МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ
Кафедра информационных систем управления

ВОЛЧЕЦКАЯ Полина Сергеевна

ТЕХНОЛОГИЯ ПОСТРОЕНИЯ АДАПТИВНЫХ СИСТЕМ ОБРАБОТКИ ГЕТЕРОГЕННЫХ ДОКУМЕНТОВ

Магистерская диссертация
специальность 1-31 80 09 Прикладная математика и информатика

Научный руководитель
Краснопрошин В. В.,
профессор, доктор технических наук

Допущена к защите
«____» _____20__ г.
Зав. кафедрой информационных
систем управления
Недзьведь Александр Михайлович
профессор, доктор технических наук

Минск, 2024

РЕФЕРАТ

Магистерская диссертация, 64 с., 17 рисунков, 1 таблица, 28 источников, 1 приложение.

Ключевые слова: ГЕТЕРОГЕННЫЕ ДОКУМЕНТЫ, АНАЛИЗ ТЕКСТА, ПРИНЯТИЕ РЕШЕНИЙ.

Объект исследования: новые типы документов, формирующиеся в результате информатизации общества, и совершенствование процессов их обработки.

Предмет исследования: методы обработки новых типов документов в системах принятия решений.

Цель работы: разработка средств для сокращения затрат на анализ и улучшение текстов, а также принятие решений.

Результаты: система, обеспечивающая сокращение объема ручного труда при анализе, оценке и коррекции документов, т.е. дополнение сервисов SAS.

Практическая значимость: разработанный комплекс алгоритмов и программ обеспечивает значительное сокращение времени и средств на анализ новых типов документов.

РЭФЕРАТ

Магістарская дысертацыя, 64 с., 17 малюнкаў, 1 табліца, 28 крыніц, 1 дадатак.

Ключавыя словы: ГЕТЭРАГЕННЫЯ ДАКУМЕНТЫ, АНАЛІЗ ТЭКСТУ, ПРЫНЯЦЦЕ РАШЭННЯЎ.

Аб'ект даследавання: новыя тыпы дакументаў, якія фармуюцца ў выніку інфарматызацыі грамадства, і ўдасканаленне працэсаў іх апрацоўкі.

Прадмет даследавання: метады апрацоўкі новых тыпаў дакументаў у сістэмах прыняцця рашэнняў.

Мэта работы: распрацоўка сродкаў для скарачэння затрат на аналіз і паляпшэнне тэкстаў, а таксама прыняцце рашэнняў.

Вынікі: сістэма, якая забяспечвае скарачэнне аб'ёму ручной працы пры аналізе, адзнацы і карэкцыі дакументаў, г.зн. дадатак сэрвісаў SAS.

Практычнае значэнне: распрацаваны комплекс алгарытмаў і праграм забяспечвае значнае скарачэнне часу і сродкаў на аналіз новых тыпаў дакументаў.

ANNOTATION

Master's thesis, 64 p., 17 illustrations, 1 table, 28 sources, 1 application.

Key words: HETEROGENEOUS DOCUMENTS, TEXT ANALYSIS, DECISION MAKING.

Object of research: new types of documents emerging as a result of informatization of society, and improvement of their processing processes.

Subject of research: methods for processing new types of documents in decision-making systems.

Purpose of the work: to develop tools to reduce the costs of analyzing and improving texts, as well as decision making.

Results: a system that reduces the amount of manual labor in the analysis, evaluation and correction of documents, i.e. addition of SAS services.

Practical significance: the developed complex of algorithms and programs provides a significant reduction in time and money for analyzing new types of documents.

ОГЛАВЛЕНИЕ

ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ И СОКРАЩЕНИЙ 6
ВВЕДЕНИЕ 7
ГЛАВА 1. АНАЛИЗ ПРОБЛЕМЫ И ПОСТАНОВКА ЗАДАЧИ 10
1.1 Анализ актуальных проблем обработки документов 10
1.2 Постановка задачи 14
1.3 Выводы 17
ГЛАВА 2. РАЗРАБОТКА МОДЕЛЕЙ И АЛГОРИТМОВ 18
2.1 Понятийный базис 18
2.2 Модели 20
2.3 Алгоритмы 23
2.4 Выводы 33
ГЛАВА 3. РАЗРАБОТКА ПРОГРАММНОГО ИНСТРУМЕНТАРИЯ 34
3.1 Архитектура целевой системы 34
3.2 Реализация архитектуры на основе микросервисного подхода 35
3.3 Методика использования системы 36
3.4 Выводы 37
ГЛАВА 4. АПРОБАЦИЯ РАЗРАБОТАННОЙ СИСТЕМЫ 38
4.1 Постановка прикладной задачи 38
4.2 Решение прикладной задачи на основе технологии 38
4.3 Выводы 46
ЗАКЛЮЧЕНИЕ 48
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 49
ПРИЛОЖЕНИЕ А 52

ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ И СОКРАЩЕНИЙ

В тексте магистерской диссертации используются следующие обозначения и сокращения:

БД – база данных

ИИ – искусственный интеллект

ИТ – информационные технологии

ЛПР – лицо, принимающее решение

ПО – программное обеспечение

СУБД – система управления базами данных

Big Data – большие данные

IT – Information technology (Информационные технологии)

SAS – Statistical Analysis System (Система статистического анализа)

ВВЕДЕНИЕ

Принятие решений в информационном обществе основано на результатах анализа потоков документов, количество и объем которых постоянно увеличивается. Качество решений зависит от качества входных документов и эффективности средств их обработки. Критическим параметром является время синтеза решения [1].

Для крупных компаний разработаны достаточно эффективные дорогостоящие системы для анализа текста, например, SAS Visual Text Analytics. В государственных организациях, а также средних и малых компаниях проблема работы с документами не решена в должной мере.

В ходе написания магистерской диссертации выявлены проблемы, затрудняющие принятие решений на основе постоянно растущего потока документов различного типа в государственных и частных организациях и предложено их теоретическое и практическое решение с учетом свойств новой информационной среды XXI в.

Рассмотрим проблемы, требующие первоочередного решения.

Во-первых, глобальная информатизация, изменение характера и сложности прикладных задач в XXI в привело к необходимости построения новой модели документов (например, в умном контракте в системах blockchain Ethereum [2]), обеспечивающих автоматизацию обработки гетерогенных текстов.

Во-вторых, постоянно растет количество случаев перехвата и несанкционированного изменения контента документов при их пересылке по электронным глобальным и локальным каналам связи [2].

В-третьих, обнаружены следующие негативные факторы:

– постоянно увеличивающийся объем человеческого труда корректоров, редакторов, нормировщиков, преподавателей негативно влияет на качество проверки правильности оформления и других требований к входным документам, а также приводит к периодическому увеличению финансовых затрат;

– сложность адаптации новой терминологии, которая в XXI в. постоянно изменяется по мере технологических и других изменений в области IT;

– субъективность и высокий уровень усталости персонала при работе с документами приводят к тому, что результат коррекции не гарантирует на 100% исправление ошибок различного рода;

– значительные финансовые и другие потери за счет несанкционированного изменения или уничтожения документа при передаче по каналам связи;

– неполнота комплекта компонентов, штатно поставляемых вместе с текстом документа.

В результате на вход специализированных систем обработки типа SAS часто поступают априори нештатные по структуре и терминологии документы, снижающие эффективность решений, принимаемых на их основе. Более того, несанкционированные изменения цифровой части, связанной с финансами, могут привести к банкротству компании.

В качестве основы для комплексного решения указанных проблем были взяты, в частности, результаты, полученные на кафедре интеллектуальной обработки документов МФТИ (руководитель Логинов В.В.), университетах Calgary, Michigan, а также корпоративные документы компании SAS.

В первой главе проанализированы литературные источники и проблемы организаций, связанных с обработкой документов. Выявлены основные нерешенные вопросы, определена актуальная тема исследования, сформулирована задача для комплексного автоматизированного решения наиболее важных проблем.

Во второй главе предложена новая модель документа, решающая проблему формирования гетерогенных документов, характерных для современных задач. Разработан комплекс алгоритмов применения этой модели, включая алгоритмы обеспечения безопасности.

В третьей главе разработана архитектура целевой системы, которая реализована в форме программного инструментария, обеспечивающего автоматизацию работы с новым типом документов в соответствии с

разработанными алгоритмами. Также разработана методика применения разработанной системы.

В четвертой главе проведена апробация разработанного программного инструментария в процессе решения практической задачи и выполнена оценка эффективности результата.

ГЛАВА 1. АНАЛИЗ ПРОБЛЕМЫ И ПОСТАНОВКА ЗАДАЧИ

1.1 Анализ актуальных проблем обработки документов

В XXI веке наблюдается эволюция процессов обработки документов в связи с ростом количества оперативных задач принятия решений и заменой бумажных носителей на цифровые (электронные) формы достаточно сложной структуры практически во всех сферах человеческой деятельности.

Как показал анализ процессов работы с документами в организациях различного типа (БГУИР, БГУ, проектные фирмы, ИТ-компании, рекрутские компании, анализирующие тысячи резюме претендентов на рабочее место и т.д.), современные документы для наиболее полного представления результатов постепенно интегрируют чужеродные (не текстовые) компоненты. Например, документ в Ethereum включает текстовую часть и исполнительную (EXE) часть. Постоянный рост объема презентационного материала (в частности, в дипломах, диссертациях), требует дополнение основного текста внешними pptx-компонентами. Служебные документы в определенных службах в обязательном порядке включают внешние компоненты, представленные в графическом (jpg), аудио (mp3) и видео (mp4) компоненты. При этом требования к документам постоянно изменяются, как и терминология в быстро развивающихся областях, включая ИТ. Увеличение объема финансовых потерь в результате несанкционированного доступа, изменение или уничтожение документов привело к необходимости поиска различных способов защиты документов, включая интеграцию специальных знаков, слов и т.д. Соответственно, увеличивается нагрузка на персонал и требования к их знаниям новых требований и стандартов для объективной оценки качества входных документов и их готовности к автоматизированной обработке и принятия эффективных решений.

Для успешной адаптации традиционных форм документов в формы, удобные для быстрой и качественной обработки (например, для построения умных контрактов в Ethereum), выполняется широкий спектр

междисциплинарных исследований природы документов и методов их трансформации.

В целом, решается фундаментальная проблема: как при лавинообразном росте объема входных документов и описанном выше фундаментальном усложнении их структуры увеличить скорость обработки до уровня, достаточного до оперативного принятия решений на основе результатов анализа качественного контента.

Одним их первых, кто понял необходимость переосмысления самого понятия “документ” в современной интерпретации и создания теории цифровых документов как основы для повышения скорости обработки и качества принятия решений в политике и бизнесе был M.Buckland [13]. Его результаты послужили основой для исследований ученых ведущих университетов, включая И.Н.Кузнецова [1], M.Buckland [14], G.Chakraborty [15], T.Jade [19], D.Khurana [22], P.Scifleet [25], M.Talafidaryani [27], C.Vasquez [28] и др. Основное внимание в их исследованиях уделяется лингвистическому и статистическому анализу документов и визуализации результатов для принятия релевантных решений. На основе полученных результатов разрабатываются специализированные системы, автоматизирующие различные процессы обработки документов.

Для автоматизации обработки контента документов различного типа в 2015-2023 гг. был разработан ряд систем. В целом они нацелены на конкретные задачи из различных предметных областей. Рассмотрим наиболее популярные системы, которые можно рассматривать как основу для исследований и усовершенствования.

Linkurious – предназначен для визуализации и анализа графиков, обнаружения мошенничества, отмывания денег, разведки или кибербезопасности.

Orion Magic – используется для поиска, анализа, организации и визуализации отчетности. Позволяет выполнять поиск по файлам, документам, веб-страницам и базам данных.

IBM Text Mining – реализует операции извлечения знаний и высококачественной информации из текстовых массивов. Для этого используются операторы выявления шаблонов, их статистической обработки и определения тенденций.

Oracle Text – выполняет высокоэффективный полнотекстовый поиск и анализ текстовой информации на основе лингвистических подходов и методов.

Megaputer PolyAnalyst – позволяет проводить классификацию и кластеризацию текстов, извлекать сущности и факты, определять тематику и тональность, а также решать ряд других прикладных аналитических задач по работе с документами на русском, английском и других языках.

M-Brain Intelligence Plaza – ИТ-платформа для управления потоками текстов о рынках и конкурентах для отделов аналитики, продаж, маркетинга, менеджмента. Хранение в облаке, структурирование и внутрикорпоративная рассылка информации по темам.

Главред – предназначен для очистки текста от словесного мусора и проверки на соответствие заранее определенному стилю.

Нетрудно заметить, что все перечисленные системы имеют узконаправленный набор функций, ограниченный статистическими и лингвистическими операциями.

Несмотря на функциональное разнообразие, указанные и другие системы используют отдельные элементы двухфазной технологии аналитической обработки. На первой фазе (ETL) производится автоматизированный анализ отдельных документов, структуризация их контента и формирование хранилищ исходной и аналитической (WareHouse) информации. На второй фазе (OLAP, Text Mining, Data Mining) выполняется извлечение в оперативном режиме знаний из (WareHouse) или из полученной по SQL-запросу подборки документов.

В качестве более эффективного подхода можно рассматривать систему SAS Text Analytics [19], которая подходит к анализу с системной точки зрения. Основное отличие подхода SAS заключается в том, что задачи анализа текста трансформируется в одну из четырех типовых технических задач:

(1) статистический анализ текста; (2) категоризацию; (3) извлечение фактов; (4) оценку эмоциональной окраски контента. Указанные задачи решаются тремя программными пакетам: SAS Text Miner, SAS Content Categorization и SAS Sentiment Analysis (рисунок (1.1)).

