

Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет»
Кафедра ИИТ

Лабораторная работа №1

За шестой семестр

По дисциплине: «Естественно-языковой интерфейс ИС»

Тема: «Разработка автоматизированной системы формирования словаря естественного языка»

Выполнил:

Студент 3 курса

Группы ИИ-23

Романюк А. П

Проверил:

Булей Е. В.

Ход работы

Список слов, упорядоченный по алфавиту и включающий только лексемы с дополнительно оформленными записями о месте и роли данного слова в составе предложения. К такой информации относится описание того, каким членом предложения может быть данное слово и в какой форме (падеж, число, время и т.п.). Например, если это существительное в именительном падеже, то оно может выступать в роли подлежащего; если это существительное в родительном падеже, то оно может быть дополнением; если это прилагательное, то оно может быть определением и т.п.

Лексический анализатор с ролями слов

Александр Сергеевич Пушкин
Полное собрание стихотворений

Стихотворения 1809–1811 гг.

Dis moi, pourquoi l'Escamoteur
fist-il siffié par le parterre?
Hélas! c'est que le pauvre auteur
L'escamota de Moliére.

Перевод

Выбрать файл

Обработать текст

Сохранить результат

Загрузить словарь

Поиск...

Искать

Сбросить поиск

	Лексема	Словоформа	Часть речи	Падеж	Число	Время	Роль в предложении	Частота
1	а	а	союз	-	-	-	Союз	339
2	абамелек	абамелек	имя существительное	именительный	единственное	-	Подлежащее	1
3	август	августом	имя существительное	творительный	единственное	-	Дополнение	1
4	август	августу	имя существительное	дательный	единственное	-	Дополнение	1
5	август	августов	имя существительное	родительный	множественное	-	Дополнение	1
6	авдей	авдей	имя существительное	именительный	единственное	-	Подлежащее	3
7	авель	авеля	имя существительное	родительный	единственное	-	Дополнение	1

Код программы:

Gui.py

```
import re

from PySide6.QtWidgets import (QApplication, QFileDialog, QHBoxLayout,
                                QHeaderView, QMainWindow, QMessageBox,
                                QPushButton, QTableWidgetItem,
                                QTextEdit, QVBoxLayout, QWidget)

from data_manager import DataManager
from text_processor import TextProcessor

class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Лексический анализатор с ролями слов")
        self.setGeometry(100, 100, 800, 600)
        self.text_processor = TextProcessor()
        self.original_data = None
        self.init_ui()

    def init_ui(self):
        """Инициализация пользовательского интерфейса с улучшенным расположением элементов."""
        # Поле для ввода текста
        self.text_edit = QTextEdit(self)
        self.text_edit.setPlaceholderText(
            "Введите текст или выберите файл для обработки..."
```

```

)
self.text_edit.setMinimumHeight(100)

# Кнопки управления файлами
self.open_file_button = QPushButton("Выбрать файл", self)
self.process_button = QPushButton("Обработать текст", self)
self.save_button = QPushButton("Сохранить результат", self)
self.load_button = QPushButton("Загрузить словарь", self)

self.open_file_button.clicked.connect(self.open_file_dialog)
self.process_button.clicked.connect(self.process_text)
self.save_button.clicked.connect(self.save_results)
self.load_button.clicked.connect(self.load_results)

# Поле поиска и кнопка
self.search_edit = QTextEdit(self)
self.search_edit.setPlaceholderText("Поиск...")
self.search_edit.setFixedHeight(30) # Делаем поле поиска маленьким
self.search_button = QPushButton("Искать", self)
self.search_button.clicked.connect(self.search_results)

# Таблица для отображения результатов
self.table = QTableWidgetItem(self)
self.table.setColumnCount(8)
self.table.setHorizontalHeaderLabels(
    [
        "Лексема",
        "Словоформа",
        "Часть речи",
        "Падеж",
        "Число",
        "Время",
        "Роль в предложении",
        "Частота",
    ]
)
#self.table.horizontalHeader().setSectionResizeMode(QHeaderView.Stretch)

# Размещение кнопок в горизонтальный layout
file_buttons_layout = QHBoxLayout()
file_buttons_layout.addWidget(self.open_file_button)
file_buttons_layout.addWidget(self.process_button)
file_buttons_layout.addWidget(self.save_button)
file_buttons_layout.addWidget(self.load_button)

search_layout = QHBoxLayout()
self.reset_search_button = QPushButton("Сбросить поиск", self)
self.reset_search_button.clicked.connect(
    lambda: self.search_edit.setText("") or self.search_results()
)

search_layout.addWidget(self.search_edit, 2)
search_layout.addWidget(self.search_button, 1)
search_layout.addWidget(
    self.reset_search_button, 1
) # Новая кнопка сброса поиска

# Основной layout
layout = QVBoxLayout()
layout.addWidget(self.text_edit)
layout.addLayout(file_buttons_layout) # Кнопки работы с файлами
layout.addLayout(search_layout) # Поиск
layout.addWidget(self.table) # Таблица занимает оставшееся пространство

container = QWidget()
container.setLayout(layout)
self.setCentralWidget(container)

def process_text(self):
    """Обработка текста и вывод результатов в таблицу."""
    text = self.text_edit.toPlainText().lower()
    if text:
        words = self.extract_words(text)

