Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет Информационных технологий и управления

Кафедра Интеллектуальных информационных технологий

**ОТЧЁТ**

по дисциплине «Естественно-языковой интерфейс интеллектуальных систем»

Лабораторная работа №1

Тема: Разработка автоматизированной системы

формирования словаря естественного языка

Выполнили: Демидовец Д.В.

Козырев Д.А.

гр. 221703

Проверил: Крапивин Ю.Б.

Минск 2025

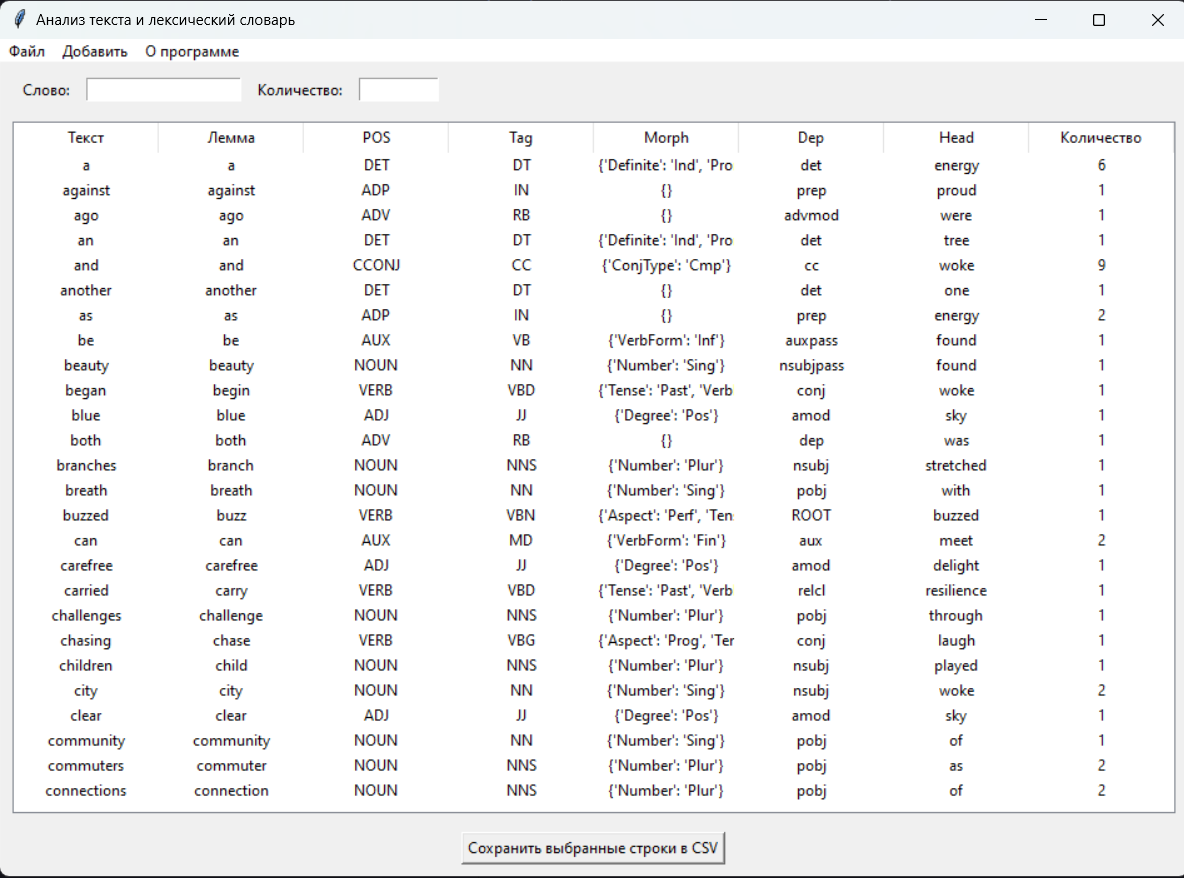
**Цель работы:**

Освоить принципы разработки прикладных сервисных программ для решения задачи автоматического лексического и лексико-грамматического анализа текста естественного языка.

**Задание:**

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.