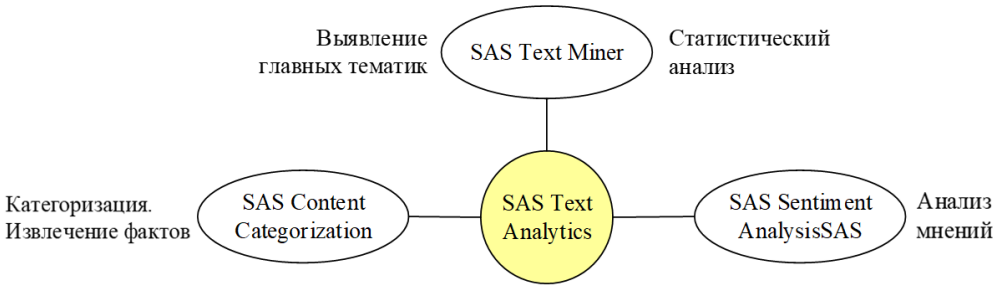

Рисунок 1.1 – Сервисы SAS Text Analytics

При очевидных успехах данного междисциплинарного направления, включая системы, аналогичные SAS, остаются нерешенными ряд проблем, которые прямо влияют на объем ручного труда и стоимость обработки документов. В первую очередь, к ним можно отнести следующие:

– “проблема универсальной модели”: большое разнообразие жестких (сфокусированных на один тип документа) моделей документов, отражающих специфические корпоративные стандарты (например, в Bitcoin, Ethereum и др.) при ориентации на один естественный язык (английский, китайский и др.). Попытки построения унифицированной модели для разных бизнес-процессов с адаптацией к языку находятся на ранней стадии развития;

– “проблема объективности”: сложность интерпретации и понимания результатов статистического и лингвистического анализа текста большого объема и различной структуры для объективной оценки качества его подготовки;

– “проблема мультиязычности”: большой объем экспертных знаний и ручного труда на проверку и коррекцию контента документов, представленных на разных языках, с целью их доведения до уровня соответствия государственным или корпоративным требованиям;

– “проблема затрат”: значительные затраты времени и средств на доведение качества документов до уровня, достаточного для последующей автоматизированной пакетной обработки;

– “проблема неполноты”: значительные потери времени на анализ полноты присланных материалов, семантически являющихся частью документа.

– “проблема безопасности”: большие финансовые, материальные и моральные потери в случае несанкционированного изменения контента документа.

Обобщая вышесказанное, можно утверждать, что решение указанных проблем существенно увеличит скорость преобразования исходных raw-документов в защищенные гетерогенные fit-документы, обеспечивающие синтез качественных управленческих решений. Ниже сформулирована формальная постановка задачи, ориентированная на комплексное решение вышеуказанных проблем.

1.2 Постановка задачи

Для постановки задачи обобщим типичные ситуации, характерные для современных организаций различного типа, включая научные и образовательные учреждения, издательства, рекрутские компании.

Пусть имеется организация, которая принимает решения на основе анализа входного потока цифровых документов $D1, D2, \ldots, Dk$, формируется авторами $A1, A2, \ldots, Ak$ (людьми или ИИ типа GPT-4). Соответственно, сцена работы с документами включает авторов документа и ЛПР, который оценивает и возможно, корректирует документ и отправляет его автору по каналу связи ch1. Документы носят различный характер и структуру (научные отчеты, докладные, диссертации, дипломы, резюме, анкеты и т.д.). Качество подготовки контента входных документов неизвестно. Документы принимаются ЛПР и отправляются авторам по одному каналу связи. Результат обработки документов ЛПР отправляется авторам по одному каналу связи.

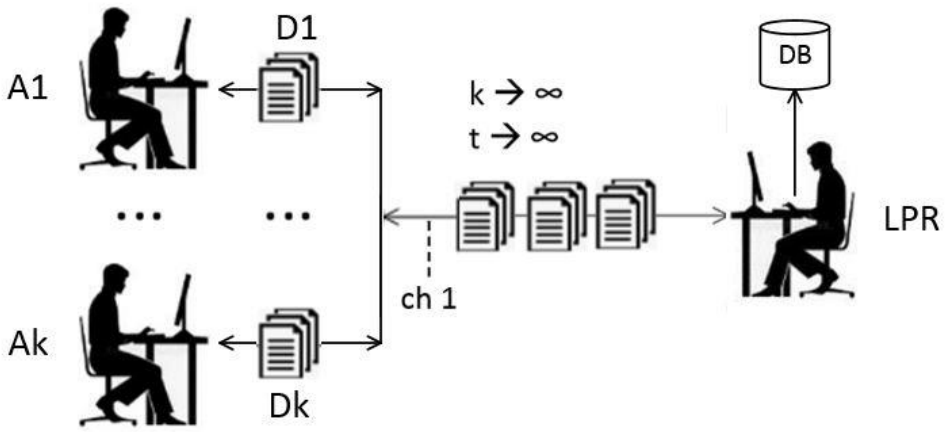

Рисунок 1.2 – Одноканальная незащищенная схема обработки документов

Дана традиционная одноканальная незащищенная схема обработки документов, ориентированная на ручной труд:

$$\text{Comp} = (\text{sys}, \text{LPR}, \text{Auth}, \text{D}, \text{V}_\text{D}, \text{t}_\text{D}, \text{z}_\text{D}), \quad \text{V} \rightarrow \infty, t \rightarrow \infty, z \rightarrow \infty, \tag{1.1}$$

где sys – система; LPR – лицо, оценивающее и корректирующее документ; Auth – автор документа; D – документ; V_D – объем документа; t_D – время обработки документа; z_D – затраты на обработку документа.

Для ускорения понимания ЛПР правильности структуры, лексики, наличия средств защиты и передачи документа в систему автоматизированной обработки компании к документам предъявляются определенные требования корпоративного и государственного уровня. Соответствующая схема представлена на рисунке 1.3.

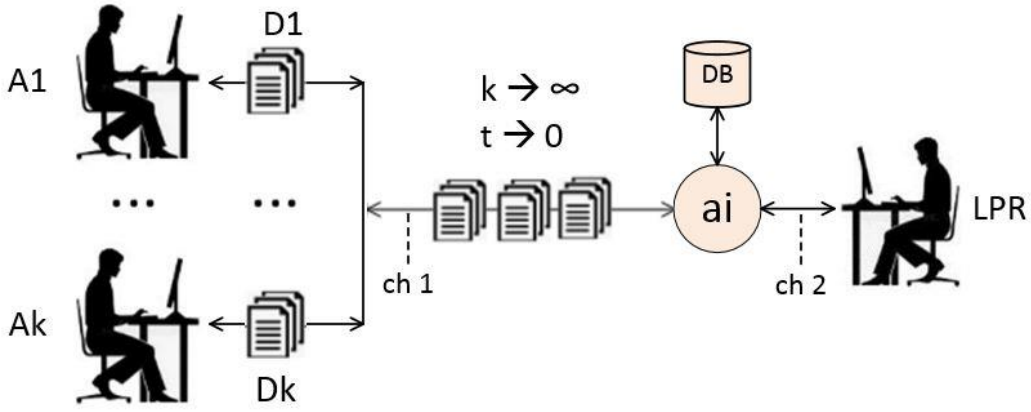

Рисунок 1.3 – Двухканальная защищенная схема обработки документов

Требуется разработать систему, обеспечивающую значительное сокращение времени и объема ручного труда на процессы преобразования входного документа неизвестного качества в документ, соответствующий государственным или корпоративным требованиям:

$$\mathrm{Comp} = (\mathrm{sysAI}, \mathrm{LPR}, \mathrm{Auth}, \mathrm{D}, \mathrm{V_D}, \mathrm{t_D}, \mathrm{z_D}), \quad \mathrm{V} \to 0, t \to 0, z \to 0, \tag{1.2}$$

где sysAI – ai-система; LPR – лицо, оценивающее и корректирующее документ; Auth – автор документа; D – документ; $\mathrm{V_D}$ – объем документа; $\mathrm{t_D}$ – время обработки документа; $\mathrm{z_D}$ – затраты на обработку документа.

Для прозрачности решения код программы должен содержать минимальное количество операторов при максимальном функциональном содержании.

Рассмотрим основные положения, которые будем использовать в качестве фундаментальной базы для решения.

В настоящее время основной подход к решению такого рода задач заключается в разработке специализированных систем ИИ, состоящих из группы микросервисов, которые обеспечивают локальную модернизацию.

Решение задачи состоит из теоретической и практической частей. В первой части необходимо разработать унифицированную адаптивную модель цифрового документа и релевантные алгоритмы для ее применения. Адаптация относится не только к документам, но и к СУБД для хранения моделей документов различного типа (т.е. модель должна обрабатываться любой современной СУБД). В второй части требуется разработать наиболее эффективную архитектуру для автоматизации применения модели и алгоритмов для пользователей с минимальным уровнем подготовки.

В целом, решение должно представлять собой замкнутый цикл: от получения документа до отправки автору защищенного результата обработки, чего не делает и одна из рассмотренных выше систем. В этом случае появится возможность обработки сотен документов за несколько минут.

Решение поставленной задачи базируется на идее делегирования некоторых интеллектуальных функций и компетенций человека искусственному

интеллекту. Решение этого вопроса позволит создать программную систему, дополняющую функционал многих систем, в частности, SAS Text Miner.

1.3 Выводы

В первой главе получены следующие основные результаты:

– на основе анализа литературных источников и анкетирования персонала из различных организаций, работающих с документами, определено, что при работе с документами основное внимание уделяется статистическому и лингвистическому анализу контента. Трудоемкие проблемы, касающиеся поиска унифицированной модели документа и автоматизации процессов улучшения качества контента, находятся на начальной стадии решения;

– сформулирована общая постановка задачи усовершенствования процессов обработки цифровых документов на основе автоматизации с целью значительного сокращения объема ручного труда, достоверной оценки качества документов, синтеза релевантных рекомендаций для их улучшения, обеспечения автоматической коррекции нарушений различного типа, приводящих к ложному восприятию как человека, так и программами обработки смысла документов;

– показано, что решение включает теоретическую и прикладную части. Теоретическая часть должна включать построение унифицированной модели документа, ориентированной на широкий спектр языков, алгоритмы ее применения для повышения эффективности работы с документами, включая их защиту. Практическая часть включает разработку архитектуры, обеспечивающей автоматизацию работы с документами на основе теоретических результатов, и соответствующее интероперабельное программное обеспечение.

ГЛАВА 2. РАЗРАБОТКА МОДЕЛЕЙ И АЛГОРИТМОВ

2.1 Понятийный базис

В литературе поставленная в гл.2 задача рассматривается с различных точек зрения с использованием часто противоречивой терминологии, что порождает полисемию, т.е. различное понимание одних и тех же терминов. Для решения проблемы полисемии необходимы некоторые базовые утверждения, отражающие эволюцию документов в различных типах организаций в XXI веке. Фактологической базой для утверждений являются документы и их свойства, относящиеся к инновационным направлениям, включая Bitcoin, потоки текстов в цифровом офисе, обеспечение информационной безопасности и других жизненно важных тенденций, прямо влияющих на качество принимаемых на различных уровнях управления решений. На основе анализа указанных особенностей современных документов можно сформулировать следующие базовые утверждения.

1. Современный документы носят гетерогенный характер, т.е. кроме традиционного текста они включают компоненты различного типа, обеспечивающие более полное восприятие контента.

2. Количество цифровых документов в государственных и частных компаниях постоянно увеличивается, что приводит к замедлению анализа контента и запаздыванию принимаемых управленческих решений (BigData).

3. Эффективным средством для ускорения принятия решений и повышения их качества являются программные системы, обеспечивающие объективный оперативный анализ входных документов.

Сформулированная в гл.1 задача исследования носит междисциплинарный характер, поэтому для ее решения с учетом представленных выше утверждений необходимо сформировать однозначно понимаемый понятийный базис, сглаживающий разногласия в трактовке терминов различными дисциплинами. Предлагаются определения, ориентированные на практическое применение в качестве базиса для разработки целевой системы.

Автор – создатель документа.

ЛПР – лицо, оценивающее качество документа и принимающее соответствующее решение.

Эксперт – лицо, знающее свойства различных типов документов и фиксирующее все происходящие с ними изменения.

Цифровой документ – гетерогенная структура, состоящая из текста и других компонентов, соответствующих определенным стандартам.

Тип цифрового документа – лингвистическая константа однозначно определяющий характер документа (диплом, курсовая работа и т.д.) и соответствие определенному классу задач, где документ используется (написание диплома, курсовой и т.д.).

Входной формат – тип представление входного документа, представленный в одном из стандартных форматов (txt, docx, pdf,).

Источник – человек или ИИ, формирующий цифровой документ с помощью определенной программы.

Контрольные точки – элементы текста, однозначно характеризующие тип, назначение, подлинность документа. Значения контрольных точек соответствуют типу документа, формируются экспертом и могут подразделяться на специфические группы, например: обязательные, желательные, нежелательные, устаревшие и другие.

Нормализованный формат – результат преобразования входного формата в унифицированный цифровой формат, удобный для математической обработки.

Нормализованный цифровой документ – документ, соответствующий цели его составления и требованиям к оформлению.

Слово состояния документа – лингвистическая переменная, определяющая текущую степень соответствия документа типу и целям.

Нормализация – преобразование входных документов различного типа (docx, pdf, txt) в нормализованный тип, удобный для обработки.

Предметная область (ПрО) – комплекс моделей, алгоритмов, программ, данных, достаточный для решения определенной задачи.

Корпоративные задачи – задачи обработки текста, актуальные для подразделений организации.

Частные задачи – задачи обработки текста, актуальные для отдельных сотрудников.

Адаптивная система – система, которая может настраиваться экспертом на анализ, оценивание, корректировку документов различного типа (научные статьи, отчеты, дипломы и др.) в рамках решения корпоративных и частных задач в условиях частых изменений требований к ним без изменения программного кода.

Данный комплекс является базой для создания концептуальных моделей, описывающих решение поставленной задачи в рамках системного подхода, до уровня программного кода. что позволяет решать поставленную задачу по принципу последовательной детализации концепций до программного кода.

2.2 Модели

2.2.1 Концептуальная модель сцены работы с документом

Как показано выше, поставленная задача носит междисциплинарный характер. Для такого рода задач обычно применяют системный подход, т.е. строят концептуальную модель, которая формирует цельный взгляд на проблему, и уточняют ее в рамках методологий различных дисциплин до уровня программного кода.

Обобщая реальные ситуации, можно утверждать, что на верхнем уровне должна находиться модель сцены, включающей всех одушевленных и неодушевленных акторов. К ним относятся: автор документа (Auth), эксперт (Exp), ЛПР (LPR), документ (doc), искомая система (sys) и связывающие их коммуникации (com). Будем считать, что эксперт знает все особенности и требования, предъявляемые к различным типам документам. Представим соответствующую сцену кортежем:

$$Scene = (Auth, Exp, LPR, doc, sys, com) \quad (2.1)$$

где Auth – автор документа; Exp – эксперт, знающий “горячие” требования к документу; LPR – лицо, оценивающее и корректирующее документ doc; sys – искомая система; com – коммуникации для общения участников.

