«Балтийский государственный технический университет «ВОЕНМЕХ» им. Д.Ф. Устинова»
(БГТУ «ВОЕНМЕХ» им. Д.Ф. Устинова»)

Факультет И (шифр) Информационные и управляющие системы (наименование)

Кафедра И5 (шифр) Информационные системы и программная инженерия (наименование)

Дисциплина Научно-исследовательская работа в семестре

ОТЧЁТ
О НАУЧНО-ИССЛЕДОВАТЕЛЬКОЙ РАБОТЕ

РАССМОТРЕНИЕ СПОСОБОВ АНАЛИЗА СТРУКТУРЫ ДОКУМЕНТОВ И СПОСОБОВ ИСПОЛЬЗОВАНИЯ СТРУКТУРЫ ДОКУМЕНТОВ ДЛЯ ПОСТРОЕНИЯ ПОЛЬЗОВАТЕЛЬСКИХ ИНТЕРФЕЙСОВ

Выполнил студент группы И9М33
Магомедов И.Н.
Фамилия И.О.

РУКОВОДИТЕЛЬ

к.т..н. доцент
Ракова И. К.
Фамилия И.О. Подпись

Оценка _____

«_____» _____ 20___ г.

Санкт-Петербург
2019 г.

РЕФЕРАТ

Отчёт 36 с., 2 рис., 29 источников.

Ключевые слова: способы анализа структуры документов, анализ структуры документов, использование структур документов, способы применение структуры документа в пользовательских интерфейсах.

Цель работы: исследование способов анализа и применения структур документов.

В ходе выполнения работы было проведено рассмотрение способов анализа структур документа и способ использования структур документов для построение пользовательских интерфейсов.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

При разработке программного обеспечения появилась потребность в анализе структуры документа и выделения из неё дополнительной информации, для последующего использования результатов в создании нового документа. Программа позволяет анализировать входные документы и компилировать на основе анализа новый документ для его дальнейшего использования.

Для улучшения анализа документа имеет смысл использовать способы структурного анализа, так как они позволяют выделить основные части текстового документа, что увеличивает производительность и качество обработки документов, а также позволяет создавать на основе структурного анализа выдержки или шаблоны основных документов.

В данной работе рассмотрены способы анализа структур документа и способ использования структур документов для построение пользовательских интерфейсов.

1 Рассмотрение способов анализа структуры документов на естественных языках

Прежде всего стоит определиться с возможными структурами документов. Структуры документов могут определены:

— форматом текстового документа (разметка, деление стилей и текста на отдельные файлы и т.д.);

— форматированием и содержанием текста.

Для первого имеет смысл применять способы анализа идея которых основывается на анализе спецификации языков разметки или языков используемых в организации хранения данных форматов текстовых документов. Для второго имеет смысл применять способы анализа идея которых основывается на анализе текста и его свойств.

Рассмотрим способы анализа форматов текстовых документов и способы анализа текста и его свойств.

1.1 Способы анализа форматов текстовых документов

Стоит выделить способы организации хранения текстовой информации и системы используемые для анализа текста организованного тем или иным способом. Ниже рассмотрены некоторые системы обработки естественно языковых (ЕЯ) текстов в рамках направлений, в которых они развивались [1].

1.1.1. Системы на базе разметки

Одна из крупнейших ветвей систем обработки ЕЯ текстов основана на представлении лингвистической информации в форме разметки (рисунок 1) в одном из стандартных форматов, например SGML и XML. При этом, различные компоненты системы обычно обмениваются размеченными документами в текстовой форме.

Рисунок 1: Системы на базе разметки

Архитектура LT-NSL [2] была разработан в 1996 году как система для обработки больших корпусов текста, возможно содержащих сложную лингвистическую информацию.

Для представления лингвистической информации в системе LT-NSL используется разметка в документе на языке SGML. Такое решение мотивировано тем, что:

– SGML – четко определенный язык, позволяющий задавать структурную информацию в тексте;

– доступ приложения к тексту в SGML может быть осуществлен с выбором необходимого уровня абстракции. В частности, приложение может просто игнорировать узлы, не затрагиваемые конкретным видом анализа;

– SGML способствует формальному описанию используемой нотации и предоставляет утилиты для проверки соответствия документов этому формальному описанию.

В процессе разработки системы авторы столкнулись с тем, что использование полного SGML как внутреннего представления данных неэффективно, поскольку требует больших затрат на его анализ и проверку корректности. Для решения этой проблемы было использовано решение в котором документ сначала проверяется на корректность и приводится к специальной упрощенной форме. При дальнейшей обработке предполагается, что документ находится в этой форме и заведомо корректен, что позволяет значительно упростить его обработку. Для разбора документов в упрощенной форме реализован специальный программный интерфейс, позволяющий

осуществлять это более эффективно чем обычные парсеры.

Использование разметки в тексте для представления лингвистической информации приводит к проблеме перекрывающихся элементов разметки. В LT-NSL эта проблема решается путем использования ссылок на разметку, находящуюся вне документа.

Одной из важных особенностей архитектуры является то, что текст вместе с лингвистической информацией рассматривается как поток данных, обрабатываемый с использование компонентов, соединяемых через каналы (pipe) UNIX. Каждый компонент выполняет свою специфическую задачу и работает как фильтр, т.е. принимает на вход поток данных и выдает модифицированный поток на выходе. Такой подход позволяет не хранить документ в памяти полностью, что важно при обработке больших корпусов текстов.

В процессе обработки текста используется специальный язык запросов для выделения конструкций текста, аналогичный современному XPath, позволяющий извлекать элементы по описанию пути к ним от корня документа и задавать дополнительные требования, такие как необходимость наличия подэлементов или текста, удовлетворяющего регулярному выражению.

Другой системой, использующей разметку для представления лингвистической информации является Wraetlic NLP suite [3], разрабатываемая в 2000-2006 годах. Использование разметки мотивировано возможностью легко объединять компоненты в систему используя стандартные средства операционной системы, такие как каналы.

