СОДЕРЖАНИЕ

Введение…….……………………………………………………………………..4

1 Исследование методов и проблем интеграции способов анализа нарративных источников.…..……………………………………………….....7

1.1 Основные понятия и определения обработки текстовых данных…....…7

1.2 Проблемы интеграции различных способов анализа текстов...............…9

1.3 Базовые принципы анализа нарративных источников...…………….....11

2 Формирование методов анализа текстов в процессе создания системы на основе концепции облачных вычислений………..………………….............13

2.1 Методы и принципы анализа текстовых данных...…….…….…........…13

2.2 Обработка текстов нарративных источников..………………...…..……16

2.3 Анализ проектной деятельности в рамках исторической информатики...………...………………………………………………..…19

3 Разработка архитектурных решений и программная реализация системы анализа нарративных источников..……..…….……………...….….…….….22

3.1 Функциональная модель системы…………………………………….….22

3.2 Модели представления системы и их описание………………….…......25

3.3 Информационная модель системы и ее описание………………..……..27

3.4 Обоснование выбора технических приёмов программирования……....30

3.5 Результаты тестирования разработанной системы и оценка

выполнения задач…..…………………...………………………...………32

4 Технико-экономическое обоснование эффективности разработки и использования системы анализа нарративных источников..….….…….….33

4.1 Характеристика программного продукта ...………………………….….33

4.2 Расчет затрат на разработку программного продукта....…...….……….34

4.3 Расчет экономического эффекта от разработки и реализации программного продукта……………………………………..……..……..39

5 Обеспечение комфортных условий труда сотрудников отдела разработок предприятия «САМСОЛЮШНС»..….……………………..…………….….41

Заключение…………………………………………………………………….....48

Список использованных источников ……………………………………...….50

Приложение А Описание программных модулей…..........................................51

Приложение Б Описание алгоритмов программных модулей..........................61

Приложение В Листинг программы....................................................................65

**ВВЕДЕНИЕ**

С возникновением и развитием информационного общества главным продуктом производства становятся информация и знания, что непосредственно влияет на возрастание роли самой информации, увеличение числа людей, занятых информационными технологиями, а вопрос о новых подходах в обработке данных стоит как никогда остро.

По результатам исследований, мировой объем информации увеличивается более чем в два раза каждые два года [1]. Цифровая Вселенная и Большие объемы данных стали одними из движущих сил фундаментальных изменений в социальной жизни, технологиях, науке и экономике. Между тем, растут и инвестиции корпоративного сектора в эту сферу. Большие объемы данных открывают новые возможности и изменяют роль IT, но, наряду с этим возникает проблема в анализе и восприятии всей этой информации человеком с целью получить требуемые, нужные данные. Именно на этом этапе и возрастает значение автоматизации обработки данных, преимущественно текстовой информации.

Сама идея не является новой. Она зародилось в тот самый момент, когда был создан первый компьютер, и родилась из самой квинтэссенции кибернетики. С развитием последней, развивалась и идея. Начиналось все с самых простых элементов, прототипов текстовых редакторов, когда к ЭВМ подключили терминалы для ввода-вывода информации. На текущий момент в области компьютерной обработки текстов сформировалось не одно направление, однако наиболее значимым является так называемый «text analysis» (грубым синонимом является «text mining», который хоть и обладает схожим смыслом, но в современном контексте не идентичен [2]) или анализ текста.

Под анализом текста в настоящее время понимают процесс получения высококачественной информации из текста на естественном языке. Под естественным языком тут понимается язык, который используется для общения людей, а не выведенный искусственно (например, любой язык программирования или эсперанто). Полученная высококачественная информация может быть интерпретирована в различном стиле и зависит от конкретных нужд и реализаций самого анализа. Это может быть и простой список ключевых слов, и автореферат, содержащий основные части источника, и даже текст, являющийся результатом качественной оценки и интеллектуальных выводов самой системы или машины (что является на данный момент наиболее перспективным, но трудно реализуемым направлением).

Цель дипломной работы состоит в объединение трех различных подходов в анализе нарративных источников (контент-анализ, дискурс-анализ и психоанализ) и их реализации на основе облачных вычислений.

Исходя из этого, выстраивается ряд задач, а именно:

* проанализировать классификацию текстовой информации;
* дать характеристику основных процессов обработки текстовых данных;
* ознакомиться с основными подходами в анализе нарративных источников;
* определить исходные и результирующие данные для различных подходов анализа;
* рассмотреть случаи дублирования и взаимного использования промежуточных данных для каждого из подходов;
* выявить проблемы в исследуемых методах анализа текстов нарративных источников и разработать предложения по совершенствованию их работы.

Объектом исследования являются текстовые данные нарративных источников.

Предметом исследования являются модели и методы обработки текстовых данных нарративных источников в рамках исторической информатики.

Структурно дипломная работа состоит из введения, трех глав и заключения.

В первой главе даны основные понятия, входящие в область компьютерной обработки текстов; рассмотрены различные подходы и методы анализа нарративных источников, а так же проблемы их интеграции.

Во второй главе проанализированы три основных метода анализа в рамках исторической информатики.

Третья глава работы заключается в разработке архитектурных решений и обзор их программной реализации в системе анализа нарративных источников.

В приложении приведены документы, отражающие функциональную модель и спецификацию разрабатываемой системы.

При написании дипломной работы были изучены труды авторитетных специалистов в области анализа и обработки текстовых данных. Отдельного упоминания заслуживает, пожалуй, работа Джастина Миртенау и Тима Финина, опубликованная университетом Мэриленда, под названием «Delta TFIDF: An Improved Feature Space for Sentiment Analysis». Остальные специалисты в этой области, такие как: Джойдип Гош, Хироши Митода, Михаэль Стэйнбах, Дэн Стейнберг и другие, которые были включены в тезисы конференции «IEEE International Conference on Data Mining» 2006 года в публикации «Top 10 algorithms in data mining». Помимо всего прочего, теоретической базой для данной работы служит публикация «Научный дискурс историка: социальная обусловленность и методология исследования».

1. **ИССЛЕДОВАНИЕ МЕТОДОВ И ПРОБЛЕМ ИНТЕГРАЦИИ СПОСОБОВ АНАЛИЗА НАРРАТИВНЫХ ИСТОЧНИКОВ**
   1. **Основные понятия и определения обработки текстовых данных**

Непосредственно понятие обработки текстовых данных является очень широким. Под ним можно понимать, например, форматирование или представление текста, чем сегодня занимаются всевозможные текстовые процессоры, такие как Лексикон или Microsoft Word. Однако эти области не представляют интереса в рамках нашей работы, по этой причине мы ограничимся рассмотрением двух терминов: text mining (или же text data mining) и text analysis, которые имеют слабые аналоги в русском языке, поэтому здесь и далее по тексту будет использовано лишь англоязычное написание. Необходимо так же отметить, что в рамках нашей работы рассматривались лишь тексты, написанные на естественном языке, то есть языке, используемом людьми в общении и не созданном искусственно. Такие языки не сконструированы целенаправленно, что в значительной степени осложняет их обработку.

Далее возникает вопрос: чем отличается text mining от text analysis? Вероятно, наиболее простым ответом будет: ничем. Тем не менее, это будет не совсем правдой и это можно будет увидеть дальше.

Text mining, или, как упоминалось выше, text data mining, представляет собой процесс получения высококачественной информации из текста. Такая высококачественная информация получается в большинстве случаев при помощи паттернов статического (машинного) обучения. Не заостряя на этом внимания, скажем лишь, что примером подобного распознавания может служить классификация, когда каждому входному значению задается свой класс с определенными атрибутами: элемент текста может быть отнесен к словам или знакам. Как правило, text mining включает в себя процессы структурирования текста (обычно парсинг с некоторыми лингвистическими методами), дальнейшее применение к нему шаблонов, и позже вычисление, агрегирование и интерпретация выходных данных. Общая схема данного подхода проиллюстрирована ниже. Пунктирной линией обозначены процессы исследования и анализа, которые имеют неограниченную свободу действий и поэтому опущены на данном рисунке:



Рисунок 1.1.1 – Общая схема процесса Text mining

В свою очередь text analysis включает в себя получение информации, лексический анализ с целью определения частотного распределения слов, аннотации, извлечение информации, визуализацию и прочее. Конечной целью является превратить текст в данные для анализа при помощи программ обработки естественных языков [2]. Как видно, данный процесс является расширением, а не простым синонимом для термина text mining. Если же прибегнуть к более простому определению, то можно говорить о том, что text analysis описывает набор лингвистических, статистических, а так же техник машинного обучения, которые создают модель и структурируют информацию текстовых источников для бизнес-аналитики и исследований.

Впервые, эти трудоемкие подходы впервые возникли в середине 1980-х годов, но развитие технологий открыло для них значительные возможности лишь в прошлом десятилетии. Столь раннее появление было обусловлено необходимостью работы с большим объемом текста в «неструктурированных» документах. Повышенный интерес еще более ускорил рост этой области. Так, например, сейчас не является редкостью межъязыковой data mining, когда источники имеют различную лингвистическую базу [3].

В наши дни, компьютерный анализ текста является популярным направлением, и, стоит отметить, что ни одна крупная корпорация не обошла его стороной, среди них Apple, Microsoft, Google, и лидер в этом направлении IBM. Эти компании интенсивно используют данный подход в различных своих продуктах, в том числе и специализированных. Кроме того, программ, нацеленных на анализ текста, существует большое множество (WordStat, TextAnalyst, STATISTICA Text Miner и другие), но, как правило, такие продукты являются узкоспециализированными по своему применению и нацелены на решение конкретных задач, в то время как комплексные программы не имеют профессиональной реализации.

Как говорилось ранее, целью данного проекта является объединение трех различных подходов в анализе нарративных источников (контент-анализ, дискурс-анализ и психоанализ) и их реализация на основе облачного сервиса. Однако на этом этапе возникает проблема интеграции подходов, различных по своей сути, и соответственно, обладающих различными требованиями к реализации и анализируемым данным. На этом основании первостепенной задачей является именно неконфликтное объединение всех трех подходов в одной системе с возможностью им параллельного применения в анализе текстов, что и будет рассмотрено далее.

* 1. **Проблемы интеграции различных способов анализа текстов**

Основной трудностью при интеграции и совместном использовании различных по своей природе подходов является повторное использование данных. Для того чтобы понять, о каких же данных идет речь, приведем стандартную классификацию по этапам использования:

* входные данные: преимущественно сам текст или же определенные словари;
* промежуточные данные: результат структурирования, парсинга и применения шаблонов; обычно базовые единицы текста, такие как слова, приведенные к начальной форме, с их атрибутами;
* результирующие данные: список слов, словарь или же визуальное представление.