2.2.2 Концептуальные модели участников сцены

Для практической реализации сцены необходимо утонить компоненты сцены (состав и отношения участников):

$$Auth = (idAuth, doc, grdE<V, U>, com) \quad (2.2)$$

где idAuth – идентификатор автора; doc – созданный автором документ; grdE <V,U> – оценка V документа экспертом или системой и соответствующее управляющее решение U, com – коммуникации для диалога с автором.

$$Exp = (idExp, knExp, com) \quad (2.3)$$

где idExp – идентификатор эксперта внешнего эксперта; knExp – актуальные знания эксперта о конкретных типах документов; com – коммуникации для диалога с автором.

$$LPR = (idLpr, knLpr, com) \quad (2.4)$$

где idLpr – лицо, оценивающее и корректирующее документ в организации; KnLpr – знания лица; com – коммуникации для доступа к ЛПР.

$$doc = (idDoc, req, K1,K2,…,Kn, V, U) \quad (2.5)$$

где idDoc – идентификатор документа определенного типа; req – требования; K1,K2,…,Kn – компоненты документа определенного типа.

Модель целевой системы:

$$sys = (ai, lib, knMOD, dbIN, dbOUT, com) \quad (2.6)$$

где ai – управляющая система; lib – библиотека программных модулей, реализующих функции обработки текста; knMOD – база данных с моделями различных типов документов; dbIN – база данных входных документов; dbOUT – база данных выходных документов, com – коммуникации в рамках системы.

В условиях быстрых изменений технологий, увеличения количества угроз и новых средств защиты документов классификация компонентов K в [2.5] является достаточно сложной задачей.

На основе анализа документов в структурах blockchain, системах документооборота частных и государственных организаций, включая университеты, автором выделены следующие компоненты.

2.2.3 Концептуальная модель сцены работы с документом

Целевая система применяется в рамках некоторой сцены, участники которой представлены в следующем кортеже:

$$mScene = (user, doc, sys, com) \quad (2.7)$$

где user – пользователь системы, работающий с документами; doc – документы; sys – система для работы с документами; com – коммуникации.

Реализация сцены возможна как в локальном, так и в сетевом варианте.

2.2.4 Концептуальная модель документа

Модель документа основана на представленных в п.2.1 определениях и обобщает различные типы документов, используемых в образовательных и бизнес-структурах:

$$mD = (id, type, cont, k1,k2,\ldots kn, e1,e2,\ldots,ek, V, U, dsw) \quad (2.8)$$

где id – идентификатор документа; type – тип документа; cont – контент документа; k1,k2,…kn – ключевые точки; e1,e2,…,ek – внешние документы, дополняющие документ; V – оценка документа; U – управляющее решение, соответствующее оценке; dsw – слово состояния документа.

Модель (2.8) носит концептуальный характер, для ее практической реализации необходима конкретизация каждого параметра. Для этого требуется машинно-ориентированная цифровая структура, допускающая формирование, хранение, коррекцию и эффективную пересылку по сети. В качестве такой структуры предлагается использовать формат JSON. Соответствующий вариант представлен на рисунке 2.1.

data_file.json > ...

```json
{
    "ModDip": {
        "docType": "диплом",
        "docComplex": "диплом, рецензия, программа, презентация",
        "docId": "D03",
        "dateModel": "01.01.2024",
        "auth": "Иванов Иван Иванович",
        "structWords": "Введение, Глава 1, Глава 2, Глава 3, Глава 4, Заключение, Список литературы",
        "badWords": "рис:рисунок, таб:таблица",
        "docVol": "50",
        "minSymbols": "25000",
        "minWords": "7000",
        "secSymbols": "***",
        "w": "хорошее, среднее, плохое",
        "u": "допустить в работу, локальная автоматическая коррекция, вернуть автору",
        "uCrit": "0, 5, 10"
    }
}
```

Рисунок 2.1– JSON-модель документа

Аналогичная модель формируется для каждого типа документов, которые подвергаются контролю. Для применения модели необходима компьютерная система, модель которой рассмотрена ниже.

2.2.5 Концептуальная модель целевой системы

Модель целевой системы состоит из управляющей системы и ресурсов моделей, алгоритмов, входных и выходных документов:

$$mSys = (ai, lib, dbMOD, dbIN, dbOUT, com) \quad (2.9)$$

где ai – управляющая система; lib – библиотека программных модулей, управляемая ai; dbMOD – база данных с моделями различных типов документов; dbIN – база данных входных документов; dbOUT – база данных выходных документов.

Библиотека mod может расширяться по мере появления новых типов документов и новых задач их обработки.

2.3 Алгоритмы

2.3.1. Алгоритм построения модели документа

Алгоритм построения модели документа представлен ниже.

```
def submit():
    global docType, docComplex, docId, dateModel, auth, structWords, badWords,
docVol, minSymbols, minWords, secSymbols, w, u, uCrit

    docType = docType_entry.get()
    docComplex = docComplex_entry.get()
    docId = docId_entry.get()
    dateModel = dateModel_entry.get()
    auth = auth_entry.get()

    structWords = structWords_entry.get()
    badWords = badWords_entry.get()

    docVol = int(docVol_entry.get())
    minSymbols = int(minSymbols_entry.get())
    minWords = int(minWords_entry.get())
    secSymbols = secSymbols_entry.get()

    w = w_entry.get()
    u = u_entry.get()
    uCrit = uCrit_entry.get()
```

Модели документов различного типа формирует и постоянно корректирует эксперт. Модель строится следующим образом: экспертом вносятся данные о документе определенного типа в специальную форму, на основе введенных данных формируется JSON-модель документа.

2.3.2. Алгоритм чтения и первичного анализа документа

Ниже представлен алгоритм чтения и первичного анализа документа.

```
global documentPath, text, isDoc
documentPath = filedialog.askopenfilename(filetypes=[("DOCX Files",
"*.docx")])
if documentPath:
    import os
    if os.path.isfile(documentPath):
        import docx2txt
        text = docx2txt.process(documentPath)
        isDoc = 1
        messagebox.showinfo("Успешно", f"Загрузка документа: {documentPath}")
    else:
        messagebox.showwarning("Внимание!", "Файл не найден.")
else:
    messagebox.showwarning("Внимание!", "Путь к документу не введен.")
```

Чтение входного документа выполняется автоматически и начинается анализ контента. Проверяется, существует ли файл.

2.3.3. Алгоритм анализа контента

Алгоритм анализа контента представлен ниже.

```
# Highlighting bad words and their replacements
pairs = d['ModDip']['badWords'].split(',')
words = []

for i in range (len(pairs)):
words.append(pairs[i].split(':'))

# Input document analysis
k = 0
j = 0
for i in range (len(words)):
    if words[i][0] in text:
        k = k+1
        badZam.append(words[i][0])
        goodZam.append(words[i][1])
        j = j+1
```

Анализ документа является обязательным для современных аналитических систем. Из введенной экспертом модели выделяются некорректные слова и их замены. Входной документ проверяется на наличие некорректных слов, запоминаются все вхождения в паре со словом-заменой.

Доступен также статистический анализ текста документа.

```
import spacy
from collections import Counter

nlp = spacy.load('en_core_web_sm')
doc = nlp(text)

sentences = list(doc.sents)
word_count = len(doc)
unique_words = len(set([token.text for token in doc if not token.is_punct]))

word_frequencies = Counter([token.text.lower() for token in doc if not
token.is_punct])

text_st_an = text
punc = '''!()--[]{};:'"\,<>./?@#$%^&*_~1234567890'''
for symb in text:
    if symb in punc:
        text_st_an = text_st_an.replace(symb, "")
```

```
    amSymbols = len(text_st_an)
    text1 = tk.Label(new_window7, text=f"Количество символов: {amSymbols}",
anchor="w")
    text1.pack(padx=(20, 0), pady=(0, 5), anchor="w")

    amWords = word_count
    text1 = tk.Label(new_window7, text=f"Количество слов: {amWords}", anchor="w")
    text1.pack(padx=(20, 0), pady=(0, 5), anchor="w")

    unAmWords = unique_words
    text1 = tk.Label(new_window7, text=f"Количество уникальных слов: {unAmWords}",
anchor="w")
    text1.pack(padx=(20, 0), pady=(0, 5), anchor="w")

    amSent = len(sentences)
    text1 = tk.Label(new_window7, text=f"Количество предложений: {amSent}",
anchor="w")
    text1.pack(padx=(20, 0), pady=(0, 5), anchor="w")

    text2 = tk.Label(new_window7, text="Прочие результаты см. в разделе
ВИЗУАЛИЗАЦИЯ РЕЗУЛЬТАТА", anchor="w")
    text2.pack(padx=(20, 0), pady=(0, 5), anchor="w")

    isStat = 1
```

В тексте подсчитывается количество символов, слов, количество уникальных слов, количество предложений.

2.3.4. Алгоритм оценивания документа.

Алгоритм оценивания документа представлен ниже.

```
# Highlighting document quality assessments and corresponding management
decisions
marks = d['ModDip']['w'].split(',')
decisions = d['ModDip']['u'].split(',')

cr = [int(x.strip()) for x in d['ModDip']['uCrit'].split(',')]

if (cr[0] < k < cr[1]):
text01 = tk.Label(new_window3, text=f"Оценка качества документа: {marks[0]}",
anchor="w")
text01.pack(padx=(20, 0), pady=(0, 5), anchor="w")

text02 = tk.Label(new_window3, text=f"Управляющее решение: {decisions[0]}",
anchor="w")
text02.pack(padx=(20, 0), pady=(0, 10), anchor="w")

if (cr[1] < k < cr[2]):
```

```
text11 = tk.Label(new_window3, text=f"Оценка качества документа: {marks[1]}",
anchor="w")
text11.pack(padx=(20, 0), pady=(0, 5), anchor="w")

text12 = tk.Label(new_window3, text=f"Управляющее решение: {decisions[1]}",
anchor="w")
text12.pack(padx=(20, 0), pady=(0, 10), anchor="w")

if (k > cr[2]):
text21 = tk.Label(new_window3, text=f"Оценка качества документа: {marks[2]}",
anchor="w")
text21.pack(padx=(20, 0), pady=(0, 5), anchor="w")

text22 = tk.Label(new_window3, text=f"Управляющее решение: {decisions[2]}",
anchor="w")
text22.pack(padx=(20, 0), pady=(0, 10), anchor="w")
```

Автоматизация оценивания качества контента необходима для передачи функций ЛПР (преподавателя) системе для многократного ускорения получения качественного объективного результата. Оценивание качества контента производится на основе введенных экспертом критериев принятия решения в модель документа.

2.3.5. Алгоритм коррекции контента

Ниже представлен алгоритм коррекции контента.

```
def perform_correction():
    global text, badZam, goodZam

    X = text
    for i in range (len(badZam)):
        X = X.replace(badZam[i], goodZam[i])

    messagebox.showinfo("Подтверждено!", "Успешно")

bad_zam_text = tk.Label(new_window4, text=f"Некорректные элементы: {badZam}",
anchor="w")
bad_zam_text.pack(padx=(20, 0), pady=(0, 10), anchor="w")

good_zam_text = tk.Label(new_window4, text=f"Предлагается заменить на:
{goodZam}", anchor="w")
good_zam_text.pack(padx=(20, 0), pady=(0, 10), anchor="w")

correction_button = tk.Button(new_window4, text="Корректировать текст",
command=perform_correction)
correction_button.pack(padx=(20, 0), pady=(0, 10))
```

При коррекции контента некорректные элементы исправляются на соответствующие им замены. Автоматизация коррекции контента необходима для передачи функций ЛПР (преподавателя) системе для многократного ускорения получения качественного документа из сырого (raw) входного текста.

2.3.6. Алгоритм визуализации результатов

Алгоритм визуализации результатов представлен ниже.

```
    minSymbols = int(d['ModDip']['minSymbols'])
    minWords = int(d['ModDip']['minWords'])

    # Symbol count visualization
    if (amSymbols < minSymbols):
        ost = minSymbols - amSymbols
        label1 = 'Недостающий объем:\n' + str(ost) + ' символов'
        label2 = 'В тексте:\n' + str(amSymbols) + ' символов'
        labels=[label1, label2]
        values=[ost, amSymbols]
        colors=["#aac3e3","#386cb0"]
        explode =[0,0.1]
        plt.pie(values, labels=labels, colors=colors, explode=explode,
autopct='%1.1f%%')
        plt.title('Количество символов. \nМинимальный необходимый объем: ' +
str(minSymbols) + ' символов')
        plt.axis('equal')
        plt.show()

    else:
        label = 'В тексте:\n' + str(amSymbols) + ' символов'
        labels=[label]
        values=[amSymbols]
        colors=["#386cb0"]
        plt.pie(values, labels=labels, colors=colors, autopct='%1.1f%%')
        plt.title('Количество символов. \nМинимальный необходимый объем: ' +
str(minSymbols) + ' символов')
        plt.axis('equal')
        plt.show()

    # Word count visualization
    if (amWords < minWords):
        ost = minWords - amWords
        label1 = 'Недостающий объем:\n' + str(ost) + ' слов'
        label2 = 'В тексте:\n' + str(amWords) + ' слов'
        labels=[label1, label2]
        values=[ost, amWords]
        colors=["#aac3e3","#386cb0"]
        explode =[0,0.1]
```

```
        plt.pie(values, labels=labels, colors=colors, explode=explode,
autopct='%1.1f%%')
        plt.title('Количество слов. \nМинимальный необходимый объем: ' +
str(minWords) + ' слов')
        plt.axis('equal')
        plt.show()

    else:
        label = 'В тексте:\n' + str(amWords) + ' слов'
        labels=[label]
        values=[amWords]
        colors=["#386cb0"]
        plt.pie(values, labels=labels, colors=colors, autopct='%1.1f%%')
        plt.title('Количество слов. \nМинимальный необходимый объем: ' +
str(minWords) + ' слов')
        plt.axis('equal')
        plt.show()


    def count_freq(my_list):
        unique_words = []
        counts = []

        # Create a list of unique words (excluding punctuation):
        for item in my_list:
            # Check if the item is not punctuation
            if item not in string.punctuation:
                if item not in unique_words:
                    unique_words.append(item)

        # Count the frequency of each word (excluding punctuation):
        for word in unique_words:
            count = 0
            for i in my_list:
                # Check if the word is not punctuation
                if i not in string.punctuation:
                    if word == i:
                        count += 1
                        counts.append(count)

        # Create a dataframe with the words and their frequencies:
        df = pd.DataFrame({"word": unique_words, "count": counts})
        df.sort_values(by="count", inplace = True, ascending = False)
        df.reset_index(drop=True, inplace = True)
        return df

    text_an = text
    punc = '''!()--[]{};:'"\,<>./?@#$%^&*_~1234567890'''
    for symb in text:
        if symb in punc:
```

```
        text_an = text_an.replace(symb, "")

text_df = count_freq(text_an.split())

# Word count visualization
text_top10 = text_df.iloc[:10]

fig, ax1 = plt.subplots()

ax1.set_facecolor("yellow")

ax1.barh(text_top10["word"], text_top10["count"],
        color = "#386cb0",
        edgecolor = "#2b548a")

ax1.set_title("Частотный анализ текста")

for ax in fig.axes:
    ax.invert_yaxis()
    ax.grid(False)
    ax.title.set_color('black')
    ax.tick_params(axis='x', colors='black')
    ax.tick_params(axis='y', colors='black')
    ax.set_facecolor('#ffffff')
    ax.spines["top"].set_visible(False)
    ax.spines["right"].set_visible(False)
    ax.spines["left"].set_visible(False)

for i in ax.patches:
    plt.text(i.get_width()+0.2, i.get_y()+0.5,
        str(round((i.get_width()), 2)),
        fontsize=10, fontweight='bold',
        color='grey')

fig.patch.set_facecolor('#ffffff')
plt.grid(which='major')
fig.tight_layout()

plt.show()
```

На экран выводятся диаграммы, отражающие, насколько обрабатываемый текст соответствует указанным требованиям, покрывает ли объем введенного текста минимальный необходимый объем. Также выводятся результаты частотного анализа текста.

