МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**  
**Факультет** **компьютерных технологий и прикладной математики**

**Кафедра информационных технологий**

**КУРСОВАЯ РАБОТА**

**МОДУЛЬ ОПРЕДЕЛЕНИЯ КОРРЕКТНОСТИ СТРУКТУРЫ ТЕХНИЧЕСКОЙ ДОКУМЕНТАЦИИ ПО ГОСТу**

Работу выполнила \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Д.А.Арабова

Направление подготовки 01.03.02 Прикладная математика и информатика курс 3

Направленность Программирование и информационные технологии

Научный руководитель

канд. пед. наук, доц.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Н.Ю.Добровольская

Нормоконтролер  
канд. пед. наук, доц.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А.В.Харченко

Краснодар

2023

**СОДЕРЖАНИЕ**

Введение 3

1. Морфологический анализ 4

1.1. Основные понятия морфологического анализа 4

1.2. Лексическая неоднозначность 6

1. Реферативный обзор сервисов морфологического анализа 9
2. Проектирование модуля определения корректности структуры технической документации по ГОСТу ...……………………………………... 15

3.1. О технической документации и её структуре 15

3.2. Подходы к морфологическому анализу 17

3.3. Реализация модуля определения корректности структуры технической документации по ГОСТу 21

Заключение 28

Список использованных источников 29

**ВВЕДЕНИЕ**

Компьютерная лингвистика — это направление лингвистики, в котором компьютерные технологии применяют для работы с языком.

На текущем этапе морфологический анализ представляет интерес, так как решает задачу определения леммы и ее грамматических характеристик.

Техническая документация — это набор документов, используемых при проектировании (конструировании), изготовлении и использовании объектов техники: зданий, сооружений, промышленных изделий, включая программное и аппаратное обеспечение.

Все программные продукты содержат техническую документацию, которая имеет строгую структуру, определяемую ГОСТами. Для людей, составляющих эту документацию, одной из важнейших задач является соблюдение требований. Для того чтобы минимизировать человеческий труд, при проверке корректности структуры документов, необходимо использовать все возможности компьютерной лингвистики для решения задачи.

В курсовой работе проектируется и реализуется модуль определения корректности структуры технической документации по ГОСТу.

**1 Морфологический анализ**

* 1. **Основные понятия морфологического анализа**

Морфологические модули (парсеры) выполняют функции, связанные с морфологическим анализом выделенных словоформ. Основной задачей морфологического анализа является определение по заданной словоформе леммы (нормальной формы слова) или основы слова, а при необходимости, и набора характеризующих ее морфологических характеристик (тегов), к числу которых относится род, число, одушевленность, падеж, лицо и др.

Морфологический анализ является одной из самых разработанных областей компьютерной лингвистики. Для русского языка она имеет ряд особенностей, что связано с его высокой флективностью и наличием большого числа исключений. К особенностям можно отнести наличие 9 падежей у существительных, кратких форм у прилагательных, отсутствие форм настоящего времени у глаголов совершенного вида и др. Все это значительно усложняет создание морфологических процессоров русского языка. Одна из существенных проблем – омонимия, когда у одной словоформы (к примеру: стекла, плачу) может оказаться несколько различных лемм и/или наборов морфологических характеристик [1].

Задачей морфологического анализа является определение по словоформе нормальной формы, от которой образована данная словоформа, и набора параметров, приписанных к данной словоформе. При этом может оказаться, что одной словоформе может быть сопоставлено несколько таких пар [2].

Морфологическому анализу предшествует подготовительный этап: текст нужно разбить на предложения, а в каждом предложении выделить слова, знаки препинания и другие элементы текста — смайлики, числа, формулы, таблицы и пр. Этот этап называется токенизацией (tokenization), а выделенные в результате единицы (слова, числа, знаки препинания и пр.) называются токенами (tokens) [3].