**Интерфейс:**

****

**Структура хранения данных:**

****

**Части речи(POS -- Part of speech):**

| Тег | Расшифровка | Пример |
| --- | --- | --- |
| ADJ | Adjective — прилагательное | big, quick, beautiful |
| ADP | Adposition — предлог или послелог | in, on, at, under |
| ADV | Adverb — наречие | very, quickly, silently |
| AUX | Auxiliary verb — вспомогательный глагол | is, have, do |
| CCONJ | Coordinating conjunction — сочинительный союз | and, but, or |
| DET | Determiner — определитель | a, the, this, my |
| INTJ | Interjection — междометие | wow, ouch, hey |
| NOUN | Noun — существительное | cat, house, apple |
| NUM | Numeral — числительное | one, two, 3, fifty |
| PART | Particle — частица | not, 's, to |
| PRON | Pronoun — местоимение | I, you, he, she, it |
| PROPN | Proper noun — имя собственное | London, Alice, Microsoft |
| PUNCT | Punctuation — пунктуация | ., !, ? |
| SCONJ | Subordinating conjunction — подчинительный союз | although, because |
| SYM | Symbol — символ | $, %, +, = |
| VERB | Verb — глагол | run, eat, sleep |
| X | Other — другое / неизвестно | (редкие случаи) |
| SPACE | Пробелы | (пустые символы) |

**Синтаксическая роль(Dep):**

| Значение (dep\_) | Описание по-русски | Пример |
| --- | --- | --- |
| ROOT | Главный член предложения (глагол-сказуемое) | She eats apples. *(eats = ROOT)* |
| nsubj | Подлежащее (именное) | Tom eats apples. *(Tom = nsubj)* |
| nsubjpass | Подлежащее в пассивном залоге | The cake was eaten by Tom. *(cake = nsubjpass)* |
| obj | Прямое дополнение | She eats apples. *(apples = obj)* |
| iobj | Косвенное дополнение | She gave him a book. *(him = iobj)* |
| csubj | Подлежащее, представленное придаточным | That she left surprised me. *(That she left = csubj)* |
| ccomp | Придаточное как дополнение | He said that he was tired. *(that he was tired = ccomp)* |
| xcomp | Несамостоятельное дополнение | She wants to sleep. *(to sleep = xcomp)* |
| obl | Обстоятельство | He sleeps in the room. *(in the room = obl)* |
| advmod | Наречное обстоятельство | He runs quickly. *(quickly = advmod)* |
| amod | Атрибутивное прилагательное | The red apple. *(red = amod)* |
| det | Определитель (артикль, указательное слово) | The dog barked. *(The = det)* |
| compound | Составное существительное | Credit card payment. *(Credit = compound)* |
| nummod | Числительное модификатор | Two cats. *(Two = nummod)* |
| appos | Приложение | My friend, a doctor, helped me. *(a doctor = appos)* |
| acl | Определительное придаточное | The book that you gave me. *(that you gave = acl)* |
| relcl | Относительное придаточное предложение | The man who came yesterday. *(who came = relcl)* |
| conj | Сочинительная связь | Apples and oranges. *(oranges = conj)* |
| cc | Союз | Apples and oranges. *(and = cc)* |
| mark | Подчинительный союз | She left because it rained. *(because = mark)* |
| case | Предлог | In the room. *(in = case)* |
| aux | Вспомогательный глагол | She is running. *(is = aux)* |
| auxpass | Вспомогательный глагол пассива | The cake was eaten. *(was = auxpass)* |
| cop | Связочный глагол (быть, становиться) | She is a teacher. *(is = cop)* |
| punct | Знак препинания | Hello, ! *(punct)* |
| parataxis | Параллельная конструкция | "Let it be," he said. *(Let it be = parataxis)* |
| discourse | Слова-связки или вводные слова | Well, I don't know. *(Well = discourse)* |
| expl | Формальное подлежащее | It is raining. *(It = expl)* |
| vocative | Обращение | John, come here! *(John = vocative)* |
| neg | Отрицание | He does not know. *(not = neg)* |

В английском языке морфологические признаки у слова зависят от его **части речи**. **Основные признаки:**

1. Для существительных (NOUN, PROPN):

* Number — число: Sing (единственное) / Plur (множественное)

2. Для глаголов (VERB, AUX):

* Tense — время: Past, Pres, Fut
* Mood — наклонение: Ind (изъявительное), Imp (повелительное), Subj (сослагательное)
* VerbForm — форма глагола: Fin (личная форма), Inf (инфинитив), Part (причастие), Ger (герундий)
* Aspect — вид: Perf (совершенный) / Prog (продолженный)
* Voice — залог: Act (действительный) / Pass (страдательный)
* Person — лицо: 1, 2, 3
* Number — число: Sing, Plur