2.3.7 Внешние сервисы

2.3.7.1 Синтез надежных паролей

Средство синтеза надежных паролей является важнейшим элементом системы кибербезопасности компании. Ниже представлен вариант генератора, обеспечивающий синтез пароля с заданным количеством символов.

```
# Password generation service
print ('Сервис генерации пароля')
import string
import random
def generate_password(length):
    characters = string.ascii_letters + string.digits + string.punctuation
    password = ''.join(random.choice(characters) for _ in range(length))
    return password
n = int(input("Длина пароля: "))
parol = generate_password(n)
print("Ваш пароль:", parol)
print ('Сервис генерации пароля отработал')
```

После генерации ключа часто возникает необходимость зашифровать. Количество заданных символов не должно быть меньше восьми.

2.3.7.2 Шифрование документов

Сервис надежного шифрования является необходимым элементом защиты информации при организации безопасного документооборота. В частности, он в обязательном порядке используется при пересылке документов по глобальной сети, где существует вероятность несанкционированного доступа к документу. Ниже представлен код сервиса для шифрования любого типа документов.

```
# Encryption service
print('Активен сервис шифрования.')
from cryptography.fernet import Fernet
# Key generation
key = Fernet.generate_key()
with open('mykey.key', 'wb') as mykey:
     mykey.write(key)
with open('mykey.key', 'rb') as mykey:
     key=mykey.read()
# Encryption
f=Fernet(key)
with open('t1.txt', 'rb') as original_file:
   original=original_file.read()
encrypted = f.encrypt(original)
```

```
with open('t2.txt', 'wb') as encrypted_file:
    encrypted_file.write(encrypted)
# Decryption
f=Fernet(key)
with open('t2.txt', 'rb') as encrypted_file:
    encrypted = encrypted_file.read()
decrypted = f.decrypt(encrypted)
with open('t3.txt', 'wb') as decrypted_file:
    decrypted_file.write(decrypted)
print('Сервис шифрования отработал.')
```

Сервис обеспечивает шифрование документов любого типа.

2.3.7.3 Хэширование документов

Сервис хэширования является необходимым элементом защиты информации при организации безопасного документооборота.

```
# Hashing service
import hashlib
print ('Активен сервис хэширования')
def hash_file(filename):
   h = hashlib.sha1()
   with open(filename,'rb') as file:
       chunk = 0
       while chunk != b'':
           chunk = file.read(512)
           h.update(chunk)
   return h.hexdigest()
fname = input('Введите имя файла с документом:\n')
hashmes = hash_file(fname)
print('Хэш документа ', fname, '=', hashmes)
print ('Сервис хэширования отработал.')
```

Любое нарушение значения хэша является признаком несанкционированного изменения контента пересылаемых документов.

2.3.7.4 Отправка результата автору или утверждение документа ЛПР

Отправка результата коррекции и оценивания автору является обязательной операцией организации отношений автора и ЛПР. Чем быстрее автор получит объективно оцененный результат, тем быстрее наступит решение с ним связанных проблем автора и ЛПР.

Разработан следующий комплекс алгоритмов, ориентированный на реализацию в форме интероперабельных микросервисов:

– алгоритм формирования модели документа и построения соответствующей базы знаний на основе формализованных экспертных знаний;

– алгоритм чтения входного документа, определения его типа, оценки его качества и синтеза соответствующего управляющего решения на основе базы знаний;

– алгоритм коррекции документа в соответствии с управляющим решением;

– алгоритм визуализации результатов обработки документа для корректора;

– алгоритм записи откорректированного документа в выходную базу данных и отправка автору;

– служебные алгоритмы для автоматизации различных методов анализа документа для широкого спектра задач (статистический анализ, сентимент анализ и т.д.);

– алгоритм управления сервисами системы.

2.4 Выводы

Во второй главе получены следующие основные результаты:

– сформулированы базовые утверждения, отражающие результаты эволюции цифровых документов в 2000-2024 гг.;

– сформирован понятийный базис, соответствующий результатам изменений, включающий основные термины и определения, с целью их однозначного понимания и использования в качестве базы;

– разработана унифицированная модель современного гетерогенного документа, допускающая расширение при появление новых задач;

– разработаны алгоритмы построения модели документа любого типа, оперативного анализа контента, оценки качества документа, автоматической коррекции с целью доведения до корпоративных стандартов. В целом, комплекс алгоритмов выполняет работу высококвалифицированного специалиста.

ГЛАВА 3. РАЗРАБОТКА ПРОГРАММНОГО ИНСТРУМЕНТАРИЯ

3.1 Архитектура целевой системы

Разработка любой программной системы предполагает выполнение важного предварительный этап – построение архитектуры.

Архитектура программной системы представляет собой схему из конечного множества программных модулей и отношений между ними посредством информационных потоков для реализации общей концептуальной схемы (в данном случае она описана моделями (2.1)-(2.8)).

Архитектура необходима для построения целостного взгляда на программное решение задачи и распределение составляющих ее подзадач задач участникам группы разработки.

Ниже представлен вариант архитектуры (рисунок 3.1) для системы, обеспечивающей применение разработанных в гл. 2 моделей и алгоритмов для автоматизированного решения проблем, сформулированных в гл.1.

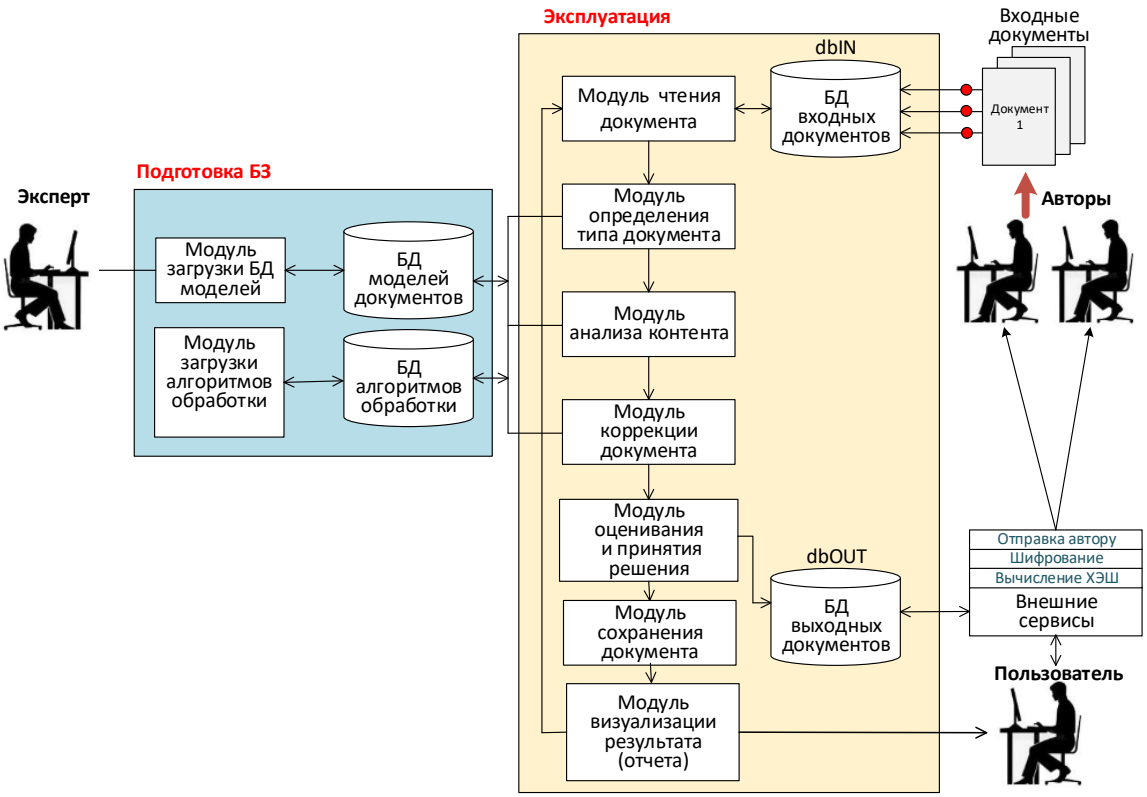

Рисунок 3.1 – Архитектура целевой системы

Данный вариант архитектуры предполагает наличие пять акторов: Эксперта (для создания модели документов), Автора (создателей документов). ЛПР (лиц, оценивающих поток входных документов и принимающих решение по их дальнейшей обработке) и документ. В ряде случаев эксперта заменяет ЛПР, что не противоречит принципу обратной связи, но повышает уровень субъективности при оценивании документа.

Фактически архитектура реализует принцип обратной связи между автором и ДПР с использованием знаний эксперта.

Для реализации архитектуры будем использовать микросервисный подход и язык высокого уровня Python 3.12. Причина такого выбора – большое количество внешних библиотек, позволяющих значительно сократить количество операторов в коде программ и обеспечить однозначное понимание смысла кода. Главное требование к коду – краткость, т.е. минимально возможное количество операторов, достаточных для реализации определенной подзадачи.

3.2 Реализация архитектуры на основе микросервисного подхода

3.2.1 Управляющая программа

Управляющая программа служит для вызова сервисов, обеспечивающих доступ к сервисам, реализующим доступ к функциональности системы.

Реализован через интерактивные элементы на странице главного меню. Нажатие кнопки влечёт за собой вызов соответствующего миркросервиса.

Разработаны следующие микросервисы:

- сервис построения модели документа определенного типа;
- сервис чтения входного документа;
- сервис анализа контента;
- сервис оценивания контента;
- сервис коррекции контента;
- сервис визуализации результатов;

– внешние сервисы (синтез надежных паролей, шифрование документов, хэширование документов, отправка результата автору или утверждение документа ЛПР).

3.3 Методика использования системы

Для решения практических задач на основе разработанного ПО предлагается методика, включающая две части: подготовительную и эксплуатационную.

Подготовительная часть предназначена для эксперта, который формирует предметную область для работы с документами и несет ответственность за ее правильность и актуальность (шаги 1-5).

Эксплуатационная часть включает сервисы для обработки документов с целью приведения их к виду, соответствующего стандартам (шаги 6-11).

В целом методика состоит из одиннадцати шагов. Фактически шаги 1-11 автоматизируют выполнение весьма трудоемких операций, выполняемых профессионально подготовленным специалистами.

Технология применения системы может быть описана последовательно в 11 шагах.

Шаг 1. Получение нового типа документа от администрации

Шаг 2. Анализ требований

Шаг 3. Поиск эксперта

Шаг 4. Построение JSON-модели предметной области задачи.

Шаг 5. Сохранение модели в базе

Шаг 6. Получение документа от автора

Шаг 7. Анализ документа

Шаг 8. Оценивание

Шаг 9. Коррекция

Шаг 10. Отправка автору или в базу

Шаг 11. Возврат на Шаг 6.

3.4 Выводы

В третьей главе получены следующие основные результаты:

– разработана архитектура системы для автоматизации процессов работы с документами;

– разработана библиотека программных модулей для реализации архитектуры на языке Python, программные компоненты которой обеспечивают построение JSON-модели предметной области для различных типов документов, анализа и обработки контента совместимой с СУБД MySQL, MongoDB;

– предложена методика применения системы для решения прикладных задач, актуальных для современных организаций.

ГЛАВА 4. АПРОБАЦИЯ РАЗРАБОТАННОЙ СИСТЕМЫ

4.1 Постановка прикладной задачи

Для апробации разработанного ПО необходимо решить прикладную задачу. В качестве таковой возьмем актуальную для ВУЗов проблему сокращения времени, которое преподаватели тратят на первом этапе проверки качества дипломных работ выпускников, включая оценку их соответствия требованиям ВУЗа и государственным стандартам.

Требования к документам, включая элементы структуры, форматирование, международную терминологию, быстро изменяются в процессе информатизации и компьютеризации. Дипломы могут быть представлены на белорусском, русском или английском языке в формате txt или docx. Среднее время проверки и коррекции преподавателем (экспертом) одной страницы в среднем занимает 1 минуту. Средний объем диплома – 50 страниц. Всего проверка одного диплома занимает 40 минут (2400 секунд)

Требуется разработать систему, обеспечивающую автоматическое распознавание типа, языка документа, автоматический анализ, коррекцию, формирование модифицированного документа и оценить эффективность, т.е. сокращение затрат времени по сравнению с человеком.

Для построения предметной области будем использовать стандарты, рекомендованные ВАК Республики Беларусь.

4.2 Решение прикладной задачи на основе технологии

Для решения будем использовать методику, предложенную в гл.3.

Шаг 1. Получение нового типа документа от администрации.

Шаг 2. Анализ требований.

Шаг 3. Поиск эксперта.

Старт Главного меню (рисунок 4.1.).

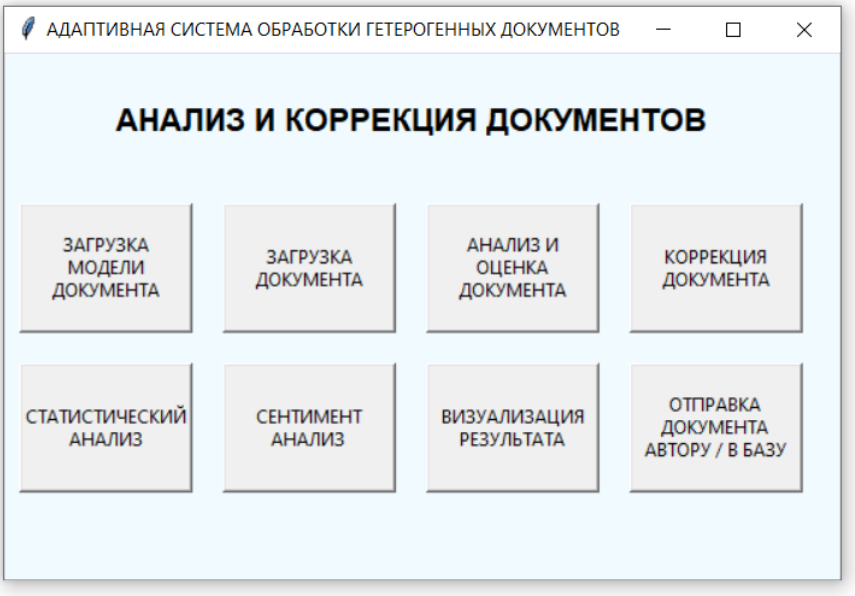

Рисунок 4.1 – Главное меню системы

Система предлагает выбрать для запуска один из микросервисов создания модели или обработки документа. После выбора одного из предложенных вариантов происходит запуск соответствующего микросервиса.

