МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Кафедра «Прикладная математика и информатика»
(наименование)

01.03.02 «Прикладная математика и информатика»
(код и наименование направления подготовки / специальности)

Компьютерные технологии и математическое моделирование
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему «Применение методов обработки естественного языка для автоматического аннотирования текстов»

Обучающийся А.С. Николаев
(Инициалы Фамилия) (личная подпись)

Руководитель М.А. Тренина
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Консультант к.филол. н., доцент, Н.В. Ященко
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2025

Аннотация

Тема бакалаврской работы – «Применение методов обработки естественного языка для автоматического аннотирования текстов».

Аннотирование тестов, позволяющее выделять и структурировать значимую информацию, является основой систем обработки естественного языка в областях от информационного поиска и анализа новостей до машинного перевода и разработки чат-ботов. Поэтому тема работы актуальна.

Целью выпускной квалификационной работы является создание системы автоматического аннотирования текстов методами обработки естественного языка.

Задачи работы:

- провести анализ существующих алгоритмов автоматического аннотирования текстов;
- выполнить проектирование математической модели;
- осуществить программную реализацию приложения.

Работа содержит введение, три раздела, заключение и перечень используемой литературы и источников.

Объем работы – 50 страниц текста, 11 рисунков, 7 таблиц и 20 источников.

Abstract

The topic of the bachelor's thesis is ‘Application of natural language processing methods for automatic annotation of texts’.

Test annotation, which allows to highlight and structure meaningful information, is the basis of natural language processing systems in areas ranging from information retrieval and news analysis to machine translation and chatbot development. Therefore, the topic of the work is relevant.

The purpose of the graduate qualification work is to create a system of automatic annotation of texts by methods of natural language processing.

Tasks of the work:

– to analyse existing algorithms for automatic annotation of texts;

– to perform the design of the mathematical model;

– to perform the software implementation of the application.

The work contains an introduction, three sections, conclusion and a list of literature and sources used.

The volume of the work is 49 pages of text, 11 figures, 7 tables and 20 sources.

Содержание

Введение

В эпоху стремительного роста объемов текстовой информации автоматизация процессов ее обработки становится критически важной задачей. Одним из ключевых направлений является аннотирование текстов, позволяющее выделять и структурировать значимую информацию, что значительно упрощает дальнейший анализ и использование. Применение аннотирования находит широкое применение в различных областях, от информационного поиска и анализа новостей до машинного перевода и разработки чат-ботов. В связи с этим, разработка эффективных и автоматизированных систем аннотирования текстов приобретает особую актуальность.

Объектом исследования является процесс автоматического аннотирования тестов.

Предметом исследования является программное обеспечение автоматического аннотирования текстов.

Целью выпускной квалификационной работы является создание системы автоматического аннотирования текстов методами обработки естественного языка. Для достижения этой цели необходимо решить следующие задачи:

- провести анализ существующих алгоритмов автоматического аннотирования текстов;
- выполнить проектирование математической модели;
- осуществить программную реализацию приложения.

Работа содержит введение, три раздела, заключение и перечень используемой литературы и источников.

В первом разделе проведен анализ современных методов и инструментов, применяемых для автоматического аннотирования текстов методами обработки естественного языка. На основе анализа определены варианты для более подробного исследования, иллюстрирующие применение

специализированных и универсальных средств обработки естественного языка для аннотирования текстов на русском языке.

Во втором разделе представлено исследование моделей и алгоритмов аннотирования текста с использованием специализированных и универсальных средств обработки естественного языка. Рассмотрены модели, методы и алгоритмы специализированной библиотеки SpaCy, универсальных библиотек NLTK и Pymorphy2. Приведены алгоритмы использования словарей и регулярных выражений при аннотировании текста.

В третьем разделе разработано программное обеспечение и выполнено экспериментальное исследование эффективности аннотирования текста с использованием специализированных и универсальных средств обработки естественного языка. В рамках третьего раздела выполнено проектирование алгоритмов программ и выбор средств реализации, разработаны программы для аннотирования текста с использованием универсальных и специализированных средств. По результатам экспериментальной проверки выполнено совершенствование программы аннотирования универсальными средствами, в результате чего качество аннотирования универсальной программой существенно повысилось.

В заключении представлены результаты выпускной квалификационной работы.

1 Анализ существующих методов, алгоритмов и инструментов аннотирования текста

1.1 Обзор методов и инструментов аннотирования текста

NLP-аннотирование текста – это процесс добавления метаданных к тексту, чтобы сделать его более понятным для компьютеров. Этот процесс включает несколько этапов (таблица 1), каждый из которых предназначен для извлечения определенного типа информации из текста.

Таблица 1 – Этапы аннотирования текста

Этап	Описание	Пример вход)	Пример (выход)	Методы и инструменты
Токенизация	Разбиение текста на токены (слова, символы, пунктуация)	"Мама мыла раму."	["Мама", "мыла", "раму”,	Пробелы, Regex, NLTK, spaCy, UDPipe, Subword
Частеречная разметка	Определение части речи каждого токена	"Мама мыла раму."	[("Мама", "NOUN"), ("мыла", "VERB"), ("раму”,	HMM, CRF, BiLSTM, Трансформеры,
Лемматизация	Приведение слова к словарной форме (лемме)	"Мыла, мыл, моет"	["мыть”, "мыть", "мыть"]	Словари, правила,
	Определение и классификация именованных сущностей	"Путин посетил Москву вчера."	[("Путин", "PER"), ("Москву", "LOC"), ("вчера", "DATE")]	Правила, CRF, BiLSTM-CRF, Трансформеры,

Проведем анализ методов и инструментов, применяемых на каждом этапе, с целью подготовить базу для выбора инструментов и методов в зависимости от конкретной задачи, требуемой точности, доступных вычислительных ресурсов и объема данных.

Токенизация имеет целью разбиение текста на отдельные значимые единицы (токены), такие как слова, знаки препинания, числа.

Рассмотренные методы и инструменты токенизации показаны в таблице 2.

Простая токенизация на основе пробелов разделяет текст на токены по пробелам. Это очень быстрое и простое в реализации решение, но оно плохо работает с русским языком, так как пунктуация часто примыкает к словам (например, "привет, мир!"), не учитывает сокращения, составные слова (например, "Санкт-Петербург") и переносы строк. Простая токенизация на основе пробелов может использоваться как базовый этап для дальнейшей модификации и улучшений в задачах, где не требуется высокая точность токенизации.

Токенизация на основе регулярных выражений (Regex-based tokenization) использует регулярные выражения для определения границ токенов. Этот подход позволяет более гибко учитывать знаки препинания, сокращения, сложные словоформы, значительно лучше справляется с пунктуацией и некоторыми другими особенностями языка, позволяет настроить правила для специфических случаев. Вместе с тем, токенизация на основе регулярных выражений требует знания регулярных выражений и времени на разработку и отладку правил, потенциально сложна в поддержке и расширении набора правил.

Библиотека NLTK (Python) [18] – это широко известная библиотека для обработки естественного языка. Она содержит множество инструментов для различных задач NLP, но изначально разработана для английского языка и требует адаптации для русского языка состоящей в написании собственных регулярных выражений или использовании предобученных моделей [1].

Библиотека spaCy (Python) [15] – это современная библиотека для NLP, ориентированная на скорость и эффективность.[2] Она имеет предобученные модели для русского языка (ru_core_news_sm, ru_core_news_md, ru_core_news_lg), которые включают токенизацию [14].