Лингвистическая разметка представляется в форме XML, при этом проблема представления пересекающейся разметки решена путем использования внешних XML-узлы и ссылок в документе. Для обработки такой разметки реализован специальный модуль, позволяющий извлекать участки документа по ссылкам.

Преимуществом использования XML является возможность простого создания инструментов для визуализации результатов в виде XSLT-преобразований, генерирующих данные для браузера.

Для увеличения эффективности, модули могут быть объединены в один процесс и использовать программный интерфейс на Java для доступа к обрабатываемым данным, что исключает накладные расходы, связанные с генерацией и разбором XML.

Система имеет модульную архитектуру, предоставляя разработчику возможность расширять его новой функциональностью. Большинство модулей реализовано на Java, но некоторые написаны на C из соображений производительности или же использования платформо-зависимых библиотек.

Система содержит модули для графематического и морфологического анализа, извлечения именованных сущностей, классификации и поверхностного синтаксического анализа.

1.1.2. Системы на базе аннотаций

Значительное влияние на ЕЯ-системы оказал проект TIPSTER [4], проводимый в 1991-1998 годах. Целью проекта было проведение исследований и разработок в области извлечения информации. Одним из направлений проекта было создание стандартной архитектуры для систем, осуществляющих извлечение информации из текстов на естественном языке, позволяющей решить следующие задачи:

- предоставление программного интерфейса для задач управления документами в системе и извлечения информации из них;
- поддержка одноязычных и многоязычных приложений;
- обмен модулями различных разработчиков;
- работоспособность в различном аппаратном и программном окружении;
- масштабируемость на различные объемы документов и потоки их

обработки;

- небольшое время отклика приложений;
- возможность использования многоуровневой системы безопасности.

В модели данных TIPSTER центральным объектом является документ, который является:

- хранилищем информации о тексте в виде атрибутов и аннотаций;
- атомарным элементом коллекций;
- атомарным элементом в операциях выделения.

Документы организованы в коллекции, которые также могут иметь атрибуты. Коллекции представляют хранилище для документов в рамках архитектуры TIPSTER. Аннотации хранят информацию о фрагментах документов. Каждый фрагмент представляется в виде набора отрезков, каждый из которых состоит из пары чисел, указывающих позиции его начала и конца в документе. Хотя большинство аннотаций привязываются только к одному отрезку текста, поддержка набора отрезков позволяет ссылаться на разрывные фрагменты текста.

Архитектура TIPSTER не позволяет аннотациям менять текст (например, вставлять символы), поскольку это требует представления позиций в тексте, позволяющих вставки.

Каждая аннотация имеет тип (например, «ЋлексемаЛ», «ЋпредложениеЛ» и т.п.) и набор атрибутов. Атрибуты могут содержать как простые значения (например, часть речи), так и более сложные, например, ссылки на другие аннотации.

В рамках архитектуры TIPSTER предполагается, что приложение осуществляет доступ к аннотациям очень часто. Для того, чтобы упростить такой доступ введено понятие набора аннотаций (AnnotationSet), который позволяет искать аннотации по позиции или типы, последовательно перебирать

их, добавлять, удалять и т.п. Каждый документ содержит набор аннотаций, представляющий все аннотации этого документа.

Основные процедуры обработки текстов реализуются в приложениях в виде компонентов, которые генерируют новую разметку в форме аннотаций. Компонент принимает на обработку документ (с некоторой разметкой, сгенерированной предыдущими компонентами или без нее) и добавляет новые аннотации в разметку. Идеи, заложенные в TIPSTER были успешно использованы во многих ЕЯ-системах (рисунок 2), некоторые из которых будут рассмотрены ниже.

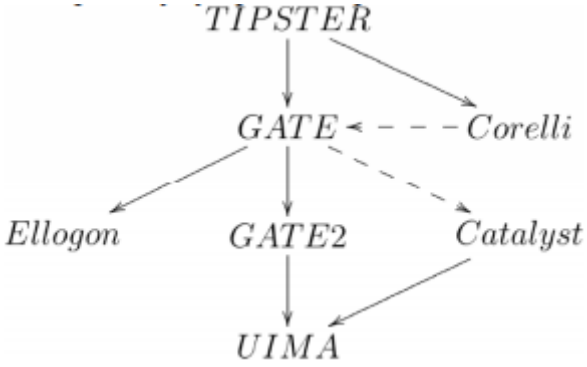

Рисунок 2: Системы основанные на TIPSTER

Одной из наиболее известных систем для автоматической обработки текстов на естественном языке, основанных на идеях проекта TIPSTER, является GATE [5] (General Architecture for Text Engineering). GATE предоставляет для приложений общую модель представления, хранения и обмена данными между компонентами приложения, а также графические инструменты для управления данными, их визуализации и анализа.

Первая версия GATE была представлена в 1996 году и предназначалась для решения следующих задач:

– обмен информацией между компонентами в максимально общей теоретико-нейтральной форме;

– интеграция компонентов, написанных на различных языках программирования на различных платформах;

– тестирование и доработка лингвистических компонентов и систем, построенных из них через единообразный графический интерфейс.

GATE использует модель данных, разработанную в рамках проекта TIPSTER, однако, позволяет осуществлять преобразование из аннотаций в XML-разметку и обратно, что позволяет интегрировать приложения со стандартными инструментами по обработке XML-документов.

Система состоит из трех слоев:

– GDM (Gate Document Manager) - хранилище текстов и сопутствующей лингвистической информации, предоставляющее компонентам единообразный интерфейс для доступа и манипуляций с данными.

– CREOLE (Collection of REusable Objects for Language Engineering) - набор переиспользуемых лингвистических компонентов для различных задач анализа текстов. Компоненты используют GDM для доступа к данным. Некоторые из этих компонентов были специально разработаны для использования в GATE, другие являлись обертками вокруг уже существующих.