Шаг 4. Построение JSON-модели документа определенного типа (например, диплома).

На данном этапе эксперт строит модель определенного документа в интерактивном режиме (Рисунок 4.2).

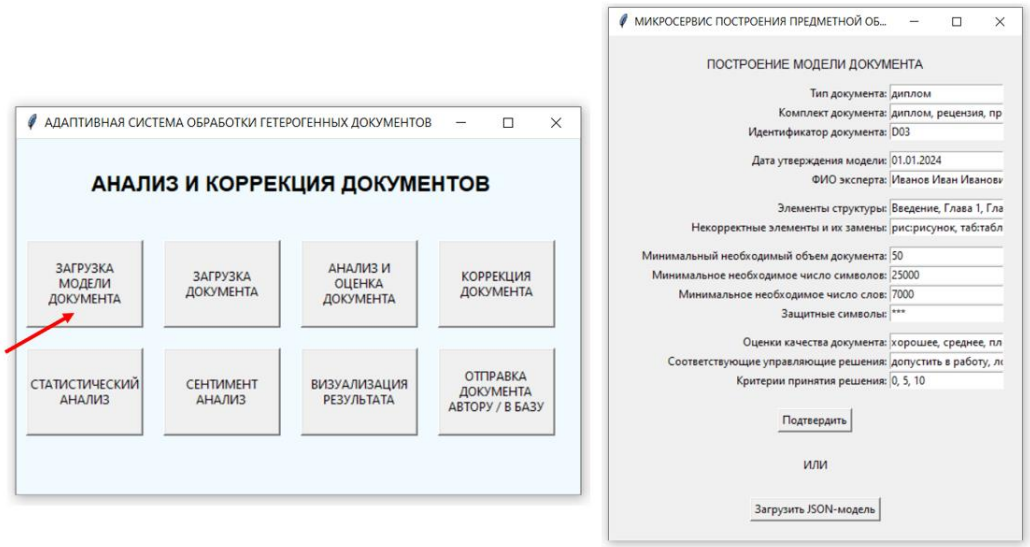

Рисунок 4.2 – Построение модели документа

Эксперт заполняет все поля формы для создания модели, либо загружает JSON-файл уже созданной ранее модели.

Шаг 5. Сохранение модели в базе.

После окончания ввода параметров микросервис формирует JSON-модель данного типа документа (рисунки 4.3, 4.4).

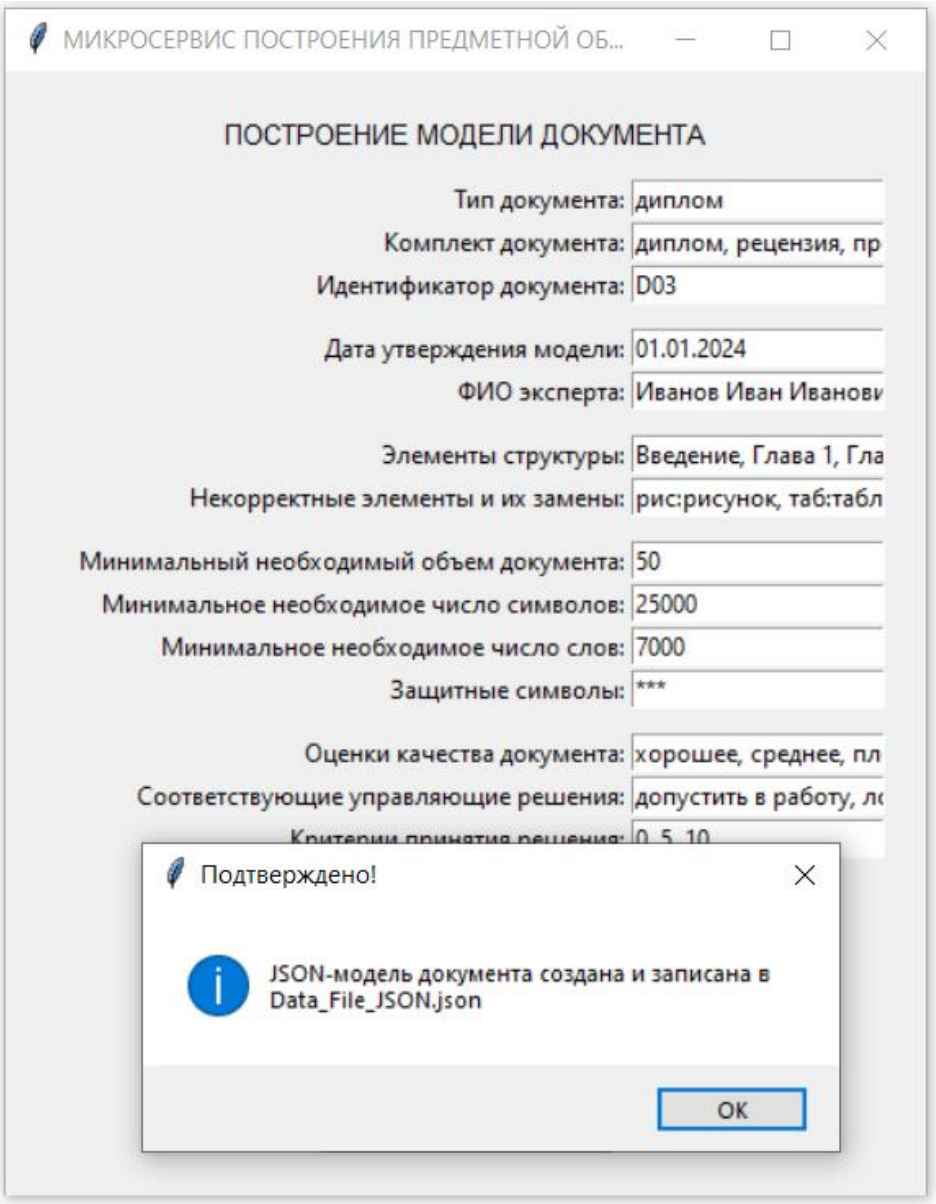

Рисунок 4.3 – Формирование JSON-модель документа

Система подтверждает, что JSON-модель документа сформирована, записана в файл с именем Data_File_JSON.json.

data_file.json > ...

```json
{
    "ModDip": {
        "docType": "диплом",
        "docComplex": "диплом, рецензия, программа, презентация",
        "docId": "D03",
        "dateModel": "01.01.2024",
        "auth": "Иванов Иван Иванович",
        "structWords": "Введение, Глава 1, Глава 2, Глава 3, Глава 4, Заключение, Список литературы",
        "badWords": "рис:рисунок, таб:таблица",
        "docVol": "50",
        "minSymbols": "25000",
        "minWords": "7000",
        "secSymbols": "***",
        "w": "хорошее, среднее, плохое",
        "u": "допустить в работу, локальная автоматическая коррекция, вернуть автору",
        "uCrit": "0, 5, 10"
    }
}
```

Рисунок 4.4 – Сформированная JSON-модель документа

В соответствии с созданной моделью документа будет произведен дальнейший анализ.

Шаг 6. Получение документа от автора

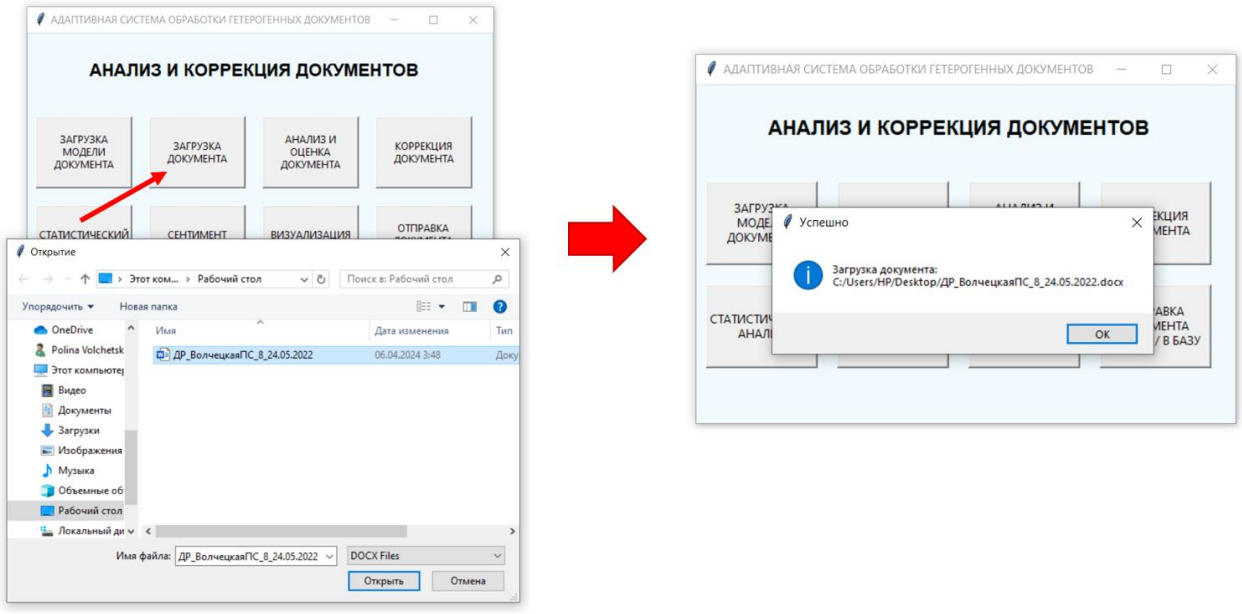

Рисунок 4.5 – Чтение входного документа

Загружается документ для обработки.

Шаг 7. Анализ документа

Аналитическая обработка документа. Оценка качества документа (принятие решения) исходя из статистики ошибок (рисунок 4.6).

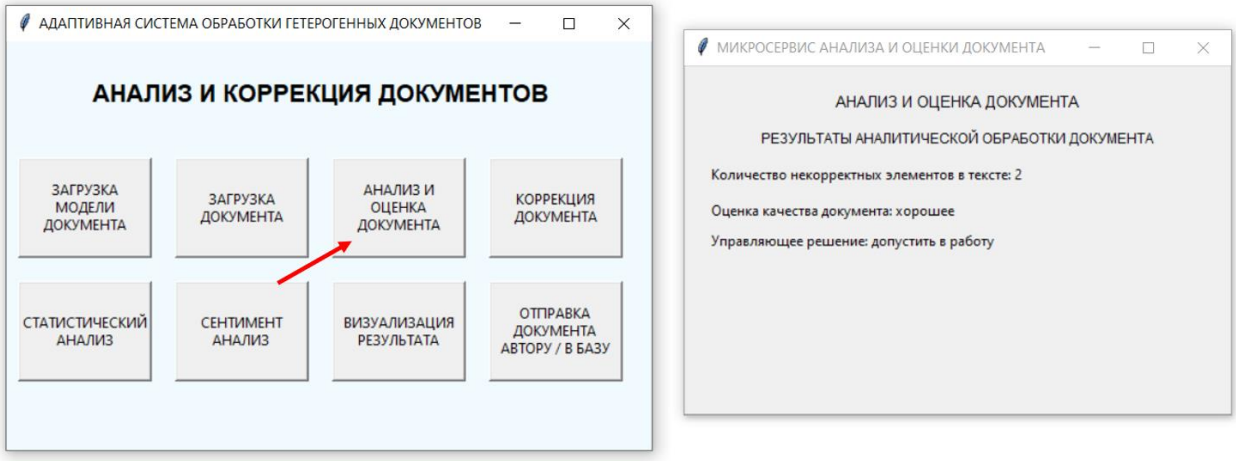

Рисунок 4.6 – Результаты аналитической обработки текста документа

На экран выводятся результат анализа текста документа, оценка качества документа, управляющее решение.

Статистический анализ документа.

Система производит статистический анализ текста документа (рисунок 4.7).

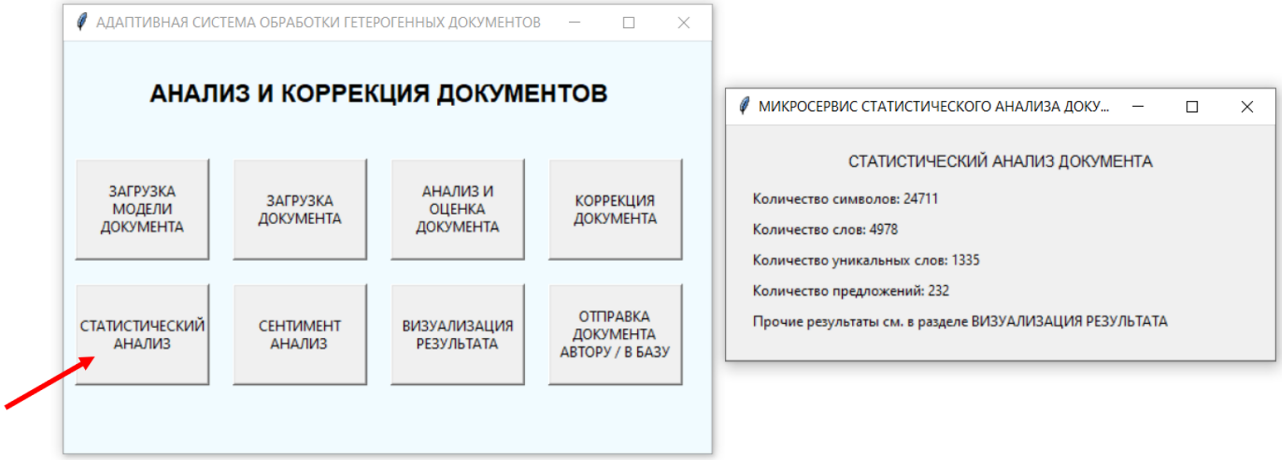

Рисунок 4.7 – Запуск микросервиса статистического анализа текста

Результаты статистического анализа текста выводятся на экран (рисунок 4.7, рисунок 4.9, рисунок 4.10).

