уникальный идентификатор предложения;
- «TEXT_ID» – идентификатор текста, к которому принадлежит предложение;
- «START_POS» – позиция начала предложения в тексте;
- «LENGTH» – количество символов в предложении;
- «TEXT» – текст предложения.

Таблица «DESCRIPTOR_TYPES» представляет собой словарь дескрипторов:

- «DESCRIPTOR_TYPE_ID» – уникальный идентификатор типа дескриптора;
- «TYPE_NAME» – наименование типа.

«DESCRIPTORS» – таблица с информацией о выявленных графематических дескрипторах. Поля таблицы:

- «DESCRIPTOR_ID» – уникальный идентификатор дескриптора;
- «TEXT_ID» – идентификатор текста, к которому принадлежит дескриптор;
- «SENTENCE_ID» – идентификатор предложения, к которому принадлежит дескриптор.
- «DESCRIPTOR_TYPE_ID» – идентификатор типа дескриптора;

– «START_POS» – позиция первого символа дескриптора в исходном тексте;
– «LENGTH» – количество символов в дескрипторе;
– «TEXT» – текст дескриптора.

Таблица «DESCRIPTORS_FULL_WORDS» содержит расшифровки дескрипторов, которые являются сокращениями или аббревиатурами. Поля таблицы:

– «DESCRIPTOR_ID» – идентификатор дескриптора;
– «FULL_WORD» – расшифровка сокращения;

6 Внутреннее представление данных

6.1 Хранение исходного текста.

Так как размер входного текстового файла может достигать нескольких гигабайт и не помещаться в оперативную память компьютера, его необходимо читать и анализировать частями. Главная задача здесь – не потерять графематические дескрипторы и предложения, начало которых находится в одной анализируемой части, а конец – в следующей за ней.

На вход графематике подаются путь до текстового файла и размер одной анализируемой части (количество символов в ней). Далее на основании размера файла рассчитывается количество анализируемых частей. После анализа каждой части вычисляется последняя корректно проанализированная позиция в текстовом буфере. Обозначим её LastValidPos. Вычисляется она на основании следующих правил:

- если текущий анализируемый буфер текста является последним, то LastValidPos совпадает с позицией последнего символа в анализируемой части;
- если текущий анализируемый буфер текста не является последним, то LastValidPos присваивается позиции последнего символа в последнем выделенном в буфере предложения, конец которого расположен в 200 и более символах от конца текстового буфера.

Все выявленные графематические дескрипторы и предложения, позиция конца которых находится после LastValidPos, удаляются, а часть текстового буфера, находящаяся после LastValidPos сохраняется и конкатенируется к началу следующего анализируемого буфера.

6.1 Представление словаря сокращений и аббревиатур

В процессе проектирования работы со словарём сокращений и аббревиатур требовалось решить две задачи:

– минимизировать время загрузки словаря при старте программы;
– обеспечить минимальное время поиска по словарю;

С целью сокращения времени, требующегося на загрузку словаря при старте программы, было принято решение реализовать механизмы экспорта и импорта словаря в бинарный файл, содержащий в себе сериализованные структуры программы, в которых хранится словарь. Загрузка словаря из такого файла занимает значительно меньше времени.

Второй и наиболее важной задачей является организация хранения словаря и поиска в нём элементов с минимальными временными затратами. Ввиду сравнительно небольшого объёма словаря (20000 записей) вопрос о минимизации объёма используемой для хранения словаря оперативной памяти отходит на второй план. Так как необходимо построить словарь только один раз (после этого он сериализуется и хранится в бинарном файле), то скорость вставки в него элементов так же не имеет значения. Операция удаления элементов применяться не будет [19].

В качестве структуры для хранения и дальнейшего поиска были рассмотрены следующие варианты:

– префиксное дерево (Trie, бор, нагруженное дерево);
– конечный автомат, алгоритм поиска – Ахо-Корасик.

Префиксное дерево представляет собой структуру данных, реализующую интерфейс ассоциативного массива, позволяющего хранить пары «ключ, значение». Как правило, в роли ключа выступают строки. Префиксное дерево отличается от обычных n-арных деревьев, тем, что в его узлах не хранятся ключи. Вместо этого каждый узел содержит одно символьную метку, а ключом, который соответствует некому узлу, является путь от корня дерева до этого узла [20, 21].

На рис. 3 отображен пример хранения словаря сокращений и аббревиатур в виде префиксного дерева, который состоит из следующих сокращений:

– докт. псих. н. – доктор психологических наук;
– д. псих. н. – доктор психологических наук;
– д. пед. н. – доктор педагогических наук;
– с. - северный;
– с. – сельский;
– т. п. – тому подобный;
– т. д. – так далее.