Таблица 2 – Инструменты и методы токенизации

Метод/ Инструмент	Точность	Скорость	Простота реализации	Адаптируемость	Ресурсы	Язык	Контролируемость	Примечания
Пробелы	Низкая	Высокая	Высокая	Низкая	Низкие		Высокая	Плохо для русского из-за пунктуации.
	Средняя	Средняя	Средняя	Средняя	Низкие		Высокая	Требует разработки regex, но более гибкий, чем пробелы.
	Средняя	Средняя	Средняя	Средняя	Низкие		Высокая	Требует адаптации для русского.
	Высокая	Высокая	Простая	Средняя	Средние		Низкая	Хорошо работает из коробки, меньше контроля.
	Высокая	Средняя	Простая	Средняя	Средние		Низкая	Предобученная модель, хорошая точность, как и spaCy, меньше контроля.
	Высокая	Средняя	Средняя	Высокая	Средние		Средняя	Актуально для нейронных сетей, справляется с неизвестными словами, требует больше ресурсов

Эта библиотека обеспечивает хорошую производительность, но требует установки модели для русского языка и может быть избыточной, если нужна только токенизация.

Библиотека UDPipe [20] представляет собой конвейер обработки текста, предоставляющий предобученные модели для многих языков, включая русский. Она выполняет токенизацию, POS-теггинг, морфологический анализ и синтаксический анализ в рамках одного конвейера, но при этом может быть медленнее, чем специализированные библиотеки и требует установки модели [19].

Метод Subword tokenization (Byte Pair Encoding (BPE) [17], WordPiece, SentencePiece [7]) разбивает слова на более мелкие подслова и этим позволяет работать с неизвестными словами (OOV – out-of-vocabulary), которые не встречаются в обучающем корпусе. Этот метод уменьшает размер словаря, что важно для языков с богатой морфологией, таких как русский.[5]

Частеречная разметка (POS-теггинг) имеет целью присвоение каждому токену в тексте его части речи (например, существительное, глагол, прилагательное).

Рассмотренные методы и инструменты частеречной разметки показаны в таблице 3.

Методы, основанные на правилах, используют заранее заданные правила для определения части речи слова. Они могут быть очень точными для конкретных случаев, заданных правилами, но при этом их сложно реализовать для русского языка из-за его сложной морфологии и большого количества исключений. Методы, основанные на правилах, требуют значительных усилий по разработке и поддержке правил и плохо масштабируются.

Статистические методы, такие как скрытые марковские модели (Hidden Markov Models (HMM)) моделируют последовательность слов и частей речи как марковский процесс.[10]

Таблица 3 – Инструменты и методы частеречной разметки

Метод/ Инструмент	Точность	Скорость	Простота реализации	Адаптируемость	Ресурсы	Язык	Контролируемость	Примечания
Правила	Низкая	Высокая	Сложная	Низкая	Низкие		Высокая	Очень сложно поддерживать для русского языка из-за его морфологии.
	Средняя	Средняя	Средняя	Средняя	Низкие		Средняя	Моделирует последовательность слов и частей речи как марковский процесс
	Высокая	Средняя	Сложная	Средняя	Средние		Средняя	Требует обучения на размеченных данных.
	Высокая	Средняя	Сложная	Высокая	Средние		Средняя	Требует обучения и подбора гиперпараметров.
Трансформеры	Отличная	Средняя	Сложная	Высокая	Высокие		Средняя	Требует обучения/ дообучения, самые требовательные к ресурсам.
	Высокая	Высокая	Простая	Средняя	Средние		Низкая	Хорошо работает из коробки, меньше контроля.
	Средняя	Средняя	Средняя	Средняя	Низкие		Высокая	Требует дополнительной обработки для POS-теггинга.

К преимуществам такого подхода относятся простота реализации и обучения, но модели требуют больших объемов данных и не очень хорошо учитывают контекст.[9]

Метод Conditional Random Fields (CRF) [8] учитывает контекст слова при определении части речи и в силу этого более точен, чем HMM т.к. обеспечивает гибкость в выборе признаков. Методы CRF более сложны в реализации и обучении, чем HMM.[11]

Широко используются в частеречной разметке текстов рекуррентные и трансформенные нейронные сети.

BiLSTM [6] – это двунаправленная рекуррентная нейронная сеть, которая обрабатывает текст в обоих направлениях, учитывая контекст слева и справа от слова. Она хорошо справляется с задачей POS-теггинга, учитывая контекст с обеих сторон слова, но требует больших объемов данных для обучения.

Нейронные сети BERT [3] и RoBERTa [13] представляют собой предобученные модели, основанные на архитектуре трансформеров. Они показывают отличные результаты для POS-теггинга за счет того, что учитывают глобальный контекст, но требуют значительных вычислительных ресурсов и нуждаются в дообучении на специфических данных для достижения оптимальной точности.

Библиотека spaCy использует предобученные модели для русского языка (ru_core_news_sm/md/lg). Она обеспечивает хорошую точность POS-теггинга при легкости использования и высокой производительности. Но точность может быть недостаточной для некоторых специфических задач.

Библиотека pymorphy2 [4] предназначена для морфологического анализа. Требует дополнительной обработки результатов для получения POS-тегов. [12]

Рассмотренные методы и инструменты лемматизации показаны в таблице 4.

Таблица 4 – Инструменты и методы лемматизации

Метод/ Инструмент	Точность	Скорость	Простота реализации	Адаптируемость	Ресурсы	Язык	Контролируемость	Примечания
Словари/ Правила	Средняя	Высокая	Средняя	Низкая	Низкие		Высокая	Хорошо для простых случаев, плохо для сложных и неизвестных слов.
Стат. методы	Высокая	Средняя	Сложная	Средняя	Средние		Средняя	Требуют больших размеченных данных для обучения.
	Высокая	Высокая	Простая	Средняя	Средние		Низкая	Хорошо работает из коробки, меньше контроля.
	Высокая	Средняя	Простая	Средняя	Низкие		Высокая	Специализирован для русского, хорошая точность.
	Высокая	Средняя	Средняя	Средняя	Средние		Низкая	Коммерческий продукт, но предоставляет API.
	Высокая	Средняя	Простая	Средняя	Средние		Низкая	Предобученная модель, хорошая точность, как и spaCy, меньше контроля.
Словари/ Правила	Средняя	Высокая	Средняя	Низкая	Низкие		Высокая	Хорошо для простых случаев, плохо для сложных и неизвестных слов.

Лемматизация – это приведение слова к его словарной форме (лемме). Например, "бегу", "бежишь", "бегут" к "бежать".

Методы лемматизации основанные на словарях и правилах используют словари и правила морфологии для определения леммы слова. Они могут быть очень точными, но требуют больших усилий по разработке и поддержке словарей и правил.

Статистические методы обучаются на больших корпусах текстов с леммами, могут обрабатывать неизвестные слова, но требуют больших объемов данных для обучения.

Библиотека pymorphy2 это одна из лучших библиотек для лемматизации русского языка. Применением данной библиотеки обеспечивается высокая точность, подробный морфологический анализ и поддержка разных словарей, но она может быть медленнее, чем другие библиотеки.

Библиотека spaCy с включением переобученных моделей ru_core_news_sm/md/lg обеспечивает легкость использования, хорошую производительность, но точность может быть ниже, чем у pymorphy2.

Библиотека mystem3 (Яндекс) представляет собой коммерческий продукт от Яндекса обеспечивающий высокую точность и скорость. Но использование этого инструмента требует лицензии.

Библиотека UDPipe выполняет лемматизацию в рамках общего процесса обработки текста. Это комплексное решение, но оно может быть медленнее, чем специализированные библиотеки.

Распознавание именованных сущностей (NER) имеет целью идентификацию и классификацию именованных сущностей в тексте (например, имена людей, названия организаций, географические объекты).

Рассмотренные методы и инструменты распознавания именованных сущностей показаны в таблице 5.