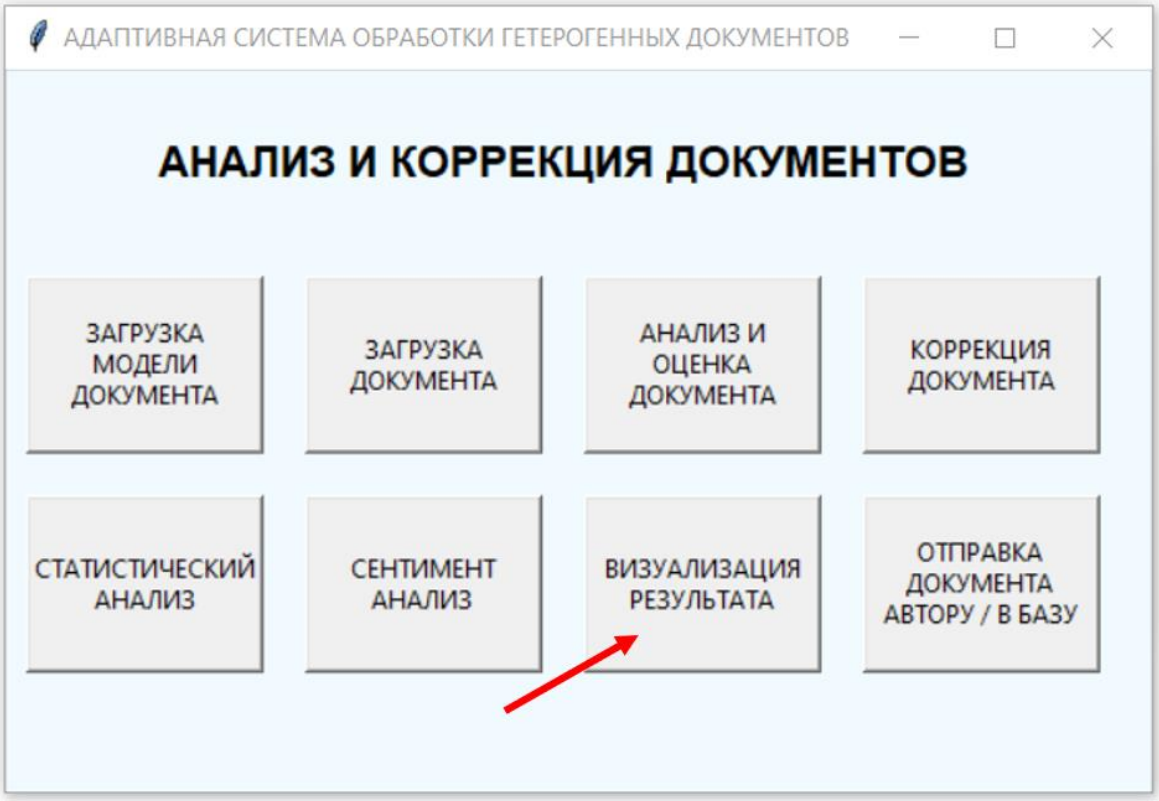

Рисунок 4.8 – Запуск микросервиса визуализации результата статистического анализа текста

После нажатия кнопки «Визуализация результатов» отображаются следующие результаты:

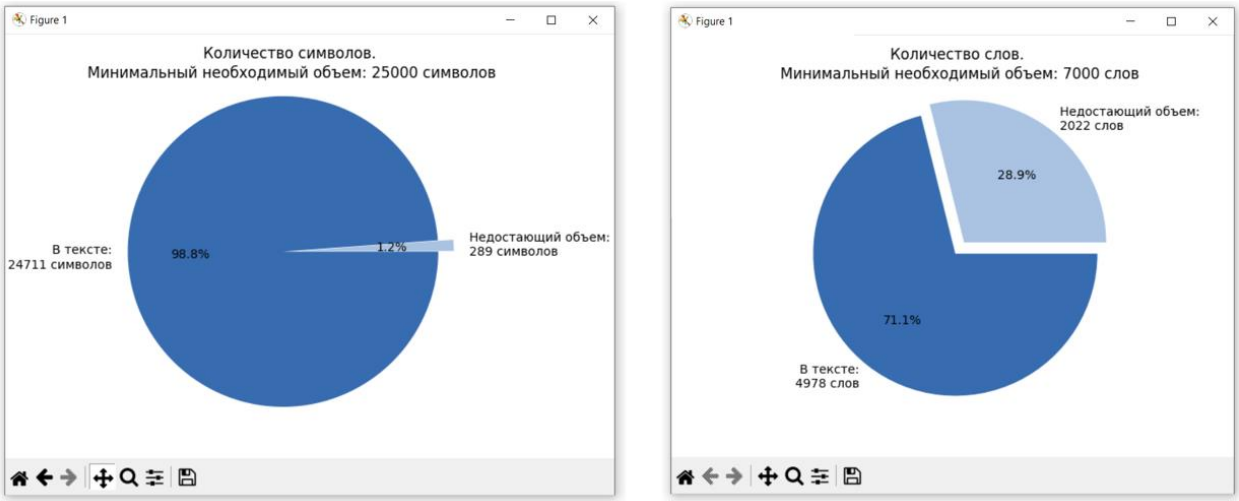

Рисунок 4.9 – Результаты статистического анализа текста

Таким образом, приведенный в примере текст не соответствует минимальному необходимому объему. Необходимо дополнить текст, чтобы он содержал более 25 000 символов и более 7000 слов.

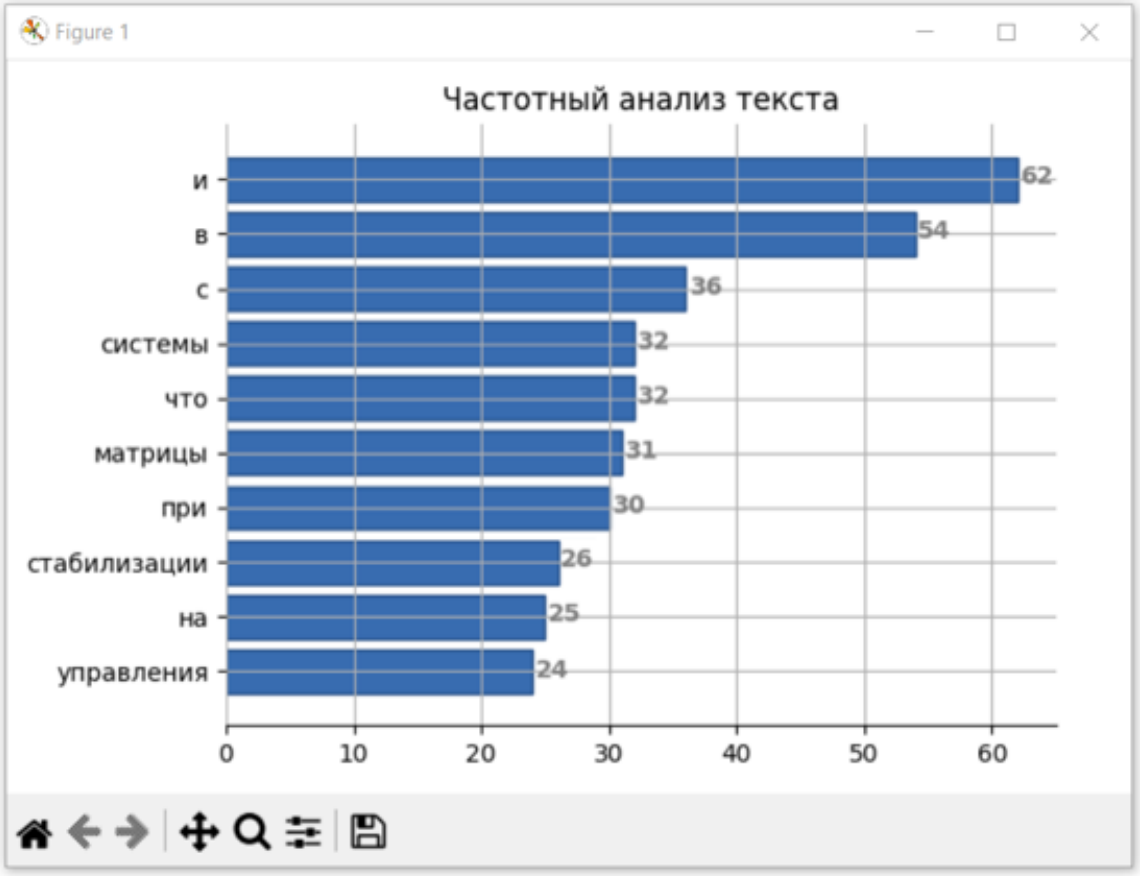

Рисунок 4.10 – Диаграмма частотного анализа текста

Диаграмма частотного анализа текста демонстрирует наиболее часто встречающиеся слова в документе.

Шаг 9. Коррекция

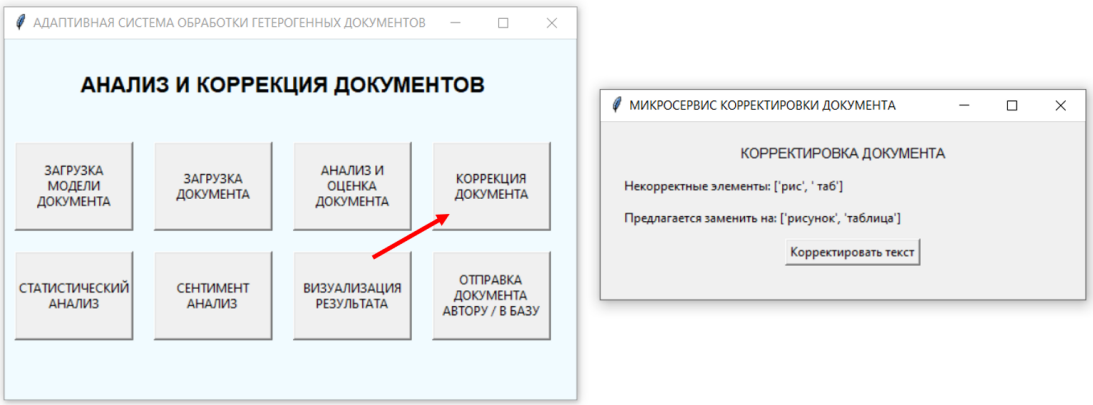

Рисунок 4.11 – Результаты аналитической обработки текста документа

Система автоматически производит улучшение качества документа, согласно построенной экспертом модели документа.

Шаг 10. Отправка автору или в базу

После автоматической коррекции и вывода откорректированного текста на экран предлагается записать исправленный текст в файл и отправить автору (рисунок 4.12).

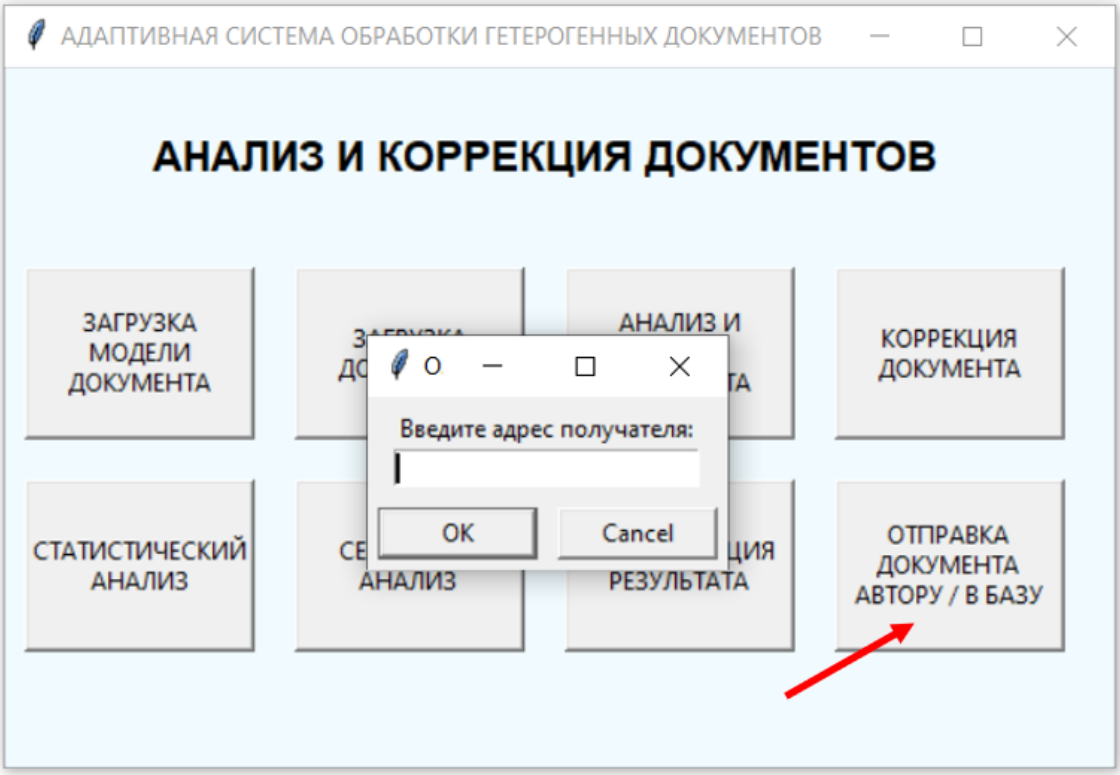

Рисунок 4.12 – Сохранение результатов в БД

После завершения ввода документ отправляется по указанному адресу.

Этап 8. Шифрование, хэширование, отправка.

Таким образом, документ можно считать защищенным.

Апробация системы

Качество разработанного ПО определим на примере решения следующей практической задачи. Дано: 3 дипломные работы объемом 50 страниц в среднем каждая, поступивших корректору на предмет оценки их соответствия требованиям организации.

Затраты времени на проверку в общем достаточно велики (таблица 1), т.к. количество документов достигает нескольких сотен в год. Объем документов

достаточно велик (50-150 страниц), поэтому проверка осуществляется, как правило, в два этапа. При этом 100% соответствия стандартам не гарантируется.

Таблица 4.1 – Затраты времени на проверку

	Курсовая	Диплом	Магистерская	Кандидатская
Затраты времени на проверку 1(мин)	20	25	30	40
Затраты времени на проверку 2(мин)	15	15	25	30
Всего:	35	40	55	70

В результате измерений определено, что на проверку одного диплома корректор тратит в среднем 40 минут. Всего затраты времени составили 119 минут (7140 секунд).