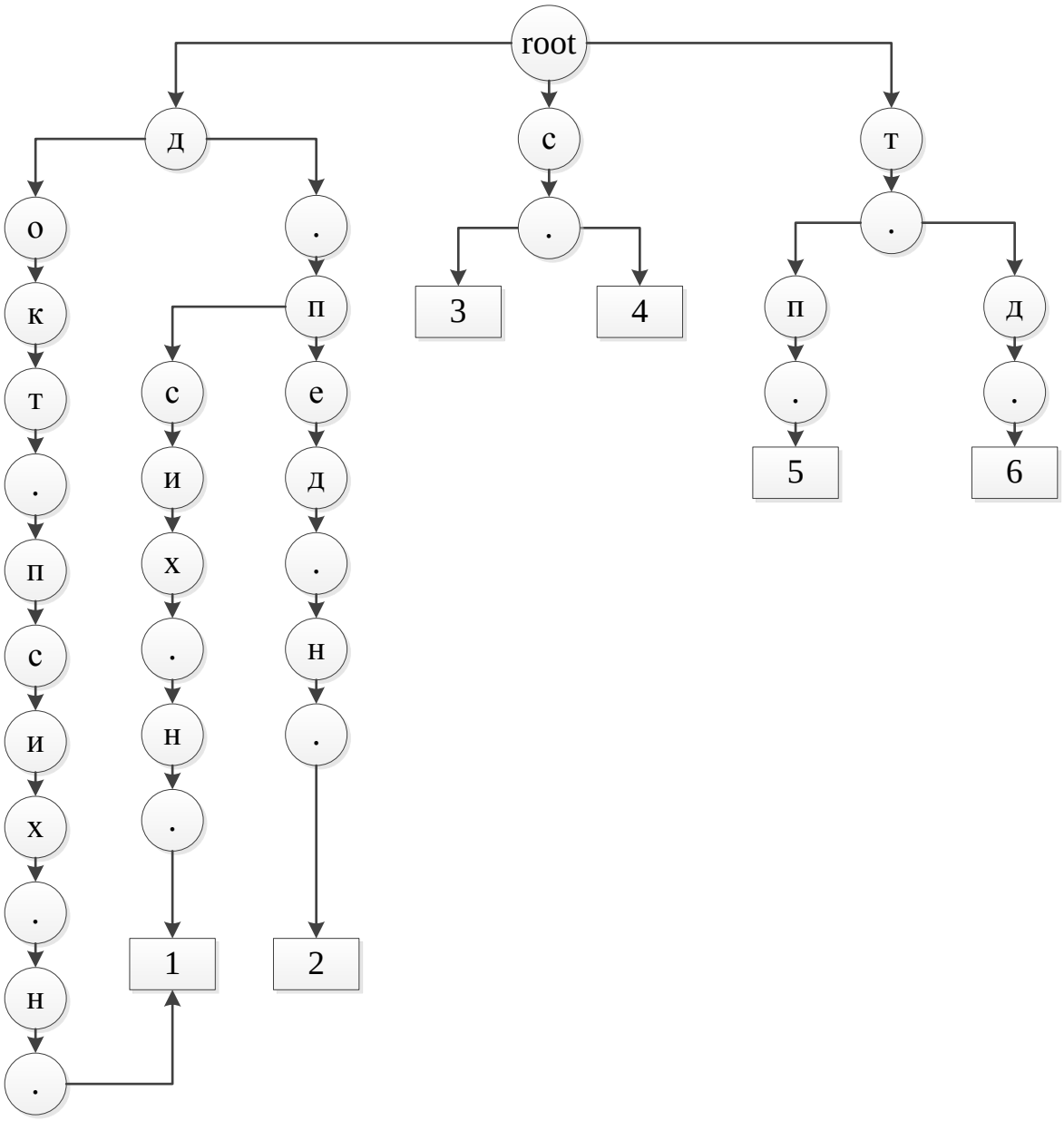

Рисунок 3 – Префиксное дерево

Как видно из рис. 3, разные ключи (цепочки символов) могут приводить к одному и тому же полному варианту сокращения, и, наоборот, один ключ может содержать в себе несколько расшифровок одного сокращения.

Алгоритм поиска сокращений и аббревиатур с использованием префиксного дерева состоит из следующих шагов.

Шаг 1. Обозначим s – строка с текстом, в котором необходимо произвести поиск, i – текущая позиция в строке с текстом, $iStart$ – позиция начала предполагаемого сокращения в тексте, $curNode$ – указатель на текущий узел дерева. Присвоим $i = 0$, $iStart = 0$, $curNode$ указывает на корневой узел дерева (root).

Шаг 2. Если i больше или равна длине s, то переходим к шагу 5. Если $i + (i - iStart)$ больше или равно длине s, то переходим к шагу 3. Среди потомков curNode ищем узел, символьной меткой которого является $s[i]$. Если такой узел найден, то присваиваем метку $curNode$ ему, увеличиваем i на 1 и повторяем шаг 2. Если искомый узел не найден, переходим к шагу 3.

Шаг 3. Начиная с $curNode$ поднимаемся на уровень вверх до тех пор, пока curNode не будет содержать одну или более расшифровок сокращения или curNode не будет равен корню дерева. В случае если узел с расшифровкой найден, значит мы нашли сокращение, переходим к шагу 4. Иначе, если мы пришли в корень дерева, то $i = i + 1$, $iStart = i$, $curNode$ присваиваем корень дерева и переходим к шагу 2.

Шаг 4. Последовательно проверяем найденное сокращение с другими ранее найденными сокращениями на предмет их пересечения. Если пересечение найдено, то сохраняем максимальный по длине элемент, второй удаляем. Если новый элемент не пересекается ни с одним ранее найденным, то добавляем его в список найденных сокращений. Далее $i = i + 1$, $iStart = i$, curNode присваиваем корень дерева и переходим к шагу 2.

Шаг 5. Конец.

Сложность поиска в префиксном дереве составляет $O(2 * H)$, где H – это длина строки, в которой ведётся поиск.

Вторым рассмотренным алгоритмом поиска стал Ахо-Корасик. Этим алгоритмом в качестве структуры для хранения словаря используется конечный автомат [23, 24].

Пусть $x[1] \dots x[m]$ – строка символьных меток, называемая ключевым словом. Бором ключевого слова называется диаграмма переходов с $m + 1$ состоянием, в которой каждое состояние соответствует префиксу ключевого слова. Для $1 \leq s \leq m$ существует переход из состояния $s - 1$ в состояние s по символу $x[s]$. Бор для ключевого слова «т. д.» отображен на рис. 4.

Рисунок 4 – Пример бора для ключевого слова «т. д.»

Каждое состояние в боре соответствует префиксу одного или нескольких ключевых слов. Начальное состояние соответствует пустой строке, а состояние, соответствующее полному ключевому слову, является заключительным.

Диаграмма переходов для набора сокращений «д. псих. н.», «докт. псих. н.», «с.», «т. д.», «т. п.» отображена на рис 5. Узлы, выделенные синим цветом, являются заключительными состояниями.