Первая категория нам мало интересна, но нам важно ее содержание для каждого отдельно взятого подхода. Таким образом, для рассматриваемых методов анализа необходимо выявить такие промежуточные и результирующие данные, которые могли бы быть использованы в качестве входных и промежуточных для других методов. Причем, эти данные должны быть качественно новыми, позволяющими расширить возможности анализа, а не просто оптимизировать очередной процесс.

В основе решения данной проблемы в рамках нашей работы лежит модель сочетаемости нетрадиционных методов исторического познания текста нарративного источника. Она была разработана, сформулирована и описана научными сотрудниками БГУ Сидорцовым В.Н. и Приборовичем А.А. Более детально с ней можно ознакомиться в их работе «Научный дискурс историка: социальная обусловленность и методология исследования» [4]. Ее общую структуру можно увидеть на рисунке ниже:



Рисунок 1.2.1 – Модель сочетаемости нетрадиционных методов исторического познания текста нарративного источника

Подробнее рассмотрим приведенную иллюстрацию. На ней изображен круг, который разделен на 2 части прямой нарратива – сути, определяющей текст. Деление является условным и чисто визуальным, разграничивающие лишь приведенные методы (верхняя часть) и отдельные их элементы (нижняя часть). В рамках как модели, так и данной работы, рассматриваются три основных метода анализа текстов нарративных источников: контент-анализ, дискурс-анализ, психолингвистический анализ. Все три метода будут подробно рассмотрены в следующих главах, по этой причине не будем их пока подробно описывать. Каждый их методов представлен стрелкой, вдоль которой расположены основные его ключевые элементы. Так же от каждой из стрелок отходит круг, представляющий из себя результирующие или промежуточные данные. Как видно, некоторые них имеют ссылки на другие оси, что и представляет собой один из видов интеграции в данной модели: ключевые слова должны быть использованы в качестве входных данных при дискурс-анализе.

В нижней части модели стоит обратить внимание на рефлексию, обозначенную красным квадратом, и ментальные операции, показанные как синий круг. Их присутствие в верхней части указывает на принадлежность к психолингвистическому анализу, а пунктирные стрелки – на их использования в других методах. Прочие элементы не представляют для нас особого интереса и поэтому не будут рассмотрены.

Далее рассмотрим базовые принципы, которыми мы будем руководствоваться при разработке нашей системы.

* 1. **Базовые принципы анализа нарративных источников**

Разрабатываемая система являет собой облачный сервис, который предоставляет ряд инструментов по анализу текстов и работе с результатами этого анализа, в том числе позволяет производить их миграцию между этапами, хранение с последующим восстановлением и множество других действий на основе пользовательских проектов. Такая система нацелена в первую очередь не на повседневное использование и широкие пользовательские группы, а на экспертов, а так же тех, кто, так или иначе, использует анализ текстов в своей работе (историки, лингвисты). Подобная система позволяет не только ускорить процесс анализа и сократить объем выполняемой работы, но и производить обучение в ходе работы с ней, предоставляя не только результаты, но и данные с промежуточных этапов.

Теперь рассмотрим принципы построения системы более подробно. Под облачным сервисом можно понимать систему, реализующую принцип SaaS (Software as a Service), в основе которой лежат облачные вычисления. Сам продукт можно условно разделить на сервер и тонкого клиента. Сервер отвечает лишь за бизнес-логику и производит предварительную обработку (структуризацию, парсинг и прочее) текста и извлечение из него необходимой информации, а так же за хранение этой самой информации. Хранение этой самой информации и словарей происходит в базе данных, к которой сервер имеет непосредственный доступ.

Тонкий клиент выполняется в браузере и средствами HTML5 и JavaScript позволяет достичь большой степени автономности, убрав необходимость постоянного обмена информацией, используя лишь синхронизацию. Формат JSON обеспечивает обратную совместимость и предоставляет возможность использовать данные полученные сервером на клиенте, а так же передавать их по сети. Кроме того, HTML5 LocalStorage позволяет хранить данные на жестком диске, что обеспечивает легкую синхронизацию между вкладками или же восстановление последней сессии без необходимости обращения к серверу. За визуализацию и дальнейшее изменение полученных результатов, так же отвечает клиент. Внесенные изменения регистрируются локально и отправляются на сервер лишь во время синхронизации. Кроме того, код клиента базируется на трех основных фреймворках: jQuery, Underscore.js и Backbone.js. Это позволяет так же реализовать паттерн MVC на стороне клиента.

С целью дальнейшего понимания работы системы, рассмотрим методы и принципы анализа текстовых данных.

1. **ФОРМИРОВАНИЕ МЕТОДОВ АНАЛИЗА ТЕКСТОВ В ПРОЦЕССЕ СОЗДАНИЯ СИСТЕМЫ НА ОСНОВЕ КОНЦЕПЦИИ ОБЛАЧНЫХ ВЫЧИСЛЕНИЙ**
   1. **Методы и принципы анализа текстовых данных**

Эффективность же применения новых методов исторического познания, в том числе таких вышеупомянутых, зависит от наличия компьютерных решений, основанных на возможности реализации исследовательских методик как на уровне обработки первичной информации (нормализация текста, морфологический анализ), так и на уровне аналитического обзора результатов статистически-поискового изучения текста. Отметим, что к общим принципам работы компьютерных систем в данном случае является реализация компонентов компьютерного (автоматического) анализа текста.

Обратимся к затронутым в теме проблематики интеграции трем методам анализа текстов и рассмотрим их для начала в общей концепции анализа. Особенности в рамках исторической информатики будут затронуты в следующих пунктах.

Первым и, пожалуй, основным методом является контент-анализ. Исходными данными для этого подхода служат непосредственно тексты в их необработанном, неструктурированном виде. Эти тексты объединяются и анализируются как единый документ, который проходит этап первоначальной обработки (алгоритм будет рассмотрен в следующем пункте). В процессе такой «фильтрации» участвует значительное количество всевозможных словарей и алгоритмов. В результате работы формируется список так называемых значимых слов, которые имеют полезную информацию и не дублируются. Для значимых слов вычисляются различные статистические атрибуты, например частота и вес в тексте, а так же формируется конкорданс (ограниченный набор значимых слов, располагающихся слева и справа от элемента). Из значимых слов на основании вычисленных статистических величин определяют ключевые, которые группируются по смысловым блокам. Для ключевых слов вычисляется один из важнейших показателей – z-оценка. z-оценка схожа с корреляцией и определяет силу связи между двумя категориями (ключевыми словами). Вычисляется она следующим образом:

 (2.1.1)

где *K* – число пар А…В или В…А в радиусе *S* элементов;

*E* – ожидаемое число вхождений;

*q* – величина, равная *(1 – p).*

Категория А содержит соседствующие слова, категория В содержит рассматриваемые слова. Величина *p* вычисляется по формуле:

 (2.1.2)

где *Z* – общее число слов в тексте (значимых или ключевых);

*Fn* – число слов категории А в тексте;

*Fс* – число слов категории В в тексте.

Величина Е вычисляется по формуле:

 (2.1.3)

где *S* – допустимое число слов между А и В.

Рассмотренная величина играет ключевую роль в контент-анализе. Она не используется в дальнейших расчетах, тем не менее, является важным критерием при составлении категорий (смысловых блоков), а так же при анализе полученной информации уже человеком.

Как можно увидеть из названия, контент или от английского «content» переводится как содержание или нарратив, поэтому сам текст рассматривается с точки зрения содержания. Все данное описание можно увидеть и на схеме, представленной в уже упомянутой работе «Научный дискурс историка: социальная обусловленность и методология исследования»:

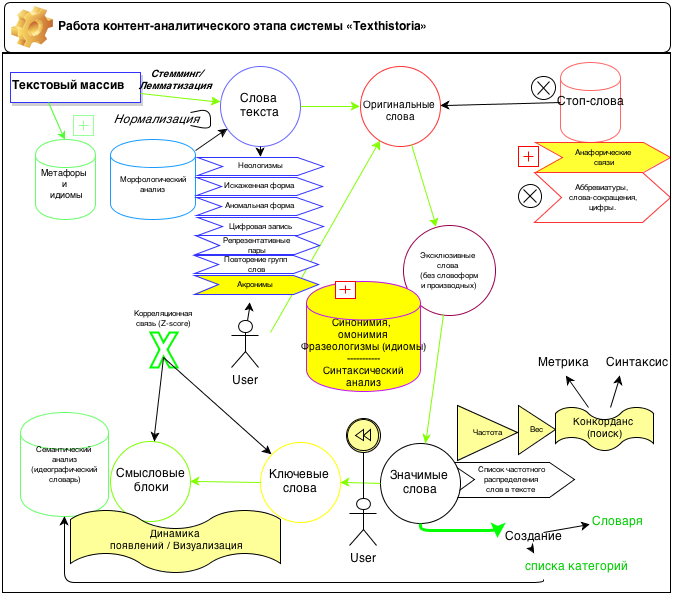


Рисунок 2.1.1 – Работа контент-аналитического этапа системы «Texthistory»

Другие два методы анализа, а именно дискурс-анализ и психолингвистический анализ нельзя в общих чертах представить на схеме.

Дискурс-анализ в нашем случае помимо всего прочего будет использовать данные, раннее полученные на этапе контент-анализа: это значимые слова и ключевые слова. Используя алгоритмы критического дискурса, мы можем выделить значимые категории и дать им анализ в соответствии со словарем. Словарь, как правило, содержит несколько трактовок, которые зависят от особенностей конкорданса, из которых выбирается наиболее подходящая.

Психолингвистический этап проводит эмоциональную оценку текста, а так же выделенных личностей дискурса. Этот метод представляет использование теории Фрейда на практике. Данный этап требует непосредственного выполнения двух предыдущих. Психолингвистический анализ состоит из двух различных направлений: анализ личности и анализ текста, причем оба направления мало связаны между собой. Для текста определяется эмоциональная окраска, его общая направленность, а так же рассматриваются отдельные предложения и слова. Личность анализируется с позиции стимулов. На основании этих стимулов дается характеристика ее действиям и словам, если таковые есть.

