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**

**ПОМОЩНИК ВВОДА ДАННЫХ НА ОСНОВЕ ГРАФЕМАТИЧЕСКОГО АНАЛИЗА**

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

2022

**СОДЕРЖАНИЕ**

Введение 4

1. Базовые принципы компьютерной лингвистики 5

1.1. Основные понятия компьютерной лингвистики 5

1.2. Графематический анализ 6

1. Технологии реализации синтаксического анализа 9

2.1. Реферативный обзор сервисов синтаксического анализа 9

2.2. Технология предварительного синтаксического анализа 12

1. Проектирование сервиса ввода данных на основе графематического анализа……………………………………………………..……………………. 19

3.1. О возможных ошибках при вводе паспортных данных 20

3.2. Основные методы нахождения ошибок 22

3.3. Реализация сервиса проверки корректности персональных данных 22

Заключение 28

Список использованных источников 29

**ВВЕДЕНИЕ**

Компьютерная лингвистика — это область знаний, которая занимается компьютерным моделированием владения естественным языком и решением прикладных задач автоматической обработки текстов и речи.

На текущем этапе графематический анализ представляет интерес, так как решает задачу синтаксического анализа предложения.

Сегодня особое внимание уделяется компьютерному моделированию языка в решении задач искусственного интеллекта, фундаментальным принципам моделирования языка, т. е. направлениям, которые связаны с моделированием общения человека и компьютера. Это позволяет выделять такие задачи и направления, как проблемы лингвистического обеспечения современных автоматизированных информационных систем, автоматическую обработку естественного языка, создание словарных и текстовых процессоров.

Графематический анализ применяется при проверке корректности различных данных, вводимых в информационные системы. Пользователь может не знать способ написания либо иметь множество его альтернатив, а также допустить в нем ошибки — это все приводит к неоднозначности написания, что в официальных документах не допускается. Синтаксический анализ предложения, выделение лексем и их приведение к формальному виду решает эту задачу.

В курсовой работе рассматривается пример помощника ввода данных на основе графематического анализа.

**1 Базовые принципы компьютерной лингвистики**

* 1. **Основные понятия компьютерной лингвистики**

## Компьютерная лингвистика – будучи одним из направлений прикладной лингвистики, изучает лингвистические основы информатики и все аспекты связи языка и мышления, моделирование языка и мышления в компьютерной среде с помощью компьютерных программ, а ее интересы лежат в области:

## 1) оптимизации коммуникации на основе лингвистических знаний

## 2)создание естественно-языкового интерфейса и типологий понимания языка для общения человека с машиной

## 3) создание и моделирование информационных компьютерных систем.

## Объект компьютерной лингвистики – анализ языка в его естественном состоянии в процессе использования людьми в различных ситуациях общения, а также анализ того, как особенности языка могут быть сформулированы [2].

Основную задачу компьютерной лингвистики можно сформулировать, как построение моделей и соответствующих им алгоритмов и программ для автоматической обработки текста [3].

Языкознание изучает общие законы естественного языка – его структуру и функционирование, и включает такие области:

Фонология – изучает звуки речи и правила их соединения при формировании речи;

Морфология – занимается внутренней структурой и внешней формой слов речи, включая части речи и их категории;

Синтаксис – изучает структуру предложений, правила сочетаемости и порядка следования слов в предложении, а также общие его свойства как единицы языка.

Семантика и прагматика – тесно связанные области: семантика занимается смыслом слов, предложений и других единиц речи, а прагматика – особенностями выражения этого смысла в связи с конкретными целями общения;

Лексикография описывает лексикон конкретного ЕЯ – его отдельные слова и их грамматические свойства, а также методы создания словарей.

Эти разделы соответствуют этапам автоматического анализа текста, который заключается в извлечении грамматической и семантической информации из заданного текста на естественном языке.

Отражение уровневой организации языка в архитектуре систем компьютерного анализа естественно-языкового (ЕЯ) текста:

* фонетический уровень;
* графематический уровень;
* морфологический уровень;
* синтаксический уровень;
* семантический уровень [4].

**1.2 Графематический анализ**

Применение систем машинного перевода позволит существенно уменьшить затраты времени и сил человека на анализ и перевод текста. Этап графематического анализа в данных системах является начальным и качество, а главное «глубина» его проведения оказывает большое влияние на конечный результат. Этот этап позволяет решить множество проблем, наиболее важная из них является поиск конца предложения. Поиск по точке оказывается, крайне неэффективен в случае наличия в предложении сокращений, отсюда вытекает ещё одна проблема – это устранение тех самых сокращений с заменой их на целые слова или словосочетания, а также борьба с их возможной неоднозначностью. Также применение графематического анализа позволяет ввести или упростить введение новых подэтапов, наличие которых существенно повлияет на качество перевода и удобство его восприятие. Таким подэтапом можно назвать замену чисел, записанных словами на эквивалентные им числовые выражения.

Этап графематического анализа.

Перед началом описания этапа, следует отметить, что на вход графематического анализа подаётся исходный текст на естественном языке, не проходивший машинной обработки. Анализ начинается с выделения абзацев, если они отсутствуют, то текст принимается как один абзац. Здесь же происходит удаление пустых строк в конце и начале входного текста. Каждый абзац делится на предложения, которые в свою очередь разбивается на слова и знаки препинания, последние приравниваются к словам, но с единственным параметром в виде части речи. На каждом шаге разбиения запоминается последовательность следования абзацев, предложений, слов или знаков препинания. Так же при разбиении мы удаляем все пробелы и в итоге для каждого предложения мы получаем массив слов и знаков препинания, идущих в том же порядке что и в исходном предложении. На завершающем шаге этапа каждый массив слов предложения проходит через этап морфологии, где для каждого слова определяется набор параметров, включающих: часть речи, начальную форму, род, число, падеж и т. д.