– GGI (Gate Graphical Interface) - графический интерфейс, который позволяет использовать ресурсы GDM и CREOLE для интерактивного создания и тестирования компонентов и приложений. Интерфейс позволяет создание, просмотр и редактирование коллекций документов, которые управляются GDM, а также отображение результатов работы компонентов в форме аннотаций.

Использование GATE в различных проектах по обработке текстов на естественном языке и извлечению информации показало [6] преимущества ее использования, выраженные в том, что GATE:

– способствует переиспользованию лингвистических компонентов, что уменьшает усилия, необходимые для интеграции и разработки приложений;

– способствует объединению усилий в лингвистических исследований за счет представления общей базы для разработки компонентов и приложений;

– облегчает сравнение алгоритмов и их реализаций в виде компонентов;

– объединяет лучшие элементы подхода TIPSTER с возможностью экспорта аннотаций в XML (а также импорта из него);

– предоставляет удобный графический интерфейс.

Так же были выявлены некоторые недостатки GATE, в частности:

– не поддерживаются компоненты, представляющих источники данных;

– не поддерживается генерация текста;

– реализация базы данных неэффективна;

– графический интерфейс иногда сложен и неочевиден;

– модель обработки не масштабируется на большое число компонентов;

– невозможно добавить новые компоненты для виузализации данных;

– для интеграции различных компонентов необходима совместимость их схем аннотаций.

Ellogon [7] – многоязыковая платформа для создания приложений обработки текстов на естественном языке, использующая идеи из проектов TIPSTER и GATE. Платформа была разработана в 2002 году как имеющая низкое потребление ресурсов и поддерживающая обработку текстов на различных языках (за счет использования Unicode). Ellogon предоставляет инфраструктуру для:

– управления, хранения и обмена текстами вместе с лингвистической информацией;

– создания, управления и интеграции лингвистических компонентов;

– поддержки взаимодействия между различными компонентами;

– визуализации текстовой и лингвистической информации.

Ellogon состоит из трех подсистем: ядра, графического интерфейса и

подключаемых компонентов. Ядро реализует всю базовую функциональность по управлению текстовой и лингвистической информацией, используемой в других частях системы. Подключаемые компоненты выполняют конкретные задачи лингвистической обработки. Ядро Ellogon написано на C++ из соображений эффективности, однако имеет программный интерфейс для других языков, таких как Tcl и Java.

В соответствии с архитектурой TIPSTER, лингвистическая информация представляется в форме аннотаций, причем одна аннотация может быть связана с несколькими непрерывными промежутками в тексте. Каждый компонент в Ellogon имеет набор предусловий и постусловий. Предусловия описывают лингвистическую информацию, которая должна быть представлена в документе для того, чтобы компонент мог осуществить его обработку, а постусловия описывают то, какая информация добавляется компонентом в текст в процессе обработки. Эта информация используется для определения порядка обработки документа. Также, каждый компонент определяет набор параметров, которые определяют его поведение и могут быть отредактированы пользователем в графическом интерфейсе и набор визуализаторов позволяющих представлять обрабатываемую информацию в графическом интерфейсе.

Для быстрой разработки компонентов Ellogon предоставляет специальные средства в графическом интерфейсе, позволяющие генерировать заготовки для новых компонентов, перезагружать компоненты после их редактирования и т.п. Компоненты могут быть реализованы как на C++ и работать напрямую с ядром так и на других языках с использованием программного интерфейса для дополнительных языков. Ключевыми возможностями Ellogon являются:

- полная поддержка Unicode и различных языков в ядре и графическом интерфейсе;
- портируемость на большинство платформ;
- совместимость с другими ЕЯ системами, такими как GATE за счет

предоставления специальных оберток для их программного интерфейса;

– сжатие данных в памяти, позволяющее значительно уменьшить использование памяти приложением;

– возможность работы в качестве веб-сервера, предоставляющего функциональность приложения через протокол HTTP.

Система Ellogon доступна для свободного использования и может быть загружена с сайта системы.

Для исправления недостатков первой версии GATE, был произведен полный редизайн системы и в 2001 году разработана новая версия GATE [8], реализованная на языке Java. Переход на Java позволил использовать представление текста в Unicode, что обеспечило поддержку обработки текстов на различных языках.

Одним из важнейших отличий стала поддержка компонентов трех типов:

– языковые ресурсы, которые хранят и предоставляют сервисы для доступа к различным типам лингвистических ресурсов, таким как документы, корпуса и онтологии; Хотя предоставляемые по умолчанию ресурсы предназначены для доступа к текстовой информации, система не содержит ограничений на использование ресурсов другого типа, что позволяет реализовать ресурсы для обработки информации других типов;

– обрабатывающие ресурсы — лингвистические компоненты, которые осуществляют различные лингвистические задачи, такие как выделение лексем, морфологический анализ, и т.п;

– визуальные ресурсы — графические компоненты, предоставляющие возможности визуализации и редактирования обрабатываемой информации или редактирования процесса выполнения приложения.

Каждый компонент GATE 2 имеет набор свойств, которые управляют его функционированием, например, для большинства обрабатывающих компонентов одним из таких свойств является ссылка на обрабатываемый

языковой ресурс. Выполнение приложения состоит в последовательном применении всех обрабатывающих компонентов к соответствующим им языковым ресурсам.

Модель данных из проекта TIPSTER была модифицирована в целях совместимости с форматом Atlas.

Также, в GATE 2 был введен специальный язык JAPE [9], позволяющий описывать процесс обработки разметки в компонентах в форме набора правил, основанных на регулярных выражениях. Использование этого языка значительно упростило создание многих компонентов для обработки естественного языка.

С использованием JAPE процесс обработки документа описывается в форме набора правил, каждое из которых состоит из двух частей, одна из которых определяет условия применения, а другая совершаемые действия.