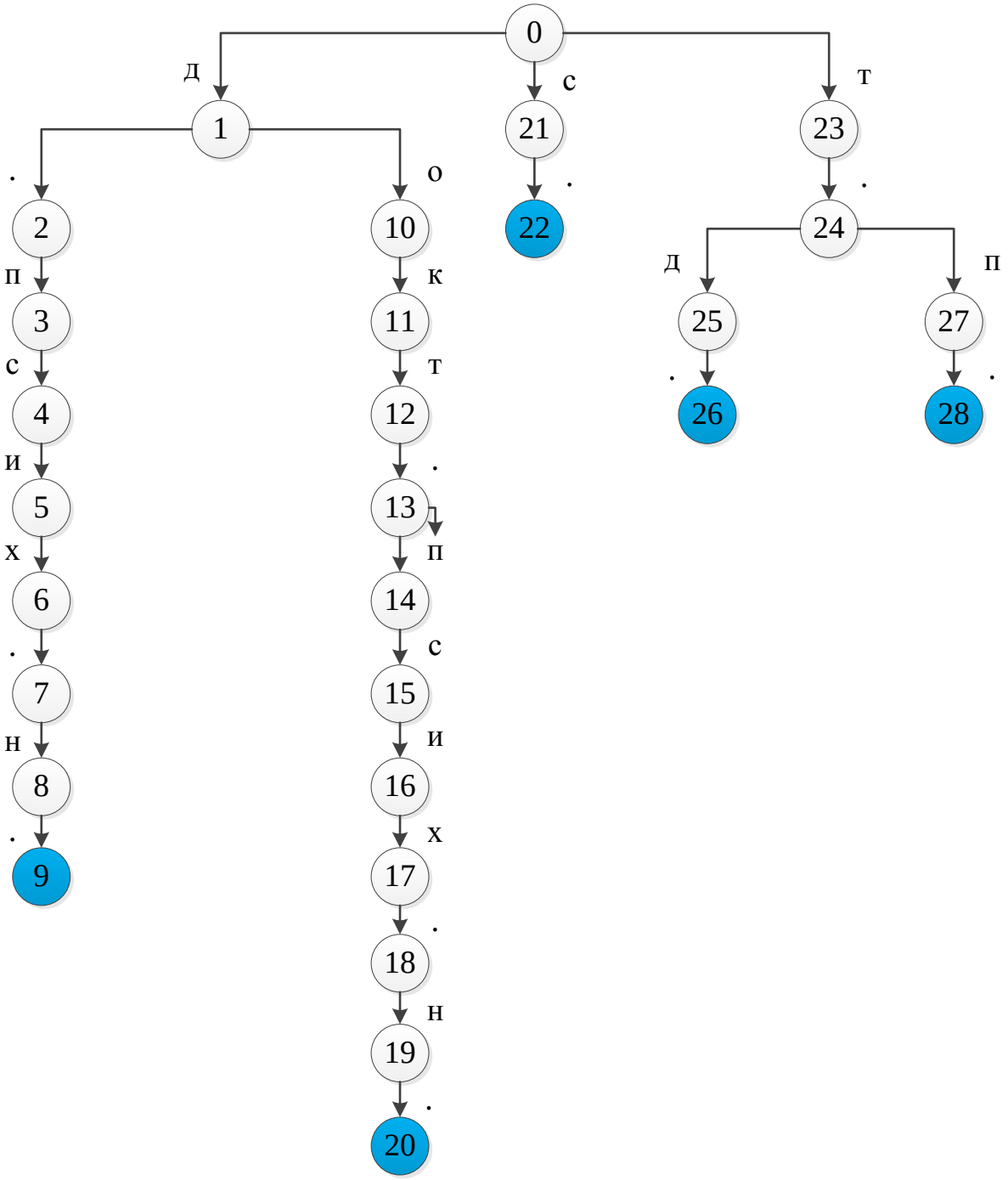

Рисунок 5 – Диаграмма переходов

Определим для бора функцию переходов g, которая отображает пары «состояние, символ» в состояние, как $g(s, x[i+1]) = s'$, если состояние s соответствует префиксу $b[1] \dots b[i+1]$. Если s_0 – начальное состояние, то $g(s_0, a) = s_0$ для всех входных символов a, которые не являются начальными символами ни одного из ключевых слов. Далее определим, что $g(s, a) = fail$

для любого не определенного перехода. Для стартового состояния $fail$ – переходов не имеется.

Допустим, что состояния s и t представляют префиксы u и v некоторых ключевых слов. $f(s) = t$ тогда и только тогда, когда v – наидлиннейший собственный суффикс u, который является префиксом некоторого ключевого слова. Функция отказа f для приведённой на рис. 5 диаграммы приведена в табл. 4.

Таблица 4 – Функция отказа f.

s	$f(s)$
1	*0*
2	*0*
3	*0*
4	*21*
5	*0*
6	*0*
7	*0*
8	*0*
9	*0*
10	*0*
11	*0*
12	*23*
13	*24*
14	*27*
15	*21*
16	*0*
17	*0*
18	*0*
19	*0*
20	*0*
21	*0*
22	*0*
23	*0*
24	*0*
25	*1*
26	*2*
27	*0*
28	*0*

Например, состояния 4 и 21 представляют префиксы «д. пс» и «с». $f(4) = 21$, поскольку «с» - наидлиннейший собственный суффикс «д. пс», который одновременно является префиксом некоторого ключевого слова. Функция f может быть вычислена обходом бора в ширину.

Так как $g(s_0,) \neq fail$ для любого символа, то цикл гарантированно завершается. После присвоения $f(s') = g(s, a)$, если $g(s, a)$ – заключительное состояние, мы также делаем состояние s' заключительным, при условии, что оно уже не является таковым. Функция отказа вычисляется за время пропорциональное сумме длин ключевых слов.

Пусть g – функция переходов, f – функция отказа для некоторого множества ключевых слов $K = x_1, \dots, .x_k$. Алгоритм Ахо-Корасик использует g и f для определения того, содержит ли строка $y[1] \dots y[n]$ подстроку, являющуюся ключевым словом. Состояние s_0 – стартовое состояние диаграммы переходов K, а F – множество заключительных состояний.