3. Для прилагательных (ADJ) и наречий (ADV):

* Degree — степень сравнения:  
  Pos (обычная), Cmp (сравнительная), Sup (превосходная)

4. Для местоимений (PRON):

* Case — падеж: Nom (именительный), Acc (объектный), Gen (притяжательный)
* Person — лицо
* Number — число

**Структурная схема приложения:**

****

**Использованные библиотеки:**

* Библиотека **spaCy** применяется для обработки текста на естественном языке. С её помощью выполняется разбиение текста на токены, лемматизация (приведение слова к начальной форме), определение частей речи, морфологических признаков, синтаксических зависимостей и связи слов в предложении.
* Библиотека **tkinter** используется для создания графического интерфейса пользователя. С её помощью реализованы окно программы, меню, кнопки, поля для ввода, таблица для отображения токенов, а также обработка пользовательских событий, например, двойной клик по строке таблицы для редактирования данных.
* Модуль **dataclasses** применяется для удобного хранения информации о лексемах. С помощью декоратора @dataclass был создан класс TokenData, который описывает структуру данных для хранения текста слова, его леммы, части речи, морфологических признаков и других свойств.
* Библиотека **csv** используется для сохранения результатов работы программы в CSV-файл. Она позволяет экспортировать выбранные пользователем строки из таблицы в виде структурированного текстового файла с разделителями.
* Ещё одна библиотека — **python-docx** — служит для работы с документами формата DOCX. Она применяется как для чтения исходного текста из Word-файла, так и для сохранения результатов анализа в новый DOCX-документ с добавлением заголовков и отформатированных абзацев.

**Описание алгоритмов:**

**1. Обработка текста из файла:**

Основной алгоритм для извлечения лексем из текста реализован в функции extract\_token\_info:

1. Пользователь загружает текст из DOCX-файла, текст передаётся в функцию extract\_token\_info.
2. Передадим входной текст в морфологический анализатор и получим список лексем.
3. Инициализируем пустой словарь для хранения информации о лексемах.
4. Последовательно переберём каждую лексему:
   * Если лексема состоит только из букв — приведём её к нижнему регистру.
   * Если лексема ранее не встречалась — создадим новую запись с основными признаками: начальная форма, часть речи, морфологические параметры, синтаксическая роль, головное слово и установим счётчик в 1.
   * Если лексема уже есть в словаре — увеличим её счётчик вхождений на единицу.
5. После завершения обработки текста — преобразуем словарь лексем в список.
6. Отсортируем список лексем по алфавиту без учёта регистра.
7. Вернём отсортированный список лексем как результат.

**2. Фильтрация лексем:**

Для удобства пользователя реализована функция auto\_filter, которая автоматически обновляет таблицу при вводе текста в поля фильтра. Алгоритм работает так:

1. Получим текстовый фильтр и числовой фильтр от пользователя. Приведём текст к нижнему регистру и удалим пробелы.
2. Если оба фильтра пусты — скопируем исходный список лексем в результирующий и перейдём к пункту 7.
3. Если оба фильтра заполнены и числовой фильтр корректен, отберём из списка лексемы, которые содержат текстовый фрагмент и совпадают по счётчику вхождений. Перейдём к пункту 7.
4. Если заполнен только текстовый фильтр — отберём лексемы, в которых текст или лемма содержат указанный фрагмент. Перейдём к пункту 7.
5. Если заполнен только числовой фильтр и он корректен — отберём лексемы с подходящим счётчиком вхождений. Перейдём к пункту 7.
6. Если фильтры заданы некорректно или не удовлетворяют условиям — скопируем исходный список лексем в результирующий.
7. Обновим отображение списка лексем в пользовательском интерфейсе.