Методы, основанные на правилах, используют заранее заданные правила и словари для идентификации именованных сущностей.

Таблица 5 – Инструменты и методы выделения именованных сущностей

Метод/ Инструмент	Точность	Скорость	Простота реализации	Адаптируемость	Ресурсы	Язык	Контролируемость	Примечания
Правила	Низкая	Высокая	Сложная	Низкая	Низкие		Высокая	Очень сложно поддерживать и расширять, низкая точность.
	Средняя	Средняя	Сложная	Средняя	Средние		Средняя	Требует обучения на размеченных данных, лучше, чем правила.
	Высокая	Средняя	Сложная	Высокая	Средние		Средняя	Требует обучения, лучше, чем CRF.
Трансформеры	Отличная	Средняя	Сложная	Высокая	Высокие		Средняя	Требует дообучения, самые современные, самые требовательные к ресурсам.
	Высокая	Высокая	Простая	Средняя	Средние		Низкая	Хорошо работает из коробки, меньше контроля.
	Высокая	Средняя	Средняя	Средняя	Средние		Средняя	Разработана специально для русского языка.
	Высокая	Средняя	Сложная	Высокая	Высокие		Средняя	Фреймворк, требует настройки и обучения, может быть очень мощным.

Они могут быть очень точными для конкретных типов сущностей, но их сложно поддерживать и расширять. Также они требуют значительных усилий по разработке правил и словарей.

Статистические методы, такие как CRF часто используется для NER, учитывая контекст слова и другие признаки. Такие методы более точные, чем методы, основанные на правилах, но требуют больших объемов данных для обучения.

BiLSTM-CRF является комбинацией двунаправленной рекуррентной нейронной сети и CRF. Этот инструмент хорошо справляется с задачей NER, но требует больших объемов данных для обучения.

Нейросети – трансформеры (BERT, RoBERTa) представляют собой предобученные модели, требующие дообучения на размеченных данных для NER. Они показывают отличные результаты для NER, поскольку учитывают глобальный контекст, но требуют значительных вычислительных ресурсов и нуждаются в дообучении на специфических данных для достижения оптимальной точности.

Библиотека spaCy содержит предобученные модели включают NER и обеспечивает такие преимущества как легкость использования, хорошая производительность, но точность может быть недостаточной для некоторых специфических типов сущностей.

Библиотека Natasha (Python), разработанная специально для русского языка, обеспечивает хорошую точность NER для русского языка, простоту использования, и может быть менее гибкой, чем другие библиотеки.

DeepPavlov – это фреймворк для разработки диалоговых систем, включающий NER для русского языка покрывающий широкий спектр возможностей для NLP. Эта библиотека более сложна в использовании, чем другие.

1.2 Выбор вариантов аннотирования текста для реализации

В результате анализа методов и библиотек для аннотирования русского текста, выбраны два варианта, каждый из которых обладает специфическими достоинствами и ограничениями.

Первый вариант, базирующийся на библиотеке spaCy, представляет собой мощное и современное решение, демонстрирующее высокую точность, скорость и простоту использования. SpaCy предоставляет готовые к применению модели для русского языка, что позволяет оперативно приступить к аннотированию без необходимости сложной настройки и интеграции различных инструментов.

Второй вариант, основанный на сочетании NLTK, Pymorphy2, кастомных словарей и регулярных выражений, является более гибким и настраиваемым подходом. Несмотря на большую трудоемкость реализации и потенциально более низкую скорость работы, этот вариант предоставляет гораздо больший контроль над процессом аннотирования, позволяя учитывать тонкости и нюансы конкретного текста.

Краткая характеристика выбранных вариантов реализации аннотирования представлена в таблице 6.

Выбор между специализированным подходом с использованием spaCy и универсальным, но более трудоемким вариантом на основе NLTK, Pymorphy2 и собственных правил (словари/Regex) отражает компромисс между точностью, скоростью разработки и уровнем контроля над процессом аннотирования. SpaCy, благодаря предобученным моделям для русского языка, предлагает высокую точность на всех этапах: от токенизации до распознавания именованных сущностей. Это достигается за счет использования сложных алгоритмов машинного обучения и больших объемов данных, на которых эти модели были обучены.

Таблица 6 – Краткая характеристика выбранных вариантов реализации аннотирования

Критерий	NLTK + Pymorphy2 + Словари/Regex	
Точность (Accuracy)	Токенизация: Хорошо. POS: Средняя (зависит от качества Pymorphy2). Лемматизация: Хорошо (Pymorphy2). NER: Низкая (зависит от словарей и Regex, легко сделать ошибки).	Токенизация: Отлично. POS: Отлично. Лемматизация: Отлично. NER: Хорошо (зависит от модели, но обычно лучше, чем словари/Regex).
Скорость(Speed)	Медленная (особенно NER с Regex и поиском по словарям).	Быстрая.
Простота реализации	Сложная (много кода, интеграция разных инструментов, разработка правил NER).	Простая (несколько строк кода для выполнения аннотирования).
Адаптируемость	Высокая (можно легко добавлять новые правила, словари, изменять параметры).	Средняя (требуется переобучение модели для адаптации к новым данным, сложнее изменить логику работы).
Необходимые ресурсы	Низкие (небольшие библиотеки, не требующие большого объема памяти).	Средние (требуются ресурсы для загрузки и работы с большими моделями).
Поддержка языка	Хорошая (Pymorphy2 специально разработан для русского языка). NLTK требует адаптации.	Отличная (существуют модели spaCy, специально обученные для русского языка).
Контролируемость	Высокая (полный контроль над логикой работы, правилами и параметрами).	Низкая (ограниченный контроль над внутренними процессами моделей, можно только выбирать разные модели и параметры).

Кроме того, spaCy значительно выигрывает в скорости и простоте реализации, позволяя получить результаты аннотирования, сопоставимые по качеству с более сложными подходами, за существенно меньшее время и с меньшими усилиями.

В то же время, универсальный подход с использованием NLTK, Pymorphy2 и собственных правил, несмотря на свою трудоемкость и необходимость интеграции различных инструментов, предоставляет более высокую степень адаптируемости и контроля над процессом. Этот подход позволяет точно настраивать правила аннотирования, добавлять новые словари и адаптировать параметры отдельных инструментов под конкретные нужды проекта. Несмотря на то, что точность NER зависит от качества созданных правил и словарей, гибкость настройки позволяет учитывать специфические особенности предметной области, что может быть критически важно для определенных задач.

Выбор именно этих двух вариантов обусловлен стремлением найти оптимальный баланс между точностью, скоростью и адаптируемостью, что позволит провести всестороннее исследование их относительной эффективности.

Выводы по разделу.

Дано определение аннотирования текста методами NLP и проведен обзор современных методов и инструментов, применяемых на каждом этапе аннотирования.

В результате анализа определены наборы методов и инструментов иллюстрирующие два подхода к организации приложений для аннотирования – с использованием специализированных инструментов и с использованием набора интегрированных универсальных инструментов обработки естественного языка. Сформулированы два решения для дальнейшей разработки – специализированное и универсальное.

2 Модели и алгоритмы реализуемых решений

2.1 Математическая модель аннотирования текста

В ходе аннотирования выполняется маркировка элементов текста с предопределенными категориями или типами.

Модель содержит: Входной документ (текст), состоящий из n токенов (слов, фраз или предложений): $D = \{t_1>, t_2, ..., t_n\}$, где t_i - i-ый токен в документе D. Множество предопределенных категорий (меток) для аннотирования: $C = \{c_1, c_2, ..., c_m\}$. Метки (labels) для каждого токена в документе: $Y = \{y_1, y_2, ..., y_n\}$, где $y_i \in C$, y_i - метка, присвоенная токену t_i. $F(t_i)$ - набор признаков (features) для токена t_i. θ - параметры модели, которые нужно обучить. $P(y_i \mid F(t_i), \theta)$ - вероятность присвоения метки y_i токену t_i, учитывая его признаки и параметры модели.