По видимым причинам должна быть четкая последовательность выполнения (методы могут применяться не полностью, а лишь для получения требуемых промежуточных данных): контент-анализ предшествует дискурс-анализу, после которого только следует психолингвистический анализ.

Стоит так же отметить, что в соответствии с уже сложившейся методологией компьютерного анализа текста к его основным элементам относится следующее:

* классификация (classification, categorization);
* кластеризация (clustering);
* извлечение фактов, понятий (feature extraction);
* реферирование и аннотирование (summarization);
* ответ на запросы (question answering);
* тематическое индексирование (thematic indexing);
* поиск по ключевым словам (keyword searching) .

Все перечисленные элементы, так или иначе, присутствуют на каждом из трех этапов. Далее перейдем к рассмотрению непосредственно анализа текстов, вместе с которым рассмотрим ряд этих элементов более подробно.

* 1. **Обработка текстов нарративных источников**

Как мы говорили ранее, под обработкой будем понимать именно процесс анализа текстов или text analysis, описание которого было дано выше.

Этот процесс в общих чертах можно задать обобщенным алгоритмом, который, как правило, содержит следующие подзадачи:

* получение информации или определение корпуса: сбор и идентификация текстовых данных из различных источников, таких как вэб, база данных, файлы для последующего анализа;
* применение статистических методов, шаблонов или методов обработки естественных языков. Для нашего случая можно говорить о нормализации текста и его парсинге, что относится ко второй группе;
* распознавание именованных сущностей: под этим процессом понимается использование статистических методов с целью выделения в тексте аббревиатур, имен, названий, так как такие сущности играют отдельную роль в общем процессе анализа;
* распознавание шаблонных сущностей: такие сущности не являются словами, имеют свой паттерн и могут быть заданы обычным регулярным выражением, например телефонные номера, адреса электронной почты, единицы системы измерений и прочее; в нашем случае они исключаются из анализа, за исключением дат;
* выделение кореференции: это позволяет избежать лишних дубликатов, так например в одном приложении может говориться о человеке с указанием его имени, а в другом – при помощи местоимения «он»;
* вычисление связей: применение статистических методов для вычисления различных показателей и определения связи между различными сущностями; наиболее популярными показателями является корреляция и z-оценка, которая была приведена в прошлой главе;
* сентимент анализ: влючает в себя несколько задач, таких как определения окраски текста или предложения (положительная/отрицательная), мнений, эмоций;
* квантитативный анализ: применение численный методов и исследование текста в формализованном виде с целью статистического измерения содержания.

Необходимо отметить, что не каждый из упомянутых трех методов анализа нарративных источников включает в себя описанные задачи или использует порядок следования. Так, сентимент анализ используется лишь при психолингвистическом анализе. Квантитативный анализ применяется лишь в начале, после парсинга, и не привязан к какому-либо методу. По этой причине мы не стали использовать нумерацию при перечислении.

При решении каждой из задач, как правило, используют определенные, уже устоявшиеся, технологии. Детально рассмотрим каждую из данных технологий на понятном примере.

Работа программных систем в группе компрессия текста ориентирована на решение проблемы – анализа возрастающего количество текстовой информации с ее дальнейшей компрессией (смыслового сжатия текста в виде аннотации, реферата). В настоящее время из обзора научных конференций по проблемам автоматического аннотирования DUC (Document Understanding Conference) и текстового реферирования TSC (Text Summarization Challenge) видно, что за рубежом придается большое значение исследованиям в области обзорного реферирования. Разработано достаточно много методов автореферирования. Их можно разделить на методы квазиреферирования (статистические, позиционные, индикаторные) и методы генерирования рефератов. Первые основаны на экстрагировании, т. е. выделении из текста наиболее информативных фрагментов, передающих основной смысл текста; вторые – на выделении наиболее существенной информации из текста документов.

В основе программных систем группы кластеризации заложен математический метод кластерного анализа, основная задача которого состоит в стилистической группировке текстового массива информации. Такая группировка может быть как одноуровневой, так и иерархической, когда кластеры расположены по схожести содержания. Принадлежность текста к некоторому кластеру на определенном уровне иерархической классификации может быть однозначной (каждый рассматриваемый текст принадлежит только одному кластеру, т. е. одному создателю), или неоднозначной (каждый рассматриваемый текст может принадлежать нескольким создателям).

Тексты, выбранные для кластеризации, представляются в виде векторов значений соответствующих признаков (категорий, ключевых слов). В качестве значений признаков, по основному варианту, чаще всего выступают частоты появления определенных слов в текстах. Альтернативой частотам слов могут быть индикаторы появления слов (0 и 1) или значения, учитывающие среднюю встречаемость слова во всех текстах. В качестве меры близости текстов, представленных векторами значений признаков, обычно используется скалярное (числовое) произведение векторов. Другой вариант состоит в том, что в качестве значений признаков, включённых в векторы, выступают частоты появления опрёделенных буквосочетаний в словах текстов.

Основная задача программных систем компьютерного индексирования состоит в распознавании словоформ в тексте, а также выделение наиболее значимых слов (ключевых слов, дескрипторов), составления словаря. В соответствии с выявленным смысловым полем (наборах концептов) система дает возможность определить тематическую направленность текста.

Технология извлечение фактов (feature extraction) основана на использовании специальных семантико-лингвистических методов, которые позволяют выявить факты в рассматриваемом текстовом массиве. Под фактом системы feature extraction понимаются структурированное представление фрагментов текста документа в смысле значения факта: его суть, время и место совершения, участники. Современные системы позволяют также учесть прямое, косвенное, а также транзитивное (через общий третий объект связи) определения значения фактов.

Системы feature extraction содержат следующие возможности:

* автоматическое определение прямых и косвенных связей понятий в анализируемом тексте;
* формирование групп текстов (концептов), связанных между собой некой общностью;
* построение многомерных частотных распределений фактов и др.

Имея общее представление о работе системы и применяемых ею технологий, мы плавно переходим к описанию ее функций и их реализации на практике.

* 1. **Анализ проектной деятельности в рамках исторической информатики**

При разработке продукта предполагается реализация основных функций системы:

* предоставление инструментов анализа в качестве облачного сервиса;
* независимое управление проектами для пользователей;
* единая точка входа для глобальных открытых и глобальных приватных документов;
* идентичное представление данных вне зависимости от порядка анализа;
* сохранение результатов с их последующей загрузкой и обновлением.

Теперь к интересующим нас трем методам, которые были подробно описаны в предыдущих пунктах, соотнесем компоненты компьютерного анализа текста, учитывая особенность применяемой в практике реализации методов терминологии, в рамках исторической информатики.

Для выполнения методики контент-анализа компьютерные системы должны содержать следующие компоненты обработки текста: нормализация, стемматизация, словарная лемматизация, поиск, реферирование текста, категоризация, визуализация (в разделе «Компьютерные инструменты проведения качественно-количественного анализа текстов исторических документов на основе метода контент-анализа» нами было уже дано описание работы каждого компонента).

Для компьютерного исполнения дискурс-анализа достаточно использование результатов работы таких компонентов как: контент-анализ (выделение ключевых категорий, доминанты текста, а также сравнение их в контексте использования); поиск дискурсивных слов, метафор, коллокаций и прочих семантических, лексико-граматических средств построения дискурса.

Описание компонентов компьютерного исполнения приемов психолингвистического анализа текста начнем с перечисления групп программных продуктов, составленных по реализуемым целям исполнения методов психолингвистики. К ним отнесем среды, участвующие в:

* + получении дополнительной информации об участниках и ситуации

речевого взаимодействия, в частности выявление имплицитных (скрытых) содержаний речевого взаимодействия;

* + определении прогноза относительно особенностей восприятия того или иного вербального материала адресатом, а так же психологического состояния адресанта;
  + реализации психолингвистических методик, практикуемых в дистантной форме психотерапевтических исследований;
  + определении уровней структуры языковой личности (вербально-семантический, лингво-когнитивный, мотивационный).

Как можно видеть, охват данных трех подходов позволяет полностью покрыть область интересов исторической информатики. Ранее не единожды говорилось, что систем, реализующих какой либо из подходов, существует достаточно большое количество, однако, основная ценность заключается именно в возможности совместного их использования. Выше были описаны проблемы, с которыми можно столкнуться при реализации такого гибридного подхода, и были приведены способы их решения.

Рассмотренная гибридная модель позволяет не только совместно использовать подходы, но и использовать выходные данные одного из типов анализа в качестве входа на другом. Это можно представить в виде непрерывного потока, состоящий из нескольких этапов:

* на первом этапе, текстовый массив поступает на обработку;
* на втором этапе, происходит процесс контент-анализа, где будут выделены только интересующие нас сущности и вычислены различные статистические показатели для них и для данных в целом;
* на третьем этапе участвуют уже отдельные сущности, выделенные на прошлом шаге. При помощи словарей, а так же полученной ранее статистики, выделяют другие сущности в рамках дискурс-анализа;
* на четвертом этапе используются данные, как со второго, так и с третьего этапов. При проведении психоанализа так же используются определенные правила и словари.

Такая гибридная модель накладывает строгие рамки на реализацию системы. Разрабатываемый продукт должен обладать определенными чертами. Задачи, которые решает система анализа нарративных источников в большей части относятся к самим подходам, однако так же затрагивают область представления и хранения результатов:

1. предоставление глобальной точки доступа к документам;
2. гибкая настройка анализа через проекты;
3. независимость выполнения аналитических подходов от их порядка следования;
4. вариативность представления данных;
5. реализация принципа «обучения через использования» (одновременное обучение системы и пользователя при работе с системой);
6. единое хранение результатов анализа с привязкой к проектам.

Данные ограничения так же влияют и на задачи, поставленные на проектирование системы:

1. объединить различные подходы анализа с возможностью их неконфликтного одновременного использования;
2. обеспечить возможность получения оперативного доступа к данным анализа без необходимости его повторного проведения;
3. свести к минимуму участие пользователя при анализе текста.

Далее необходимо рассмотреть модели представления системы, чтобы иметь четкие представления о принципах ее работы и внутренней структуре.

1. **РАЗРАБОТКА АРХИТЕКТУРНЫХ РЕШЕНИЙ И ПРОГРАММНАЯ РЕАЛИЗАЦИЯ СИСТЕМЫ АНАЛИЗА НАРРАТИВНЫХ**
   1. **Функциональная модель системы**

Рассмотрим подробно функциональную модель системы анализа нарративных источников. Для этого воспользуемся диаграммой, построенной по стандарту IDEF0.