Сложность такого поиска составляет $O(n + m + z)$, где n – длина строки, в которой ведётся поиск, m – суммарная длина всех ключевых слов, z – количество появлений шаблонов в y. В худшем случае $z = n * k$, но на практике он встречается крайне редко [22, 24, 25].

7. Программная реализация

Для программной реализации системы графематического анализа был выбран язык C++. Для экспортирования словаря в бинарный и текстовый архивы использовались следующие библиотеки проекта Boost: «boost_serialization» и «boost_archive». Для поиска текстовых конструкций с использованием шаблонов была применена библиотека «boost_regex», предоставляющая функциональность поиска по регулярным выражениям. В качестве СУБД для экспортирования результатов графематического анализа была использована MySQL [26, 27, 28].

7.1 Структура классов программы

Для хранения морфологических характеристик слов из словаря сокращений была использована система морфологического анализа «Кросслейтор» и её внутренние структуры данных. Сделано это с целью упрощения дальнейшей интеграции двух систем и реализации следующих этапов анализа текста на естественном языке.

Структура программы состоит из двух основных классов:

- «shortDictionary»;
- «grafematika».

Класс «shortDictionary» является служебным классом и отвечает за работу со словарём сокращений и аббревиатур. Класс предоставляет следующую функциональность:

- парсинг исходного текстового файла словаря и его хранение во внутренних структурах в памяти;
- сохранение словаря в бинарный и текстовый архивы с использованием библиотек boost_serialization и boost_archive;
- загрузка словаря из этих архивов;
- поиск в тексте сокращений и аббревиатур по словарю.

Публичные функции класса «shortDictionary»:

- shortDictionary(LanguageDictionary2010*);
- int loadShortDctFromFlatfile(std::string, std::string);
- void printTree();
- int saveShortDctToArfile(std::string, int);
- int loadShortDctFromArfile(std::string, int);
- int searchShortWordsDict(std::string, std::vector<descriptor>&).

Конструктор класса в качестве входного параметра принимает указатель на экземпляр класса «LanguageDictionary2010» системы морфологического анализа «Кросслейтор».

Функция «loadShortDctFromFlatfile» осуществляет загрузку словаря сокращений и аббревиатур из текстового файла. Входными параметрами являются путь до текстового файла со словарём и путь до текстового файла, в который будут сохранены записи словаря, в процессе разбора которых возникли ошибки. Возвращаемые значения: 0 – успешная загрузка словаря, 1 – ошибка. В том случае, если хотя бы одна запись была загружена успешно, и не произошло никаких непредвиденных ошибок, будет возвращено значение 0.

Функция «printTree» выводит хранящееся в памяти программы дерево словаря обходя его в глубину.

Функция «saveShortDctToArfile» осуществляет сохранение находящегося в памяти программы дерева словаря сокращений и аббревиатур в файл, представляющий собой архив. Для создания файла используются библиотеки «boost_serialization» и «boost_archive». Входные параметры: путь, по которому нужно сохранить файл словаря, и тип архива. Тип архива может принимать значения 0 и 1, которые означают бинарный и текстовый формат соответственно. Возвращаемые значения: 0 – успешный экспорт словаря, 1 – ошибка.

Функция «loadShortDctFromArfile» производит загрузку словаря из файла архива в память программы. Входные параметры: путь до файла, который необходимо загрузить и тип архива. Значения типа архива аналогичны его значениям в функции «saveShortDctToArfile». Возвращаемые значения: 0 – успешная загрузка словаря, 1 – ошибка.

Функция «searchShortWordsDict» осуществляет поиск всех непересекающихся сокращений в тексте. Входные параметры: текст, в котором необходимо произвести поиск сокращений, и ссылка на вектор графематических дескрипторов, который необходимо заполнить найденными сокращениями. В случае возникновения ошибки возвращается значение 1, иначе 0.

Приватные функции класса «shortDictionary»:

- int parseShortDctString(std::string, ShortDctElement*);
- int parseShortDctStringLeftPart(std::string, ShortWordsCollection*);
- int parseShortDctStringRightPart(std::string, FullWordsCollection&);
- bool compareRegistry(std::string, ShortWordsCollection*);
- bool addDescriptor(std::vector<descriptor>&, descriptor&, size_t&).

Функции «parseShortDctString», «parseShortDctStringLeftPart», «parseShortDctStringRightPart» осуществляют парсинг одной записи из словаря сокращений и аббревиатур. Внутри функции «parseShortDctString» производится вызов функций «parseShortDctStringLeftPart» и «parseShortDctStringRightPart». Если хотя бы одна из функций возвращает значение, не равное 0, то разбираемая запись из словаря считается некорректной и не добавляется в дерево словаря, а заносится в отдельный текстовый файл, содержащий записи некорректного формата.

Функция «compareRegistry» производит сравнение регистра символов сокращения, найденного в тексте, с сокращением из словаря. Входными параметрами являются подстрока текста, являющаяся сокращением и

коллекция слов, из которых состоит соответствующее ей сокращение в словаре. Функция возвращает «true», если регистры всех символов совпадают, иначе «false».

Функция «addDescriptor» добавляет новое сокращение в список найденных в процессе поиска сокращений в тексте. Входные параметры:

- указатель на вектор с найденными ранее графематическими дескрипторами (в него так же добавляется новый дескриптор),
- найденный дескриптор, который необходимо добавить;
- указатель на переменную, в которую помещается позиция нового дескриптора при его добавлении.

Если два сокращения пересекаются между собой, сохраняется наиболее длинное из них. Функция возвращает «true», если новый дескриптор был добавлен, иначе – «false».

Класс «grafematika» отвечает за графематический анализ текста, сохранение и выдачу результатов этого анализа, а так же предоставляет интерфейс взаимодействия при интеграции с другими программными продуктами.

Публичные ункции класса «grafematika»:

- grafematika();
- int loadShortDct(std::string, std::string, int);
- int saveShortDct(std::string, int);
- int loadPatterns(std::string);
- int analyze(std::string, size_t);
- void getDescriptors(std::vector<descriptor>&);
- void getSentences(std::vector<sentence>&);
- std::string getText();
- int expToDb(std::string, std::string, std::string, std::string, uint32_t).