```

```

        sorted_lemmas = self.text_processor.process_text(words)
        self.original_data = sorted_lemmas
        self.update_table(self.original_data)
    else:
        QMessageBox.warning(self, "Ошибка", "Поле ввода текста пусто!")

    @staticmethod
    def extract_words(text):
        """Извлекает слова из текста."""
        pattern = re.compile(r"[а-яА-Я]+")
        return pattern.findall(text)

    def open_file_dialog(self):
        """Открытие диалога выбора файла."""
        file_dialog = QFileDialog(self)
        file_dialog.setNameFilter("Текстовые файлы (*.txt *.rtf *.pdf *.doc *.docx)")
        file_dialog.setViewMode(QFileDialog.Detail)
        if file_dialog.exec():
            file_paths = file_dialog.selectedFiles()
            if file_paths:
                file_path = file_paths[0]
                self.load_file(file_path)

    def load_file(self, file_path):
        """Загрузка текста из файла."""
        try:
            text = self.read_file(file_path)
            self.text_edit.setText(text)
        except Exception as e:
            QMessageBox.critical(self, "Ошибка", f"Не удалось загрузить файл: {str(e)}")

    def load_results(self):
        """Загрузка данных из JSON в таблицу."""
        filename, _ = QFileDialog.getOpenFileName(
            self, "Открыть файл", "", "JSON файлы (*.json)"
        )
        if filename:
            try:
                data = DataManager.load_data(filename)
                self.original_data = data
                self.update_table(self.original_data)
                QMessageBox.information(self, "Успех", "Данные успешно загружены.")
            except Exception as e:
                QMessageBox.critical(
                    self, "Ошибка", f"Не удалось загрузить данные: {str(e)}"
                )

    def search_results(self):
        """Поиск данных по ключевому слову"""
        keyword = self.search_edit.toPlainText().strip()
        if not keyword:
            self.update_table(self.original_data)
            return
        filtered_data = DataManager.search_data(self.original_data, keyword)
        self.update_table(filtered_data)

    def read_file(self, file_path):
        """Чтение текста из файла."""
        if file_path.endswith(".txt"):
            with open(file_path, "r", encoding="utf-8") as file:
                return file.read()
        elif file_path.endswith(".pdf"):
            import PyPDF2

            with open(file_path, "rb") as file:
                reader = PyPDF2.PdfReader(file)
                return "".join(page.extract_text() for page in reader.pages)
        elif file_path.endswith(".docx"):
            import docx

            doc = docx.Document(file_path)
            return "\n".join([para.text for para in doc.paragraphs])
        elif file_path.endswith(".rtf"):

```

```

import striprt

with open(file_path, "r", encoding="utf-8") as file:
    rtf_text = file.read()
    return striprt.rtf_to_text(rtf_text)
else:
    raise ValueError("Неподдерживаемый формат файла")

def save_results(self):
    """Сохранение результатов в JSON в формате, аналогичном выходному формату process_text."""
    filename, _ = QFileDialog.getSaveFileName(
        self, "Сохранить файл", "", "JSON файлы (*.json)"
    )
    if filename:
        try:
            DataManager.save_data(self.original_data, filename)
            QMessageBox.information(self, "Успех", "Данные успешно сохранены.")
        except Exception as e:
            QMessageBox.critical(
                self, "Ошибка", f"Не удалось сохранить данные: {str(e)}"
            )

def update_table(self, sorted_lemmas):
    """Обновление таблицы с результатами."""
    self.table.setRowCount(0)
    for lemma_info in sorted_lemmas:
        lemma = lemma_info["lemma"]
        for form_info in lemma_info["word_forms"]:
            row_position = self.table.rowCount()
            self.table.insertRow(row_position)
            self.table.setItem(row_position, 0, QTableWidgetItem(lemma)) # Лексема
            self.table.setItem(
                row_position, 1, QTableWidgetItem(form_info["word"])
            ) # Словоформа
            self.table.setItem(
                row_position, 2, QTableWidgetItem(form_info["pos"])
            ) # Часть речи
            self.table.setItem(row_position, 3, QTableWidgetItem(form_info["case"]))
            self.table.setItem(
                row_position, 4, QTableWidgetItem(form_info["number"])
            )
            self.table.setItem(
                row_position, 5, QTableWidgetItem(form_info["tense"])
            )
            self.table.setItem(
                row_position, 6, QTableWidgetItem(form_info["role"])
            ) # Роль в предложении
            self.table.setItem(
                row_position, 7, QTableWidgetItem(str(form_info["count"]))
            ) # Частота
    self.table.resizeColumnsToContents()

if __name__ == "__main__":
    app = QApplication([])
    window = MainWindow()
    window.show()
    app.exec()

```

Вывод: в ходе выполнения лабораторной работы освоил принципы разработки прикладных сервисных программ для решения задачи автоматического лексического и лексико-грамматического анализа текста естественного языка.