Функциональная модель на основе стандарта IDEF0 позволяет учесть все условия поставленной задачи и автоматизировать её. Исходя из принципов стандарта IDEF0, необходимо определить главную функцию проекта, затем осуществить её декомпозицию.

В нашем случае, основной функцией является непосредственно анализ текста. Ее мы и положим за основу контекстного уровня, что можно увидеть на диаграмме, приведенную на рисунке 3.1.1.

Как видно, на входе мы имеем массив текстовых данных. Под таким массивом подразумевается любой набор документов (из поддерживаемых системой форматов) в любом представлении.

На выходе системы получаются статистические характеристики текста, представленный в различном формате, автореферат, как результат смысловой компрессии текста, а так же конкорданс, представляющий собой процесс динамической работы с элементами текста в контексте документов.

В элементы управления входят методология анализа текста, определяющая сам процесс анализа и структура проекта, которая влияет на процесс косвенно и в большей степени задействована на этапе подготовки и обработки результатов.

В качестве механизмов управления выступают сам пользователь, минимальное участие которого требуется практически на каждом этапе, а так же сервер, представляющий собой как вычислительную машину, так и абстракцию самого сервиса.



Рисунок 3.1.1 – Диаграмма контекстного уровня: процесс анализа текста

При декомпозиции контекстного уровня мы получим четыре основных блока, а именно:

* подготовить проект и источник: формируется набор правил анализа и порядок анализа источников, подключенных к проекту в качестве текстовых массивов;
* выполнить анализ: различные этапы анализа, как контент-анализ, дискурс-анализ и психоанализ;
* сформировать результаты анализа: на основе данных анализа формируются результаты в графическом представлении, табличном и других, так же осуществляется конкорданс;
* сохранить состояние проекта: проект сохраняет свое состояние с определенными параметрами и результатами.

Декомпозиция контекстного уровня представлена ниже:



Рисунок 3.1.2 – Декомпозиция контекстного уровня

Так же можно представить детализированную диаграмму анализа текста, которая имеет вид:



Рисунок 3.1.3 – Декомпозиция выполнения анализа

А так же следующий уровень декомпозиции, в качестве детализированной диаграммы произведения контент-анализа:



Рисунок 3.1.4 – Декомпозиция проведения контент-анализа

После того, как сформирована функциональная модель, следующим этапом следует поставить задачи проекта и рассмотреть все методы ее решения с целью формирования наиболее рационального подхода при разработке программного продукта. Все эти моменты мы рассмотрим в следующей главе.

* 1. **Модели представления системы и их описание**

При описании моделей представления системы анализа нарративных источников мы руководствовались спецификацией языка моделирования UML версии 2.4.1, последним из утвержденных конечных спецификаций на данный момент. Однако все представленные диаграммы имеют обратную совместимость вплоть до версии 2.0.

UML — язык графического описания для объектного моделирования области разработки программного обеспечения. UML является языком широкого профиля, это открытый стандарт, использующий графические обозначения для создания абстрактной модели системы, называемой UML-моделью [5]. UML был создан для определения, визуализации, проектирования и документирования в основном программных систем. UML не является языком программирования, но в средствах выполнения UML-моделей как интерпретируемого кода возможна кодогенерация.При помощи UML можно разработать детальный план создаваемой системы.

Приведем краткое описание диаграммы вариантов использования системы анализа нарративных источников. Все прочие диаграммы можно просмотреть в приложение в разделах, а их описание будет рассмотрено в следующей главе, описывающей модели представления.

В разработанной системе можно выделить лишь одного актера, которым будет являться сам пользователей системы. Это обусловлено тем, что облачный сервис предоставляет инструменты и все сопутствующие процессы анализа полностью автоматизированы и выполняются системой. Поэтому сам пользователь одновременно является обычным участником процесса, экспертом и модератором. Сама диаграмма приведена ниже.



Рисунок 3.2.1 – Диаграмма вариантов использования.

Остальные диаграммы стандарта UML, в которых были отражены модели представления системы включают следующие:

1. диаграмма состояний (рисунок А.1);
2. диаграмма последовательностей (рисунок А.2);
3. диаграмма классов (рисунок А.3 – А.7);
4. диаграмма компонентов (рисунок А.8);
5. диаграмма развертывания (рисунок А.9).

Главное предназначение диаграммы состояний ― описать возможные последовательности состояний и переходов, которые в совокупности характеризуют поведение пользователя при работе. На рисунке А.1 представлен процесс создания проекта и выполнение анализа.

Диаграмма последовательности изображает те объекты, которые непосредственно участвуют во взаимодействии и не показываются возможные статические ассоциации с другими объектами. Ключевой момент – это динамика взаимодействия объектов во времени. Каждое сообщение на диаграмме соответствует вызову метода класса. На рисунке А.2 представлена упрощенная работа пользователя и системы при проведении анализа текста.

Диаграмма классов служит для представления статической структуры модели системы в терминологии классов объектно-ориентированного программирования.

Диаграмма компонентов описывает особенности физического представления системы и позволяет определить архитектуру разрабатываемой системы, установив зависимости между программными компонентами. Общее соотношение компонент можно увидеть на рисунке А.8.

Диаграмма развертывания предназначена для визуализации элементов и компонентов программы, существующих лишь на этапе ее исполнения (runtime). Представляются только компоненты-экземпляры программы, являющиеся исполнимыми файлами или динамическими библиотеками. Как видно из рисунка А.8, система состоит из сервера базы данный с MySQL Server и сервера приложений с Tomcat 7 и самим приложением в виде \*.war архива.

На следующем этапе мы опишем информационную модель нашей системы.

* 1. **Информационная модель системы и ее описание**

Как говорилось ранее, в основе анализа текстов лежит гибридный подход. Его смысл заключается и в том, что в процессе анализа используются не только правила и самообучение, но и словари, без которых невозможно реализовать необходимые этапы. Примером такого словаря может служить частотный словарь, который формируется на основе анализа тысяч источников с целью определения числа вхождения слов (лемм) на миллион лемм текста. Для упрощения учета информации необходима его частичная или полная автоматизация. Однако, рассмотрим информационную модель системы подробнее. Она разработана с использованием методологии стандарта IDEF1.x. На рисунках ниже представлены логическая (рисунок 3.3.1) и физическая (рисунок 3.3.2) модели:



Рисунок 3.3.1 – Логическая модель системы



Рисунок 3.3.2 – Физическая модель системы

Сущность meta хранит в себе мета-информацию о самой базе, так как является актуальным обновление источников информации. Сущности user, project и source отвечают за хранение информации и пользователях, проектах и источниках соответственно. Эти сущности связаны с пользовательскими аккаунтами и являются базовыми. Остальные элементы на рисунках связаны с определенными словарями. Так сущности abramov\_\* определяют словарь синонимов Абрамова, opcorpora\_\* определяют открытый словарь лемм русского языка. Словари независимы от пользователей, а по этой причине в базу могут добавляться или удаляться сущности, которые определяют словарь по мере необходимости и развития разрабатываемой системы.

Теперь проанализируем нашу модель на соответствие нормальным формам:

* Переменные отношения находятся в первой нормальной форме тогда и только тогда, когда в любом допустимом значении отношения каждый его кортеж содержит только одно значение для каждого из атрибутов. Это условия соблюдается по условию, так как в нашем случае используется реляционная база данных, а это исходит из самого понятия «отношение» [6].
* Переменные отношения находятся во второй нормальной форме тогда и только тогда, когда они находится в первой нормальной форме и каждый неключевой атрибут неприводимо (функционально полно) зависит от ее потенциального ключа. Функционально полная зависимость означает, что атрибут функционально зависит от всего составного ключа, но при этом не находится в функциональной зависимости от какой-либо из входящих в него атрибутов (частей). Как видно, все сущности не находятся в функциональной зависимости от атрибутов, входящих в составной ключ.
* Переменные отношения находятся в третьей нормальной форме тогда и только тогда, когда они находится во второй нормальной форме и отсутствуют транзитивные функциональные зависимости неключевых атрибутов от ключевых. Составленная модель так же соответствует третьей нормальной форме.

После того, как мы затронули работу системы, ее составные части и модели представления, перейдем к обоснованию оригинальных решений, так как каждый продукт должен обладать данными решениями в целях повышения конкурентоспособности и обеспечения уникальности.

* 1. **Обоснование выбора технических приёмов программирования**

Главным новаторским решением стало использование передовых технологий и подходов к проектированию систем.

Использование в качестве основного фреймворка Spring версии 3.1, обеспечило не только тесную интеграцию с Hibernate и Apache Tiles, но и позволило упростить разработку, а так же наделить продукт авангардными способами защиты информации, доступа в систему через OpenID а так же всесторонне использовать AJAX подход. Использование HTML5 на стороне клиента позволило избежать использования небезопасных элементов, и расширить взаимодействие с клиентом, а так же значительно снизить нагрузку сервиса, снизив количество запросов к минимуму, путем использования HTML5 LocalStorage и SessionStorage.

Однако основным оригинальным решением является не столько использование самих технологий, сколько подход, для которого они используются. Основной идеей было создать систему на основе облачного сервиса.

Данный подход далеко не новый, однако его использование в области анализа текста является уникальным. Существуют похожие сервисы, однако, они, как правило, реализуют лишь один из подходов и не обеспечивают хранение и управление данными, что является лишь инструментом, который предоставляется через сеть интернет. Общая схема взаимодействия на основе технологии облачных вычислений:



Рисунок 3.4.1 – Технология облачных вычислений.

Облачные вычисления – это технология, основанная на системе интернет, которая позволяет удаленным дата центрам управлять информационными сервисами и приложениями. Облачные сервисы позволяют удаленным пользователям управлять файлами, информацией и приложениями без надобности устанавливать дополнительное программное обеспечение. Для этого требуется лишь доступ в интернет и браузер [7].

Такие решения, как правило, состоят из трех слоев:

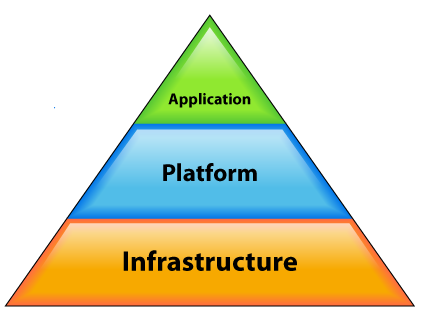


Рисунок 3.4.2 – Три уровня технологии облачных вычислений.