Считается, что для каждого токена существует его начальная (или нормальная) форма (также называемая леммой). От этой начальной формы создаются все остальные формы слова путем флексии, то есть некоторых изменений этой начальной формы, например (рисунок 1):

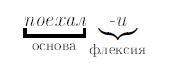


Рисунок 1 – Флексия

Образование новых слов или их форм происходит на уровне комбинирования морфов — минимальных значащих единиц языка. Морфы делятся на корневые (корень слова) и служебные: префикс (приставка), суффикс, флексия (окончание), постфикс. Носителем основного смысла слова является корень, а служебные, в общем случае, придают дополнительный смысл. Разбиение слова на морфы называется морфемным разбором, например (рисунок 2):

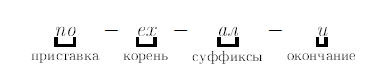


Рисунок 2 – Морфемный разбор

Некоторые служебные морфы (например, приставки и суффиксы) отвечает за образование новых слов, другие (например, окончания) — за образование форм слов. Изменение формы слова привязывается к набору грамматических параметров (тегов): часть речи, род, число, падеж, притяжательность и прочее.

Словоформой называется группа (кортеж), состоящую из токена, связанной с ним начальной формы и множества грамматических параметров. Например, словоформой является множество ⟨кошачьего, кошачий, {прил, муж. род, ед. ч., род. п} ⟩, которое содержит в себе строку «кошачьего», связанную с начальной формой «кошачий», и которая характеризуется следующими параметрами: прилагательное в мужском роде, единственном числе и родительном падеже. Лексемой называется множество всех словоформ, связанных с данной начальной формой.

Исходя из данных определений, можно сказать, что задачей морфологического анализа (лемматизации) является нахождение в словаре слово формы по ее токену. Задача морфологического синтеза прямо противоположная: по словоформе необходимо вернуть ее токен.

Морфологический анализ – это получение леммы или основы (псевдоосновы) заданного токена, а при необходимости, морфологических параметров;

Морфологический синтез— это генерация нужной словоформы слова или всей его парадигмы по нормальной форме (или основе) и морфологическим характеристикам [4].

**1.2 Лексическая неоднозначность**

Лексическая неоднозначность включает в себя несколько явлений.

Омонимия — это явление, при котором два слова сходны по написанию и звучанию, но различные по смыслу. Различают лексическую (совпадение форм разных лексем), грамматическую (совпадение токенов одной лексемы), синтаксическую (различные корректные трактовки одной и той же последовательности слов) и другие виды омонимии. Например, песчаная коса, косая коса и косой в соответствующих формах будут именно омонимами. Лексическая неоднозначность, помимо омонимии, включает в себя также различные варианты анализа слова в рамках одной лексемы: токен косой может быть проанализирован одновременно как именительный падеж мужского рода и творительный падеж женского рода в рамках одной лексемы ⟨косой, прилагательное⟩.

Снятие (или разрешение) омонимии — этап анализа текста, на котором проводится выбор единственного варианта морфологического анализа для каждого токена.

Если словарь не содержит данный токен, будем называть этот токен не словарным. Если с данным токеном связано более одной словоформы, будем называть такой токен неоднозначным. Словоформа содержит в себе лемму, часть речи и набор грамматических параметров. Следовательно, неоднозначность может проявить себя в одной из этих частей или их комбинации. Помимо не словарных и однозначных слов введём для токенов ещё четыре класса омонимии:

* неоднозначные по параметрам – в анализе присутствуют

словоформы с различными множествами грамматических параметров, но совпадающей леммой и частью речи. Например, прилагательное косой может выражать как именительный падеж мужского рода, так и творительный падеж женского рода.

* неоднозначные по части речи — в анализе присутствуют

словоформы, совпадающие по лемме, но отличающиеся по части речи. Так как части речи не совпадают, то наборы параметров у словоформ также будут отличаться. В связи с этим сравнение параметров проводиться не может. Например, прилагательное и существительное раненый.

* неоднозначные по лемме — в анализе присутствуют

словоформы, отличающиеся по лемме, но имеющие одинаковую часть речи. Здесь параметры могут как совпадать, так и отличаться. Примером совпадающих параметров является токен смели — третье лицо единственного числа от лемм сметь и смолоть (ударение у токенов будет различаться, но мы не увидим этого в тексте). Примером различающихся параметров будет являться токен вина, соответствующий именительному падежу единственного числа леммы вина и именительному падежу множественного числа леммы вино.

* неоднозначные по части речи и лемме — в анализе

присутствуют словоформы, отличающиеся как по лемме, так и по части речи. Сравнение параметров здесь также проводиться не может. Примером здесь может служить токен стекло, соответствующий именительному падежу единственного числа существительного стекло и третьему лицу среднего рода прошедшего времени глагола стечь [5].