**3. Редактирование и удаление лексем:**

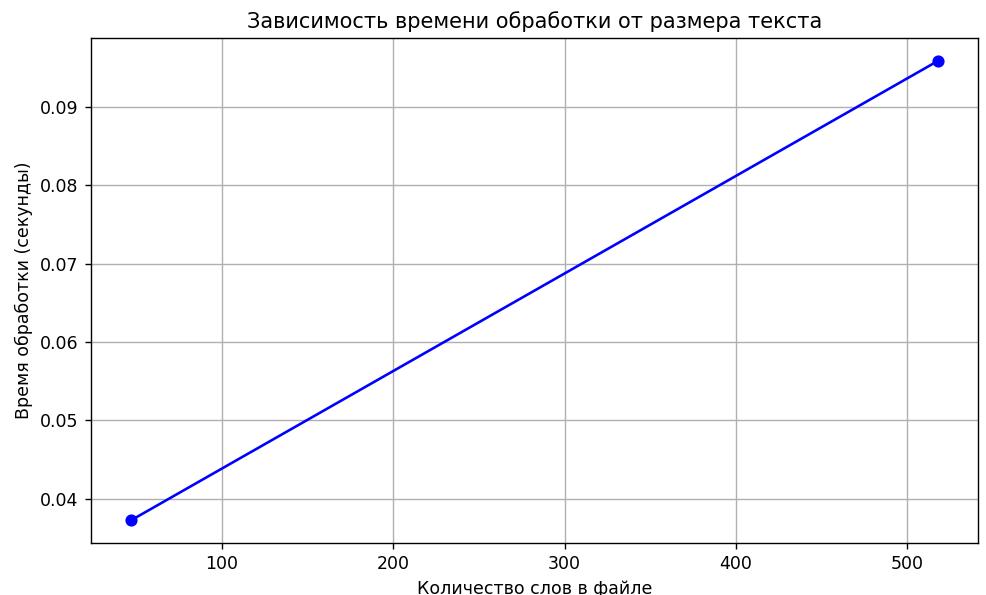
Программа поддерживает ручное изменение и удаление лексем с помощью диалогового окна:

1. Определим выбранную лексему в таблице по текущему выделению.
2. Если элемент не выбран — завершим работу алгоритма.
3. Считаем данные выбранной лексемы и попробуем создать объект с её параметрами.
4. Если данные некорректны — покажем сообщение об ошибке и завершим работу алгоритма.
5. Откроем диалоговое окно для изменения или удаления лексемы.
6. Если пользователь подтвердил действие:
   1. Если выбрана операция удаления — удалим лексему из списка.
   2. Если введены новые данные:
      1. Найдём индекс редактируемой лексемы в списке.
      2. Заменим старую лексему на новую.
      3. Скопируем актуальный список для отображения.
      4. Обновим отображение списка лексем.

**4. Отображение данных в графическом интерфейсе**

1. При запуске программы создается главное окно и задаются его свойства, после чего размещаются основные элементы: меню, текстовые поля, таблицы и кнопки.
2. Происходит связывание действий пользователя с обработчиками событий.
3. Таблица отображается при загрузке и обновляется после фильтрации и редактирования данных.
4. Для редактирования, просмотра текста и вывода сообщений открываются отдельные диалоговые окна.
5. Процесс ожидания новых действий пользователя продолжается, пока запущен цикл событий.
6. Цикл событий прерывается, как только пользователь закрывает окно программы.

**Тест производительности:**

****

В ходе эксперимента было проведено измерение времени выполнения программы при обработке текстов разного объёма. Для теста были использованы два файла short\_text.docx с размером текста примерно в 50 слов и large\_text.docx с размером примерно в 500 слов. Полученный график демонстрирует прямую зависимость времени выполнения от размера текста: чем больше слов содержит документ, тем больше времени требуется на его обработку.

Такой результат объясняется тем, что для каждого слова выполняется последовательный анализ с использованием морфологического разборщика, что линейно увеличивает общее время выполнения по мере роста количества входных данных. Этот результат подтверждает предсказуемость и стабильность работы программы при увеличении размера текстовых файлов.

**Вывод:**

В ходе выполнения лабораторной работы было разработано приложение для анализа текста на английском языке, позволяющее обрабатывать документ формата .docx и автоматически извлекать лингвистическую информацию о каждом слове. В результате работы программа формирует структурированный словарь, содержащий основную информацию о словах: лемму, количество вхождений, часть речи, падеж, число и другие грамматические характеристики. Разработанная структура хранения данных обеспечивает наглядное и удобное представление информации, что значительно упрощает последующую обработку и анализ текста. В перспективе разработанный функционал можно расширять и адаптировать для других языков, что увеличит область применения в образовательной, исследовательской и редакторской практике.