В первую очередь нас интересует верхний уровень, уровень приложения. Приложение этого уровня реализует SaaS бизнес модель, когда программное обеспечение предоставляется как услуга. Это дает сразу несколько положительных элементов:

* приложение приспособлено для удаленного использования;
* одним приложением пользуется несколько клиентов (приложение коммунально);
* техническая поддержка приложения включена в оплату;
* модернизация и обновление приложения происходит оперативно и прозрачно для клиентов.

После того, как описаны уникальные решения, давайте рассмотрим, как работает сама система, приведя в пример несколько алгоритмов.

* 1. **Результаты тестирования разработанной системы и оценка выполнения задач**

Для описания полученных результатов выделим главные критерии, на основании которых проведем тестирование программы:

1. оценка реализованной базы данных;
2. дизайн пользовательского интерфейса;
3. реализация основных функций для работы с данными;
4. реализация веб-сервиса.

Созданная база данных является реляционной, т.е. ее отличительной возможностью является связь данных нескольких таблиц, которая осуществляется с помощью ключевых полей. Преимущество данной базы состоит в ее модульности и возможности наращивания без существенных затрат памяти. Так же данный тип баз данных позволяет существенно ускорить процесс анализа текста при использовании словарей, поэтому можно сказать, что база данных была выбрана правильно.

Интерфейс сервиса достаточно функционален и эстетичен, а так же интуитивно понятен. Общие черты схожи с популярными сервисами, на подобии сервисов компании Google, что однозначно ускорит процесс обучения пользователя за счет ассоциации элементов и действий.

Веб-интерфейс реализован на стандарте HTML5 с использованием JavaScript и XmlHttpReruest, а так же на основе методологии AJAX, что позволило свести к минимуму перерисовку пользовательского интерфейса и применять его только для значительных изменений.

Так же были реализованы все основные функции для данного сервиса, которые были затронуты при построении диаграммы вариантов использования системы.

1. **ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ ЭФФЕКТИВНОСТИ РАЗРАБОТКИ И ИСПОЛЬЗОВАНИЯ СИСТЕМЫ АНАЛИЗА НАРРАТИВНЫХ ИСТОЧНИКОВ**
   1. **Характеристика программного продукта**

Программы компьютерного анализа в современном мире пользуются заслуженно большой популярностью, как среди частных лиц, например исследователей, лингвистов, так и среди компаний. Значимым является тот факт, что идеи таких программ были высказаны еще в середине прошлого века, что было обусловлено необходимостью в них. В настоящее время широкое распространение получили программы анализа текстовой информации и средства бизнес-аналитики. Разрабатываемый продукт относится к первой категории.

«Система обработки текстов нарративных источников на основе интеграции лингвистических и статистических методов анализа» представляет собой продукт, призванный решить следующие задачи:

* увеличить скорость обработки и уменьшить трудоемкость анализа текстовых данных;
* сократить расходы на оплату машинного времени и других ресурсов при обработке и анализе данных;
* автоматизировать обработку и хранение промежуточных данных;
* осуществлять визуализацию и форматирование конечных данных.

Разработка проектов таких программных средств связана со значительными затратами ресурсов (трудовых, материальных, финансовых, временных). В связи с этим, создание и реализация каждого проекта программного обеспечения нуждается в соответствующем технико-экономическом обосновании (ТЭО). Это позволит рационализировать разработку, снизить затраты и максимизировать прибыль.

* 1. **Расчет затрат на разработку программного продукта**

Основная заработная плата исполнителей на наш программный продукт рассчитывается по формуле:

|  |  |
| --- | --- |
|  | (4.2.1) |

где *n* − количество исполнителей, занятых разработкой наше программного продукта;

*Tчi* − часовая тарифная ставка i-го исполнителя (тыс. руб.);

*Фп* − плановый фонд рабочего времени i-го исполнителя (дн.);

*Tч* − количество часов работы в день (ч);

*К* − коэффициент премирования.

Коэффициент премирования 1,5. Для расчета заработной платы месячная тарифная ставка 1-го разряда на предприятии установлено на уровне шестьсот тысяч белорусских рублей.

Таблица 4.2.1 – Расчет заработной платы

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Категория исполнителя | Разряд | | Тарифный коэффициент | Часовая тарифная ставка, тыс. руб. | Трудоемкость, дн. | Основная заработная плата, тыс. руб. |
| Техник-программист | 10 | 2.48 | | 8,91 | 32 | 2281,006 |
| Итого с премией (50%), Зо | - | - | | - | - | 3421,509 |

Дополнительная заработная плата на наш программный продукт (*Зд*) включает выплаты, предусмотренные законодательством о труде (оплата отпусков, льготных часов, времени выполнения государственных обязанностей и других выплат, не связанных с основной деятельностью исполнителей), и определяется по нормативу в процентах к основной заработной плате:

|  |  |
| --- | --- |
| , | (4.2.2) |

где *Зд* − дополнительная заработная плата исполнителей (тыс. руб.);

*Нд* − норматив дополнительной заработной платы равный 10%.



Отчисления в фонд социальной защиты населения и обязтельное страхование (*Зсз*) определяются в соответствии с действующими законодательными актами по нормативу в процентном отношении к фонду основной и дополнительной зарплаты исполнителей, определенной по нормативу, установленному в целом по организации:

|  |  |
| --- | --- |
| , | (4.2.3) |

где *Нсз* − норматив отчислений в фонд социальной защиты населения и на обязательное страхование (34 + 0,6%).



Расходы по статье «Машинное время» (*Рм*) включают оплату машинного времени, необходимого для разработки и отладки программного продукта, которое определяется по нормативам (в машино-часах) на 100 строк исходного кода (*Hмв*) машинного времени, и определяются по формуле:

|  |  |
| --- | --- |
|  | (4.2.4) |

где *Цм* − цена одного машино-часа. Рыночная стоимость машино-часа компьютера со всеми необходимым оборудованием (5 тыс. руб. / ч);

Тпр – время работы над программным продуктом (256 ч).



Расходы по статье «Научные командировки» (*Рнк*) на программное средство определяются по формуле:

|  |  |
| --- | --- |
| , | (4.2.5) |

где *Нрнк* – норматив расходов на командировки в целом по организации (%). Норматив на командировки − 10 % от основной заработной платы.

,

Расходы по статье «Прочие затраты» (Рпр) на программное средство включают затраты на приобретение и подготовку специальной научно-технической информации и специальной литературы. И определяются по формуле:

|  |  |
| --- | --- |
| , | (4.2.6) |

где *Hп­з* − норматив прочих затрат в целом по организации равен 20%



Затраты по статье «Накладные расходы» (*Рн*), связанные с необходимостью содержания аппарата управления, вспомогательных хозяйств и опытных (экспериментальных) производств, а также с расходами на общехозяйственные нужды (*Рн*), и определяют по формуле:

|  |  |
| --- | --- |
| , | (4.2.7) |

где *Pн* – накладные расходы на программный продукт (тыс. руб.);

*Нрн* – норматив накладных расходов в целом по организации,100%.



Общая сумма расходов по смете (*Ср*) на программный продукт рассчитывается по формуле:

|  |  |
| --- | --- |
|  | (4.2.8) |



Кроме того, организация-разработчик осуществляет затраты на сопровождение и адаптацию программного продукта (Рса), которые определяются по формуле:

|  |  |
| --- | --- |
| , | (4.2.9) |

где Нрса – норматив расходов на сопровождение и адаптацию 10%.



Общая сумма расходов на разработку (с затратами на сопровождение и адаптацию) как полная себестоимость программно продукта (Сп) определяется по формуле:

|  |  |
| --- | --- |
| , | (4.2.10) |



* 1. **Расчет экономического эффекта от разработки и реализации программного продукта**

Рассматриваемое программное обеспечение имеет лишь целевых заказчиков. Так же, как говорилось ранее, на текущий момент на рынке существует достаточное количество подобных систем, преимущественно иностранных аналогов.

По этим причинам, будем применять следующий подход: определим необходимый для себя уровень прибыльности (рентабельности) ПО, а затем, исходя из него и на основании суммы затрат на разработку, самостоятельно определим цену реализации ПО и предложим ее заказчику (либо используем ее как «отправную точку» в переговорах с заказчиком). Очевидно, что если предложенная цена окажется ниже средней цены на схожие программы, то, при прочих равных условиях, заказчик предпочтет приобрести разрабатываемую нами систему.

Такой подход целесообразно применять, если на рынке существует множество специализированных организаций, способных предложить заказчику схожее программное обеспечение, или организация-разработчик только выходит на данный рынок и заинтересована в привлечении клиентов. В этом случае прибыль рассчитывается по формуле:

|  |  |
| --- | --- |
| , | (4.3.1) |

где *Ур* – запланированный норматив рентабельности, % (30%);

*Сп* – полная себестоимость программно продукта;

*N* – количество копий, которое планируется продать за год (5 ед.).



Так как компания-разработчик (Иностранное Частное Предприятие «САМСОЛЮШНС») является резидентом Парка высоких технологий, то она освобождена от уплаты НДС, налога на прибыль. Цена в этом случае будет рассчитываться по формуле:

|  |  |
| --- | --- |
| , | (4.3.2) |



Теперь сравним стоимость со схожими программными разработками. Разрабатываемых продукт идет первым в списке и назван, как «Texthistory».

Таблица 4.3.1 – Цена различных продуктов категории анализа текстов

|  |  |
| --- | --- |
| Название продукта | Цена одной копии, тыс. руб. |
| Texthistory | 2871,164 |
| STATISTICA | 6011,750 |
| SAS | 51900,0 |
| WordStat | 25906,75 |

Как видно из приведенной таблицы, наиболее подходящие аналоги имеют значительную стоимость, в чем уступают «Texthistory». Продукт «SAS» представляют собой модульную систему, причем анализ текста не является его основной задачей. Пакет программ «STATISTICA» обладает значительным функционалом в области непосредственно статистики, что так же ограничивает ее круг применения. Наиболее близким аналогом является программа «WordStat», однако, ее стоимость превышает разработанный в рамках данного проекта аналог почти в 9 раз.

Исходя из всего вышеперечисленного, можно сделать вывод, что заявленная цена не только ниже средней по рынку, но и является самой низкой. Это, несомненно, дает значительное конкурентное преимущество продукту, а значит реализация данного проекта имеет экономическую целесообразность.