1. **Реферативный обзор сервисов морфологического анализа**

# Рассмотрим примеры сервисов, осуществляющих морфологический разбор слова.

Для первого примера возьмем онлайн-сервиса (рисунок 3).

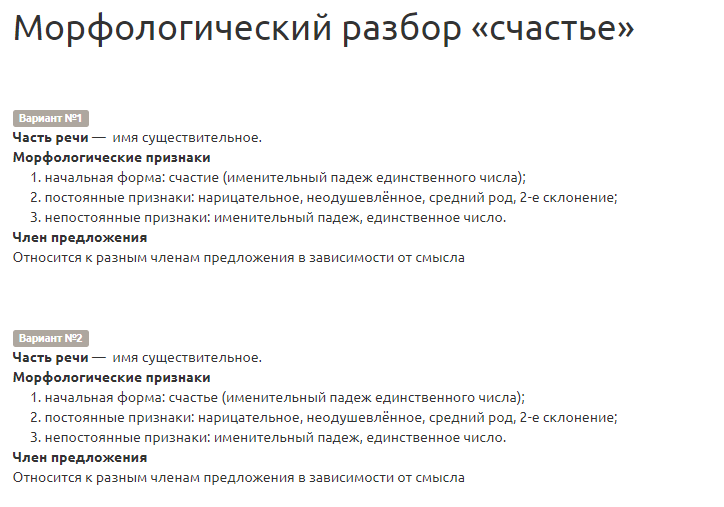


Рисунок 3 – Пример разбора слова в онлайн-сервисе

На рисунке 3 показан разбор слова: счастье. Сервис выполнил морфологический разбор, а также указал морфологические признаки слова. Рассматриваемый онлайн-сервиса располагает следующими возможностями:

* морфологический разбор слова,
* морфемный разбор слова,
* склонение слов,
* фонетический разбор слова.

# Существуют технологии, позволяющие выполнять морфологический разбор слова. К ним относится система pymorphy2, система TreeTagger, система Диалинг–AOT, система Snowball, система MyStem .

Система Диалинг-АОТ.

Проект АОТ включает в себя практически все этапы автоматического анализа текстов на ЕЯ, в том числе и морфологический. Проект разрабатывался при поддержке группы лингвистов из РГГУ, основным разработчиком выступал А. Сокирко. Морфологический процессор выложили в открытый доступ в 2004 году. Лексической основой служит словарь Зализняка.

Морфологический процессор АОТ предоставляет все функции полного морфологического анализа, включая нахождение леммы и морфологических характеристик словоформы, а также синтез словоформ.

АОТ базируется на словарной морфологии, в качестве словаря используется русский грамматический словарь А.А.Зализняка, который включает более 161 тыс. лемм. Логическая структура компьютерного словаря представляется в виде нескольких таблиц: лемм, флексий, приставок, морфологических характеристик. Таблица лемм содержит список псевдооснов слов со ссылками на таблицы флексий и приставок. В таблице флексий каждому из окончаний слов соответствует ссылка на соответствующие морфологические характеристики. Морфологический разбор слова по словарю состоит в поиске максимально совпадающей псевдоосновы в таблице лемм, поиск соответствующих приставки и окончания, а затем нахождение по таблице окончаний соответствующих морфологических характеристик. Синтез осуществляется похожим образом: после нахождения псевдоосновы по морфологическим характеристикам определяется вся парадигма и возвращается пользователю.

Для физического (бинарного) представления словаря используется структура конечного автомата. Стоит отметить, что хранение морфологических характеристик сильно увеличивает число состояний автомата и, как следствие, время поиска в нем, поэтому в АОТ характеристики находятся в отдельной таблице, а сам автомат хранит ссылки на них. Итоговый размер словаря составляет около 9 МБ, что является небольшим значением для современных компьютеров. Если словоформа не найдена в словаре, то в этом случае в АОТ работает морфологическое предсказание. Первым шагом предсказания является попытка найти существующую словоформу языка, которая имела бы максимально общее окончание со входным словом. Если при этом длина левой (неузнанной) части слова не превышает определенного размера (5 символов), а длина общего окончания со словарной словоформой не меньше 4 символов, тогда слово предсказывается по найденной правой части, т. е. берутся морфологические характеристики найденной словоформы. Если же такой подход не сработал, то ищется наиболее длинное совпадающее окончание, пример рисунок 4.