Цель – присвоить каждой метке каждому токену в документе, максимизируя вероятность правильного аннотирования.

$$Y^* = \underset{Y}{argmax}\, P(Y|D, \theta), \quad (1)$$

где $P(Y \mid D, \theta)$ – вероятность присвоения меток Y документу D с параметрами θ.

Эта модель описывает общий процесс автоматического аннотирования текста с использованием методов машинного обучения и обработки естественного языка. Описание конкретных используемых моделей в рамках специализированного и универсального решений выполнено в следующих подразделах.

2.2 Модели и алгоритмы специализированного решения

Аннотирование текста с использованием spaCy на русском языке – это процесс, который позволяет автоматически извлекать лингвистическую информацию из текста, делая его пригодным для дальнейшего анализа и применения в различных NLP-задачах

Для русского языка обычно используется либо модель ru_core_news_lg либо ru_core_news_sm. Модель lg это большая модель, обеспечивающая более высокую точность, но требующая больше ресурсов. Модель sm – маленькая модель, более быстрая, но менее точная.

Взаимосвязи основных моделей при аннотировании русского текста библиотекой spaCy показаны на рисунке 1.

Процесс аннотирования начинается с преобразование текста в объект Doc. Объект Doc содержит токенизированную версию текста и все аннотации, полученные в результате обработки. Объект Doc можно итерировать для доступа к отдельным токенам. Каждый токен имеет ряд атрибутов:

- text – текст токена (слово или знак препинания);
- lemma_ – лемма токена (нормальная форма слова);
- pos_ – часть речи (NOUN - существительное, VERB – глагол, ADJ – прилагательное и т.п.);
- tag_ – более детальный морфологический тег, который предоставляет более точную информацию о грамматических характеристиках слова (падеж, число, род);
- dep_ – тип синтаксической зависимости между токеном и его родителем в синтаксическом дереве;
- shape_ – форма токена ("Xxxxx" для слова с заглавной буквы, "dd" для двузначного числа);

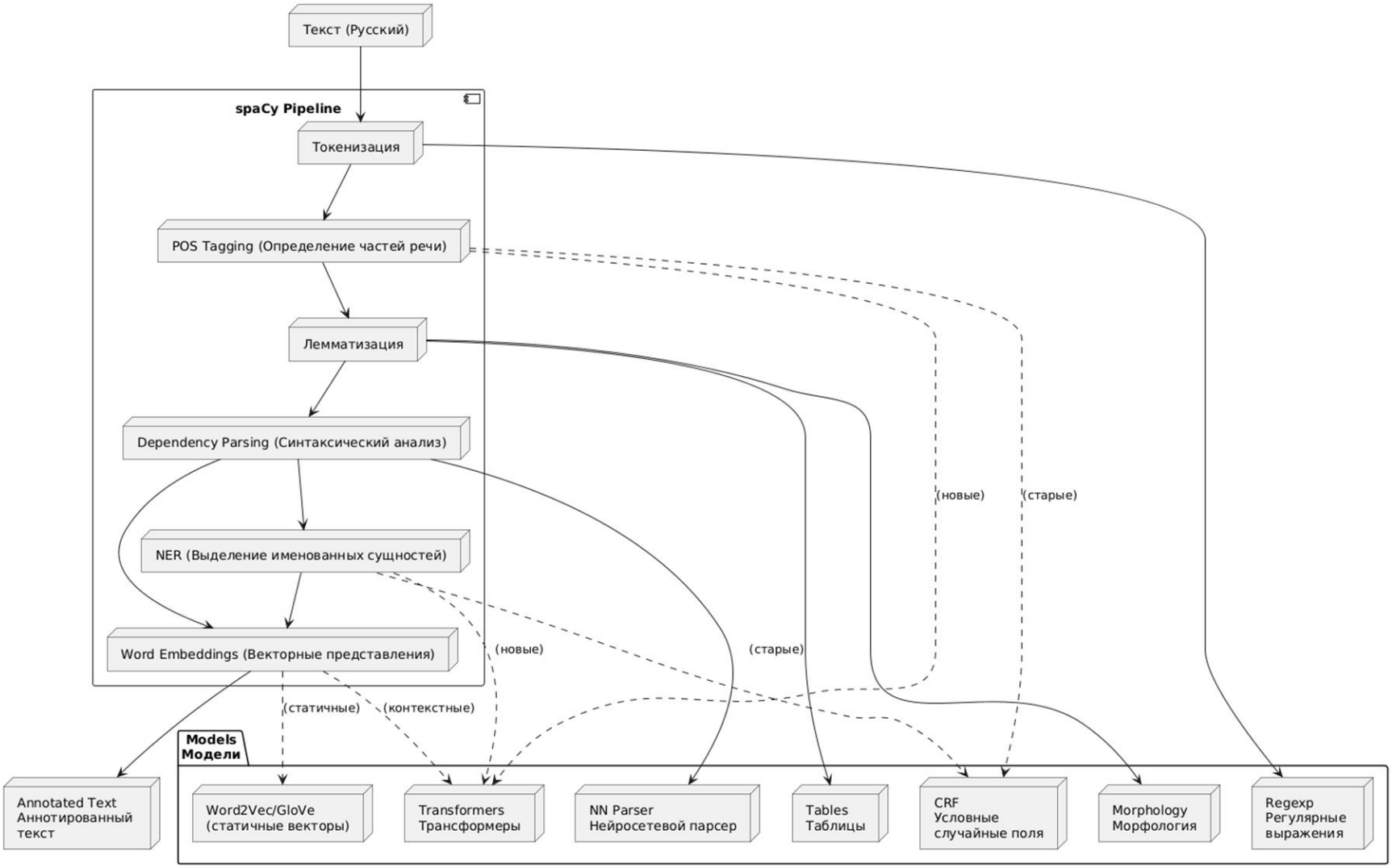

Рисунок 1 – Схема использования моделей при аннотировании специализированной библиотекой spaCy

– is_alpha – true, если токен состоит только из букв;

– is_stop – true, если токен является стоп-словом ("и", "в", "на" и т.п.).

Обработка предложений в spaCy также разделяет текст на предложения.

Выделение именованных сущностей (Named Entity Recognition, NER) предусматривает вывод следующих атрибутов:

– ent.text – текст именованной сущности;

– ent.label_– тип сущности (PER – человек, ORG – организация, LOC – местоположение и т.п.).

SpaCy использует различные статистические модели и алгоритмы для выполнения лингвистического анализа.

В модуле Токенизация используются:

– правила на основе регулярных выражений для определения границ слов и знаков препинания;

– статистические модели, обученные на большом корпусе текстов для обработки сложных случаев, таких как сокращения, аббревиатуры и т.д.

В модуле Part-of-Speech Tagging (определение частей речи) используются:

– трансформерные модели (Transformer-based models), которые используют архитектуру Transformer, такую как BERT или RoBERTa для понимания контекста и определения наиболее вероятной части речи для каждого слова;

– Conditional Random Fields (CRF) – вероятностная модель, которая учитывает контекст окружающих слов для определения оптимальной последовательности тегов частей речи и вычисляет условную вероятность последовательности тегов, учитывая входную последовательность слов.

В модуле лемматизации используются:

– табличные методы, которые используют таблицы соответствий между словами и их леммами;

– правила на основе морфологического анализа, которые применяют правила для удаления окончаний и преобразования слов в их нормальную форму.