Входными параметрами функции «loadShortDct» являются:

– путь до словаря, используемого системой морфологического анализа «Кросслейтор»;
– путь до словаря сокращений и аббревиатур;
– формат файла словаря сокращений и аббревиатур.

Формат может принимать следующие значения:

– 0 – бинарный архив;
– 1 – текстовый архив;
– 2 – плоский текстовый файл.

Если файл со словарём представлен в виде плоского текстового файла, функция «loadShortDct» создает экземпляр класса «LanguageDictionary2010» системы «Кросслейтор» и вызывает функцию «loadShortDctFromFlatfile» из класса «shortDictionary» для его загрузки. После загрузки словаря объект класса «LanguageDictionary2010» удаляется. Если файл представлен в виде архива, то его загрузка производится путём вызова функции «loadShortDctFromArfile» класса «shortDictionary» с передачей ей необходимых параметров. Возвращаемые значения: 0 – успешная загрузка словаря, 1 – ошибка.

Функция «saveShortDct» осуществляет запуск функции «saveShortDctToArfile» из класса «shortDictionary», передавая ей необходимые параметры, и возвращает её значение. Такая схема реализована с целью использования одного класса «grafematika» в качестве интерфейса взаимодействия при интеграции с другими продуктами. Возвращаемые значения: 0 – успешный экспорт словаря, 1 – ошибка.

Функция «loadPatterns» отвечает за загрузку шаблонов поиска. Входным параметром является путь до файла с шаблонами. Возвращаемые значения: 0 – успешная загрузка шаблонов, 1 – ошибка.

Входными параметрами функции «analyze» являются: путь до текстового файла, который необходимо проанализировать и размер одного анализируемого буфера текста (количество символов в одном буфере). При запуске функции

производится поэтапный графематический анализ текста, содержащегося в текстовом файле. В случае возникновения ошибок возвращается значение 1, иначе 0.

Функция «getDescriptors» принимает указатель на вектор графематических дескрипторов и заполняет его найденными в процессе анализа дескрипторами. Элементы в векторе отсортированы по позиции начала дескриптора в тексте.

Функция «getSentences» принимает указатель на вектор предложений и заполняет его найденными в процессе анализа предложениями. Элементы в векторе отсортированы по позиции начала предложения в тексте.

UML диаграммы структур «descriptor» и «sentence» отображены на рис. 6.

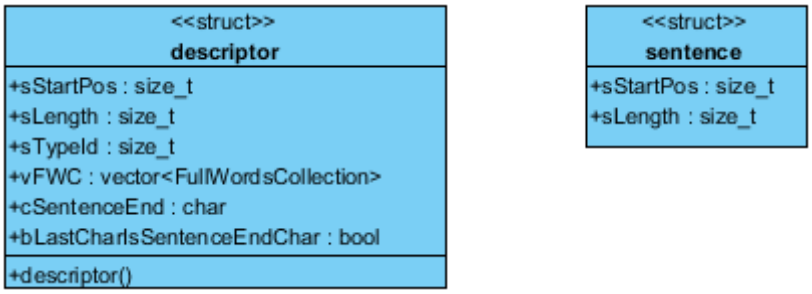

Рисунок 6 – UML диаграмма для структур «descriptor» и «sentence»

Функция «getText» возвращает строку, содержащую исходный текст с выделенными в нём графематическими дескрипторами.

Функция «expToDb» производит экспорт результатов графематического анализа в базу данных. Входные параметры:

- хост или IP адрес компьютера, на котором располагается база данных;
- имя пользователя БД;
- пароль;
- имя схемы БД;
- порт для подключения [29].

Приватные функции класса «grafematika»:

- int openFile();
- int closeFile();
- int readBufferFromFile(size_t);
- int loadRegExp(std::string);
- size_t getLastAnalyzedBufferPos();
- int addDescriptor(size_t, descriptor, std::vector<descriptorPos>&, bool);
- int analyzeBuffer();
- int searchShortWords();
- int searchDescriptorsByPattern();
- int searchOtherDescriptors().

Функции «openFile» и «closeFile» осуществляют соответственно открытие и закрытие входного текстового файла, анализ которого осуществляется. Функция «openFile» так же рассчитывает количество буферов, на которые будет разделен анализируемый текст. Расчет производится исходя из размера файла и количества символов в одном буфере.

Функция «readBufferFromFile» осуществляет чтение из файла текстового буфера. На вход подаётся номер буфера, который необходимо прочесть Непечатаемые символы при чтении игнорируются.

Функция «loadRegExp» производит загрузку файла с шаблонами, по которым осуществляется поиск. Входной параметр – путь до файла с шаблонами.

Функция «getLastAnalyzedBufferPos» возвращает последнюю корректно проанализированную позицию в текстовом буфере.

Функция «addDescriptor» добавляет графематический дескриптор в список дескрипторов и помещает в вектор с позициями дескрипторов позиции тех дескрипторов, с которыми пересекается добавляемый, предварительно удалив все элементы вектора. Входные параметры:

- позиция первого символа дескриптора;

- графематический дескриптор;
- указатель на вектор с позициями дескрипторов;
- флаг того, добавлять ли новый дескриптор или только найти пересекающиеся с ним элементы. Значение true означает, что необходимо провести добавление нового дескриптора. Если значение равно false, то добавление не производится, а лишь проводится проверка на пересечение.

Функция «analyzeBuffer» производит полный графематический анализ загруженного текстового буфера.

Функция «searchShortWords» производит поиск сокращений и аббревиатур в анализируемом буфере и их добавление в список найденных графематических дескрипторов. В данной функции происходит вызов «searchShortWordsDict» из класса «shortDictionary».

Функция «searchDescriptorsByPattern» производит поиск в текущем текстовом буфере текстовых конструкций по загруженным ранее шаблонам и их добавление в список найденных графематических дескрипторов.