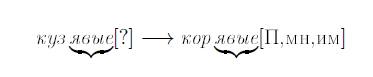


Рисунок 4 – Нахождение словоформы языка, имеющей максимальное общее окончание со входным словом

В настоящий момент морфопроцессор проекта АОТ является полностью открытым и распространяется под лицензией LGPL. Однако проект не поддерживается и не имеет удобных средств для пополнения словаря.

Система TreeTagger.

TreeTagger позиционируется как система для определения частей речи слов с возможностью настройки на любой естественный язык при наличии словаря и размеченного корпуса. Она была разработана в 1996 году в университете Штутгарта Хельмутом Шмидтом, и на данный момент доступна для множества языков, включая русский.

Процессор TreeTagger позволяет определять части речи слов и другие морфологические характеристики, а также их нормальную форму. Основной упор в данном процессоре сделан на разрешение морфологической омонимии и предсказание характеристик неизвестных слов.

TreeTagger базируется на словарной морфологии и использует словарь английского языка из проекта Penn TreeBank, содержащий более 2 млн. словоформ. Объем русского словаря неизвестен, однако по объему бинарного файла можно судить о сопоставимости размера словаря с английской версией. В процессе анализа используются 2 вида словарей: словоформ и суффиксов (имеются в виду флексии). Структуры данных, используемые для словарей, похожи на те, что используются в проекте АОТ, и также являются вариациями минимальных конечных автоматов. Автомат суффиксов (флексий) строится из всех флексий слов длиной до пяти символов. При этом каждому из суффиксов приписывается соответствующая флективная часть речи на основе взвешенной энтропии Шенона. При этом узлы, имеющие значение энтропии меньше определенного порога, удаляются из автомата.

Для снятия частеречной омонимии в TreeTagger используются решающие деревья для частей речи, обученные на размеченном корпусе. В узлах такого дерева находятся предикаты с ответом «да» или «нет» для двух предшествующих слов. При этом в листьях хранятся значения вероятностей для возможных ответов. Построение дерева происходит рекурсивно, с помощью модифицированного алгоритма ID3. На каждом шаге для двух предыдущих слов проверяются предикаты на равенство всем возможным частям речи, при этом для определения предиката, наилучшим образом разбивающего пространство признаков, используется правило максимизации энтропии Шенона. Для определения части речи входного слова достаточно, используя информацию о предыдущих словах, пройти по дереву от корня до листьев и выбрать наиболее вероятное значение.

В настоящий момент TreeTagger распространяется в виде бинарного файла, код самого процессора является закрытым. Проект поддерживается, для него создаются новые словари под различные языки. Синтез словоформ в TreeTagger отсутствует.

Система Pymorphy2.

Pymorphy2 — морфологический процессор с открытым исходным кодом, предоставляет все функции полного морфологического анализа и синтеза словоформ.

Процессор базируется на словарной морфологии и использует словарные данные проекта OpenCorpora. Словарь содержит около 250 тыс. лемм, а также является полностью открытым и регулярно пополняемым. Словарь, как и в проекте АОТ, логически представляет собой структуру из трёх таблиц, однако словарные данные хранятся в едином автомате. Для бинарного представления используется автомат с оптимизацией по памяти, что позволяет иметь в нем не более чем 232 различных связей, однако для задачи морфологического анализа данное ограничение не является существенным. Итоговый размер словаря составляет около 7 МБ.

В процессе морфологического синтеза, по исходной словоформе и тегам выполняется поиск нормальной формы слова, а затем перебор всех возможных пар ⟨окончание, теги⟩ в найденной лексеме, пока не будет найдена пара с заданными морфологическими тегами. После этого от нормальной формы отсекается её окончание, а найденное окончание приписывается к полученной псевдооснове.

Для анализа неизвестных слов в Pymorphy2 используются несколько методов, которые применяются последовательно. Изначально от слова отсекается префикс из набора известных префиксов и если остаток слова найден в словаре, то отсеченный префикс приписывается к результатам разбора. Если этот метод не сработал, то аналогичные действия выполняются для префикса слова длиной от 1 до 5, даже если такой префикс является неизвестным. Затем, в случае неудачи, словоформа разбирается по окончанию. Для этого используется дополнительный автомат всех окончаний, встречающихся в словаре с имеющимися разборами. В процессе построения из автомата удаляются редкие окончания и разборы. Метод анализа по окончанию аналогичен тому, что используется в процессоре АОТ.