Левая часть правила задает так называемый шаблон, который используется для выделения последовательности аннотаций. Шаблон может содержать элементы, сопоставляемые с аннотациями, а также операторы регулярных выражений (?,*,+,|). При этом в шаблоне могут быть заданы ограничения на свойства аннотаций, входящих в выделяемую последовательность, например Token.kind == number.

Правая часть Jape-правил описывает преобразования, применяемые к разметке в точке сопоставления левой части правила. Обычно такими преобразованиями является добавление новых аннотаций, с заданием им тех или иных свойств. Новые аннотации добавляются к фрагментам текста, соответствующим меткам, указанным в левой части правила.

Правила Jape применяются следующим образом: производится поиск в тексте последовательности аннотаций, соответствующей левой части правила, затем меткам ставятся в соответствие фрагменты текста, к которым в последствии добавляются аннотации правой части.

Система GATE 2 доступна для свободного использования и может быть загружена с сайта системы.

Catalyst [10] – это архитектура, выделенная из вопросно-ответной системы Qanda в 2002 году и предназначенная для решения задач обработки текстов на естественном языке и извлечения информации. Исходная система Qanda впоследствии послужила прототипом для разработки других систем на базе архитектуры.

Основными причинами для разработки архитектуры послужил тот факт, что существующие системы плохо масштабировались при росте объемов обрабатываемой информации и требований к скорости ее обработки.

Для представления данных Catalyst использует модель, основанную на аннотациях TIPSTER.

Важным отличием системы Catalyst от других, является использование концепции потоков данных для интеграции компонентов. Каждый компонент соединяется с другими каналами, по которым передаются аннотации, упорядоченные в соответствии с их позициями в документе (а для совпадающих позиций - по типам аннотаций).

Каждый компонент объявляет какие типы аннотаций необходимы ему для обработки и какие аннотации он генерирует на выходе. Такая информация позволяет не передавать между компонентами аннотации, которые не будут использован и значительно уменьшить кол-во информации, передаваемой по каналам, что важно при построении хорошо масштабируемых распределенных системы.

Такой подход к интеграции компонентов имеет ряд преимуществ:

– многие ошибки, связанные с неправильной организацией приложения могут быть выявлены на этапе сборки;

– отсутствуют накладные расходы на преобразование разметки в какой-либо стандартный формат (например, на базе XML);

– разработчики компонентов могут работать непосредственно с аннотациями, а не с разметкой в каком-либо формате;

– система может работать как на одной машине, так и распределенно. Компоненты могут быть реплицированы для увеличения производительности;

– код компонентов упрощается поскольку в них не требуется проверка данных на корректность (она выполняется на этапе сборки приложения).

Поскольку отладка и мониторинг распределенной системы может представлять значительные сложности Catalyst предоставляет возможности для распределенного логирования и мониторинга. Использование распределенного логирования позволяет собирать информацию со всех или указанных процессов в системе. Собранная информация включает события начала и завершения обработки, пользовательские сообщения, сообщения об ошибках и т.п. Логирование может быть включено при запуске системы или в процессе ее функционирования. Система мониторинга позволяет отслеживать конфигурацию и состояние приложения, включая потоки передаваемых данных в процессе его функционирования. Предоставляется информация об активных компонентах, потоках данных, количестве буферизованных сообщений и т.п.

Работа с Catalyst состоит из написаний компонентов для фреймворка и сборке приложений из компонентов. Catalyst разработан так, чтобы в первую очередь упростить вторую задачу - сборка приложения представляет из себя размещение компонентов и соединение их каналами.

Компоненты для фреймворка могут быть разработаны непосредственно с использованием модели аннотаций или представлять из себя обертки вокруг уже существующих компонентов.

1.1.3. Системы интеграции поверхностной и глубокой обработки

Система SProUT [11; 12] (Shallow Processing with Unification and Typed Feature Structures) была разработана для задач поверхностной обработки текста в 2002-2004 годах. Мотивацией для разработки системы послужила

необходимость в системе, которая позволяла бы гибко интегрировать различных обрабатывающих компонентов и в то же время представляла бы хорошее соотношение между производительностью и выразительностью используемого формализма.

Идеей системы было объединение формализмов конечных преобразователей, для которых существуют эффективные алгоритмы, и унификационных грамматик, позволяющих естественным образом выражать синтаксические и семантические ограничения. В качестве средства для такого объединения использовалась машина с конечным числом состояний, работающая над типизированными структурами признаков. Таким образом правила преобразований в правой части содержат регулярное выражения над структурами признаков, а правая часть представляет выходную структуру признаков. При этом ограничения на равенство признаков заменяются их унифицируемостью.

Описанный формализм был расширен путем добавления функциональных операторов и возможности вызова дополнительных правил в процессе сопоставления. Функциональные операторы позволяют расширять формализм путем подключения новых функций, используемых для вычисления значений в результирующей структуре. Использование вызова дополнительных правил позволяет вызывать в левой части правил сопоставление других правил (или, возможно, того же самого правила) тем самым расширяя выразительность формализма до контекстно-свободного ценой небольшого снижения эффективности сопоставления (поскольку такой вызов приводит к порождению нового процесса сопоставления).

Ядро системы состоит из четырех основных компонентов - инструментария для обработки конечных машин, компилятора регулярных выражений, интерпретатора формализма XTDL и пакета типизированных структур признаков. С использованием этих компонентов разработаны

переиспользуемые компоненты для обработки лингвистической информации. Компоненты легко интегрируются внутри системы поскольку имеют унифицированное представление данных в виде типизированных структур признаков.

Компоненты осуществляют обработку последовательно, но возможна более сложная конфигурация путем использования специально разработанного языка описания процесса обработки.

Система Whiteboard [13; 12] была разработана в 2000-2002 годах и предполагала возможность интеграции лингвистических компонентов для поверхностной и глубокой обработки текста.