Определение сокращения.

В текстах многих языков для экономии времени и места широко используется применение сокращений, аббревиатур и тому подобное. Как правило, при неполном написании слова последним символом ставится точка, например при написании инициалов имени и отчества «Пушкин А. С.» или в предложениях, указывающих на время и место «… в 2004 **г.** (году) в **г.** (городе) Москве …». Отсюда при анализе может возникнуть масса ошибок, так «Пушкин А. С.» можно разделить на два предложения «Пушкин А.» и «С.», что приведет к потере смысла и обрыву предложения. Во втором примере мы ясно видим два одинаковых по написанию, но разных по значению сокращения, то есть возникает проблема неоднозначности замены. Для пресечения возникновения подобных ошибок этап графематического анализа имеет в своём составе этап замены сокращений, который занимается поиском, анализом и соответственно заменой сокращений на слова или словосочетания.

Этап замены сокращений.

Основой работы для данного этапа являются правила, служащие для анализа точек, апострофов, других знаков препинания, присутствие которых во входном тексте может означать наличие сокращения. Следует отметить, что на вход данного этапа поступает весь текст в виде массива слов и знаков препинания. Каждое правило создаётся на основе грамматики данного естественного языка, с учётом особенностей дальнейших этапов работы и является уникальным, его примерный формат показан на рисунке 1.

Как показано на рисунке 1 левая, и правая часть правила состоит из отдельных ячеек 1, 2, 3, … каждая ячейка содержит только одно слово или знак препинания и набор характеризующих его параметров. В пределах части их формат одинаков а, следовательно, каждая ячейка имеет строго определённое число параметров наличие или отсутствие, которых необходимо указать. Важным условием при составлении правила является сохранение порядка следования ячеек таким же, как порядок слов и знаков при использовании сокращения. Каждому естественному языку, для которого должен выполняться графематический анализ, формируется свой набор правил по выделению сокращений. Эти правила объединяются в коллекции, по одной на каждый язык.

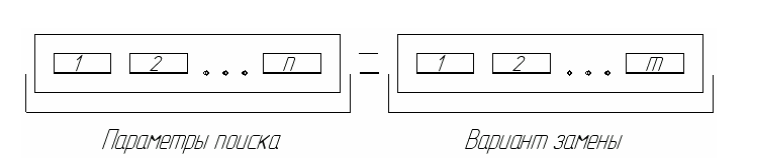


Рисунок 1 – Формат правил

Работа правил осуществляется следующим образом. При встрече во входном потоке сокращения к нему и близлежащим словам поочерёдно применяют правила из коллекции. Например, если под «г.» подразумевали (город), то левая часть будет описано так, в первой ячейке «г» — это не заглавная буква и не число. Во второй ячейки «.» — это не заглавная буква и не число. В третьей ячейке будет говориться, что следующее слово должно как минимум идти с большой буквы. Вариант замены или правая часть правила будет содержать одну ячейку с записью: «город» плюс параметры существительное, с маленькой буквы и т. д. Если же под «**г.»** понимается «год», то применится другое правило, где в параметрах поиска добавится ячейка с условием, что перед «г» должно стоять числительное. При замене удаляется сокращённое слово или массив слов и знаков препинания, а на их место ставятся целые слова или словосочетания.

Преобразование чисел и числовых выражений.

Отдельный подэтап предназначен для преобразования числительных, записанных словами в аналогичные цифровые записи. Благодаря проведению графематического анализа работу данного этапа удалось существенно упростить за счёт того, что имеется возможность отлавливания числительных на начальной стадии анализа. Этап работает со списками числительных, которые включают строковые числительные и числовой эквивалент. Для каждого языка существуют списки числительных, хранящиеся в отдельных базах. Формат записи, примерно следующий (рисунок 2):

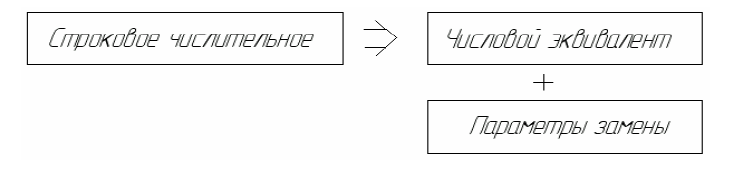


Рисунок 2 – Формат записи правил числительных

На вход поступает текст прошедший графематический анализ, выделяем слова или последовательность слов с частью речи числительные. Используя списки числительных, заменяем найденные слова на числовые выражения, то есть алгоритм этапа производит преобразование текста во внутреннее представление программы машинного перевода.

Этот модуль графематического анализа выполняет две основные функции: полное поэтапное разбиение текстов составные единицы и определение границ предложений.

Поэтапное разбиение текстов даёт возможность отслеживания изменений на каждом этапе обработки. Результат этапа является массив слов с параметрами, такой способ представления данных оптимален для дальнейшего анализа.

Точное определение границ предложений необходимо для сохранения логического смысла текста при переводе, что является основополагающей задачей систем машинного перевода [5].

Рассмотренные особенности графематического анализа позволяют взять его за основу проверки корректности вводимых данных, например паспортных данных.