Разрешение мофромонимии построено на основе корпусной статистики. Если слово имеет несколько вариантов разбора, то среди всех выбирается наиболее вероятный. Вероятности определяются по следующей формуле (рисунок 5):

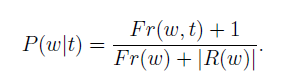


Рисунок 5 – Формула определения вероятности

В приведенной формуле, 𝐹𝑟(𝑤) — количество раз, которое словоформа 𝑤 встретилась в корпусе, а 𝐹𝑟(𝑤, 𝑡) — количество раз, которое эта словоформа встретилось с тегом 𝑡. |𝑅(𝑤)| — число разборов, полученных от анализатора для словоформы 𝑤.

В настоящее время Pymorphy2 поддерживается, при этом происходит постоянное пополнение корпуса OpenCorpora, что улучшает характеристики точности и полноты морфологического разбора [6].

**3** **Проектирование модуля определения корректности структуры технической документации по ГОСТу**

**3.1 О технической документации и её структуре**

Техническая документация — это документация, которая используется при проектировании, изготовлении и эксплуатации каких-либо технических объектов: зданий, сооружений, промышленных товаров, программного и аппаратного обеспечения.

Техническую документацию разделяют на несколько видов [7]:

* конструкторская документация
* эксплуатационная документация
* ремонтная документация
* технологическая документация
* документы, определяющие технологический цикл изделия
* документы, дающие информацию, необходимую для организации производства и ремонта изделия

Техническая документация демонстрирует и позволяет проследить правильность хода процесса, своевременно выявить отклонения или сбои и предупредить выпуск некачественной продукции или выполнение услуг. Также техническая документация необходима при оформлении договоров, сертификатов соответствия и при прохождении инспекционных проверок в компании надзорными органами.

К программным относят документы, содержащие сведения,

необходимые для разработки, изготовления, сопровождения и

эксплуатации программ.

Примеры технических документов:

* текст программы,
* описание программы,
* описание применения,
* руководство системного программиста,
* руководство оператора [8].

Каждый файл должен соответствовать ГОСТу, а также содержать обязательные части. Программный документ состоит из следующих условных частей:

* титульной,

Титульная часть состоит из листа утверждения и титульного листа.

Правила оформления листа утверждения и титульного листа

устанавливаются по ГОСТ 19.104–78.

* информационной,

Информационная часть должна состоять из аннотации и

содержания. Необходимость включения информационной части в различные виды программных документов установлена соответствующими стандартами ЕСПД на эти документы. В аннотации приводят сведения о назначении документа и краткое изложение его основной части [9].

Содержание включает перечень записей о структурных элементах основной части документа, в каждую из которых входят: обозначение структурного элемента (номер раздела, подраздела и т. п.), наименование структурного элемента, адрес структурного элемента на носителе данных (например, номер страницы, номер файла и т. п.);

* основной,

Состав и структура основной части программного документа устанавливаются стандартами ЕСПД на соответствующие документы.

* регистрации изменений

О каждом изменении программного документа в этой части делается запись в соответствии с требованиями ГОСТ 19.603–78.

Проверка корректности структуры документа упростит и ускорит нахождение ошибок, обязательных частей, а также проверку на соответствие ГОСТу.

Настоящий стандарт устанавливает общие требования к оформлению программных документов для вычислительных машин, комплексов и систем независимо от их назначения и области применения и предусмотренных стандартами Единой системы программной документации (ЕСПД) для любого способа выполнения документов на различных носителях данных [10].

Также стоит сказать о задачах, возникающих при разработке программы

* задача корректного составления структуры документа,
* задача считывания строки,
* задача проверки, написанного текста,
* задача выделения ошибок,
* задача визуализации результата

Все это и многое другое послужило импульсом для создания столь нужного приложения.