Такая интеграция компонентов для поверхностного и глубокого анализа проблематична в связи с различиями в их производительности и точности. Одно из возможных решений состоит в том, чтобы выполнять анализ параллельно, используя результаты глубокого анализа при их наличии. Однако, для больших наборов данных такой подход приводит к рассинхронизации работы компонентов. Авторы системы предложили решение, основанное на анализе данных с использованием компонентов поверхностного анализа для определения участков, которые необходимо обработать с помощью компонентов глубокого анализа. Кроме того, использование результатов поверхностного анализа на таких участках может быть использовано в качестве дополнительной эвристики для компонентов глубокого анализа, увеличивая производительность их работы.

Для представления лингвистической информации на поверхностном уровне в системе используется XML-разметка. При этом более сложные структуры, которые не могут быть выражены в XML хранятся отдельно и доступны компонентам, выполняющим глубокую обработку. Это позволяет, с одной стороны компонентам поверхностного анализа работать с хорошо известным представлением, а компонентам глубокого анализа не быть

ограниченными этим представлением.

Архитектура Heart of Gold [14; 15; 12] была разработана в 2004-2005 как развитие Whiteboard исправляющее ряд ее основных недостатков. Задача архитектуры состоит в том, чтобы сохранить преимущества поверхностной обработки текста (в первую очередь устойчивость и эффективность), но при этом увеличить точность и глубину анализа в тех местах, где это необходимо.

В качестве единого формата для представления данных выбран RMRS. Для компонентов, которые используют другие представления, основанные на XML, применяется преобразование данных, описываемое на языке XSLT.

Платформа работает как медиатор между приложениями и набором компонентов. Приложения посылают запросы об анализе документов центральному модулю, который рассылает запросы различным компонентам и затем осуществляет слияние результатов их обработки. При этом, результаты запросов сохраняются в базе данных, что позволяет избежать повторной обработки при получении тех же запросов. Параметры запросов указывают на анализируемый документ, участок в тексте, который необходимо проанализировать, а также необходимую глубину анализа.

В соответствии с идеей, заложенной в систему результат обработки каждого компонента представляет из себя недоспецифицированную семантическую информацию, которая может быть углублена в процессе дальнейшей обработки. В соответствии с этим, стратегия обработки запроса состоит в том, чтобы обработать запрос с помощью всех компонентов начиная с минимальной глубины до заданной в запросе, откатываясь к результату работы предыдущего компонента в случае, если какой-либо компонент не обработал запрос. Для возможности последующего анализа условий при которых был получен тот или иной результат, вместе с ним сохраняется метаинформация о переданных параметрах, времени обработки запроса и т.п.

Ядро системы написано на Java, однако компоненты и приложения могут

быть написаны на любых языках и осуществлять взаимодействие с ядром через протокол XML-RPC.

1.1.4. Системы, развивающие отдельные аспекты обработки текста

Здесь представлены другие системы, осуществляющие обработку текстов на естественном языке так или иначе интересные идеями, заложенными в них.

Система Fastus [16] была разработана в 1994 как средство для эффективного и точного извлечения информации из текстов и интересна в первую очередь тем, что являлась одной из первых систем, использующих регулярные шаблоны для решения задач извлечения информации.

При разработке системы были учтены такие особенности извлечения информации из текстов, как релевантность только небольшой части анализируемого текста, необходимость отображения информации в заранее определенное относительно пространственное представление и незначительность многих аспектов, связанных со значением и целью написания текста. При этом рассмотрение извлечения информации как подзадачи понимания текста и применение систем, предназначенных для понимания текстов приводило к относительно низкому качеству результатов при низкой эффективности обработки.

Для преодоления этой проблемы авторами было предложено использовать для извлечения информации более простой регулярный формализм вместо обычно используемых контекстно-свободных с тем, чтобы увеличить эффективность работы системы. Работа системы FASTUS состояла из четырех этапов:

- обнаружение ключевых слов, свидетельствующих о наличии в предложениях релевантной информации;
- выделение именных групп, глагольных групп и важных классов слов, таких как предлоги, союзы, и т.п;
- выделение необходимой информации по шаблонам. Шаблоны для

выделения представлялись в виде конечных автоматов, где переходы соответствовали ключевым словам или словосочетаниям заданного типа, выделенным на предыдущем этапе. При этом в шаблонах использовались конструкции, позволяющие выделять более сложные языковые конструкции (например, сложные именные группы) для представления более полной информации в результирующей структуре.

– структуры, выделенные для одного и того же предложения сливались в одну для получения наиболее полной информации об описываемом событии.

Проведенные эксперименты показали эффективность выбранного подхода для извлечения информации.

Система CAFE [17] (Cascading, Asynchonous, Feedback Envrionment) разрабатывалась в 2001 году и предназначалась для решения задач распознавания речи.

CAFE интересна в первую очередь принципами взаимодействия компонентов, основанных на идеях так называемого Ђнепрерывного пониманияЛ, в котором компоненты, обрабатывающие данные не дожидаются полных результатов предыдущего этапа, а производят анализ параллельно, при этом предоставляя компонентам предыдущего информацию, корректирующую их функционирование.

Традиционно взаимодействие компонентов, составляющих систему строится на основе предоставления каждым компонентом единственного наилучшего результата анализа на основе информации на своем уровне. Однако, во многих случаях выбор различных вариантов на определенном уровне анализа не может быть осуществлен только на основе информации этого уровня, а требует использования информации с более глубоких уровней анализа текста.

Для решения этой проблемы в CAFE используется следующая мождель

взаимодействия между компонентами, реализующими анализ на различных уровнях:

– каждый компонент предоставляет следующему уровню результаты своего анализа постепенно, по мере их получения не дожидаясь окончания обработки всего документа или предложения; Соответственно, информацию от предыдущего уровня компонент также получает постепенно и может возникать ситуация в которой новой информации еще не поступило;

– компонент передает на следующий уровень несколько наилучших вариантов анализа на своем уровне, при этом в случае отсутствии информации от предыдущего уровня компонент может продолжать передавать следующему уровню оставшиеся варианты для текущей порции данных;

– компонент передает на предыдущий уровень информацию об оценке полученных вариантов анализа, что позволяет скорректировать направление анализа на предыдущем этапе.