2. **ОХРАНА ТРУДА**

**Обеспечение комфортных условий труда сотрудников отдела разработок иностранного частного предприятия «SaM Solutions»**

В данном дипломном проекте рассматривается разработка системы обработки текстов нарративных источников на основе интеграции лингвистических и статистических методов анализа, осуществляемая в компании «SaM Solutions». Важно ясно понимать, что подобные проекты обладают высокой трудоемкостью и сложностью поставленной задачи. В свою очередь, сложность самой задачи перекладывается на реализацию, за которую ответственны исполнители, а в нашем случае – это сотрудники отдела разработок (далее в тексте **–** программисты). Поэтому, при разработке рассматриваемой системы, именно качество и скорость работы программистов имеет наибольшее влияние на конечный продукт, а так же на большинство затрат, связанных с его производством.

Давно известно, что чем больше человек ощущает внутренней потребности в работе, тем лучше идёт эта самая его работа. Если кто-то прекрасно понимает, что и как надо делать, но не видит в этом смысла, то   сотрудник из такой личности выйдет не самый лучший. Даже первоклассный специалист вместо работы будет частенько заниматься имитацией бурной деятельности. Согласно исследованиям учёных, чем выше умственная и творческая составляющие в работе, тем больше для сотрудника значит комфорт своего рабочего места. Так, например, известно, что один и тот же сотрудник, выполняющий одинаковую работу в двух разных местах, но с одним и тем же коллективом, будет иметь разную производительность труда.

Поэтому, учитывая объем работ и сложность создания самой системы, а так же прямую зависимость результатов от работы программистов, основной задачей является создание и обеспечение комфортных условий труда сотрудников, с целью повышения их производительности. Для этого вначале необходимо рассмотреть особенности умственного труда, показатели его напряженности, определить влияние элементов рабочего места, воздушной среды и организации труда на работоспособность и здоровье сотрудника. Кроме того, необходимо так же выделить требования, предъявляемые к производственной среде, и организации труда с целью профилактики психофизиологических перегрузок сотрудника, а также способы и средства их обеспечения при разработке программного продукта.

Нельзя с определенностью говорить о том, что какое-либо из требований является наиболее важным. Однако в данном разделе нас интересует непосредственно вопрос оптимизации условий труда программистов при разработке системы анализа.

По содержанию труд можно разделить на две группы:

* физический труд, при котором преобладает мышечная деятельность, характеризуется тяжестью труда;
* умственный труд с преобладанием умственной и творческой деятельности, который характеризуется напряженностью работы.

По причине того, что программист в большей мере подвержен умственному труду, то рассмотрим его особенности, не затрагивая другую группу.

Умственный труд заключается в переработке и анализе большого объёма разнообразной информации, и как следствие этого – мобилизация памяти и внимания, а мышечные нагрузки, как правило, незначительны. Этот труд характеризуется значительным снижением двигательной активности (гипокинезией), что может приводить к сердечно-сосудистой патологии; длительная умственная нагрузка угнетает психику, ухудшает функции внимания, памяти. Основным показателем умственного труда является напряжённость, отражающая нагрузку на центральную нервную систему [9].

Напряженность труда – это состояние человека, определяемое количественными и качественными характеристиками структуры деятельности и интенсивностью физиологических процессов, обеспечивающими выполнение деятельности [10].

Специфическая напряженность труда включает два вида – интенсивную и темповую напряженность; неспецифическая – определяется процессом деятельности и подразделяется на информационную, энергетически-силовую, мотивационную, например, напряженность ответственности, опасности и напряженность труда, обусловленную нерациональным режимом труда и отдыха, условиями обитаемости на рабочем месте. В качестве критериев напряженности умственной работы используются величины физиологических и психофизиологических показателей, к которым относятся: частота слияния световых мельканий; максимальный темп двигательных реакций; время простой и сложной сенсомоторных реакций; концентрация, распределение внимания и объема памяти [10].

По значениям наблюдаемых физиологических и психофизиологических показателей различают четыре категории напряженности умственного труда:

* ненапряженная;
* малонапряженная;
* напряженная;
* очень напряженная.

Одним из важных факторов, которые влияют на работоспособность и состояние здоровья сотрудников является организация рабочего места. Неправильная организация рабочего места приводит к общей усталости, головным болям, усталости мышц рук, болям в спине и шее. Подобные негативные моменты чаще всего возникают из-за несоответствия помещений и организации рабочих мест эргономическим требованиям и санитарно-производственным нормам.

По этим причинам, рассмотрим более подробно, какие негативные факторы влияют на трудовой процесс программистов, а также причины их возникновения.

На сам трудовой процесс оказывают влияние следующие факторы: элементы рабочего места, рабочее помещение и мебель, освещённость, а также предметы, которыми непосредственно пользуются сотрудники во время работы. При неправильной их организации данные элементы вносят дополнительную отрицательную нагрузку при долговременной, кропотливой и монотонной обработке информации, что приводит к напряжению зрения, памяти, внимания; также в процессе долгого пребывания в сидячем положении.

Правильно организовать рабочие места – это значит не только оснастить средствами и предметами труда, но и в рациональном порядке разместить их таким образом, чтобы создать уютную обстановку для высокопроизводительной работы.

На рассматриваемом нами предприятии а так же на ему подобных находится огромное количество различной офисной техники: персональные компьютеры, принтеры, сканеры и другие виды, – все они играют немаловажную роль в появлении вредных факторов в воздушной среде, а именно: повышенное содержание положительных аэроионов в воздухе; пониженное содержание отрицательных аэроионов в воздухе; повышенная или пониженная температура и влажность воздуха; повышенные уровни запыленности воздуха вокруг рабочего места; повышенное напряжение в электрической цепи, замыкание которой может произойти через тело человека; повышенная или пониженная подвижность воздуха [12].

Таким образом, вышеперечисленные факторы оказывают дополнительную нагрузку на организм сотрудника, который и без этого переносит существенные умственные нагрузки.

Для снижения утомления в процессе труда и повышения работоспособности и сохранения его здоровья, используют следующие эффективные методы:

* рациональная организация рабочего места, времени, труда и отдыха;
* создание благоприятного микроклимата;
* производственная гимнастика.

Для профилактики психофизических перегрузок программисту необходимо рассмотреть требования к производственной среде, рабочей мебели и организации труда, способы и средства их обеспечения.

К помещению предъявляются требования по освещению: оно должно иметь как естественное, так и искусственное. Оконные проемы должны быть оборудованы регулируемыми светозащитными устройствами типа: жалюзи, занавеси, внешние козырьки и др. Искусственное освещение должно осуществляться системой общего равномерного освещения. Следует ограничивать прямую блесткость от источников освещения, при этом яркость светящихся поверхностей (окна, светильники и другие), находящихся в поле зрения, должна быть не более 200 кд/м2 (кандел на метр квадратный).

Яркость светильников общего освещения в зоне углов излучения от 50 до 90 градусов с вертикалью в продольной и поперечной плоскостях должна составлять не более 200 кд/м2, защитный угол светильников должен быть не менее 40°. Светильники местного освещения должны иметь не просвечивающий отражатель с защитным углом не менее 40°.

Коэффициент пульсации не должен превышать 5%, для этого применяются газоразрядные лампы в светильниках общего и местного освещения с высокочастотными пускорегулирующими аппаратами (ВЧ ПРА) для любых типов светильников [11].

Теперь рассмотрим основные требования предъявляемые к рабочим помещениям и мебели. Рабочее помещение программистов должно способствовать созданию творческой обстановки. Этому способствует рациональное цветовое оформление с мягкими светлыми тонами. Благоприятные климатические условия на рабочем месте так же могут обеспечить системы кондиционирования воздуха, а для повышения влажности воздуха следует применять увлажнители, заправляемые ежедневно дистиллированной или прокипяченной питьевой водой.

Располагать рабочие места допускается по периметру помещения или рядами. При конструировании оборудования и организации рабочего места программиста следует обеспечить соответствие конструкции всех элементов рабочего места и их взаимного расположения эргономическим требованиям с учетом характера выполняемой деятельности, комплектности технических средств, форм организации труда и основного рабочего положения пользователя [13].

Конструкция рабочего стола должна обеспечивать оптимальное размещение на рабочей поверхности используемого оборудования с учетом его количества и конструктивных особенностей, характера выполняемой работы. При этом допускается использование рабочих столов различных конструкций, отвечающих современным требованиям эргономики.

Высота рабочей поверхности стола должна регулироваться в пределах 680-800 мм, при отсутствии такой возможности высота рабочей поверхности стола должна составлять 725 мм.

Модульными размерами рабочей поверхности стола, на основании которых должны рассчитываться конструктивные размеры, следует считать: ширину 800, 1000, 1200 и 1400 мм, глубину 800 и 1000 мм при нерегулируемой его высоте, равной 725 мм.

Рабочий стол должен иметь пространство для ног высотой не менее 600 мм, шириной - не менее 500 мм, глубиной на уровне колен -  
не менее 450 мм и на уровне вытянутых ног - не менее 650 мм [11].

Конструкция рабочего кресла должна обеспечивать поддержание рациональной рабочей позы, позволять изменять её с целью снижения статического напряжения мышц шейно-плечевой области и спины для предупреждения развития утомления.

Тип рабочего кресла должен выбираться в зависимости от характера и продолжительности работы и с учетом роста пользователя.

Рабочее кресло должен быть с подлокотниками, подъемно-поворотным регулированием по высоте и углам наклона сиденья и спинки, расстоянию спинки от переднего края сиденья, при этом регулировка каждого параметра должна быть независимой, и иметь надежную фиксацию.

Поверхность сиденья, спинки и других элементов стула (кресла) должна быть полумягкой, с нескользящим, не электризующимся и воздухопроницаемым покрытием, обеспечивающим легкую очистку от загрязнений.

Конструкция рабочего стула (кресла) должна обеспечивать:

* ширину и глубину поверхности сиденья не менее 400 мм;
* поверхность сиденья с закругленным передним краем;
* регулировку высоты поверхности сиденья в пределах 400-550 мм и углам наклона вперед до 15° и назад до 5°;
* высоту опорной поверхности спинки 300±20 мм, ширину - не менее 380 мм и радиус кривизны горизонтальной плоскости - 400 мм;
* угол наклона спинки в вертикальной плоскости в пределах 0±30°;
* регулировку расстояния спинки от переднего края сиденья в пределах 260-400 мм;
* стационарные или съемные подлокотники длиной не менее 250 мм и шириной 50-70 мм;
* регулировку подлокотников по высоте над сиденьем в пределах 230 ± 30 мм и внутреннего расстояния между подлокотниками в пределах 350-500 мм.