Требуется провести сравнительный анализ качества работы и затрат времени на проверку дипломов контролером и системой. В систему экспертом предварительно загружена модель диплома DP2024.

Затраты времени на проверку с использованием автоматизированной системы составляют ~35 секунд (20 секунд – создание модели документа экспертом; 5 секунд – статистический анализ документа; 10 секунд – корректировка документа). В то же время затраты времени на проверку диплома без использования автоматизированной системы составляют 40 минут.

Система затратила на решение 98 секунд, что в 73 раза быстрее корректора.

Таким образом, автоматизированная система позволяет сократить количественные затраты времени в ~70 раз.

4.3 Выводы

На основании анализа проблемы и результата автоматизации можно утверждать, что в условиях изменчивости структуры и требований к документам адаптивные системы являются основным средством реагирования на турбулентность среды и сокращения затрат. Применение экспертных знаний для адаптации необходимо. В целом, такого рода системы можно рассматривать как дополнение адаптивными сервисами крупных аналитических систем, аналогичных SAS.

В четвертой главе получены следующие основные результаты:

– сформулирована прикладная задача, актуальная для сокращения затрат времени на первый этап проверки качества дипломных работ;

– выполнено решение прикладной задачи на основе разработанной методики;

– выполнено сравнение результатов оценки документа разработанной системой и преподавателем. Качественные результаты проверки системы и преподавателя совпали на 100%. Количественно затраты времени в результате автоматизации совращены, в среднем, в 73 раза.

ЗАКЛЮЧЕНИЕ

В результате написания магистерской диссертации были получены следующие основные результаты:

Показано, что постоянный рост количества и объема входных документов в глобальной динамической среде приводит к увеличению ручного труда и снижению качества их проверки и падению эффективности принимаемых на их основе решений.

Показано, что структура и состав документов постоянно изменяются в процессе адаптации к изменениям и требованиям среды, что требует постоянной переподготовки персонала и увеличивает время проверки.

Известная система SAS не обеспечивают улучшение качества динамической обработки документов.

Для комплексного решения указанных проблем разработана универсальная модель документа, которая обладает свойствами адаптации к изменениям особенностей документов и автоматической коррекции на основе новых требований.

Для применения модели разработан комплекс алгоритмов, обеспечивающих создание модели документа, его анализ, коррекцию, оценку, шифрование и отправку документов.

Для практического применения разработанных моделей и алгоритмов разработана архитектура и соответствующий программный инструментарий.

Для апробации программного инструментария была решена практическая задачи и выполнена оценка эффективности процесса обработки.

В результате апробации системы показано, что скорость обработки повышена в 73 раза в сравнении с ручным трудом.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Кузнецов, И.Н. Документационное обеспечение управления / И.Н. Кузнецов – М.: Юрайт, 2015. – 611 с.

2. Виссия, Х., Краснопрошин В.В., Вальвачев А.Н. Принятие решений в информационном обществе. – СПб: ЛАНЬ, 2019. – 227 с

3. Лейн, Х. Обработка естественного языка в действии / Х.Лейн, Х.Хапке, К.Ховард – СПб.: Питер, 2020. – 576 с.

4. О порядке оформления диссертации, диссертации в виде научного доклада, автореферата диссертации и публикаций по теме диссертации [Электронный ресурс] – Режим доступа: Полное руководство по анализу текста [Электронный ресурс] – Режим доступа: https://asu-analitika.ru/analys-text. Дата доступа: 20.08.2023.

5. Плас, Д. Python для сложных задач: наука о данных и машинное обучении / Д.Плас – Питер, 2017. – 576 с.

6. Полное руководство по анализу текста [Электронный ресурс] – Режим доступа: https://asu-analitika.ru/analys-text. Дата доступа: 20.06.2023.

7. Проект ГОСТ Р эталонная модель цифрового документооборота организации [Электронный ресурс] – Режим доступа: https://www.normacs.ru/ Doclist/ doc/235BA.html. Дата доступа: 20.06.2023.

8. Электронные документы. ГОСТ Р 7.0.95-2015. Режим доступа: https: //www.ifap.ru/library/gost/70952015.pdf. Дата доступа: 20.02.2023.

9. Allen-Robertson, J. Critically assessing digital documents: materiality and the interpretative role of software / J.Allen-Robertson //Information, Communication & Society. – 2018. – № 21 (4). – P. 1732-1746.

10. Anandarajan, M. Practical Text Analytics: Maximizing the Value of Text Data / M.Anandarajan, C.Hill, T.Nolan – Springer, 2018. – 455 p.

11. Bengfort, B. Applied Text Analysis with Python: Enabling Language-Aware Data Products with Machine Learning/B.Bengfort –O'Reilly, 2018. – 334 p.

12. Boreus, K. Analyzing Text and Discourse: Eight Approaches for the Social Sciences / K.Boreus, G.Bergstrom – SAGE, 2017. – 304 p.

13. Buckland, M. What is a "digital document?" / M.Buckland //Document Numérique (Paris). – 1998. – № 2. – P. 221-230.

14. Buckland, M. Document Theory: An Introduction / M.Buckland // Conference and School on Records, Archives and Memory Studies,Zadar, 2013. – P. 223-237.

15. Chakraborty, G. Text Mining and Analysis: Practical Methods, Examples, and Case Studies Using SAS / G.Chakraborty, M.Pagolu, S.Garla – SAS Institute, 2013. – 340 p.

16. Digital Document Management: What Are the Best Technologies? Электронный ресурс. – Режим доступа: https://www.sydle.com/blog/digital-document-management-62309cc21c92a326f30454f5. Дата доступа: 05.03.2023.

17. Global economic effects of COVID-19. Congressional research service, 2021. – 110 p.

18. How to Use Chat GPT for Scientific Research Paper writing? [Электронный ресурс] – Режим доступа: https://plainenglish.io/blog/how-to-use-chat-gpt-for-scientific-research-paper-writing /Дата доступа: 20.06.2023.

19. Jade, T. SAS Text Analytics for Business Applications: Concept Rules for Information Extraction Models /T.Jade – SAS Institute, 2019. – 308 p.

20. GPT-4 Technical Report. [Электронный ресурс] – Режим доступа: https://arxiv.org/abs/2303.08774 /Дата доступа: 20.06.2023.

21. JSON vs XML in 2021. [Электронный ресурс]. – Режим доступа: https://hackr.io/blog/json-vs-xml. Дата доступа: 05.03.2023.

22. Khurana, D. Natural language processing: state of the art, current trends and challenges / D.Khurana et al. // Multimedia Tools and Applications – 2023. – Vol.82. – P. 3713–3744.

23. Natural Language Processing Crash Course for Beginners: Theory and Applications of NLP using TensorFlow 2.0 and Keras – AI Publishing LLC, 2020. – 396 p.

24. SAS Visual Text Analitics [Электронный ресурс] – Режим доступа: https://www.sas.com/en_id/software/visual-text-analytics.html. Дата доступа: 20.10.2023.

25. Scifleet, P. Practice theory & the foundation of digital document encoding /P.Scifleet, S.Williams // Proceedings of the 27th ACM international conference on Design of communication – P.213-220.

26. Seven challenges in digital document workflow management [Электронный ресурс] – Режим доступа: https://www.sas.com/en_id/software/visual-text-analytics.html. /Дата доступа: 20.09.2023.

27. Talafidaryani, M. A text mining-based review of the literature on dynamic capabilities perspective in information systems research / M.Talafidaryani //Management Research Review. – 2021. – Vol. 44. – N.2. – P.236-267.