Таким образом в системе CAFE во-первых достигается больший уровень параллелизма работы компонентов, а во-вторых существует обратная связь между этапами, которая помогает скорректировать получаемые результаты.

Система LinguaStream [18] разработана в 2003 году и основывается на использовании декларативного описания процесса обработки, который может быть представлен в виде графа.

Приложение разабатывается путем выбора компонентов, каждый из которых имеет набор параметров, входов и выходов и их соединении. Платформа основана на использовании XML и может обрабатывать любой XML-файл, сохраняя его изначальную структуру.

В платформе используются декларативные механизмы описания процесса обработки.

Система использует идею, согласно которой различные модели анализа дополняют друг друга, и, соответственно, не отдает предпочтения какому-либо

из них. Для обеспечения совместимости между различными компонентами используется унифицированное представление разметки и аннотаций в виде наборов признаков (feature sets).

Важным аспектом является возможность использования различных минимальных единиц на различных этапах анализа. Когда какая-либо модель требует наличия минимальной единицы анализа (например, лексемы), то эта единица может быть определена локально, только для соответствующего компонента. Кроме того, каждый компонент отмечает какие элементы разметки он обрабатывает. Описанные возможности позволяют определить различные точки зрения на обрабатываемый документ для каждого этапа.

Каждое приложение может быть переиспользовано в качестве компонента для создания более сложного процесса обработки.

Помимо компонентов, выполняющих конкретные задачи обработки текста, платформа включает компоненты, представляющие различные формализмы реализации задач обработки, например:

- Унификационных грамматик (на базе Prolog);
- Преобразований с конечным числом состояний;
- Грамматик, основанных на ограничениях;
- Регулярных выражений.

Система Learning Based Java [19] представляет средства для интеграции и обучения различных статистических компонентов.

В основе системы лежит представление задачи анализа текста в виде поиска набора выходных данных, максимизирующего некоторые оценочные функции и при этом удовлетворяющего заданным ограничениям.

Приложение в системе представляется как набор моделей, описывающих признаки, передаваемые на вход статистической модели. В качестве примера рассмотрим простую модель, осуществляющую выделение в качестве признаков множества слов новостной статьи:

```
/** This feature generating classifier "senses" all the
* words in the document that begin with an alphabet
* letter. The result is a bag-of-words representation
* of the document. */
discrete% BagOfWords(Post post) <- {
  for (int i = 0; i < post.bodySize(); ++i)
    for (int j = 0; j < post.lineSize(i); ++j) {
      String word = post.getBodyWord(i, j);
      if (word.length() > 0 &&
        word.substring(0, 1).matches("[A-Za-z]"))
       sense word;
    }
}
/** The label of the document. */
discrete NewsgroupLabel(Post post) <-
  { return post.getNewsgroup(); }
```

В приложении задается используемый компонент машинного обучения и модели, признаки из которых используются для обучения и распознавания объектов:

```
/** Here, we train averaged Perceptron for many
* rounds of the training data. **/
discrete NewsgroupClassifierAP(Post post) <-
learn NewsgroupLabel
  using BagOfWords
  from new NewsgroupParser("data/20news.train.shuffled")
    40 rounds
  with SparseNetworkLearner {
    SparseAveragedPerceptron.Parameters p =
```

```
    new SparseAveragedPerceptron.Parameters();
   p.learningRate = .1;
   p.thickness = 3;
   baseLTU = new SparseAveragedPerceptron(p);
  }
  progressOutput 20000
  testFrom new NewsgroupParser("data/20news.test")
end
```

1.1.5. Прочие системы

Система Corelli [20] была разработана в 1996 году и предназначена для интеграции распределенных лингвистических компонентов, реализованных на различных языках.

В системе используется модель данных проекта TIPSTER, при этом тип представляемой информации не ограничивается. Для решения проблем совместимости в системе предоставляется библиотека, осуществляющая преобразование данных.

Система состоит из центрального сервера, реализованного на Java, который отвечает за хранение документов и сопутствующей лингвистической информации и различных обрабатывающих компонентов, которые получают необходимые данные от сервера документов. Компоненты взаимодействуют с центральным сервером напрямую с использованием программного интерфейса на Java или удаленно на базе CORBA. Кроме того, компоненты система поддерживает подключение и отключение компонентов в процессе выполнения.

Поскольку взаимодействие с центральным сервером осуществляется по фиксированному протоколу, его реализация может быть заменена в соответствии с нуждами приложения. В частности, предоставляется три основных версии центрального сервера использующие для хранения данных файловую систему, специальное объектное хранилище или реляционную базу

данных. В последнем случае центральный сервер предоставляет возможности для транзакционного взаимодействия.

Система UIMA разрабатывается с 2004 года по настоящее время. Для представления данных используется модель TIPSTER.

Обработка документов осуществляется последовательно, каждый компонент добавляет аннотации в представление документа.

Для аннотаций определяется система типов, обеспечивающая проверку совместимости аннотаций между различными компонентами. В случае несовпадения систем типов, отображение между ними может быть произведено путем реализации соответствующего компонента.

Система UIMA доступна для свободного использования и может быть загружена с сайта системы.

Система OpenPipeline предоставляет возможности для автоматизированной обработки документов в серверном приложении. Для системы задается расписание выполнения работ, каждая из которых состоит из получения данных из какого-либо источника и последовательного применения определенных этапов преобразования.

Система реализована как серверное J2EE-приложение.

Система TESLA предоставляет удобный графический интерфейс на базе среды разработки Eclipse для построения приложений естественно-языковой обработки. Компоненты в системе связываются каналами в ориентированный граф.

Система имеет клиент-серверную архитектуру - графический интерфейс выступает в роли клиента и сам не выполняет задач по обработке текстов, а передает их серверу.