**3.2 Подходы к морфологическому анализу**

В зависимости от постановки задачи морфологический анализатор может возвращать разную информацию. Если нам необходимо рассчитать вектор частотности употребления слов в тексте, это может быть только лемма или лемма и часть речи. Для языков с невысокой флективностью применяется стемминг -определение основы слова путём отбрасывания окончаний из известного набора (возможно, псевдоосновы за счёт отбрасывания псевдоокончаний). Отличие стемминга от лемматизации можно продемонстрировать на примере слова выходцы: процесс лемматизации определит лемму выходец, в то время как стемминг вернёт псевдооснову выход. По функциональным возможностям морфологические процессоры делятся на несколько видов:

* выполняющие только лемматизацию или стемминг,
* определяющие часть речи,
* осуществляющие полный морфологический анализ, т. е.

лемматизацию и определение всех морфологических характеристик словоформы.

* осуществляющие морфемный анализ, выделяющие морфы,

входящие в состав слова.

Введение в морфоанализатор функции морфемного разбора расширяет его применимость. Например, с помощью данной функции может быть реализован поиск семантически близких слов разных частей речи (однокоренных), что может быть полезно в ряде задач компьютерной лингвистики, включая задачу распознавания конструкций по шаблонам. Первые морфологические анализаторы русского языка были простыми и практически не использовали словарной информации. С ростом вычислительных мощностей и объёмов оперативной памяти, а также появлением новых алгоритмов и структур данных стало возможно эффективное использование больших словарей, что значительно улучшило качество реализуемой модели.

Бессловарная морфология

Бессловарные морфологии являются одним из первых подходов к решению задачи морфоанализа русского языка. Однако бессловарными они являются лишь условно: в них отсутствуют большие словари лексических единиц, но фактически используются небольшие словари с информацией

о флексии (рисунок 6).

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 6 – Пример небольшого словаря с флексией

В первом столбце содержатся флексии словоформ, во втором – флексия нормальной формы, а в третьем — морфологические характеристики, соответствующие исходной словоформе с данной флексией. Конечно, из данной таблицы можно удалить второй столбец, договорившись, что первая запись содержит в себе псевдоокончание начальной формы.

Для бессловарной морфологии анализ сводится к поиску наиболее длинного окончания анализируемой словоформы в этом словаре и выборке соответствующих морфологических характеристик (рисунок 7)

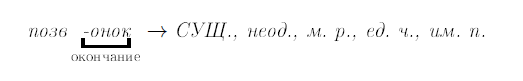


Рисунок 7 – Пример морфологических характеристик

Существенным плюсом здесь является то, что определение морфологических характеристик и нормальной формы возможно практически для любой словоформы, для которой нашлась подходящая флексия. Однако такой подход зачастую оказывается неточным из-за наличия большого количества исключений в языке.

Словарная морфология на основе словаря основ

Одним из классических подходов к морфологическому анализу русского языка является построение словарной морфологии на основе словаря основ. Основой модели выступает словарь основ, который содержит все (псевдо)основы лексем языка. Он связан со вспомогательными словарями, в которых содержится список флексий всех словоизменительных классов, для каждой из которых указан набор значений морфологических характеристик, которые она может выражать. Также, как правило, хранится дополнительная информация об особенностях словоизменения, например, о чередовании букв в основах или беглых гласных. Зачастую присутствует дополнительный словарь исключительных случаев.

Разбор словоформ происходит по следующей схеме

1. Последовательно отсекаются возможные окончания длиной от 0 до 𝑛

букв, таким образом слово разбивается на основу и флексию.

1. Для полученного окончания находится его словоизменительный класс.
2. Проверяется наличие полученной основы в словаре основ и находится

номер её словоизменительного класса.

1. В случае совпадения словоизменительных классов выбираются

соответствующие морфологические характеристики и строится лемма, которые и

являются результатом анализа.

Словарная морфология на основе словаря словоформ

Для высокофлективного языка наиболее частым подходом к решению

задачи морфологического анализа является словарная морфология на основе словаря словоформ. База морфологического процессора, построенного на такой морфологии, заключается в создании словаря всех форм языка, который может быть представлен в виде таблицы (рисунок 8).