Рабочее место должно быть оборудовано подставкой для ног, имеющей ширину не менее 300 мм, глубину не менее 400 мм, регулировку по высоте в пределах до 150 мм и по углу наклона опорной поверхности подставки до 20 градусов. Поверхность подставки должна быть рифленой и иметь по переднему краю бортик высотой 10 мм [11].

В заключение рассмотрим требования к организации режима труда и отдыха. Работа сотрудника отдела разработок относится к группе «В» – творческая работа в режиме диалога с ЭВМ. По категории тяжести и напряженности данный вид работ следует отнести ко второй категории. Программист при работе обрабатывает информацию не более 4 часов за смену. Можно сделать вывод, что при 8-ми часовой смене следует устанавливать перерывы через 2 часа от начала рабочей смены и через 1,5 – 2 часа после обеденного перерыва продолжительностью 15 минут каждый или продолжительностью 10 минут через каждый час работы [11].

Во время таких перерывов рекомендуется выполнять комплексы упражнений для глаз, пальцев, кистей рук, спины, ног, шеи и прочих частей тела, подверженных нагрузке.

Общие характеристики помещения и рабочего места, такие как освещение, параметры рабочего места и мебели в отделе разработок компании «SaM Solutions», находятся в пределах нормы. По этой причине стоит обращать особое внимание на другие показатели, которые целиком и полностью зависят от самих сотрудников: положение тела, монитора, рук, регулируемая высота рабочего кресла, а так же соблюдение регламентируемых перерывов. Таким образом, предложенные выше мероприятия позволят в полной мере обеспечить комфортные условия труда сотрудников.

**ЗАКЛЮЧЕНИЕ**

В настоящей курсовой работе была рассмотрена и спроектирована система обработки текстов нарративных источников на основе интеграции лингвистических и статистических методов анализа.

Результатом выполнения стал облачный сервис, представляющий собой систему по анализу текстовых данных. Данный подход позволил применить концептуально новые решения в создании подобных систем. В рамках стандартного разделения на клиентскую и серверную часть, клиент стал практически полностью независим от сервера, что позволяет сократить количество запросов и снизить нагрузку на последний. Таким образом, пользователю достаточно один раз произвести анализ в рамках выбранного проекта и работать с ним в режиме оффлайн без внесения качественных изменений в полученных данных. Однако из этого возникли и некоторые недостатки, проявляющиеся в обратной совместимости и наличии поддержки лишь самых последних версий браузеров на момент разработки.

Процесс разработки данного программного обеспечения представлял (и представляет, ввиду продолжения разработки) собой спиральную модель. В качестве подхода к разработке программного обеспечения в рамках спиральной модели жизненного цикла, была использована получившая в последнее время широкое распространение методология быстрой разработки приложений RAD (Rapid Application Development). Это было обусловлено, в первую очередь тем, что разработка велась всего одним программистом и имела короткий производственный график. Благодаря этому, была возможность представить рабочие версии ПО на различных конференциях, в ходе которых системабыла дана конструктивная критика и оценка системы, а так же высказаны различные важные предложения.

Необходимо отметить, что полученный программный продукт будет интересен в первую очередь специалистам, работающим в сферах, где так или иначе используется анализ текста в ходе основной работы или же сам по себе является таковой. Это могут быть лингвисты, исследователи языка, и, в первую очередь, историки. Все объясняется тем, что сервис реализует три основных подхода (контент-анализ, дискурс-анализ и психолингвистический анализ), которые используются в историческом исследовании.

В ходе реализации системы, в качестве одной из задач, была решена проблема интеграции этих подходов. Благодаря использованию таких фреймворков, как Spring Framework и Hibernate на стороне сервера, а так же непосредственно языка Java, появилась возможность грамотно манипулировать данными. Это позволило избежать ненужного дублирования результатов, и организовать их хранение и передача на другие этапы в рамках проекта, и таким образом в полной мере реализовать модель сочетаемости нетрадиционных методов исторического познания текста нарративного источника, описанную в начале.

Хотя данный продукт и находится на стадии бета-тестирвания, он уже пригоден для использования и выполняет основные задачи. Однако это является лишь малой частью его потенциала. Система уже имеет свой вектор развития до полноценной профессиональной сети, основной задачей которой будет уже не только анализа текста, но и использование результатов этого анализа. Такая сеть призвана объединить пользователей и внедрить социальную составляющую, позволяя делиться результатами исследования, источниками, создавать отчеты, статьи и публикации. Помимо всего прочего, сам сервис, обладая огромным массивом документов, который сформирован пользователями, может составлять свой корпус языка. На данный момент реализована поддержка лишь русского языка, но в дальнейшем продукт с легкостью может быть расширен до возможности анализа текстов и других языков, а так же мульти языковых источников.

На данный же момент сервис уже имеет широкий круг применения, а в скором времени будет запущен в тестовом режим в сети интернет.

**Список использованных источников**

1. World’s Data More Than Doubling Every Two Years—Driving Big Data Opportunity, New IT Roles [Электронный ресурс]. – Электронные данные. – Режим доступа: http://www.emc.com/about/news/press/2011/20110628-01.htm
2. Text Mining [Электронный ресурс]. – Электронные данные. – Режим доступа: http://en.wikipedia.org/wiki/Text\_mining#Text\_mining\_and\_text\_analytics
3. Content Analysis of Verbatim Explanations (CAVE) Research [Электронный ресурс]. – Электронные данные. – Режим доступа: http://www.ppc.sas.upenn.edu/cave.htm
4. Сидорцов В.Н., Приборович А.А. Научный дискурс историка: социальная обусловленность и методология исследования. Минск, Изд. центр БГУ. 2013 г. - 199 с.
5. UML [Электронный ресурс]. – Электронные данные. – Режим доступа: http://ru.wikipedia.org/wiki/UML
6. Нормализация отношений. Нормальные формы: 1НФ, 2НФ, 3НФ [Электронный ресурс]. – Электронные данные. – Режим доступа: http://dmtsoft.ru/bn/109/as/oneaticleshablon
7. What cloud computing means? [Электронный ресурс]. – Электронные данные. – Режим доступа: http://www.onbile.com/info/what-cloud-computing-means/
8. Положения о Парке высоких технологий, утвержденным Декретом Президента РБ от 22.09.2005 № 12
9. Wikipeadia – Физиология труда. [Электронный ресурс]. – Электронные данные. – Режим доступа: http://ru.wikipedia.org/wiki/Физиология\_труда
10. Лазаренков А. М. Охрана труда: учебник / А. М. Лазаренков. – Минск: БНТУ, 2004. – 497с.
11. СанПиН 2.2.2.9-131 РБ 2000. Гигиенические требования к видеодисплейным терминалам, персональным электронно-вычислительным машинам и организации работы.
12. Душков Б.А. и др. Основы инженерной психологии / Б. А. Душков. – Москва-Екатеринбург, 2002. ­–576 с.
13. Основы инженерной психологии. Учебник для техн. вузов / Под ред. Б.Ф. Ломова – М., 1996.

**Приложение А**

****

Рисунок А.1 – Диаграмма состояний

**Приложение А**

****

Рисунок А.2 – Диаграмма последовательностей

**Приложение А**

****

Рисунок А.3 – Диаграмма классов (package org.edloidas.text)

**Приложение А**

****

Рисунок А.4 – Диаграмма классов (package org.edloidas.web.json)

**Приложение А**

****

Рисунок А.5 – Диаграмма классов (package org.edloidas.web.controller)

**Приложение А**

****

Рисунок А.6 – Диаграмма классов (package org.edloidas.web.service)

**Приложение А**

****

Рисунок А.7 – Диаграмма классов (package org.edloidas.entity)

**Приложение А**

****

Рисунок А.7 – Диаграмма классов (package org.edloidas.entity.common)

**Приложение А**

****

Рисунок А.8 – Диаграмма компонент

**Приложение А**

****

Рисунок А.9 – Диаграмма развертывания

**Приложение Б**



Рисунок Б.1 – Алгоритм работы системы

**Приложение Б**



Рисунок Б.2 – Обобщенный алгоритм нормализации слова

**Приложение Б**



Рисунок Б.3 – Обобщенный алгоритм авторизации пользователя.

**Приложение В**

**Листинг программы**

package org.edloidas.web.controller;

import org.apache.log4j.Logger;

import org.edloidas.entity.common.Project;

import org.edloidas.entity.common.Source;

import org.edloidas.text.words.Concordance;

import org.edloidas.text.words.KeyWord;

import org.edloidas.text.words.MeanWord;

import org.edloidas.text.words.SenseBlock;

import org.edloidas.web.json.ErrorCode;

import org.edloidas.web.json.JsonData;

import org.edloidas.web.service.EntityService;

import org.edloidas.web.service.SessionService;

import org.edloidas.web.service.TextService;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.\*;

import javax.annotation.Resource;

import java.util.ArrayList;

import java.util.Collections;

import java.util.Comparator;

import java.util.List;

/\*\*

\* Content Controller, that handles requests for content analysis.

\*

\* @author Никита

\*/

@Controller

@RequestMapping(value = "/content")

@SessionAttributes({"userSession", "textService"})

public class ContentController {

/\*\*

\* Logger for common output. Uses root logger output style.

\* See log4j.properties file for more details.

\*/

private final static Logger LOGGER = Logger.getLogger(ContentController.class);

/\*\*

\* Project service for CRUD operations.

\*

\* @see org.edloidas.web.service.common.ProjectService

\*/

/\* USE Interface here.

\* Otherwise SPRING will throw BeanNotOfRequiredTypeException.

\* Spring uses the interface type to make dependency injection!

\*/

@Resource(name = "projectService")

private EntityService<Project> projectService;

/\*\*

\* Source service for CRUD operations.

\*

\* @see org.edloidas.web.service.common.SourceService

\*/

/\* USE Interface here.

\* Otherwise SPRING will throw BeanNotOfRequiredTypeException.

\* Spring uses the interface type to make dependency injection!

\*/

@Resource(name = "sourceService")

private EntityService<Source> sourceService;

/\*\*

\* Session variables holder.

\*

\* @see org.edloidas.web.service.SessionService

\*/

@Resource(name = "sessionService")

private SessionService userSession;

/\*\*

\* Text session variables holder.

\*

\* @see org.edloidas.web.service.TextService

\*/

@Resource(name = "textService")

private TextService textService;

/\*\*

\* Handles access to content section.