1.2 Способы анализа текста и его свойства

Среди возможных решений систематизации множества разнородных методик анализа текста и его свойств, наиболее известны деления по

выполняемым функциям и ориентировке [21].

А. По выполняемым функциям различают группу методов [22]:

а) способных осуществлять импорт текста и работу с ним;

б) исследования текста (работают на грамматическом, синтаксическом уровне, осуществляют разнообразный поиск в тексте, выделяют ключевые слова, индексы и пр.);

в) ориентированных на семантический анализ, создание схем категоризации, словарей, кодирования;

г) позволяющих осуществлять экспорт данных анализа (например, сам текст или схему кодирования, или используемый словарь и т.п.).

В. По объекту анализа [23]

а) методы, «ориентированные на язык» (лингвистических единиц):

- лингвистические методы;
- методы работы с данными (поиск информации, списки слов, конкорданс, индексы и пр.).

б) методы, «ориентированные на контент» или содержательный анализ:

- качественные методы, позволяющие осуществлять поиск закономерностей и различий в тексте, анализировать текст целиком (некоторые методики позволяют анализировать аудио- и видеоинформацию). В данной группе методов для проведения качественного (содержательного) исследования текста могут быть использованы и количественные данные, которые помогают организовать качественную (содержательную) информацию. Важным отличием является преимущественное использование в качестве единиц анализа тем, концептов, процессов, контекстов. При этом объем анализируемого текста может быть ограничен;
- методики анализа событий по текстовым данным;
- количественные методики, позволяющие осуществлять статистическую проверку гипотез, ориентированы на исследование больших

объемов текста:

– категориальные системы имеют встроенные или пользовательские словари, на основе которых осуществляется поиск в тексте (категории при этом могут быть как тематическими, так и семантическими), некоторые методики имеют ограничения по размеру единиц анализа;

– некатегориалъные системы на основе одновременной встречаемости слов, строк, концептов позволяют строить разнообразные графы и дендрограммы;

– системы кодирования ответов на вопросы незаконченных предложений не предусмотрены для анализа больших объемов текста, предназначены для анализа достаточно гомогенного текста и лимитированы по размеру единиц анализа текста.

Работа по систематизации существующих методов крайне необходима для объединения различных способов содержательного анализа текста, так как несколько методов могут находить свою реализацию в одной конкретной методике, где данные по ним дополняют друг друга увеличивая репрезентативность и валидность результатов.

При этом под методикой анализа текста понимается «конкретной вариант того или иного метода, направленный на решение определенного класса исследовательских задач» [24, с. 73].

Рассмотрим способы анализа текста для извлечения информации.

– Интент-анализ — метод, позволяющий реконструировать интенции (субъективная направленность на некий объект) автора по его тексту [25], поскольку для выявления и квалификации интенций опора на отдельные слова и предложения малопродуктивна.

– Контент-анализ — самый распространенный метод, имеющий множество вариаций в различных методиках, позволяющий провести качественно-количественный анализ содержания текстовых массивов с целью

последующей интерпретации выявленных числовых закономерностей.

– Фоносемантический анализ — текста или слова заключается в оценке его звучания безотносительно к его содержанию.

– Дискурс-анализ или дискурсивный анализ — совокупность методик и техник интерпретации текстов или высказываний как продуктов речевой деятельности, осуществляемой в конкретных общественно-политических обстоятельствах и культурно-исторических условиях [26].

– Нарративный анализ — это метод обобщения прошлого опыта при помощи соотнесения последовательности слов в предложении и последовательности реальных (как предполагается) событий [27].

– Экспертная оценка текста — в эту группу методов входят различные экспертизы текста, классификации которых, согласно А.А. Леонтьеву [28], можно представить в следующем виде:

а) автороведческая экспертиза, направленная на установление автора текста или выявление категориальных признаков вероятного автора: пол, возраст, национальность, место рождения, место долговременного проживания, уровень образования и пр. [29];

б) экспертиза, направленная на установление временных признаков автора текста (эмоциональное состояние и пр.);

в) экспертиза, направленная на установление тех или иных условий создания исследуемого текста (также экспертиза аутентичности записей при интервью);

г) экспертиза, направленная на установление преднамеренного искажения сведений, высказываемых в тексте;

д) экспертиза, направленная на установление определенных признаков (оскорбление, призыв и пр.).

– Морфологический анализ — направлен на определение множества морфологических интерпретаций каждого из слов текста, состоящей из таких

параметров, как лемма, морфологическая часть речи; набор общих граммем; множество наборов граммем.

– Синтаксический анализ — это метод сопоставления линейной последовательности лексем языка с его формальной грамматикой. Результатом анализа становится синтаксическая структура предложения, которая представляется в виде дерева зависимостей.

– Семантический анализ — метод, направленный на построение семантической структуры предложения, состоящей из семантических узлов и семантических отношений

2 Рассмотрение способов использования структуры документов на естественных языках для построения пользовательских интерфейсов программ обработки данных документов

Структуры документов можно использовать для:

- анализа и выделения мета информации, и построения на основе результатов анализа и выделенной мета информации пользовательского интерфейса;
- частичного копирования выделенных фактов, требуемых для построения и встраивания в пользовательский интерфейс документа;
- создание шаблона на основе выделенных структур документа.

Для первого пункта используются способы анализа текста и свойств, для дальнейшего применения результатов в разработки пользовательского интерфейса. Способы анализа текста и свойств представлены в разделе 1 данной работы.

Для второго пункта характерны схожие способы с первым пунктом, но при этом возможно частичное копирование текста.

Для третьего пункта используются свойственные первому и второму пункту способы анализа текста, но при этом способы применяются для выделение основных структурных элементов и создание из выделенных частей шаблона.

Для всех трёх пунктов применимым способы анализа текста изложенные в разделе 1 данной работы.