1. **Технологии реализации синтаксического анализа**

**2.1 Реферативный обзор сервисов синтаксического анализа**

# Рассмотрим примеры сервисов, осуществляющих синтаксический разбор предложения.

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

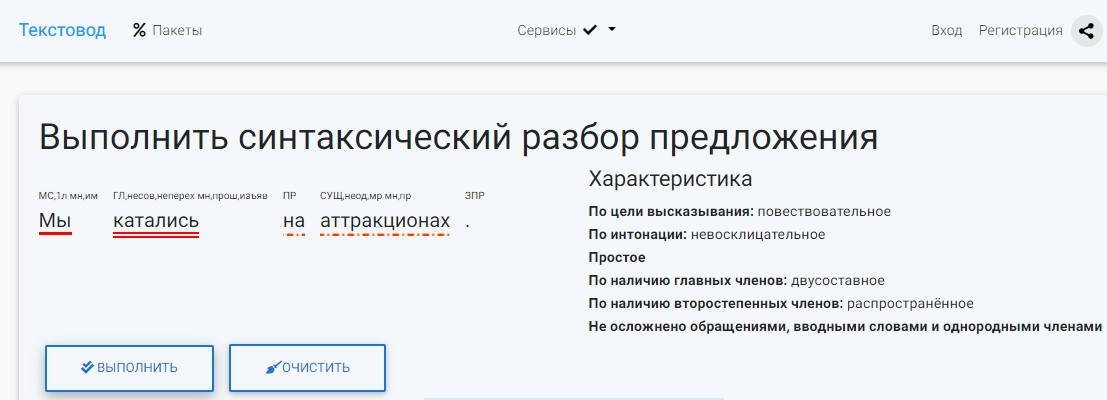


Рисунок 3 – Пример разбора предложения в онлайн-сервисе

На рисунке 3 показан разбор предложения: Мы катались на аттракционах. Сервис выполнил синтаксический разбор, а также указал характеристики предложения. Рассматриваемый онлайн-сервиса располагает следующими возможностями:

* синтаксический разбор предложения,
* морфологический разбор предложения,
* части речи в предложении,
* разбор предложения по членам,
* расстановка запятых в тексте автоматически
* проверка уникальности текста,
* проверка орфографии,
* подбор синонимов,
* подсчет количества символов,
* поиск слова в тексте и его замена.

# Существуют технологии, позволяющие выполнять синтаксический разбор предложения. К ним относится анализатор pymorphy2.

Анализатор pymorphy2 написан на языке Python. Перечислим возможности сервиса:

1. Приведение слова к нормальной форме (например, “люди -> человек”, или “гулял -> гулять”).
2. Постановка слова в нужную форму, например, во множественное число, изменение падежа слова и т. д.
3. Вернуть грамматическую информацию о слове (число, род, падеж, часть речи и т. д.)

При работе используется словарь [OpenCorpora](http://opencorpora.org/); для незнакомых слов строятся гипотезы. Для поиска шаблона (прилагательное, существительное) нужно искать пару слов в том же падеже, числе и роде. Для определения характеристик (граммем) в Python можно использовать pymorphy2.

source = '''

Вася ест кашу

# сущ гл сущ

# что/кто делает с\_чем-то

NOUN,nomn VERB NOUN,accs

Красивый цветок

ADJF NOUN

Птица сидит на крыше

# сущ гл предлог сущ

NOUN,nomn VERB NOUN,loct

'''

text = '''

Мама мыла раму

Вася разбил окно

Лара сама мыла раму

Мама мыла пластиковые окна

'''

import pymorphy2 as py

class PPattern:

def \_\_init\_\_(self):

super().\_\_init\_\_()

def checkPhrase(self,text):

def checkWordTags(tags, grams):

for t in tags:

if t not in grams:

return False

return True

def checkWord(tags, word):

variants = morph.parse(word)

for v in variants:

if checkWordTags(self.tags[nextTag].split(','), v.tag.grammemes):

return (word, v)

return None

morph = py.MorphAnalyzer()

words = text.split()

nextTag = 0

result = []

for w in words:

res = checkWord(self.tags[nextTag].split(','), w)

if res is not None:

result.append(res)

nextTag = nextTag + 1

if nextTag >= len(self.tags):

return result

return None

def parseText(pats, text):

def parseLine(line):

was = False

for p in pats:

res = p.checkPhrase(line)

if res:

print('+',line, p.tags, [r[0] for r in res])

was = True

if not was:

print('-',line)

buf = io.StringIO(text)

s = buf.readline()

while s:

s = s.strip()

if s != '':

parseLine(s)

s = buf.readline()

patterns = parseSource(source)

parseText(patterns, text)

Pymorphy2 при анализе слова возвращает массив всех возможных вариантов, того, что это за слово: «мыла» — это существительное или глагол. Поэтому наша задача проверить все эти варианты и выбрать из них такой, что характеристики слова подойдут под шаблон. Это делается в функции checkWord.  
Получаем результат разбора:

+ Мама мыла раму ['NOUN,nomn', 'VERB', 'NOUN,accs'] ['Мама', 'мыла', 'раму']  
+ Вася разбил окно ['NOUN,nomn', 'VERB', 'NOUN,accs'] ['Вася', 'разбил', 'окно']  
+ Лара сама мыла раму ['NOUN,nomn', 'VERB', 'NOUN,accs'] ['Лара', 'мыла', 'раму']  
+ Лара сама мыла раму ['ADJF', 'NOUN'] ['сама', 'мыла']  
+ Мама мыла пластиковые окна ['NOUN,nomn', 'VERB', 'NOUN,accs'] ['Мама', 'мыла', 'окна']  
+ Мама мыла пластиковые окна ['ADJF', 'NOUN'] ['пластиковые', 'окна']

**2.2 Технология предварительного синтаксического анализ**

Приведем пошаговое описание технологии синтаксического разбора.

### Выделение слов.

Разбиение сплошного текста на составляющие слова и элементы (номер телефона, дата, интернет-адрес и прочие) в общем случае не представляет трудной задачи. Алгоритмы усложнялись при выделении сложносоставных слов («диван-кровать»), окончаний цифровых числительных («2000-**го**», «в 90**-х**») и сокращений («кап. гос-ва»). Более сложными были задачи по сборке фразеологизмов, состоящих из нескольких слов, которые сами по себе имеют самостоятельное значение («на первый взгляд»). Или отладка логики отличия точки в конце предложения от точки в составе сокращения («… и проч.»). При этом за сокращением не может (которое само по себе является самостоятельным словом) следовать какое-либо имя / название с большой буквы («гор. Москва»). Предложение естественного языка состоит из словоформ и устойчивых словосочетаний. Ряд словоформ данного слова называется парадигмой.