\*

\* @return {@code String}, that represents text response of operation success.

\*/

@RequestMapping(value = "/", method = RequestMethod.GET)

String content(Model model) {

try {

if (!userSession.isLogged()) {

throw new Exception("Access denied. User not authorized.");

}

model.addAttribute("sessionUser", userSession.getUser().getName());

model.addAttribute("sessionProject", userSession.getProject().getName());

if (userSession.getProject().getId() == -1) {

return navaigateToProjects(model);

}

List<Source> sources = sourceService.getAll(new Source(userSession.getProject()));

StringBuilder sb = new StringBuilder();

for (Source src : sources) {

if (src.getStatus())

sb.append(src.getText()).append("\n");

}

model.addAttribute("text", sb.toString());

if (userSession.isUpdateText()) {

userSession.setUpdateText(false);

textService.doUpdate(sb.toString());

}

float water = (1 - (float)textService.getMeanCount()

/ (float)textService.getWordsCount()) \* 100;

model.addAttribute("textWordsCount", textService.getWordsCount());

model.addAttribute("textWordsLength", textService.getWordsLength());

model.addAttribute("textMeanCount", textService.getMeanCount());

model.addAttribute("textWater", water);

return "content";

} catch (Exception ex) {

LOGGER.error("Server error.", ex);

return "index";

}

}

/\*\*

\* Handles access to content section.

\*

\* @return {@code String}, that represents text response of operation success.

\*/

@RequestMapping(value = "/meaningful/list", method = RequestMethod.GET)

String contentMeaningfulList(Model model) {

try {

if (!userSession.isLogged()) {

throw new Exception("Access denied. User not authorized.");

}

model.addAttribute("sessionUser", userSession.getUser().getName());

model.addAttribute("sessionProject", userSession.getProject().getName());

if (userSession.getProject().getId() == -1) {

return navaigateToProjects(model);

}

model.addAttribute("meanWords", textService.getMeanWords());

model.addAttribute("count", textService.getMeanWords().size());

return "content-mean-list";

} catch (Exception ex) {

LOGGER.error("Server error.", ex);

return "index";

}

}

/\*\*

\* Handles and retrieves the AJAX request for viewing meaningful words.

\*

\* @return {@code String}, that represents text response of operation status.

\*/

/\* Using mapping produces = "application/json; charset=utf-8" is highly recommended.

\* In other case you will have bad Response with ISO character encoding. \*/

@RequestMapping(value = "/meaningful/view/{meanId}", method = RequestMethod.GET,

produces = "application/json; charset=utf-8")

public String contentMeaningfulView(@PathVariable("meanId") String meanId,

Model model) {

try {

if (!userSession.isLogged()) {

throw new Exception("Access denied. User not authorized.");

}

model.addAttribute("sessionUser", userSession.getUser().getName());

model.addAttribute("sessionProject", userSession.getProject().getName());

if (userSession.getProject().getId() == -1) {

return navaigateToProjects(model);

}

int id = Integer.parseInt(meanId);

List<Concordance> cons = textService.getConcordance(id);

int count = cons.size();

String word = "";

if (id < textService.getMeanWords().size()) {

word = textService.getMeanWords().get(id).getName();

}

model.addAttribute("cons", cons);

model.addAttribute("count", count);

model.addAttribute("word", word);

model.addAttribute("id", id);

return "content-mean-view";

} catch (Exception ex) {

LOGGER.error("Server error.", ex);

return "index";

}

}

/\*\*

\* Handles access to content section.

\*

\* @return {@code String}, that represents text response of operation success.

\*/

@RequestMapping(value = "/meaningful/graph", method = RequestMethod.GET)

String contentMeaningfulGraph(Model model) {

try {

if (!userSession.isLogged()) {

throw new Exception("Access denied. User not authorized.");

}

model.addAttribute("sessionUser", userSession.getUser().getName());

model.addAttribute("sessionProject", userSession.getProject().getName());

if (userSession.getProject().getId() == -1) {

return navaigateToProjects(model);

}

model.addAttribute("meanWords", textService.getMeanWords());

model.addAttribute("count", textService.getMeanWords().size());

return "content-mean-graph";

} catch (Exception ex) {

LOGGER.error("Server error.", ex);

return "index";

}

}

/\*\*

\* Handles access to content section.

\*

\* @return {@code String}, that represents text response of operation success.

\*/

@RequestMapping(value = "/key/list", method = RequestMethod.GET)

String contentKeyList(Model model) {

try {

if (!userSession.isLogged()) {

throw new Exception("Access denied. User not authorized.");

}

model.addAttribute("sessionUser", userSession.getUser().getName());

model.addAttribute("sessionProject", userSession.getProject().getName());

if (userSession.getProject().getId() == -1) {

return navaigateToProjects(model);

}

model.addAttribute("keyWords", textService.getKeyWords());

model.addAttribute("count", textService.getKeyWords().size());

return "content-key-list";

} catch (Exception ex) {

LOGGER.error("Server error.", ex);

return "index";

}

}

/\*\*

\* Handles and retrieves the AJAX request for adding new key word.

\*

\* @return {@code String}, that represents text response of operation status.

\*/

@RequestMapping(value = "/key/add", method = RequestMethod.POST,

produces = "application/json; charset=utf-8")

public

@ResponseBody

String contentKeyAdd(@RequestParam(value = "name", required = true) String name) {

try {

if (!userSession.isLogged())

return "{\"code\":2,\"msg\":\"Access denied.\",\"data\":\"User has no rights to do this.\"}";

JsonData json;

int id = 0;

if (textService.getKeyWords().size() > 0) {

id = textService.getKeyWords().get(textService.getKeyWords().size() - 1).getId() + 1;

}

textService.getKeyWords().add(new KeyWord(id, name));

json = new JsonData(0, "Category added.", name);

return json.toString();

} catch (Exception ex) {

LOGGER.info(ex.getMessage());

return "{\"code\":1,\"msg\":\"Server error.\",\"data\":\"See server log.\"}";

}

}

/\*\*

\* Handles and retrieves the AJAX request for adding new key word.

\*

\* @return {@code String}, that represents text response of operation status.

\*/

@RequestMapping(value = "/key/delete", method = RequestMethod.POST,

produces = "application/json; charset=utf-8")

public

@ResponseBody

String contentKeyDelete(@RequestParam(value = "id", required = true) String keyId) {

try {

if (!userSession.isLogged())

return "{\"code\":2,\"msg\":\"Access denied.\",\"data\":\"User has no rights to do this.\"}";

JsonData json;

int id = Integer.parseInt(keyId);

json = new JsonData(0, "Category not found.", keyId);

for (int i = 0; i < textService.getKeyWords().size(); i++) {

if (textService.getKeyWords().get(i).getId() == id) {

textService.getKeyWords().remove(i);

json = new JsonData(0, "Category deleted.", keyId);

break;

}

}

return json.toString();

} catch (Exception ex) {

LOGGER.info(ex.getMessage());

return "{\"code\":1,\"msg\":\"Server error.\",\"data\":\"See server log.\"}";

}

}

/\*\*

\* Handles and retrieves the AJAX request for viewing key words.

\*

\* @return {@code String}, that represents text response of operation status.

\*/

/\* Using mapping produces = "application/json; charset=utf-8" is highly recommended.

\* In other case you will have bad Response with ISO character encoding. \*/

@RequestMapping(value = "/key/view/{keyId}", method = RequestMethod.GET,

produces = "application/json; charset=utf-8")

public String contentKeyView(@PathVariable("keyId") String keyId,

Model model) {

try {

if (!userSession.isLogged()) {

throw new Exception("Access denied. User not authorized.");

}

model.addAttribute("sessionUser", userSession.getUser().getName());

model.addAttribute("sessionProject", userSession.getProject().getName());

if (userSession.getProject().getId() == -1) {

return navaigateToProjects(model);

}

int id = Integer.parseInt(keyId);

for (KeyWord word : textService.getKeyWords()) {

if (word.getId() == id) {

model.addAttribute("id", id);

model.addAttribute("word", word.getName());

model.addAttribute("meanWords", textService.getMeanWords());

model.addAttribute("meanCount", textService.getMeanWords().size());

model.addAttribute("keyMeanWords", word.getMeanWords());

model.addAttribute("keyMeanCount", word.getMeanWords().size());

List<SenseBlock> senseBlocks = textService.getSenseBlocks();

model.addAttribute("senseBlocks", senseBlocks);

model.addAttribute("senseCount", senseBlocks.size());

return "content-key-view";

}

}

LOGGER.error("KeyWord not found.");

return "content-key-list";

} catch (Exception ex) {

LOGGER.error("Server error.", ex);

return "index";

}

}

/\*\*

\* Handles and retrieves the AJAX request for adding new key word.

\*

\* @return {@code String}, that represents text response of operation status.

\*/

@RequestMapping(value = "/key/view", method = RequestMethod.POST,

produces = "application/json; charset=utf-8")

public

@ResponseBody

String contentKeyMeanUpdate(@RequestParam(value = "keyId", required = true) String keyId,

@RequestParam(value = "meanId", required = true) String meanId) {

try {

if (!userSession.isLogged())

return "{\"code\":2,\"msg\":\"Access denied.\",\"data\":\"User has no rights to do this.\"}";

JsonData json;

int keyIdVal = Integer.parseInt(keyId);

int meanIdVal = Integer.parseInt(meanId);

for (KeyWord key : textService.getKeyWords()) {

if (key.getId() == keyIdVal) {

for (int i = 0; i < key.getMeanWords().size(); i++) {

if (key.getMeanWords().get(i).getId() == meanIdVal) {

key.getMeanWords().remove(i);

json = new JsonData(3, "Category inactivated.", "inactivated");

return json.toString();

}

}

key.getMeanWords().add(textService.getMeanWords().get(meanIdVal));

json = new JsonData(3, "Category activated.", "activated");

return json.toString();

}

}

//json = new JsonData(0, "Category added.", status);

json = new JsonData(4, "Category not found.", ErrorCode.code4);

return json.toString();

} catch (Exception ex) {

LOGGER.info(ex.getMessage());

return "{\"code\":1,\"msg\":\"Server error.\",\"data\":\"See server log.\"}";

}

}

private String navaigateToProjects(Model model) {

List<Project> projects;

int count;

projects = projectService.getAll(new Project(userSession.getUser()));

count = projects.size();

model.addAttribute("projects", projects);

model.addAttribute("count", count);

return "project-list";

}

}