ЗАКЛЮЧЕНИЕ

В процессе выполнения научно-исследовательской работы были рассмотрены способы анализа структур документа и способы использования структур документов для построение пользовательских интерфейсов.

На первом этапе научно-исследовательской работы были рассмотрены способы анализа структуры документов на естественных языках. На данном этапе были определены возможные структуры текстовых документов для анализа и рассмотрены способы анализа форматов текстовых документов и способы анализа текста и его свойств.

Рассмотрение способов использования структуры документов на естественных языках для построения пользовательских интерфейсов программ обработки данных документов

На втором этапе научно-исследовательской работы были рассмотрены способы использования структуры документов на естественных языках для построения пользовательских интерфейсов программ обработки данных. Были выделены возможные способы применения структур документов.

Таким образом, все поставленные задачи в ходе выполнения научно-исследовательской работы были выполнены.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Большакова Е. И. и др. Автоматическая обработка текстов на естественном языке и компьютерная лингвистика. – 2015.

2. McKelvie D., Brew C., Thompson H. Using SGML as a basis for data-intensive NLP //Proceedings of the fifth conference on Applied natural language processing. – Association for Computational Linguistics, 1997. – С. 229-236.

3. Alfonseca E. et al. The wraetlic NLP suite //In Proceedings of the Language Resources and Evaluation Conference, LREC-2006. – 2006.

4. Bird S. et al. ATLAS: A flexible and extensible architecture for linguistic annotation //arXiv preprint cs/0007022. – 2000.

5. Bird S., Liberman M. A formal framework for linguistic annotation //Speech communication. – 2001. – Т. 33. – №. 1-2. – С. 23-60.

6. Maynard D. et al. A Survey of Uses of GATE. – Technical Report CS–00–06, Department of Computer Science, University of Sheffield, 2000.

7. Petasis G. et al. Ellogon: A new text engineering platform //arXiv preprint cs/0205017. – 2002.

8. Bontcheva D. K. et al. Gate: A unicode-based infrastructure supporting multilingual information extraction. – 2003.

9. Cunningham H., Maynard D., Tablan V. JAPE: a Java annotation patterns engine. – 1999.

10. Mardis S. et al. Qanda and the catalyst architecture //AAAI Spring Symposium on Mining Answers from Text and Knowledge Bases. – 2002.

11. Piskorski J., Schäfer U., Xu F. Shallow processing with unification and typed feature structures–foundations and applications //Knstliche Intelligenz. – 2004. – Т. 1. – №. 1.

12. Schäfer U. Integrating deep and shallow natural language processing components: representations and hybrid architectures. – 2006.

13. Crysmann B. et al. An integrated architecture for shallow and deep processing //Proceedings of the 40th annual meeting on association for computational linguistics. – Association for Computational Linguistics, 2002. – С. 441-448.

14. Callmeier U. et al. The DeepThought Core Architecture Framework //LREC. – 2004.

15. Schäfer U. Middleware for creating and combining multi-dimensional NLP markup //Proceedings of the 5th Workshop on NLP and XML: Multi-Dimensional Markup in Natural Language Processing. – Association for Computational Linguistics, 2006. – С. 81-84.

16. Appelt D. E. et al. FASTUS: A finite-state processor for information extraction from real-world text //IJCAI. – 1993. – Т. 93. – С. 1172-1178.

17. Stoness S. C. Continuous understanding: A first look at CAFE. – 2001.

18. Bilhaut F., Widlöcher A. Linguastream: an integrated environment for computational linguistics experimentation //Proceedings of the Eleventh Conference of the European Chapter of the Association for Computational Linguistics: Posters & Demonstrations. – Association for Computational Linguistics, 2006. – С. 95-98.

19. Rizzolo N., Roth D. Learning Based Java for Rapid Development of NLP Systems //LREC. – 2010. – Т. 5. – С. 313-323.

20. Zajac R., Casper M., Sharples N. An open distributed architecture for reuse and integration of heterogeneous NLP components //Proceedings of the fifth conference on Applied natural language processing. – Association for Computational Linguistics, 1997. – С. 245-252.

21. Митина Ольга Валентиновна, Евдокименко Александр Сергеевич Методы анализа текста: методологические основания и программная реализация // Вестник ЮУрГУ. Серия: Психология. 2010. №40 (216). URL: https://cyberleninka.ru/article/n/metody-analiza-teksta-metodologicheskie-osnovaniya-i-programmnaya-realizatsiya (дата обращения: 18.01.2019).

22. Alexa, M. Text Analysis Software: Commonalities, Differences and Limitations: The Results of a Review / M. Alexa, C. Zuell // Springer Netherlands, - 2000. - 34 (3) - P. 299-321.

23. Klein, H. Classification of Text Analysis Software / H. Klein // Classification and knowledge organization: Proceedings of the 20th annual conference of the Gesellschaft fur Klassifikation / In R. Klar, O. Opitz (Eds.). - University of Freiburg, 1996, Berlin, New York: Springer, 1997. - P. 255-261.

24. Леонтьев, А.А. Основы психолингвистики / А.А. Леонтьев. - М.: Смысл, 1997. -287 с.

25. Слово в действии. Интент-анализ политического дискурса / под ред. Т.Н. Ушаковой, Н.Д. Павловой. - СПб.: Алетейя, 2000. -316 с.

26. Brown, G. Discourse analysis / G. Brown, G. Yule. - Cambridge: Cambridge University Press, 1983. - 288p.

27. Labov, W. Sociolinguistic patterns / W. Labov. — Pennsylvania: University of Pennsylvania Press, 1972. - 346p.

28. Психолингвистическая экспертиза ксенофобии в средствах массовой информации: методические рекомендации для работников правоохранительных органов. — М.: Смысл, 2003. - 85 с.

29. Батов, В.И. Опыт построения методики для установления авторства текстов / В.И. Батов, Ю.А. Сорокин // Известия АН СССР, Сер. литературы и языка. - 1977. -36(4) - С. 345-347.