Например, парадигма слова "рука": [рука, руки, руке, руку, рукой, о руке]

Каждое слово в предложении определяется тройкой:

* строка словоформы/словосочетания («писал»),
* нормальная форма слова («писать»),
* набор грамматических параметров (['VERB', 'sing', 'musc', 'tran', 'past']).

Таким образом, разбиение предложения "Ясно дело, он не придет на встречу" будет иметь следующий вид:

['Ясно дело', 'он', 'не', 'придет', 'на', 'встречу']

здесь 'ясно дело' - устойчивое выражение, неизменяемое

### Определение грамматических параметров (граммем).

### Граммемой называется элемент грамматической категории; различные граммемы одной категории исключают друг друга и не выражаются вместе. Для каждой словоформы определяем набор из семи граммем:

### [ ЧАСТЬ РЕЧИ, ЧИСЛО, ЛИЦО, РОД, ПАДЕЖ, ВАЛЕНТНОСТЬ, ВРЕМЯ ] В качестве источника используем словарь [OpenCorpora](http://opencorpora.org/) и интерфейс к нему -[pymorphy2](https://habr.com/ru/post/176575/). Для поиска правила в грамматике по данному набору граммем представим их в общем виде:

'яблоки' [NOUN, plur,neut,accs] -> [NOUN,?numb,?per,?gend,accs,None,None] здесь '?' означает, что параметр может принимать произвольное значение

### Связывание слов меж собой.

### Для определения синтаксической связи между словами используем контекстно-свободную грамматику и LR-анализ.

#### Грамматика и LR-анализ.

#### Формальная грамматика — способ описания языка в виде так называемых продукций. Например:

#### a -> ab | ac

#### означает, что правило 'a' порождает 'ab' ИЛИ 'ac'.

#### Нетерминалами называются объекты, обозначающие какую-либо сущность языка (предложение, формула и т. д.). Терминалы — объекты непосредственно присутствующие в языке, соответствующего грамматике, и имеющий конкретное, неизменяемое значение (буквы, слова, формулы и др.). Контекстно-свободные грамматики, это такие грамматики, у которых левые части всех продукций являются одиночными нетерминалами. Для описания русского языка используем теорию грамматики составляющих (phrase structure grammar), которая утверждает что всякая сложная грамматическая единица складывается из двух более простых и не пересекающихся единиц, называемых её непосредственными составляющими. Выделяют следующие составляющие:

#### а) Именная группа (NP).

#### NP[case='nomn'] -> N[case='nomn'] | ADJ[case='nomn'] NP[case='nomn'] | …

#### То есть номинативная именная группа — это существительное в номинативном падеже ИЛИ прилагательное в номинативном падеже + номинативная именная группа ИЛИ другое.

#### б) Глагольная группа (VP).

#### VP[tran] -> V[tran] NP[case='ablt'] | ADJ VP[tran] | …

#### Другими словами, транзитивная глагольная группа — это транзитивный глагол + аблативная именная группа ИЛИ краткое прилагательное + транзитивная глагольная группа ИЛИ другое.

#### в) Предложная группа (PP).

#### PP -> PREP NP [case='datv'] | ...

#### Предложная группа — это предлог + именная дативная группа ИЛИ другое.

#### г) Полное предложение (S).

#### S -> NP [case='nomn'] VP[tran]

#### Полное предложение существует тогда и только тогда, когда именная и глагольная группы согласованы в числе, лице и роде.

#### def agreement(self, node\_left, node\_right):

#### ...

#### if (numb1 and numb2):

#### if (numb1 != numb2): return False;

#### if (per1 and per2):

#### if (per1 != per2): return False;

#### if (gend1 and gend2):

#### if (gend1 != gend2): return False;

#### return True;

#### Неполным предложением называется такое предложение, где опущена именная часть. Как правило, в таких предложениях глагольная группа выражена безличным глаголом. Например, "Мне хочется гулять", "Светает". Эллиптическим предложением называется предложение, где опущена глагольная часть, она заменяется знаком тире. Например, "За спиной – лес. Справа и слева – болота".

#### Для того, чтобы определить принадлежность данного предложения к языку грамматики используем алгоритм LR-анализа. Данный алгоритм предполагает построение дерева разбора снизу вверх (от листьев к корню). Ключевым элементов алгоритма является метод «переноса-свертки» (англ. shift-reduce):

#### читаем символы входной строки до тех пор, пока найдется цепочка, совпадающая с правой частью какого-нибудь из правил, найденную цепочку положить в стэк (перенос);

#### заменим найденную цепочку правилом из грамматики (свертка).

#### Если все цепочки строки были перенесены, то данное предложение принадлежит языку грамматики, и по крайней мере одно дерево разбора существует.

1. Дерево   
   Для представления синтаксической связи в предложении используется бинарное дерево, где листья — это слова (терминалы) с набором граммем, а узлы — правила (претерминалы). Корнем является предложение (нетерминал).