Функция «searchOtherDescriptors» осуществляет токенизацию текущего буфера, поиск предложений с прямой речью и выделение в нём границ предложений.

7.2 Структуры данных

UML диаграмма структур, использующихся для хранения словаря сокращений и аббревиатур представлена на рис. 7.

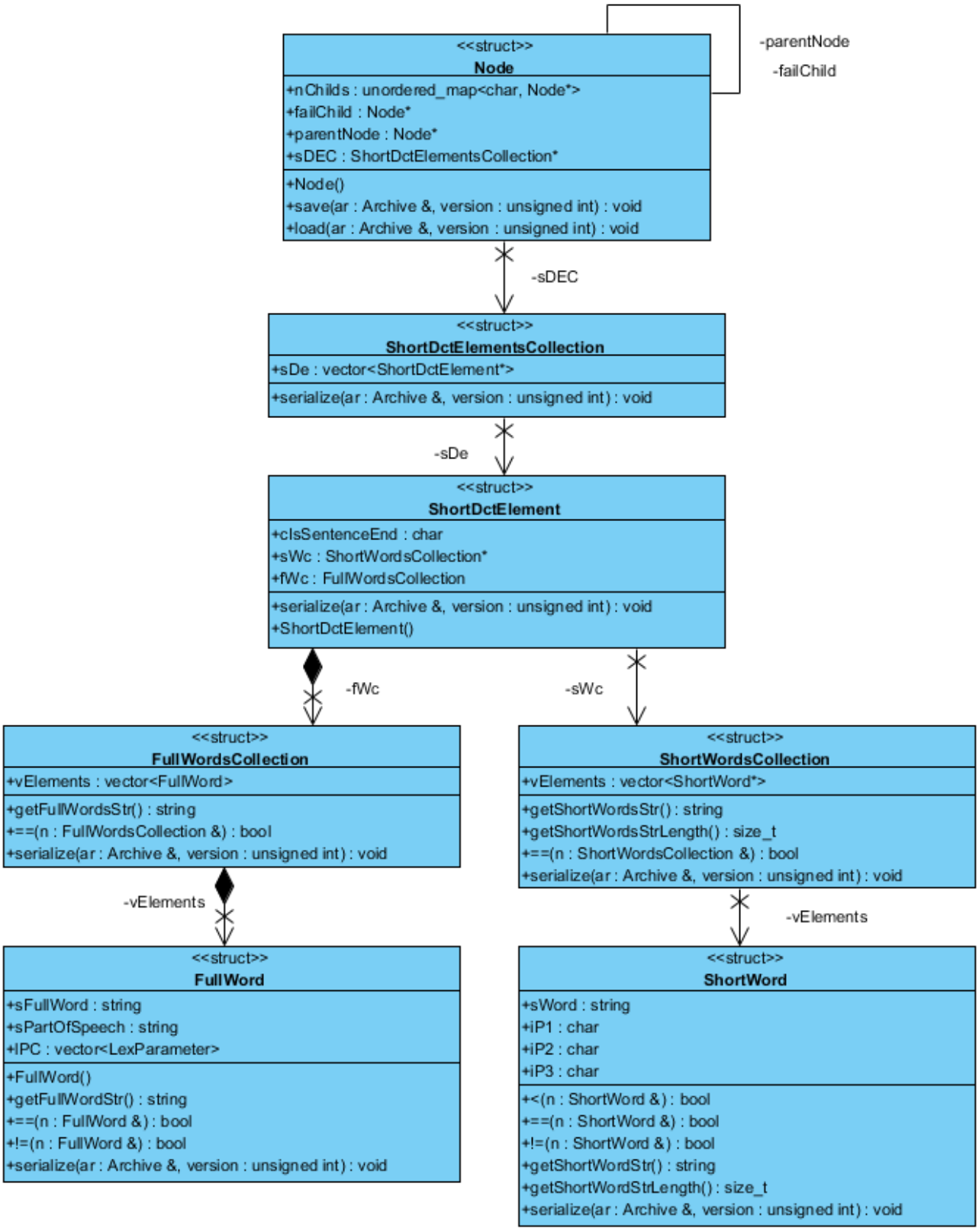

Рисунок 7 – UML диаграмма структур, в которых хранится словарь

Структура «Node» является состоянием конечного автомата. «nChilds» содержит указатели на состояния, в которые существуют переходы из текущего узла и символ, по которому осуществляется переход. «failChild» содержит

указатель на состояние, определенное функцией отказа. «parentNode» - указатель на состояние, из которого был совершен переход в текущее состояние. Данный указатель используется при вычислении «failChild». «sDEC» содержит указатель на множество сокращений и их расшифровок в случае, если состояние является заключительным. Иначе «sDEC» равно NULL.

Структура «ShortDctElementsCollection» содержит вектор указателей на сокращения и их расшифровки.

Структура «ShortDctElement» содержит информацию об одном сокращение (указатель «sWc») и его расшифровке («fWc»), а так же флаг того, может ли данное сокращение находиться в конце предложения («cIsSentenceEnd»).

«ShortWordsCollection» представляет собой список указателей на структуры «ShortWord», которые содержат в себе следующую информацию о каждом слове сокращения:

- «sWord» – слово;
- «iP1» – информация о регистре певой буквы;
- «iP2» – информация о регистре последующих букв;
- «iP3» – количество символов в слове.

Структура «FullWordsCollection» содержит в себе список структур «FullWord», которые хранят следующую информацию о каждом полном слове сокращения:

- «sFullWord» - строка с полным словом;
- «sPartOfSpeech» - часть речи;
- «lPc» - список морфологических параметров, хранящихся во внутреннем формате системы морфологического анализа «Кросслейтор».

Таким образом, i-й элемент вектора «vElements» структуры «FullWordsCollection» будет соответствовать i-му элементу вектора «vElements» структуры «ShortWordsCollection». Однако, некоторые записи словаря содержат

полное словосочетание в одном элементе «ShortWord». Поэтому количество элементов в «FullWordsCollection» и «ShortWordsCollection» для одного «ShortDctElement» может не совпадать.