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 8 – Представление словаря всех форм в виде таблицы

Определение нормальной формы и морфологических характеристик

сводится к поиску словоформы в таблице. Синтез словоформы, похожим образом, сводится сначала к поиску нормальной формы, а затем формы, соответствующей запрошенным морфологическим характеристикам. Значительной проблемой всех словарных морфологий, особенно на основе словаря словоформ, является структура хранения словаря, который может занимать несколько гигабайт. Однако, при использовании эффективной структуры данных, даже более обширный словарь словоформ становится не только хорошей теоретической моделью, но и удобным практическим средством. Достоинствами являются возможность выполнения как анализа, так и синтеза. К основным недостаткам словарных морфологий относятся: проблема анализа слов, которых нет в словаре, необходимость качественного и объёмного словаря, проблема морфологической омонимии [11].

**3.3 Реализация модуля определения корректности структуры технической документации по ГОСТу**

Выделим набор ошибок, которые необходимо распознать модулю корректности.

Ошибка 1. Отсутствие обязательных страниц.

Обязательными по ГОСТу считаются 4 страницы: титульный лист, аннотация, содержание, лист регистрации, их отсутствие или недостаток является несоответствием ГОСТу.

Ошибка 2. Недостаточное количество строк кода.

В файле технического документа «Текст программы» по ГОСТу основная часть должна содержать не менее 500 строк кода, в противном случае – несоответствие ГОСТу.

Ошибка 3. Несоответствие содержания технического документа «Описание программы» ГОСТу.

По ГОСТу содержание документа «Описание программы» должно иметь обязательные пункты и подпункты, если в файле их не обнаружено – несоответствие ГОСТу.

Для реализации проекта был использован язык Python с подключением кросс-платформенной событийно-ориентированной графической библиотеки tkinter, также библиотеки pathlib, docx, docx2txt. Программа содержит пять функций, работающих с файлами. Рассмотрим работу некоторых из них.

Первым примером является функция, считающая количество строк кода в файле.

def n\_o\_r():

doc1 = docx.Document(file1\_entry.get())

f = Path(file2\_entry.get()).stem

if f == 'ШаблонТекстпрограммы':

line\_count = 0

for paragraph in doc1.paragraphs[3:-1]:

line\_count += len(paragraph.text.splitlines())

result\_text.delete("3.0", tk.END)

result\_text.insert("3.0", "\nКоличество строк: " + str(line\_count) + "\n\n")

Функция n\_o\_r() проверяет какой шаблон был загружен, если он соответствует такому типу технической документации, как «Текст программы», то в файле отбрасываются первые три и последняя страницы, так как они являются обязательными и не содержат кода. Затем вычисляется количество строк на оставшихся страницах и результат выводится на экран.

Вторым примером является функция, проверяющая содержание файла «Описание программы» на соответствие ГОСТу.

def content\_op():

text = docx2txt.process(file1\_entry.get())

f = Path(file2\_entry.get()).stem

text = text.lower()

if f == 'ШаблонОписаниепрограммы':

search\_words = ['общие сведения', 'функциональное назначение', 'описание логической структуры',

'используемые технические средства', 'вызов и загрузка', 'входные данные', 'выходные данные']

for search\_word in search\_words:

if search\_word.lower() not in text:

return result\_text.insert("5.0", "\n Cодержаниe файла 'Описание программы' не соответсвует ГОСТу: " + "\n\n")

return result\_text.insert("5.0", "\nCодержаниe файла 'Описание программы' соответсвует ГОСТу: " + "\n\n")

Функция content\_op() проверяет какой шаблон был загружен, если он соответствует такому типу технической документации, как «Описание программы», то файл проверяется на наличие обязательных по ГОСТу разделов содержания .

Третьим примером является функция, сравнивающая исходный файл с соответствующим файлом-шаблоном, содержащим обязательные по ГОСТу страницы.

def compare\_files():

file1\_path = Document(file1\_entry.get())

file2\_path = Document(file2\_entry.get())

template\_pages = len(file2\_path.sections)

count = 0

same\_pages = []

for i in range(len(file1\_path.sections)):

for j in range(len(file2\_path.sections)):

if file1\_path.paragraphs[i].text == file2\_path.paragraphs[j].text:

same\_pages.append(i)

count += 1

result\_text.delete("0.0", tk.END)

result\_text.insert("1.0", "Одинаковые страницы: " + str(

same\_pages) + "\nКоличество одинаковых страниц: " + str(count) + "\n\n")

Функция compare\_files() разделяет загруженные файлы на секции, затем сравнивает их и выводит одинаковые секции, а также количество одинаковых страниц.

Рассмотрим работу модуля на конкретных примерах.