В модуле синтаксического анализа зависимостей используются:

– Neural Network-based Dependency Parser – нейронная сеть для предсказания синтаксических зависимостей между словами в предложении. Сеть обучается на размеченных синтаксических деревьях и предсказывает для каждого слова его "родителя" и тип зависимости;
– Transition-based parsing – алгоритм, который строит синтаксическое дерево пошагово, применяя последовательность "переходов" (действий). Нейронная сеть используется для предсказания наиболее вероятного перехода на каждом шаге.

В модуле выделения именованных сущностей (NER) используются:

– трансформерные модели (Transformer-based models);
– Conditional Random Fields (CRF).

Модели Word2Vec, GloVe, FastText создают векторные представления слов, где слова с похожим значением расположены близко друг к другу в векторном пространстве.

Модели Contextualized Word Embeddings (BERT, RoBERTa, ELMo) создают векторные представления слов, которые зависят от контекста, в котором они используются. Это позволяет учитывать многозначность слов.

Таким образом, spaCy использует широкий спектр математических моделей и методов для решения задач обработки естественного языка.

Условные случайные поля (CRF) – это графическая модель, используемая для структурированного предсказания. В контексте NLP, CRF часто применяются для задач, где выходные данные, например, теги частей речи, имеют зависимость друг от друга. В отличие от скрытых марковских моделей (HMM), CRF позволяют учитывать глобальные признаки входных данных, а не только локальные. Это делает их более мощными для задач,

требующих учета контекста, таких как выделение именованных сущностей и частеречная разметка. Модель определяет вероятность последовательности меток при заданном входном предложении, учитывая как признаки отдельных слов, так и связи между метками. Обучение CRF заключается в максимизации условной вероятности правильных меток для обучающего набора данных.

Скрытые марковские модели – это вероятностная модель, предполагающая, что наблюдаемая последовательность, например, последовательность слов в предложении, генерируется скрытой последовательностью состояний, например, последовательностью тегов частей речи. Модель состоит из матрицы переходов между состояниями, матрицы эмиссий показывающей вероятность наблюдения слова в каждом состоянии, и начальных вероятностей состояний. HMM используются для задач, где нужно определить скрытую структуру за наблюдаемыми данными, в частности, для частеречной разметки, когда нужно определить, какая часть речи скрыта за каждым словом в предложении.

Многослойные персептроны (MLP) представляют собой базовый тип нейронной сети, состоящий из нескольких слоев взаимосвязанных нейронов. Каждый нейрон получает входные данные, выполняет взвешенную сумму и применяет функцию активации для получения выходного значения. MLP используются для решения различных задач NLP, включая классификацию текстов, определение тональности и предсказание слов.

Сверточные нейронные сети (CNN) особенно эффективны при работе с данными, имеющими пространственную структуру, такими как изображения и текст. В NLP CNN используют сверточные фильтры для извлечения локальных признаков из текста, например, паттернов из последовательностей n-грамм. Эти признаки затем объединяются для получения более общего представления текста, которое можно использовать для классификации, выделения именованных сущностей и других задач.

Рекуррентные нейронные сети (RNN) разработаны для обработки последовательностей данных, таких как текст. Они имеют «память», которая

позволяет им учитывать предыдущие входы при обработке текущего входа. LSTM (Long Short-Term Memory) и GRU (Gated Recurrent Unit) – это варианты RNN, которые лучше справляются с проблемой исчезающего градиента и позволяют моделировать долгосрочные зависимости в тексте. RNN и их варианты используются для задач машинного перевода, генерации текста и анализа тональности.

Трансформеры – это современная архитектура нейронных сетей, основанная на механизме внимания. Они позволяют моделировать зависимости между словами в тексте без учета их относительного положения, что делает их более эффективными, чем RNN, для обработки длинных текстов. Трансформеры используются в различных задачах NLP, включая машинный перевод, суммаризацию текста, вопросно-ответные системы и классификацию текстов. Такие трансформеры как BERT и RoBERTa стали стандартом де-факто для многих задач NLP, благодаря своей способности захватывать контекст и сложные семантические отношения.

Применение методов линейной алгебры связано с обработкой векторных представлений слов. Векторные представления слов, получаемые алгоритмами Word2Vec, GloVe, FastText позволяют отобразить слово в многомерное пространство. Близкие по смыслу слова располагаются близко друг к другу в этом пространстве. Линейная алгебра используется для работы с этими векторами. Основные операции над векторными представлениями слов это вычисление косинусного расстояния для определения семантической близости, выполнение векторных операций сложения и вычитания для решения аналогий.

Статистические методы используются для оценки параметров вероятностных моделей CRF, HMM и нейронных сетей. Эти методы включают максимизацию правдоподобия, байесовскую оценку и стохастический градиентный спуск.

Также статистические методы используются для оценки производительности моделей NLP.

Аннотирование русского текста с использованием spaCy – это мощный инструмент, который позволяет извлекать ценную лингвистическую информацию из текста, необходимую для различных NLP-приложений.

2.3 Модели и алгоритмы решения на базе универсальных инструментов

Входной текст представляет собой текст, который нужно аннотировать.

Схема применения универсальных инструментов для аннотирования текста показана на рисунке 2.

Предобработка текста включает следующие этапы:

- токенизация (NLTK) – разбивает текст на отдельные токены (слова, знаки препинания и т.д.);
- удаление стоп-слов (NLTK) – удаляет распространенные слова, которые не несут большой смысловой нагрузки (например, "и", "в", "на");
- приведение к нижнему регистру приводит все токены к нижнему регистру для унификации.

Морфологический анализ выполняется следующими инструментами:

- pymorphy2 – проводит морфологический анализ слов (определяет часть речи, падеж, число и т.д.);
- словари совместно с pymorphy2 используются для получения информации о словах.

Аннотирование – это этап, на котором происходит основная работа по разметке текста, в частности:

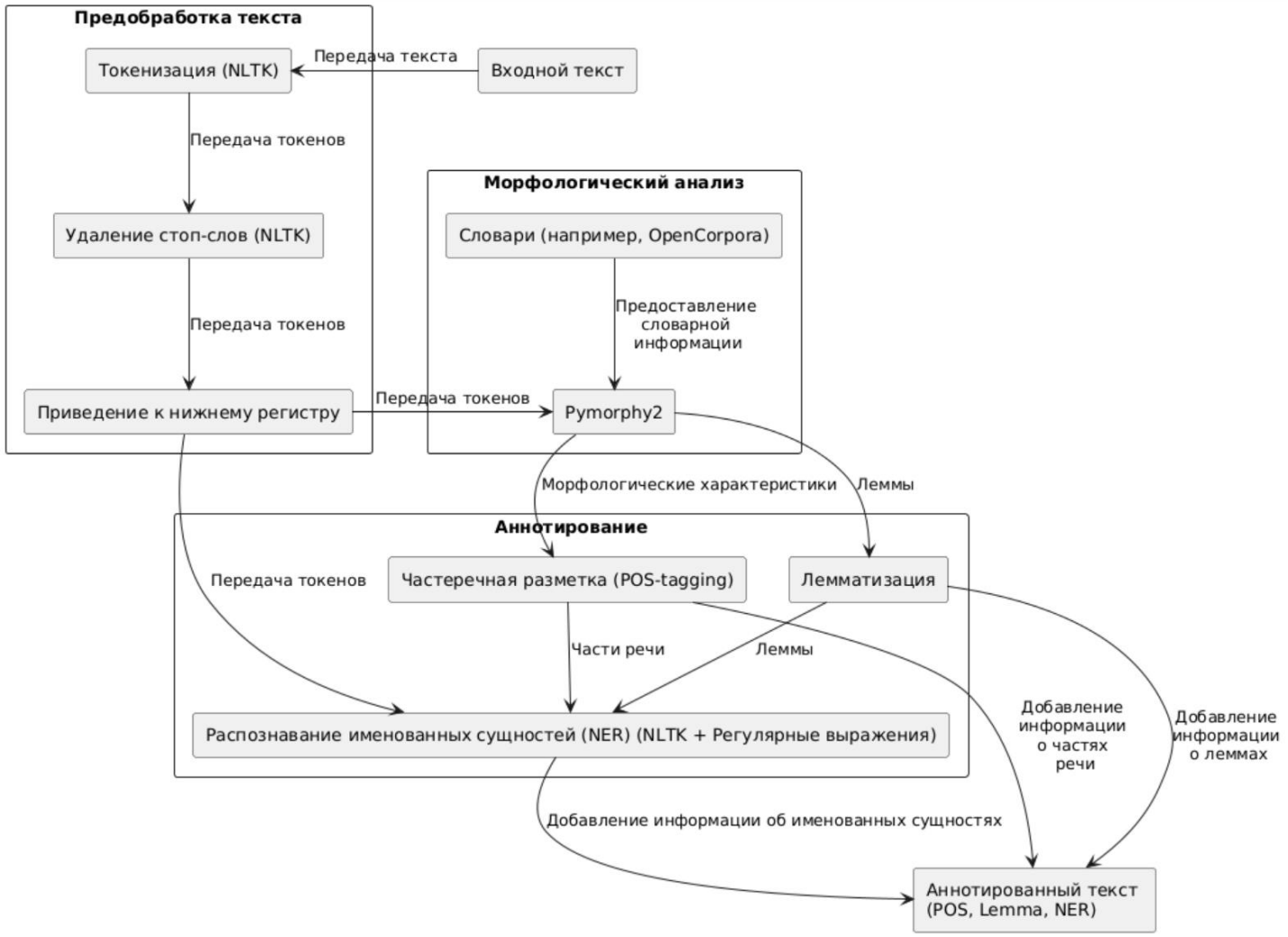

Рисунок 2 – Схема аннотирования текста с использованием универсальных NLP библиотек и методов

– частеречная разметка (POS-tagging), которая определяет часть речи каждого токена с использованием результатов работы pymorphy2;
– лемматизация, которая приводит слова к их нормальной форме (лемме) токена с использованием результатов работы pymorphy2;
– распознавание именованных сущностей (NER) (NLTK + регулярные выражения), которое определяет именованные сущности (например, имена людей, названия организаций, даты, географические названия) с использованием NLTK и регулярных выражений.

Полученный в результате аннотированный текст представляет собой текст, размеченный дополнительной информацией о каждом слове и предложении.

В рассматриваемом варианте NLTK используется для токенизации и базового NER, а Pymorphy2 для морфологического анализа.

Рассмотрим боле подробно возможности каждого из используемых в универсальном решении инструментов.

В рамках универсального подхода к аннотированию текста, библиотека NLTK (Natural Language Toolkit) играет ключевую роль, обеспечивая базовый функционал для предобработки и автоматической аннотации. Работа с NLTK начинается с загрузки текста, после чего происходит этап предобработки. На этом этапе текст подвергается токенизации и лемматизации, которые подготавливают его для дальнейшей обработки (рисунок 3).

Токенизация, как показано на рисунке 4, осуществляется модулем Tokenizer, разбивающим текст на отдельные токены. Затем, модуль Lemmatizer приводит каждый токен к его словарной форме (лемме).

Далее, в зависимости от требований к автоматической аннотации, NLTK может быть использован для выполнения POS-теггинга и NER. Модуль POSTagger присваивает каждому токену частеречный тег, определяя его грамматическую роль в предложении.

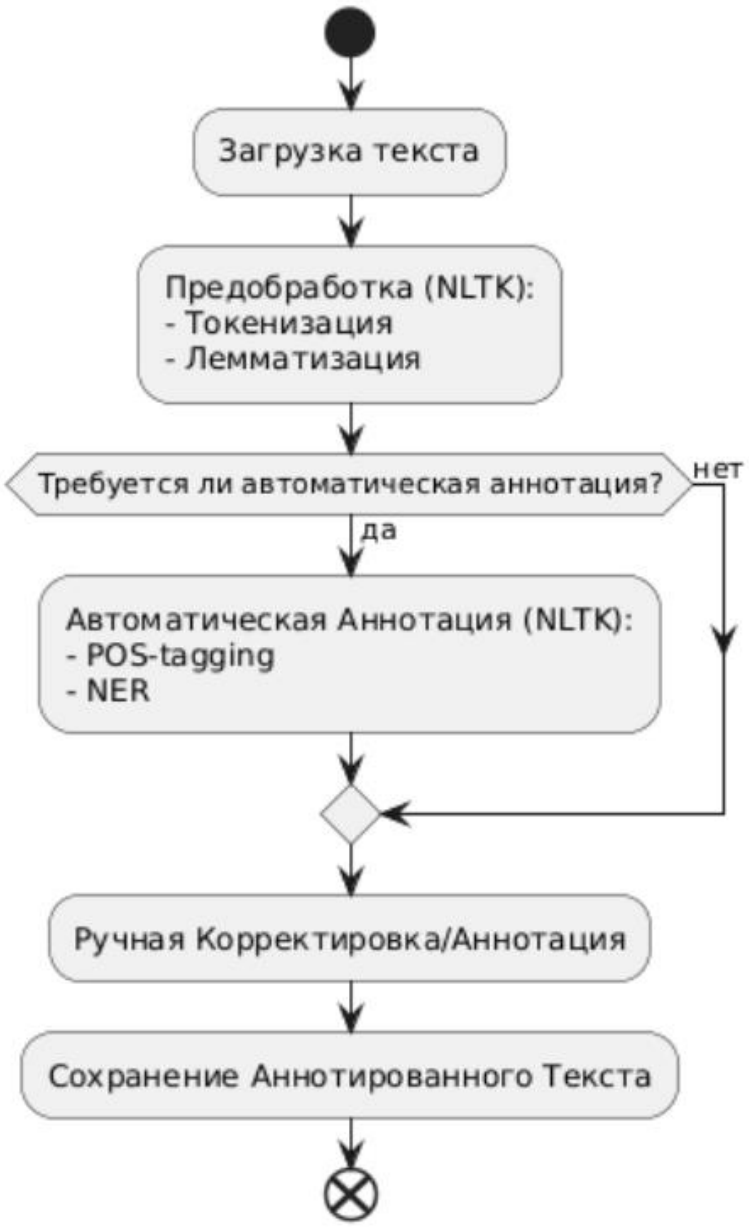

Рисунок 3 – Процесс аннотирования текста NLTK

Модуль NER выполняет распознавание именованных сущностей, идентифицируя и классифицируя такие элементы, как имена, организации и географические объекты. Результаты автоматической аннотации, полученные с помощью NLTK, часто требуют ручной корректировки и доработки, особенно для русского языка, учитывая его сложную морфологию и синтаксис. После этого, аннотированный текст сохраняется для дальнейшего использования. Диаграмма классов наглядно демонстрирует взаимосвязь

между основными модулями NLTK, используемыми в процессе аннотирования, подчеркивая модульную структуру библиотеки и её возможности для выполнения различных задач NLP.