Узел дерева определяется следующим образом:  
class Node:

def \_\_init\_\_(self, word=None, tag=None, grammemes=None, leaf=False):

self.word = word; # строка слова

self.tag = tag; # здесь тэг - претерминал,

который соотвествует промежуточному правилу грамматики

self.grammemes = grammemes; # набор граммем

self.leaf = leaf;

self.l = None;

self.r = None;

self.p = None;

Построение дерева начинается с листьев, которым присваивается строка слова или словосочетания, а также набор ее граммем.

def build(self, sent):

for word in sent:

new\_node = Node(word[0], word[1], word[2], leaf=True)

self.nodes.append(new\_node)

Далее осуществляется LR-анализ. Каждой свертке соответствует объединение двух узлов или листьев под общим предком. Узлу предка присваивается тэг-претерминал, который соответствует правилу грамматики, кроме того предок принимает граммемы главного члена группы, например, в глагольной группе V[tran] PRCL (e.g. *«хотел бы»*) признаки будут приняты от транзитивного глагола V[tran], а не от частицы PRCL; а в именной группе NP[case='nomn'] NP[case='gent'] (e.g.*«отец детей»*) признаки будут приняты от существительного в номинативе.

Важно заметить, что свертка происходит в установленном порядке:

def reduce(self):

self.reduce\_ADJ() # прилагательные

self.reduce\_NP() # именные группы

self.reduce\_PP() # предложные

self.reduce\_VP() # глагольные

self.reduce\_S() # полные и неполные предложения

Такой порядок важен, так как исключает возможность «упустить» некоторые члены предложения. Сначала формируются прилагательные вместе с модификаторами (e.g. *безумно красивый*), затем именные группы, предложные и наконец глагольные. После этого идет поиск полных/неполных предложений, если таковые отсутствуют, то дерево не имеет корня, а значит и предложение не принадлежит языку грамматики.  
Рассмотрим условный пример построение дерева (рисунок 4):

sent = "Я пишу письмо старому другу"

def build(self, sent):

for word in sent:

new\_node = Node(word[0], word[1], word[2], leaf=True)

self.nodes.append(new\_node)



Рисунок 4 – Первый шаг построения дерева

NP[case='nomn'] -> NPRO[case='nomn']

NP[case='accs'] -> N[case='accs']

NP[case='datv'] -> ADJ[case='datv'] NP[case='datv']

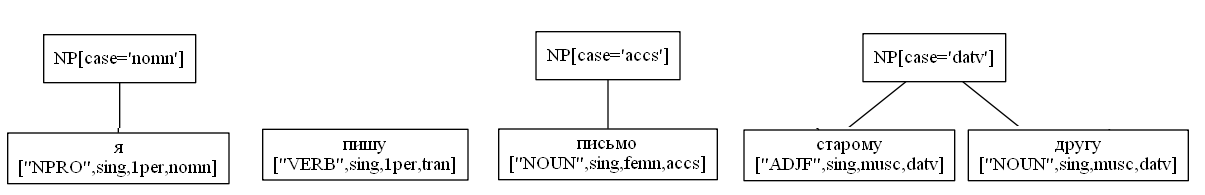


Рисунок 5 – Второй шаг построения дерева

VP[tran] -> V[tran] NP[case='accs']

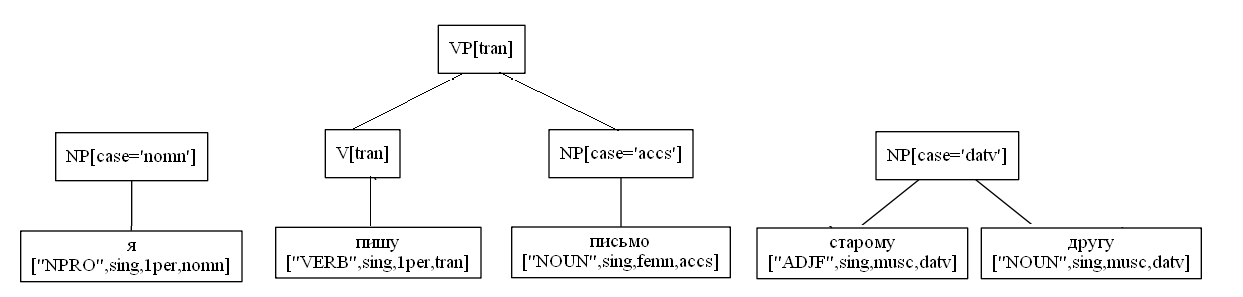


Рисунок 6 – Третий шаг построения дерева

VP[tran] -> VP[tran] NP[case='datv']

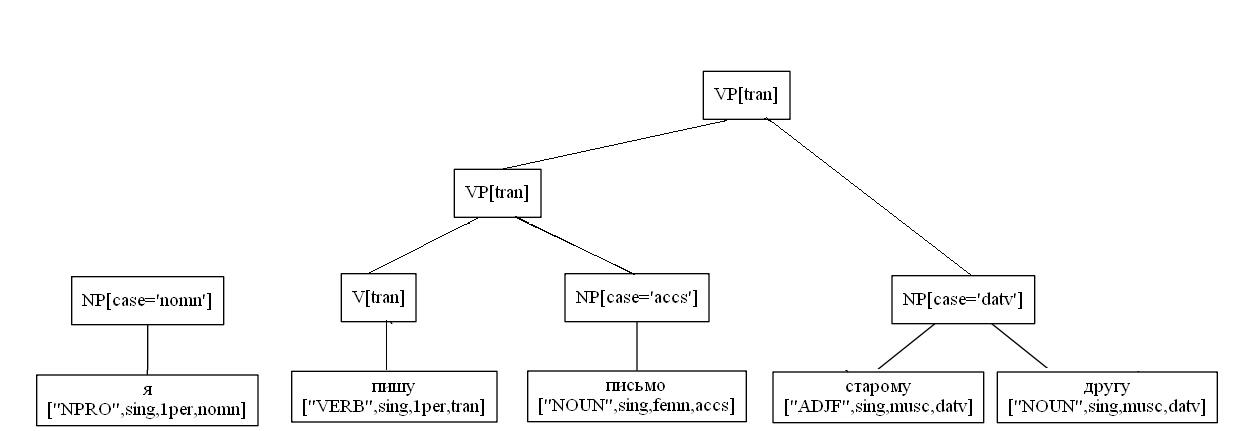


Рисунок 7 – Четвертый шаг построения дерева

S -> NP[case='nomn'] VP[tran]