8 Пример функционирования

Программный продукт представляет собой консольное приложение. На вход приложению подаются имя входного файла с текстом, анализ которого необходимо произвести и имя выходного файла, в который будет записан текст с выделенными в нем графематическими дескрипторами.

Доступные флаги запуска программы:

- db_host – хост или ip адрес БД;
- db_port – номер порта БД;
- db_user – имя пользователя БД;
- db_pass – пароль БД;
- db_schema – имя схемы БД;
- dct_type – тип словаря сокращений и аббревиатур;
- dct_path – путь до словаря;
- dct_out_type – тип архива для сохранения словаря;
- dct_out_path – путь, по которому необходимо сохранить словарь;
- pat_path – путь до файла с шаблонами;
- buf_size – размер одного буфера текста в символах.

Флаги db_host, db_port, db_user, db_pass, db_schema содержат в себе параметры подключения к СУБД MySql для экспорта результатов графематического анализа. Если хотя бы один из флагов отсутствует, экспорт не производится. Параметры не являются обязательными.

Параметры dct_type и dct_path являются обязательными параметрами. Параметр dct_type может принимать следующие значения:

- 0 – текстовый файл (в данном случае необходим парсинг словаря);
- 1 – бинарный архив;
- 2 – текстовый архив.

Параметры dct_out_type и dct_out_path являются опциональными и задаются при необходимости экспорта словаря сокращений и аббревиатур в бинарный или текстовый архив. Для dct_out_type доступны следующие значения:

- 1 – бинарный архив;
- 2 – текстовый архив.

Параметр pat_path содержит путь до файла с шаблонами поиска и является обязательным параметром.

Параметр buf_size задаёт размер одного анализируемого буфера текста в символах. Параметр не является обязательным, значение по умолчанию – 500000.

Таким образом, строка запуска программы со всеми заполненными параметрами выглядит следующим образом: «*./grafematika -db_host 217.77.59.5 -db_port 18306 -db_user graf -db_pass graf -db_schema graf -dct_type 1 -dct_path short_dct.txt -dct_out_type 2 -dct_out_path short_dct.dat -pat_path regexps.txt -buf_size 200000 in.txt out.txt*».

Результаты работы программы отображены на рис 8. и рис. 9. На рис. 8 отображены выделенные в процессе анализа предложения. Рис. 9 содержит список выделенных графематических дескрипторов.

SENTENCE_ID	TEXT_ID	START_POS	LENGTH	TEXT
154804	7	1608	218	Продолжается рост раннего алкоголизма (по сравнению с 2000-м г. болезненность ал...
154805	7	1827	297	Злоупотребление психоактивными веществами ведет к таким тяжелым последствиям...
154806	7	2125	68	В целом по России зарегистрировано около 196 000 ВИЧ-инфицированных.
154807	7	2194	67	Специалисты полагают, что их реальное количество в 5-10 раз больше.
154808	7	2262	182	За последние десятилетия число наркоманов, зарегистрированных наркологами Каре...
154809	7	2445	145	Наркомания в большей степени распространена среди мужского населения, но в посл...
154810	7	2591	167	В целом можно отметить рост потребления наркотиков, расширение географии нарко...
154811	7	2759	103	Последствия, связанные со злоупотреблением наркотиками, угрожают национальной ...
154812	7	2863	162	Это заболевания и смертность населения молодого и работоспособного возраста, ум...
154813	7	3026	82	Проблемы социально-экономического характера коснутся государства в ближайшие г...
154814	7	3109	47	Не менее остро стоит проблема с табакокурением.
154815	7	3157	144	Большинство специалистов считает, что никотинизм является вариантом токсикомани...
154816	7	3302	127	По данным ВОЗ, в начале 90-х гг. на планете существовал 1,1 млрд курильщиков, что ...
154817	7	3430	86	В большинстве стран Европы свыше 50% мужчин выкуривает в среднем по 15 сигарет ...
154818	7	3517	77	Число курящих женщин колеблется от 10 до 50%, превышая во многих странах 30%.
154819	7	3595	122	За последние годы доля курящих людей в ряде стран стала уменьшаться, но абсолютн...
154820	7	3718	82	В России резко увеличилось число курящих людей, особенно среди детей и подростков.
154821	7	3801	59	Средний возраст приобщения к курению снизился до 11-13 лет.
154822	7	3861	92	Распространенность курения среди мужского населения возросла с 53% в 1985 г до 6...

Рисунок 8 – Выделенные в тексте предложения, экспортированные в БД

DESCRIPTOR_ID	TEXT_ID	SENTENCE_ID	TYPE_NAME	START_POS	LENGTH	TEXT	FULL_WORDS_STR
2012326	7	154805	RUSSIAN_LEX	1986	2	по	NULL
2012327	7	154805	RUSSIAN_LEX	1989	6	данным	NULL
2012332	7	154805	FIO	1996	16	Воробьевой Н. И.	NULL
2012328	7	154805	RUSSIAN_LEX	1996	10	Воробьевой	NULL
2012329	7	154805	RUSSIAN_LEX	2007	1	Н	NULL
2012330	7	154805	PUNCTUATION_CHAR	2008	1	.	NULL
2012331	7	154805	RUSSIAN_LEX	2010	1	И	NULL
2012333	7	154805	PUNCTUATION_CHAR	2011	1	.	NULL
2012334	7	154805	PUNCTUATION_CHAR	2012	1	,	NULL
2012335	7	154805	RUSSIAN_LEX	2014	5	среди	NULL
2012337	7	154805	OTHER_LEX	2020	18	ВИЧ-инфициров...	NULL
2012336	7	154805	SHORT_WORD	2020	3	ВИЧ	вирус иммунодефи...
2012338	7	154805	RUSSIAN_LEX	2039	1	в	NULL
2012339	7	154805	RUSSIAN_LEX	2041	7	Карелии	NULL
2012340	7	154805	RUSSIAN_LEX	2049	2	на	NULL
2012341	7	154805	NUM_SEQ	2052	4	2001	NULL
2012343	7	154805	SHORT_WORD	2057	2	г.	год,город
2012342	7	154805	RUSSIAN_LEX	2057	1	г	NULL
2012344	7	154805	PUNCTUATION_CHAR	2058	1	.	NULL

Рисунок 9 – найденные в тексте графематические дескрипторы

Для оценки качества работы системы была проанализирована коллекция из 20 текстов. Основными показателями качества являются:

– процент корректно выделенных предложений;
– процент корректно выделенных сокращений и аббревиатур.