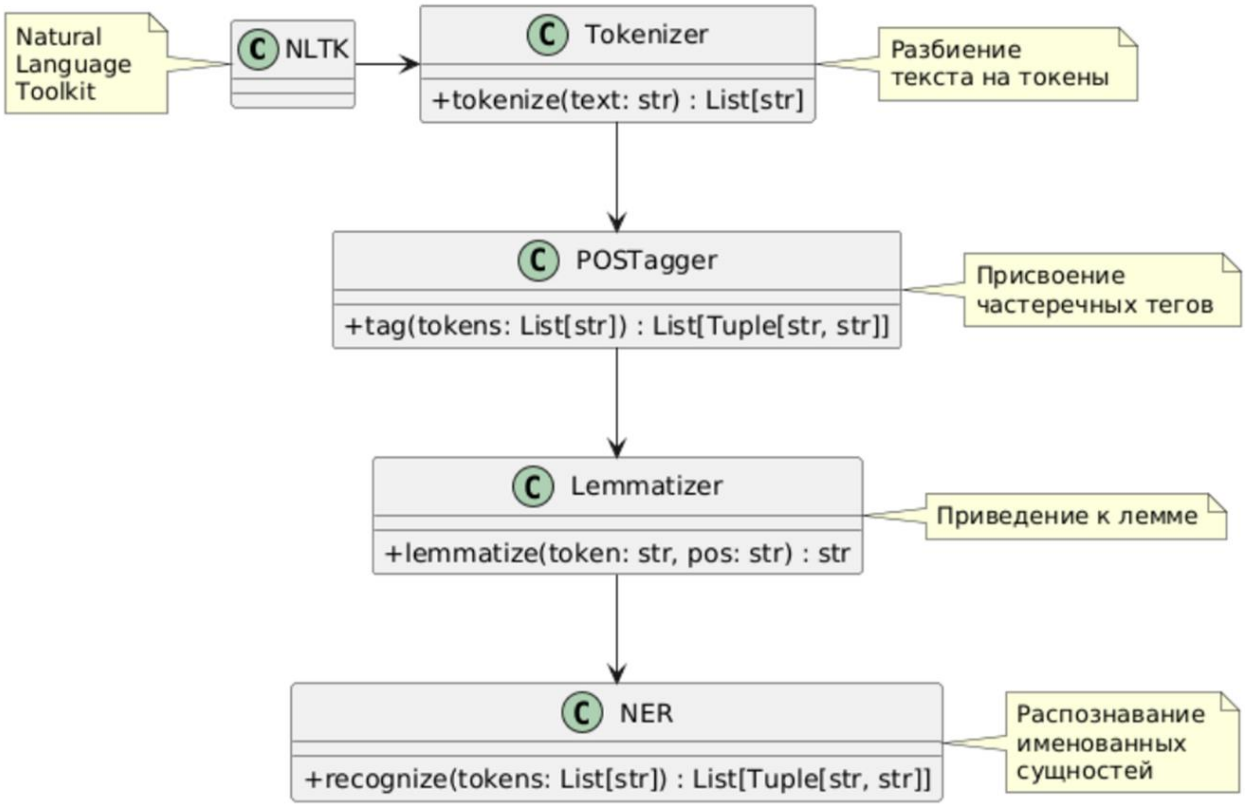

Рисунок 4 – Методы обработки текста NLTK

Не менее важным составляющим элементом универсального решения по аннотированию текстов является библиотека Pymorphy2, используемая на этапе морфологического анализа.

Общая схема применения библиотеки показана на рисунке 5.

На рисунке Pymorphy2 это основной блок, представляющий библиотеку Pymorphy2 и её внутренние процессы:

– загрузка словарей – Pymorphy2 при инициализации загружает необходимые словари из файлов. Эти словари содержат информацию о различных словах и их грамматических характеристиках;

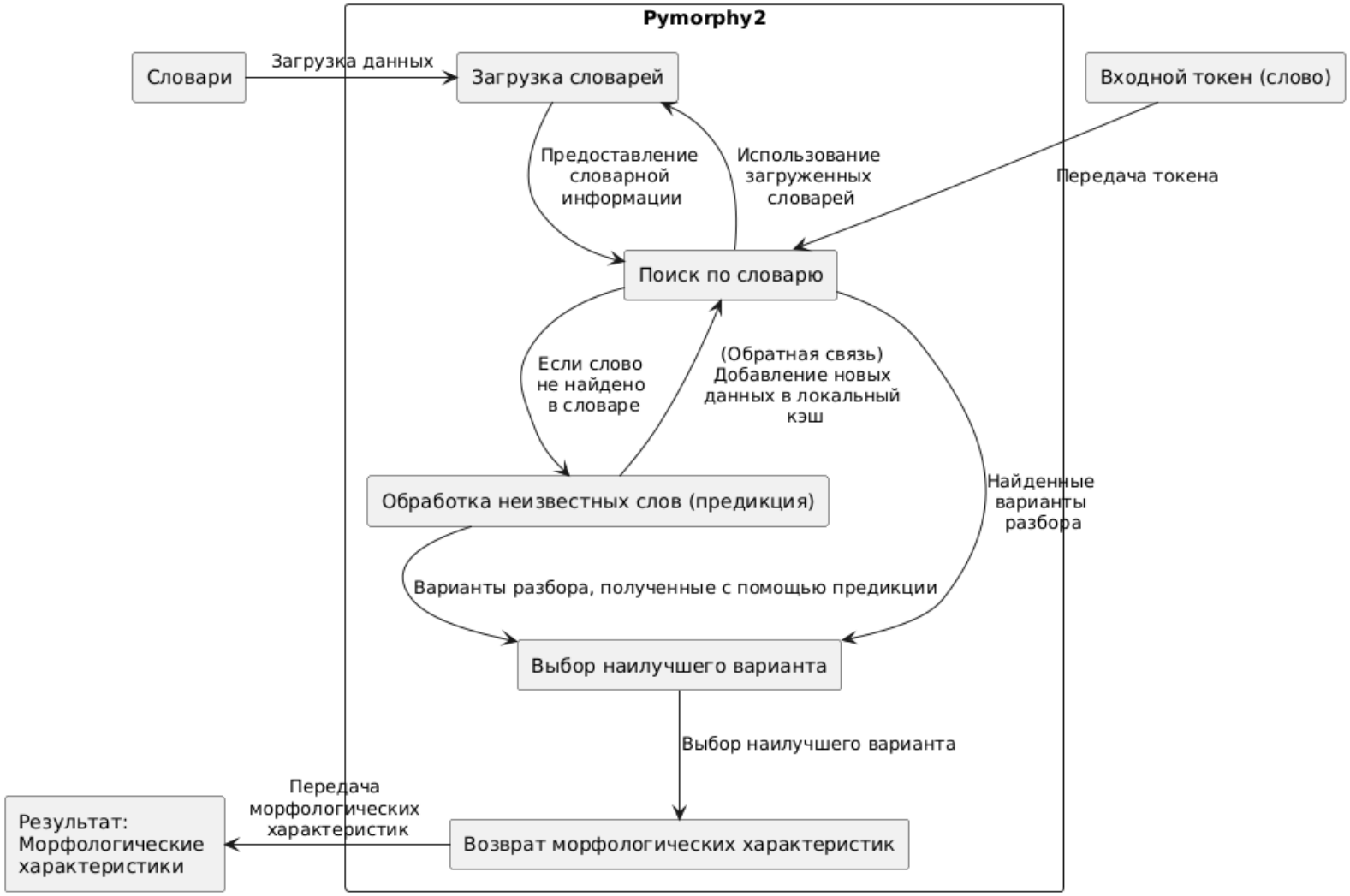

Рисунок 5 – Применение библиотеки Pymorphy2 для морфологического анализа

– поиск по словарю – попытка найти токен в загруженных словарях. Если слово найдено, из словаря извлекаются все возможные варианты разбора (падеж, род, число, часть речи и т.д.);
– обработка неизвестных слов (предикция) – если слово не найдено в словарях, Pymorphy2 применяет алгоритмы предсказания, основанные на морфологических правилах и суффиксах/префиксах, чтобы определить наиболее вероятные морфологические характеристики слова. Также может временно хранить варианты разбора;
– выбор наилучшего варианта – Pymorphy2 может найти несколько вариантов разбора для одного и того же слова. Например, слово "стекло" может быть существительным или глаголом. На этом этапе происходит выбор наиболее подходящего варианта на основе контекста (если доступен) или статистических данных. Pymorphy2 использует эвристические правила для выбора лучшего варианта;
– возврат морфологических характеристик – возвращает выбранный вариант разбора, содержащий морфологические характеристики слова (часть речи, падеж, род, число, время и т.д.).