Пример 1

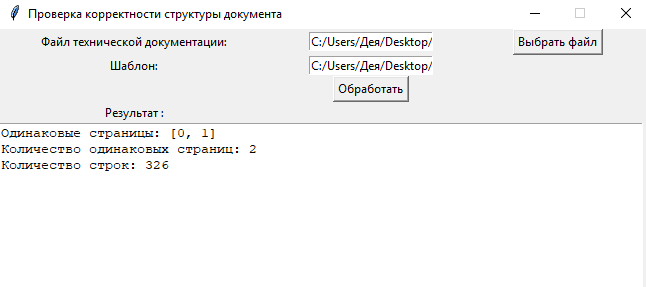


Рисунок 9 – Пример работы модуля с техническим документом «Текст программы»

На рисунке 9 можно увидеть работу модуля с техническим документом «Текст программы».

В примере 1 можно увидеть, что было проведено сравнение с шаблоном и выведены одинаковые страницы, их количество и количество строк кода в файле.

Пример 2

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Рисунок 10 – Пример работы модуля с техническим документом «Описание программы»

На рисунке 10 можно увидеть работу модуля с техническим документом «Описание программы».

В примере 2 можно увидеть, что было проведено сравнение с шаблоном и выведены одинаковые страницы и их количество, также была проведена проверка содержания на соответствие ГОСТу.

**ЗАКЛЮЧЕНИЕ**

Цель курсовой работы – проектирование и реализация модуля проверки корректности структуры технической документации по ГОСТу– достигнута.

В теоретической части курсовой работы описаны базовые принципы морфологического анализа, а также возможные лексические неоднозначности.

Выполнен реферативный обзор сервисов, реализующих морфологический анализ. Рассмотрены особенности технической документации и ее структуры, учтённые далее при реализации модуля.

В работе предложен проект и реализация модуля проверки корректности структуры технической документации. Основной функцией модуля является поиск несоответствий с ГОСТами в структуре документов.

При программной реализации функционала использовалась библиотека языка Python tkinter. При работе модуль находит такие ошибки, например, как отсутствие обязательных страниц, недостаточное количество строк кода и несоответствие содержания ГОСТу.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Сапин, А.С. Особенности построения морфопроцессора русского языка CrossMorhpy/ А.С. Сапин, Е.И .Большакова.; Новые информационные технологии в автоматизированных системах. 2017. №20.

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

3. Бочаров, В. В. Компьютерная морфология / В. В. Бочаров, О. В. Митренина Компьютерная морфология // Прикладная и компьютерная лингвистика / Николаев И.С., Митренина О.В., Ландо Т.М. (ред.), М.: URSS, 2017. C . 14–34

4. Автоматическая обработка текстов на естественном языке и анализ данных: учеб. пособие / Большакова Е. И., Воронцов К. В., Ефремова Н.Э., Клышинский Э.С, Лукашевич Н.В., Сапин А.С – М.: Изд-во НИУ ВШЭ, 2017. – 269 с.

5. Агеев, М. С. Автоматическая рубрикация текстов: методы и проблемы / М. С. Агеев., Б. В. Добров, Н. В. Лукашевич– Казань: Физико-математические науки, — 2008.— Т. 150, № 4.— С. 25–40.

6. Павлов, А. С. Метод обнаружения массово порожденных неестественных текстов на основе анализа тематической структуры /, А. С. Павлов, Б. В. Добров Вычислительные методы и программирование: новые вычислительные технологии. — 2011.—Т. 12.— С. 58–72.

7. Техническая документация: [сайт]. – 2022 – URL: <https://rossertcentr.ru/dokumentatsiya> (дата обращения: 05.05.2023).

8. ГОСТ 19.101-77. Единая система программной документации. Виды программ и программных документов.

9. ГОСТ 19.104-78. Единая система программной документации. Основные надписи.

10. ГОСТ 19.105-78. Единая система программной документации. Общие требования к программным документам.

11. Система для извлечения информации из текстов на базе лексико-синтаксических шаблонов / Большакова Е.И., Иванов К.М., Сапин А.С., Шариков Г.Ф.; Пятнадцатая национальная конференция по искусственному интеллекту с международным участием (КИИ-2016). 3-7 октября 2016 г.: Труды конференции. Том. 1 —Смоленск, Универсум, 2016, с.14-22.