Процент корректно выделенных в текстах предложений составил более 97%. При рассмотрении нескольких десятков случаев некорректного выделения предложений и исключения из них опечаток, процент правильно выделенных предложений приблизился к 98%.

Процент корректно найденных сокращений и аббревиатур составил 95%. Ошибочно найденные сокращения делятся на две группы:

– сокращения, состоящие из одного символа (например, «А» – ампер);
– сокращения, совпадающие с короткими словами, стоящими в конце предложения (например, «раз.» – разъезд, «мак.» – македонский).

Анализ текстового файла размером 1 мегабайт, содержащего более 10 тысяч предложений, занял 11 секунд на компьютере с процессором Intel Pentium III с частотой 870 МГц и 1,5 секунды на компьютере с процессором Intel Core i5 с частотой 2,4 ГГц.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

[1] Компьютерная лингвистика | Энциклопедия Кругосвет. [Электронный ресурс] URL: http://files.school-collection.edu.ru/dlrstore/0032e870-dadd-5ab2-a83e-85e032c459b2/1009220A.htm

[2] Большакова Е. И., Клышинский Э. С., Ландэ Д. В., Носков А. А., Пескова О. В., Ягунова Е. В. Автоматическая обработка текстов на естественном языке и компьютерная лингвистика: учебное пособие. М.: МИЭМ, 2011. 272 с.

[3] Задачи компьютерной лингвистики. [Электронный ресурс] URL: http://www.kompling.narod.ru/index1.html

[4] Естественный язык – Википедия. [Электронный ресурс] URL: http://ru.wikipedia.org/wiki/Естественный_язык

[5] Токенизация. [Электронный ресурс] URL: http://www.stardesign.org/bible/gloss/t/tokenization/

[6] Обработка текста – NLPub. [Электронный ресурс] URL: http://nlpub.ru/wiki/Обработка_текста

[7] Каталог лингвистических программ и ресурсов в Сети / Linguistics Software Catalogue. [Электронный ресурс] URL: http://www.rvb.ru/soft/catalogue/catalogue.html

[8] Lingsoft, Inc. – Lingsoft. [Электронный ресурс] URL: http://lingsoft.fi

[9] Home - FreeLing Home Page. [Электронный ресурс] URL: http://nlp.lsi.upc.edu/freeling/

[10] АОТ :: Технологии :: Графематика: программный интерфейс. [Электронный ресурс] URL: http://www.aot.ru/docs/graphan.html

[11] Сокирко А.В. Семантические словари в автоматической обработке текста: По материалам системы ДИАЛИНГ: Диссертация кандидата технических наук: 05.13.17 / Москва, 2001. 120 с.

[12] Графематика: программный интерфейс. [Электронный ресурс] URL: http://src.gnu-darwin.org/ports/textproc/lemmatizer/work/lemmatizer-1.0/Docs/Graphan.htm

[13] Сегментация и токенизация текста [Электронный ресурс] URL: http://solarix.ru/for_developers/docs/tokenizer.shtml

[14] Apache OpenNLP. [Электронный ресурс] URL: http://opennlp.apache.org/

[15] The Stanford Natural Language Processing Group. [Электронный ресурс] URL: http://nlp.stanford.edu/software/corenlp.shtml

[16] Twitter NLP and Part-of-Speech Tagging. [Электронный ресурс] URL: http://www.ark.cs.cmu.edu/TweetNLP/

[17] Natural Language Toolkit. [Электронный ресурс] URL: http://nltk.org/

[18] Розенталь Д. Э., Джанджакова Е. В., Кабанова Н. П. Справочник по правописанию, произношению, литературному редактированию. М: ЧеРо, 1998. 400 с.

[19] Адаманский А. Обзор методов и алгоритмов полнотекстового поиска. [Электронный ресурс] URL: http://callisto.nsu.ru/documentation/searchreview.pdf

[20] Trie, или нагруженное дерево / Хабрахабр. [Электронный ресурс] URL: http://habrahabr.ru/post/111874/

[21] Trie - Wikipedia, the free encyclopedia. [Электронный ресурс] URL: http://en.wikipedia.org/wiki/Trie

[22] Алгоритм Ахо — Корасик — Википедия. [Электронный ресурс] URL: http://ru.wikipedia.org/wiki/Алгоритм_Ахо_—_Корасик

[23] Конечный автомат — Википедия. [Электронный ресурс] URL: http://ru.wikipedia.org/wiki/Конечный_автомат

[24] Алгоритм Ахо-Корасик. [Электронный ресурс] URL: http://aho-corasick.narod.ru/

[25] Alfred V. Aho and Margaret J. Corasick. Efficient String Matching: An Aid to Bibliographic Search. Communications of the ACM. 1975. Vol. 18 Issue 6. P. 333-340.

[26] Serialization. [Электронный ресурс] URL: http://www.boost.org/doc/libs/1_53_0/libs/serialization/doc/index.html

[27] Boost. Regex. [Электронный ресурс] URL: http://www.boost.org/doc/libs/1_53_0/libs/regex/doc/html/index.html

[28] Регулярные выражения в C++: использование Boost::Regex | Блог Alno. [Электронный ресурс] URL: http://blog.alno.name/ru/2008/10/using-boost-regex

[29] MySQL 5.7 Reference Manual. MySQL C API. [Электронный ресурс] URL: http://dev.mysql.com/doc/refman/5.7/en/c-api.html