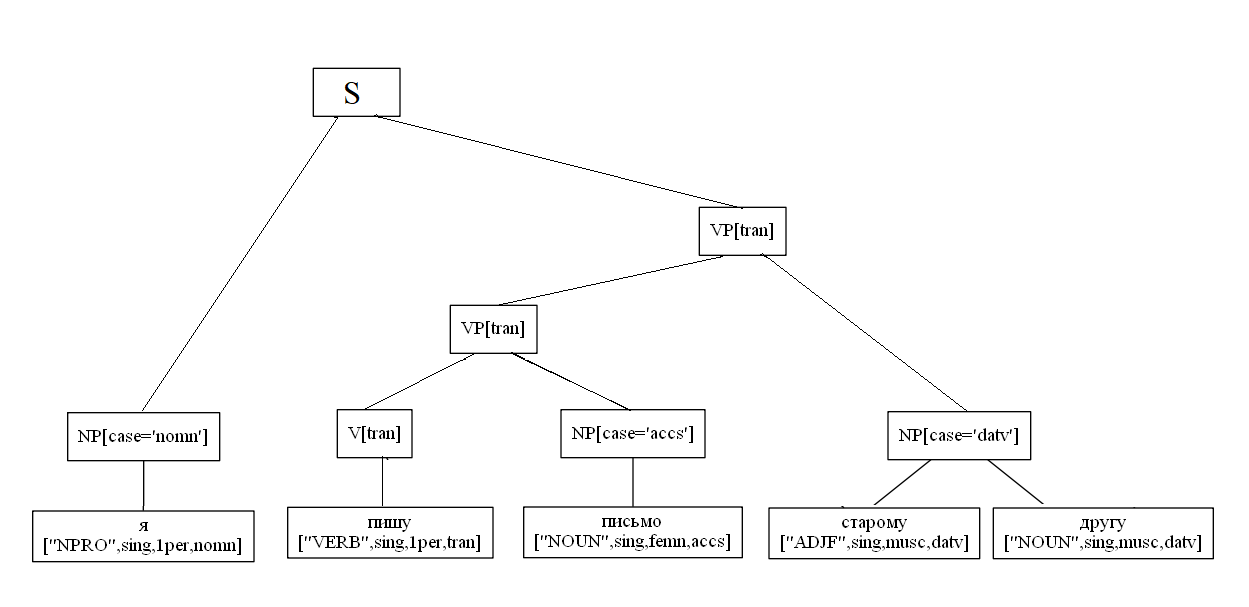


Рисунок 8 – Пятый шаг построения дерева

Конкретный пример разбора двусоставного предложения:

import analyzer

parser = analyzer.Parser()

sent = "Пустыня внемлет богу, и звезда с звездою говорит."

t = parser.parse(sent)

t[0].display()

S

NP[case='nomn']

Пустыня ['NOUN', 'sing', 'femn', 'nomn']

VP[tran]

VP[tran]

внемлет ['VERB', 'sing', '3per', 'tran', 'pres']

NP[case='datv']

богу ['NOUN', 'sing', 'datv']

S

NP[case='nomn']

звезда ['NOUN', 'sing', 'femn', 'nomn']

VP[tran]

PP

PREP

с ['PREP']

NP[case='ablt']

звездою ['NOUN', 'sing', 'femn', 'ablt']

VP[tran]

говорит ['VERB', 'sing', '3per', 'tran', 'pres'] [7].

**3 Проектирование сервиса проверки корректности ввода персональных данных**

**3.1 О возможных ошибках при вводе паспортных данных**

Информационные технологии c каждым днем больше и больше затрагивают области человеческой жизни. И на сегодняшний день, под натиском инновационных и телекоммуникационных технологий нужно вводить информационные системы в тех областях, где они не распространены и слабо развиваются и которые смогут минимизировать экономические затраты, затраты временные по обработке информации, и максимально повысить производительность труда.

Таким образом, все эти обстоятельства стали причиной создания данной технологи.

Проверка корректности заполнения паспортных данных используется во всех структурах, имеющих право на их приём и обработку, к примеру: сотрудниками банковской сферы, таможенного и паспортно-визового контроля, МВД, налоговой инспекции и это далеко не все структуры, которые могут использовать данную технологию. Это приложение упрощает работу сотрудников, исключает неправильный ввод данных и возможность ошибок, которые могли бы повлечь за собой вытекающие из этого последствия. В паспорте указывается главная и основная информация о человеке: фамилия, имя, отчество, пол, дата и место рождения, дата выдачи, место регистрация, серия и номер, код подразделения, информация об органе, выдавшем документ, о семейном положении и воинской обязанности, ранее выданных удостоверениях личности, а также в него можно вносить информацию медицинского характера.

Возможные ошибки:

* В фамилии, имени и отчестве нельзя использовать цифровые и буквенные коды, а также символы, ранги (поручик), должности и числительные.
* Серия паспорта- четыре цифры, в которой первые две цифры – ОКАТО-Код региона, а вторые две – год печати бланка. Таким образом, первая пара проверяется на наличие в базе и существование данного региона, а вторая на год в диапазоне 1997–2022 (это годы утверждения современного типа бланка)
* Номер паспорта - шесть цифр, которые уникальны. Код подразделения- шесть цифр, первые две - номер региона (используют более привычный номер субъекта в конституции), третья цифра означает уровень этого подразделения, уровней четыре:

а)0 — УФМС;

б)1 — ГУВД или МВД региона;

в)2 — УВД или ОВД района или города;

г) 3 — отделение полиции (обычно в селе или деревне).