28. Vasquez, C. Research Methods for Digital Discourse Analysis /C. Vasquez – Bloomsbury Academic, 2022. – 352 p.

ПРИЛОЖЕНИЕ А

Листинг программы на языке Python

```
import sys
import time
import tkinter as tk
from tkinter import font
from tkinter import messagebox
from tkinter import simpledialog
from tkinter import filedialog

# Declare global variables to store the entered values
d = ""

docType = ""        # Document Type
docComplex = ""     # Document complex
docId = ""          # Document ID

dateModel = ""      # Model approval date
auth = ""           # Expert's name

structWords = ""    # Structure Words
badWords = ""       # Bad words and their replacements

docVol = ""         # Min doc volume
minSymbols = ""     # Min symbols
minWords = ""       # Min words
secSymbols = ""     # Security symbols

w = ""              # Document quality ratings W
u = ""              # Corresponding control solutions
uCrit = ""          # Decision criteria


documentPath = ""
text = ""

badZam = []
goodZam = []

amSymbols = 0
amWords = 0

isMod = -1
isDoc = -1
isStat = -1


def open_window1():
```

```
    global d, docType, docComplex, docId, dateModel, auth, structWords, badWords,
docVol, minSymbols, minWords, secSymbols, w, u, uCrit, isMod, isDoc
    new_window1 = tk.Toplevel(window)
    new_window1.title("МИКРОСЕРВИС ПОСТРОЕНИЯ ПРЕДМЕТНОЙ ОБЛАСТИ ДЛЯ ДОКУМЕНТОВ
ОПРЕДЕЛЕННОГО ТИПА")
    new_window1.geometry("450x550")

    title_label_font = font.Font(size=10)
    title_label = tk.Label(new_window1, text="ПОСТРОЕНИЕ МОДЕЛИ ДОКУМЕНТА",
font=title_label_font)
    title_label.pack(pady=(20, 10))

    def create_input_label(label_text):
        frame = tk.Frame(new_window1)
        frame.pack(anchor="e", padx=(0, 20))
        label = tk.Label(frame, text=label_text)
        label.pack(side=tk.LEFT)
        entry = tk.Entry(frame)
        entry.pack(side=tk.LEFT)
        return entry

    def create_padding(height):
        padding_frame = tk.Frame(new_window1, height=height)
        padding_frame.pack()

    docType_entry = create_input_label("Тип документа:")
    docComplex_entry = create_input_label("Комплект документа:")
    docId_entry = create_input_label("Идентификатор документа:")
    create_padding(10)

    dateModel_entry = create_input_label("Дата утверждения модели:")
    auth_entry = create_input_label("ФИО эксперта:")
    create_padding(10)

    structWords_entry = create_input_label("Элементы структуры:")
    badWords_entry = create_input_label("Некорректные элементы и их замены:")
    create_padding(10)

    docVol_entry = create_input_label("Минимальный необходимый объем документа:")
    minSymbols_entry = create_input_label("Минимальное необходимое число
символов:")
    minWords_entry = create_input_label("Минимальное необходимое число слов:")
    secSymbols_entry = create_input_label("Защитные символы:")
    create_padding(10)

    w_entry = create_input_label("Оценки качества документа:")
    u_entry = create_input_label("Соответствующие управляющие решения:")
    uCrit_entry = create_input_label("Критерии принятия решения:")
```

```
    def submit():
        global d, docType, docComplex, docId, dateModel, auth, structWords,
badWords, docVol, minSymbols, minWords, secSymbols, w, u, uCrit, isMod

        docType = docType_entry.get()
        docComplex = docComplex_entry.get()
        docId = docId_entry.get()

        dateModel = dateModel_entry.get()
        auth = auth_entry.get()

        structWords = structWords_entry.get()
        badWords = badWords_entry.get()

        docVol = int(docVol_entry.get())
        minSymbols = int(minSymbols_entry.get())
        minWords = int(minWords_entry.get())
        secSymbols = secSymbols_entry.get()

        w = w_entry.get()
        u = u_entry.get()
        uCrit = uCrit_entry.get()


        # Write to DataBase.txt
        z = (docType, docComplex, docId, dateModel, auth, structWords, badWords,
docVol, minSymbols, minWords, secSymbols, w, u, uCrit)
        with open('DataBase.txt', 'w') as file:
            print(*z, file=file, sep='\n')
        print('Модель документа создана и записана в DataBase.txt')

        # JSON
        import json
        jsonMod = {
            "ModDip" : {
                "docType": docType,
                "docComplex": docComplex,
                "docId": docId,
                "dateModel": dateModel,
                "auth": auth,
                "structWords": structWords,
                "badWords": badWords,
                "docVol": docVol,
                "minSymbols": minSymbols,
                "minWords": minWords,
                "secSymbols": secSymbols,
                "w": w,
                "u": u,
                "uCrit": uCrit
            }
```

```
        }

        with open("Data_File_JSON.json", "w", encoding="utf8") as write_file:
            json.dump(jsonMod, write_file, ensure_ascii=True)
        print('JSON-модель документа создана и записана в Data_File_JSON.json\n')
        with open("Data_File_JSON.json", "r") as jsonMod:
            d = json.load(jsonMod)
        print('JSON-модель документа Data_File_JSON.json прочитана:\n')
        print(d)

        if any(value == "" for value in [docType, docComplex, docId, dateModel,
auth, structWords, badWords, docVol, minSymbols, minWords, secSymbols, w, u,
uCrit]): # Check if any input is empty
            messagebox.showwarning("Ошибка!", "Не все поля заполнены.")
        else:
            messagebox.showinfo("Подтверждено!", "JSON-модель документа создана и
записана в Data_File_JSON.json")
            isMod = 1
            new_window1.destroy()

    def open_file_dialog():
        global d, isMod
        file_path = filedialog.askopenfilename(filetypes=[("JSON Files",
"*.json")])
        if file_path:
            print("Selected File:", file_path)
            import json
            with open(file_path, "r") as jsonMod:
                d = json.load(jsonMod)
            print('JSON-модель документа прочитана:\n')
            print(d)
            isMod = 1
            messagebox.showinfo("Подтверждено!", "JSON-модель документа прочитана")
            new_window1.destroy()

    submit_button = tk.Button(new_window1, text="Подтвердить", command=submit)
    submit_button.pack(pady=(20, 25))

    title_label = tk.Label(new_window1, text="ИЛИ")
    title_label.pack()

    # Create a button to open the file dialog
    load_button = tk.Button(new_window1, text="Загрузить JSON-модель",
command=open_file_dialog)
    load_button.pack(pady=(25, 0))


def open_window2():
    global documentPath, text, isDoc
```

```
    documentPath = filedialog.askopenfilename(filetypes=[("DOCX Files", "*.docx")])
    if documentPath:
        import os
        if os.path.isfile(documentPath):
            import docx2txt
            text = docx2txt.process(documentPath)
            print('\n    -----Исходный текст:-----\n')
            print(text)
            isDoc = 1
            messagebox.showinfo("Успешно", f"Загрузка документа: {documentPath}")
        else:
            messagebox.showwarning("Внимание!", "Файл не найден.")

    else:
        messagebox.showwarning("Внимание!", "Путь к документу не введен.")


def open_window3():
    global d, documentPath, text, badZam, goodZam, isMod, isDoc

    if (isMod == -1):
            messagebox.showwarning("Ошибка!", "Не загружена модель документа.")
    if (isDoc == -1):
            messagebox.showwarning("Ошибка!", "Не загружен документ.")

    else:
        new_window3 = tk.Toplevel(window)
        new_window3.title("МИКРОСЕРВИС АНАЛИЗА И ОЦЕНКИ ДОКУМЕНТА")
        new_window3.geometry("470x300")

        title_label_font = font.Font(size=10)
        title_label = tk.Label(new_window3, text="АНАЛИЗ И ОЦЕНКА ДОКУМЕНТА",
font=title_label_font)
        title_label.pack(pady=(20, 10))

        # Highlighting bad words and their replacements
        pairs = d['ModDip']['badWords'].split(',')
        words = []

        for i in range (len(pairs)):
            words.append(pairs[i].split(':'))

        # Highlighting document quality assessments and corresponding management
decisions
        marks = d['ModDip']['w'].split(',')
        decisions = d['ModDip']['u'].split(',')

        # Input document analysis
        k = 0
        j = 0
```

```
        for i in range (len(words)):
            if words[i][0] in text:
                k = k+1
                badZam.append(words[i][0])
                goodZam.append(words[i][1])
                j = j+1

        title_label_font = font.Font(size=9)
        title_label = tk.Label(new_window3, text="РЕЗУЛЬТАТЫ АНАЛИТИЧЕСКОЙ
ОБРАБОТКИ ДОКУМЕНТА", font=title_label_font)
        title_label.pack(pady=(0, 10))

        main_text = tk.Label(new_window3, text=f"Количество некорректных элементов
в тексте: {k}", anchor="w")
        main_text.pack(padx=(20, 0), pady=(0, 10), anchor="w")

        cr = [int(x.strip()) for x in d['ModDip']['uCrit'].split(',')]
        print(cr[0])
        print(cr[1])
        print(cr[2])
        if (cr[0] < k < cr[1]):
            text01 = tk.Label(new_window3, text=f"Оценка качества документа:
{marks[0]}", anchor="w")
            text01.pack(padx=(20, 0), pady=(0, 5), anchor="w")

            text02 = tk.Label(new_window3, text=f"Управляющее решение:
{decisions[0]}", anchor="w")
            text02.pack(padx=(20, 0), pady=(0, 10), anchor="w")


        if (cr[1] < k < cr[2]):
            text11 = tk.Label(new_window3, text=f"Оценка качества документа:
{marks[1]}", anchor="w")
            text11.pack(padx=(20, 0), pady=(0, 5), anchor="w")

            text12 = tk.Label(new_window3, text=f"Управляющее решение:
{decisions[1]}", anchor="w")
            text12.pack(padx=(20, 0), pady=(0, 10), anchor="w")


        if (k > cr[2]):
            text21 = tk.Label(new_window3, text=f"Оценка качества документа:
{marks[2]}", anchor="w")
            text21.pack(padx=(20, 0), pady=(0, 5), anchor="w")

            text22 = tk.Label(new_window3, text=f"Управляющее решение:
{decisions[2]}", anchor="w")
            text22.pack(padx=(20, 0), pady=(0, 10), anchor="w")
```

```
def perform_correction():
    global text, badZam, goodZam

     X = text
         for i in range (len(badZam)):
             X = X.replace(badZam[i], goodZam[i])

    messagebox.showinfo("Подтверждено!", "Успешно")
    new_window4.destroy()


def open_window4():
    global d, documentPath, text, badZam, goodZam, isMod, isDoc

    if (isMod == -1):
            messagebox.showwarning("Ошибка!", "Не загружена модель документа.")
    if (isDoc == -1):
            messagebox.showwarning("Ошибка!", "Не загружен документ.")

    else:
        new_window4 = tk.Toplevel(window)
        new_window4.title("МИКРОСЕРВИС КОРРЕКТИРОВКИ ДОКУМЕНТА")
        new_window4.geometry("470x300")

        title_label_font = font.Font(size=10)
        title_label = tk.Label(new_window4, text="КОРРЕКТИРОВКА ДОКУМЕНТА",
font=title_label_font)
        title_label.pack(pady=(20, 10))


        bad_zam_text = tk.Label(new_window4, text=f"Некорректные элементы:
{badZam}", anchor="w")
        bad_zam_text.pack(padx=(20, 0), pady=(0, 10), anchor="w")

        good_zam_text = tk.Label(new_window4, text=f"Предлагается заменить на:
{goodZam}", anchor="w")
        good_zam_text.pack(padx=(20, 0), pady=(0, 10), anchor="w")


        correction_button = tk.Button(new_window4, text="Корректировать текст",
command=perform_correction)
        correction_button.pack(padx=(20, 0), pady=(0, 10))


def open_window7(): # 5
    global d, isDoc, documentPath, text, badZam, goodZam, amSymbols, amWords,
isStat

    if (isDoc == -1):
        messagebox.showwarning("Ошибка!", "Не загружен документ.")
```

```
    else:
        new_window7 = tk.Toplevel(window)
        new_window7.title("МИКРОСЕРВИС СТАТИСТИЧЕСКОГО АНАЛИЗА ДОКУМЕНТА")
        new_window7.geometry("470x200")

        title_label_font = font.Font(size=10)
        title_label = tk.Label(new_window7, text="СТАТИСТИЧЕСКИЙ АНАЛИЗ ДОКУМЕНТА",
font=title_label_font)
        title_label.pack(pady=(20, 10))


        import spacy
        from collections import Counter

        nlp = spacy.load('en_core_web_sm')

        doc = nlp(text)

        # Count the number of sentences, words, and unique words
        sentences = list(doc.sents)
        word_count = len(doc)
        unique_words = len(set([token.text for token in doc if not
token.is_punct]))

        # Perform word frequency analysis
        word_frequencies = Counter([token.text.lower() for token in doc if not
token.is_punct])

        text_st_an = text
        punc = '''!()--[]{};:'"\,<>./?@#$%^&*_~1234567890'''
        for symb in text:
            if symb in punc:
                text_st_an = text_st_an.replace(symb, "")

        amSymbols = len(text_st_an)
        text1 = tk.Label(new_window7, text=f"Количество символов: {amSymbols}",
anchor="w")
        text1.pack(padx=(20, 0), pady=(0, 5), anchor="w")

        amWords = word_count
        text1 = tk.Label(new_window7, text=f"Количество слов: {amWords}",
anchor="w")
        text1.pack(padx=(20, 0), pady=(0, 5), anchor="w")

        unAmWords = unique_words
        text1 = tk.Label(new_window7, text=f"Количество уникальных слов:
{unAmWords}", anchor="w")
        text1.pack(padx=(20, 0), pady=(0, 5), anchor="w")
```

```
        amSent = len(sentences)
        text1 = tk.Label(new_window7, text=f"Количество предложений: {amSent}",
anchor="w")
        text1.pack(padx=(20, 0), pady=(0, 5), anchor="w")

        text2 = tk.Label(new_window7, text="Прочие результаты см. в разделе
ВИЗУАЛИЗАЦИЯ РЕЗУЛЬТАТА", anchor="w")
        text2.pack(padx=(20, 0), pady=(0, 5), anchor="w")

        isStat = 1


def open_window5():
    global d, isDoc, documentPath, text, badZam, goodZam, amSymbols, amWords,
isStat
    import pandas as pd
    import matplotlib.pyplot as plt


    if (isDoc == -1):
        messagebox.showwarning("Ошибка!", "Не загружен документ.")

    elif (isStat == -1):
        messagebox.showwarning("Ошибка!", "Сперва проведите статистический анализ
документа.")

    elif (isMod == -1):
        messagebox.showwarning("Ошибка!", "Не загружена модель документа.")


    else:
        minSymbols = int(d['ModDip']['minSymbols'])
        minWords = int(d['ModDip']['minWords'])

        # Symbol count visualization
        if (amSymbols < minSymbols):
            ost = minSymbols - amSymbols
            label1 = 'Недостающий объем:\n' + str(ost) + ' символов'
            label2 = 'В тексте:\n' + str(amSymbols) + ' символов'
            labels=[label1, label2]
            values=[ost, amSymbols]
            colors=["#aac3e3","#386cb0"]
            explode =[0,0.1]
            plt.pie(values, labels=labels, colors=colors, explode=explode,
autopct='%1.1f%%')
            plt.title('Количество символов. \nМинимальный необходимый объем: ' +
str(minSymbols) + ' символов')
            plt.axis('equal')
            plt.show()
        else:
```

```
            label = 'В тексте:\n' + str(amSymbols) + ' символов'
            labels=[label]
            values=[amSymbols]
            colors=["#386cb0"]
            plt.pie(values, labels=labels, colors=colors, autopct='%1.1f%%')
            plt.title('Количество символов. \nМинимальный необходимый объем: ' +
str(minSymbols) + ' символов')
            plt.axis('equal')
            plt.show()

        # Word count visualization
        if (amWords < minWords):
            ost = minWords - amWords
            label1 = 'Недостающий объем:\n' + str(ost) + ' слов'
            label2 = 'В тексте:\n' + str(amWords) + ' слов'
            labels=[label1, label2]
            values=[ost, amWords]
            colors=["#aac3e3","#386cb0"]
            explode =[0,0.1]
            plt.pie(values, labels=labels, colors=colors, explode=explode,
autopct='%1.1f%%')
            plt.title('Количество слов. \nМинимальный необходимый объем: ' +
str(minWords) + ' слов')
            plt.axis('equal')
            plt.show()
        else:
            label = 'В тексте:\n' + str(amWords) + ' слов'
            labels=[label]
            values=[amWords]
            colors=["#386cb0"]
            plt.pie(values, labels=labels, colors=colors, autopct='%1.1f%%')
            plt.title('Количество слов. \nМинимальный необходимый объем: ' +
str(minWords) + ' слов')
            plt.axis('equal')
            plt.show()


        import string

        def count_freq(my_list):
            unique_words = []
            counts = []

            # Create a list of unique words (excluding punctuation):
            for item in my_list:
                # Check if the item is not punctuation
                if item not in string.punctuation:
                    if item not in unique_words:
                        unique_words.append(item)
```

```
    # Count the frequency of each word (excluding punctuation):
    for word in unique_words:
        count = 0
        for i in my_list:
            # Check if the word is not punctuation
            if i not in string.punctuation:
                if word == i:
                    count += 1
        counts.append(count)


    # Create a dataframe with the words and their frequencies:
    df = pd.DataFrame({"word": unique_words, "count": counts})
    df.sort_values(by="count", inplace = True, ascending = False)
    df.reset_index(drop=True, inplace = True)
    return df

text_an = text
punc = '''!()--[]{};:'"\,<>./?@#$%^&*_~1234567890'''
for symb in text:
    if symb in punc:
        text_an = text_an.replace(symb, "")

text_df = count_freq(text_an.split())

# Word count visualization
text_top10 = text_df.iloc[:12]

fig, ax1 = plt.subplots()

ax1.set_facecolor("yellow")

ax1.barh(text_top10["word"], text_top10["count"],
        color = "#386cb0",
        edgecolor = "#2b548a")

ax1.set_title("Частотный анализ текста")

for ax in fig.axes:
    ax.invert_yaxis()
    ax.grid(False)
    ax.title.set_color('black')
    ax.tick_params(axis='x', colors='black')
    ax.tick_params(axis='y', colors='black')
    ax.set_facecolor('#ffffff')
    ax.spines["top"].set_visible(False)
    ax.spines["right"].set_visible(False)
    ax.spines["left"].set_visible(False)

for i in ax.patches:
```

```
            plt.text(i.get_width()+0.2, i.get_y()+0.5,
                 str(round((i.get_width()), 2)),
                 fontsize=10, fontweight='bold',
                 color='grey')

        fig.patch.set_facecolor('#ffffff')
        plt.grid(which='major')
        fig.tight_layout()

        plt.show()


### Buttons actions ###

def button1_action():
    open_window1()

def button2_action():
    open_window2()

def button3_action():
    open_window3()

def button4_action():
    open_window4()

def button5_action():
    open_window5()

def button6_action():
    documentPath = simpledialog.askstring("ОТПРАВКА ДОКУМЕНТА", "Введите адрес
получателя:")

def button7_action():
    open_window7()

def button8_action():
    messagebox.showinfo("Button 8", "Сентимент анализ")


### Buttons view ###

window = tk.Tk()
window.title("АДАПТИВНАЯ СИСТЕМА ОБРАБОТКИ ГЕТЕРОГЕННЫХ ДОКУМЕНТОВ")
window.geometry("555x350")
window.configure(bg="#f1fbff")

label_font = font.Font(size=16, weight="bold")
label = tk.Label(window, text="АНАЛИЗ И КОРРЕКЦИЯ ДОКУМЕНТОВ", justify="center",
font=label_font, bg="#f1fbff")
```

```
label.grid(row=0, column=0, columnspan=4, pady=30)


button1 = tk.Button(window, text="ЗАГРУЗКА МОДЕЛИ ДОКУМЕНТА", width=15, height=5,
wraplength=100, command=button1_action)
button1.grid(row=1, column=0, padx=10, pady=10)

button2 = tk.Button(window, text="ЗАГРУЗКА ДОКУМЕНТА", width=15, height=5,
wraplength=100, command=button2_action)
button2.grid(row=1, column=1, padx=10, pady=10)

button3 = tk.Button(window, text="АНАЛИЗ И ОЦЕНКА ДОКУМЕНТА", width=15, height=5,
wraplength=100, command=button3_action)
button3.grid(row=1, column=2, padx=10, pady=10)

button4 = tk.Button(window, text="КОРРЕКЦИЯ ДОКУМЕНТА", width=15, height=5,
wraplength=100, command=button4_action)
button4.grid(row=1, column=3, padx=10, pady=10)

button7 = tk.Button(window, text="СТАТИСТИЧЕСКИЙ АНАЛИЗ", width=15, height=5,
wraplength=110, command=button7_action)
button7.grid(row=2, column=0, padx=10, pady=10)

button8 = tk.Button(window, text="СЕНТИМЕНТ АНАЛИЗ", width=15, height=5,
wraplength=100, command=button8_action)
button8.grid(row=2, column=1, padx=10, pady=10)

button5 = tk.Button(window, text="ВИЗУАЛИЗАЦИЯ РЕЗУЛЬТАТА", width=15, height=5,
wraplength=100, command=button5_action)
button5.grid(row=2, column=2, padx=10, pady=10)

button6 = tk.Button(window, text="ОТПРАВКА ДОКУМЕНТА АВТОРУ / В БАЗУ", width=15,
height=5, wraplength=100, command=button6_action)
button6.grid(row=2, column=3, padx=10, pady=10)

window.mainloop()
```