* Дата выдачи паспорта указывается в соответствии с законом в определенном промежутке, так как он выдается в 14, 20 и 45 лет, еще учитывается, что на замену паспорта дается 30 дней.
* Подразделение, которым выдан паспорт, место рождения и жительства, где корректного должны быть указаны аббревиатуры, также сокращения, если они есть. [8]

При вводе данных допущенная ошибка может привести к недействительности документов, также к различным ограничениям и проблемам с законом.

Также стоит сказать о задачах, возникающих при разработке программы

* задача корректного ввода данных,
* задача считывания строки,
* задача проверки, написанного текста,
* задача выделения ошибок,
* задача исправления ошибок,
* задача визуализации результата

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

**3.2 Основные методы нахождения ошибок**

Компьютерная технология характеризуется рядом особенностей, которые следует учитывать при оценке условий и процедур контроля. Отличия компьютерной обработки данных от неавтоматизированной, в основном, следующие:

Единообразное выполнение операций.

Компьютерная обработка предполагает использование одних и тех же команд при выполнении идентичных операций учета, что практически исключает появлению случайных ошибок, обыкновенно присущих ручной обработке. Напротив, программные ошибки (или другие систематические ошибки в аппаратных либо программных средствах) приводят к неправильной обработке всех идентичных операций при одинаковых условиях.

Потенциальные возможности появления ошибок и неточностей.

По сравнению с неавтоматизированными системами учета компьютерные системы более открыты для несанкционированного доступа, включая лиц, осуществляющих контроль. Они также открыты для скрытого изменения данных и прямого или косвенного получения информации об активах. Чем меньше человек вмешивается в машинную обработку операций учета, тем ниже возможность выявления ошибок и неточностей. Ошибки, допущенные при разработке или корректировке прикладных программ, могут оставаться незамеченными на протяжении длительного периода.

Инициирование выполнения операций в компьютере.

Компьютерная система может выполнять некоторые операции автоматически, причем их санкционирование не обязательно документируется, как это делается в неавтоматизированных системах учета, поскольку сам факт принятия такой системы в эксплуатацию администрацией предполагает в неявном виде наличие соответствующих санкций .

Раскрывая, эту тему рассмотрим различные способы проверки, вводимых данных, например:

Посимвольная проверка

Как правило такие проверки выполняются в пользовательском интерфейсе, по мере ввода данных.

Проверка отдельных значений

Для пользовательского интерфейса это проверка значения в отдельном поле, причём выполняться она может как по мере ввода (проверяется то неполное значение, которое введено к настоящему моменту), так и после завершения ввода, когда поле теряет фокус. Для программного интерфейса (API) это проверка одного из параметров, переданных в вызываемую процедуру. Для данных, получаемых из файла, это проверка какого-то прочитанного фрагмента файла.

Совокупность входных значений

Можно предположить, что в программу сначала передаются какие-то данные, после чего подаётся некоторый сигнал, который инициирует их обработку. Например, пользователь ввёл данные в форму или в несколько форм и наконец нажал кнопку «OK». В этот момент можно выполнить так называемые «семантические» проверки, нацеленные на валидацию не только отдельных значений, но и взаимосвязей между ними, взаимных ограничений.

Проверка состояния системы после обработки данных

Наконец, есть последний способ, к которому можно прибегнуть, если валидацию непосредственно входных данных выполнить не удаётся — можно попытаться их обработать, но оставить возможность вернуть всё к исходному состоянию. Такой механизм часто называется транзакционным [9].

* 1. **Проектирование сервиса ввода данных на основе графематического анализа**

Для реализации проекта был использован язык Python с подключением кросс-платформенной событийно-ориентированной графической библиотеки tkinter. Программа содержит одиннадцать функций, считывающих и исправляющих введенные персональные данные. Рассмотрим работу некоторых из них.

Первым примером является функция, обрабатывающая поле с фамилией, именем и отчеством.

def fio():   
 if not entry\_fio.get().isdigit():  
 tmp = entry\_fio.get().title()  
 entry\_fio.delete(0, END)  
 entry\_fio.insert(0, tmp)  
 else:  
 entry\_input\_error(entry\_fio)

Функция fio() проверяет введенные данные, если они не являются численными, то все первые буквы преобразуются в заглавные. Далее удаляются введенные пользователем данные и вставляется откорректированная информация.

При некорректном вводе данных будет выведено «Некорректный ввод!».

Вторым примером является функция, обрабатывающая информацию о поле.

def gender():   
 if [i for i in m\_gender\_list if i == entry\_gender.get()]:  
 entry\_gender.delete(0, END)  
 entry\_gender.insert(0, "муж.")  
 elif [i for i in w\_gender\_list if i == entry\_gender.get()]:  
 entry\_gender.delete(0, END)  
 entry\_gender.insert(0, "жен.")  
 else:  
 entry\_input\_error(entry\_gender)

Функция gender() ищет совпадение введенных данных с данными в списке m\_gender\_list. При совпадении, введенные пользователем данные удаляются и вставляется стандарт записи пола. Иначе, поиск совпадения будет происходит в списке w\_gender\_list и производится таже процедура по удалению и вставлению стандарта записи. Если совпадения не были найдены ни в одном из списков на экран будет введено «Некорректный ввод!».

Третьим примером является функция, обрабатывающая информацию о серии и номере паспорта.

def series\_and\_number():   
 if entry\_series.get().isdigit():  
 if len(entry\_series.get()) == 4:  
 entry\_series.insert(0, entry\_series.get()[:2] + " " + entry\_series.get()[2:])  
 entry\_series.delete(5, END)  
 else:  
 entry\_input\_error(entry\_series)  
 elif len(entry\_series.get()) == 5:  
 if entry\_series.get()[0:2].isdigit() and entry\_series.get()[2:3] == " " \  
 and entry\_series.get()[3:5].isdigit():  
 pass  
 else:  
 entry\_input\_error(entry\_series)  
 else:  
 entry\_input\_error(entry\_series)  
  
 if entry\_number.get().isdigit():  
 if len(entry\_number.get()) != 6:  
 entry\_input\_error(entry\_number)  
 else:  
 entry\_input\_error(entry\_number)

Функция series\_and\_numbe() проверяет введенные данные на принадлежность числовому классу, затем проверяет длина введенной серии, если оба условия удовлетворены, то данные серии корректируются и записываются в соответствии со стандартом. Также проверяется, если серия была введена через пробел, то она корректна и не изменяется. При некорректном вводе данных будет выведено «Некорректный ввод!».

Затем проверяется введенный номер, если он принадлежит числовому классу, но его длина не равна 6, будет выведено «Некорректный ввод!» и также, если введенная информация не принадлежит числовому классу, будет выведено «Некорректный ввод!».

Рассмотрим работу сервиса на конкретных примерах

Пример 1

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 9 – Введенные пользователем данные пример 1

На рисунке 9 показаны данные, введенные пользователем.

В примере 1 можно увидеть, что ошибки допущены в полях: Ф.И.О., дата и место рождения, пол, код подразделения, серия и номер, кем выдан паспорт и дата его выдачи.

Изображение выглядит как текст

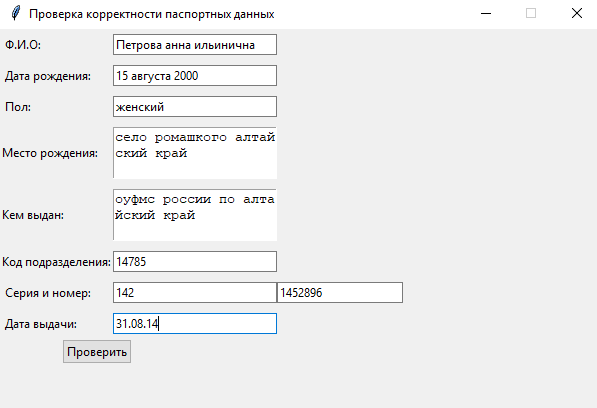
Автоматически созданное описание

Рисунок 10 – Результат работы сервиса пример 1

На рисунке 10 показан результат работы сервиса.

Все ошибки, допущенные при вводе персональных данных, были исправлены в соответствии со стандартом записи.

Пример 2



Рисунке 11 – Введенные пользователем данные пример 2

На рисунке 11 показаны данные, введенные пользователем.

В примере 2 можно увидеть, что ошибки допущены в полях: Ф.И.О., дата и место рождения, пол, код подразделения, серия и номер, кем выдан паспорт и дата его выдачи.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 12 – Результат работы сервиса пример 2

На рисунке 12 показан результат работы сервиса.

Ошибки в полях: Ф.И.О., пол, место рождения, кем выдан, дата рождения были исправлены в соответствии со стандартом записи.

В сервисе введено ограничение, поэтому в полях: дата рождения, код подразделения, серия и номер было выведено «Некорректный ввод!»

Таким образом, в курсовой работе разработан помощник ввода персональных данных, позволяющий осуществлять модификацию введенных пользователем персональных данных и приведение их к единому специальному стандарту.

**ЗАКЛЮЧЕНИЕ**

Цель курсовой работы- проектирование и реализация графематического анализатора ввода персональных данных - достигнута.

В теоретической части курсовой работы описаны базовые принципы компьютерной лингвистики, а также технологии реализации синтаксического анализа.

Выполнен реферативный обзор приложений, реализующих синтаксический анализ. Рассмотрены особенности ввода персональных данных, учтённые далее при реализации помощника.

В работе предложен проект и реализация сервиса проверки корректности ввода паспортных данных. Основной функцией помощника является модификация введенных пользователем персональных данных и приведение их к единому специальному стандарту.

При программной реализации функционала использовалась библиотека языка Python tkinter.

Разработанный сервис может являться составным элементом крупных проектов по работе с клиентами, например сотрудниками банковской сферы, таможенного и паспортно-визового контроля, МВД, налоговой инспекции.

**СПИСОК ИСПОЛЬЗОВАНЫХ ИСТОЧНИКОВ**

1. Проблема развития когнитивно-творческих способностей обучающихся в современных условиях: Коллективная монография / Под общ. ред. проф. Р.Х. Джураева. Ташкент: Шарк, 2012

2. Марчук, Ю. Н. Компьютерная лингвистика: учебное пособие / Ю.Н. Марчук. - М.: АСТ: Восток –Запад, 2007. – 317 с.

3. Касевич, В.Б. Элементы общей лингвистики / Касевич, В.Б., Маслов, О.С. // Москва: Наука, 1977 – 183 с.

4. Боярский К. К. Введение в компьютерную лингвистику. Учебное пособие. – СПб: НИУ ИТМО, 2013. – 72 с.

5. Первушин, А. Модуль графематического анализа в системе обработки русскоязычных текстов // Новые информационные технологии в автоматизированных системах. – 2012. – №15 (дата обращения 5.10.2022).

6. sshmakov: Разбор предложений по шаблонам русского языка // Habr: [сайт]. – URL: <https://habr.com/ru/post/350802/> (дата обращения 9.10.2022).

7. alexivan3232: Особенности создания синтаксического анализатора русского текста // Habr: [сайт]. – URL: <https://habr.com/ru/post/567268/> (дата обращения 9.10.2022).

8. DEADStop: Как проверить паспорт на действительность // Habr: [сайт]. – URL: <https://habr.com/ru/company/hflabs/blog/478538/> (дата обращения 13.10.2022).

9. barancev: Эссе о валидации данных // Habr: [сайт]. – URL: <https://habr.com/ru/post/72796/> (дата обращения 13.10.2022).