Словари представляют собой наборы данных, используемые Pymorphy2 для морфологического анализа. Они содержат информацию о словах, их грамматических характеристиках и правилах словоизменения.

Результат – это морфологические характеристики (набор атрибутов, описывающих грамматические свойства слова).

Использование словарей обеспечивает учет особенностей языка.

Общий словарь содержит базовую словарную информацию (леммы, части речи, морфологические признаки).

Тематический словарь (медицина, юриспруденция и др.) содержит специализированную терминологию для конкретной области. Помогает улучшить точность аннотирования в специализированных текстах.

Словарь стоп-слов содержит список слов, которые следует исключить из анализа.

Словарь синонимов содержит списки синонимов для слов. Используется для разрешения неоднозначности и расширения запросов.

Пользовательский словарь содержит информацию, специфичную для конкретного проекта или задачи. Может содержать новые термины, сокращения, аббревиатуры и т.д. Позволяет адаптировать систему аннотирования к конкретным требованиям.

Процессы аннотирования с использованием словарей выключают:

- анализ морфологических характеристик слов, который использует общие и тематические словари;
- удаление стоп-слов, которое состоит в исключении неинформативных слов с использованием словаря стоп-слов;
- лемматизация – приведение слов к нормальной форме (лемме). Использует общие и тематические словари, а также результаты морфологического анализа4
- частеречная разметка (POS-tagging) – определение частей речи для слов. Использует общие и тематические словари, а также результаты морфологического анализа;
- распознавание именованных сущностей (NER): Определение именованных сущностей (например, имена, организации, даты). Использует общие, тематические и пользовательские словари.

Схема (рисунок 6) показывает, что для аннотирования текста используются разные типы словарей, каждый из которых выполняет свою функцию.

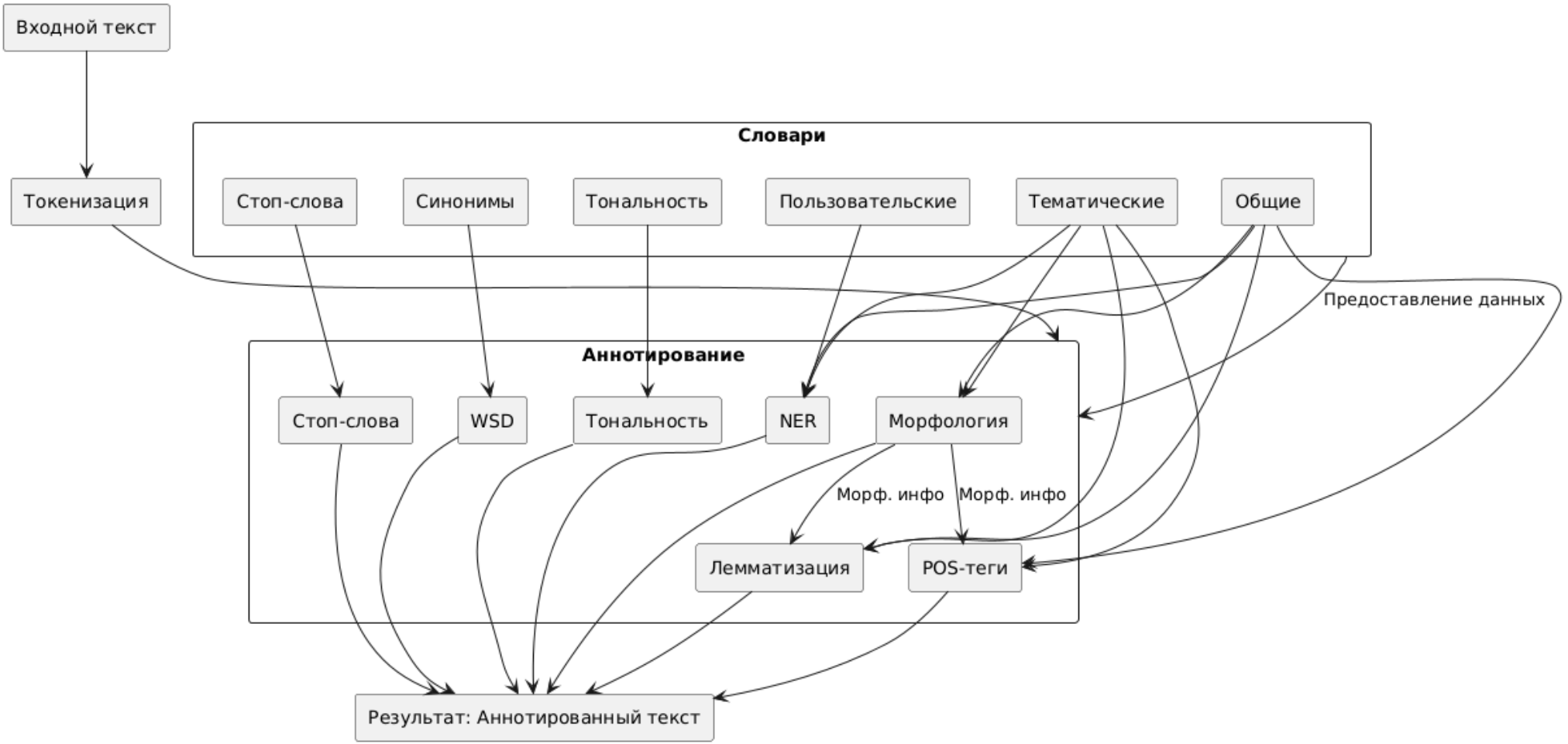

Рисунок 6 – Применение словарей

Регулярные выражения при аннотировании текста (рисунок 7) включают:

- выражения, предназначенные для поиска и распознавания дат в различных форматах (например, "ДД.ММ.ГГГГ", "ГГГГ-ММ-ДД");
- шаблоны для чисел – для распознавания чисел (целых, дробных, с разделителями);
- шаблоны для email – регулярные выражения для поиска и валидации email-адресов;
- шаблоны для URL – регулярные выражения для поиска и валидации URL-адресов;
- шаблоны для телефонных номеров – регулярные выражения для распознавания телефонных номеров в различных форматах;
- шаблоны для денежных сумм – регулярные выражения для распознавания денежных сумм с указанием валюты (например, "100$", "100 руб");
- шаблоны для аббревиатур – регулярные выражения для поиска аббревиатур (например, "и т.д.", "США");
- пользовательские шаблоны – регулярные выражения, определенные пользователем для конкретных задач (например, для поиска определенных терминов или идентификаторов).

Процессы аннотирования с использованием регулярных выражений:

- распознавание именованных сущностей (NER) – регулярные выражения используются для распознавания определенных типов именованных сущностей, которые сложно определить только с помощью словарных методов (например, даты, числа, email, URL);
- лемматизация – регулярные выражения могут использоваться для обработки особых случаев при лемматизации (например, для обработки редких словоформ или сленговых выражений);

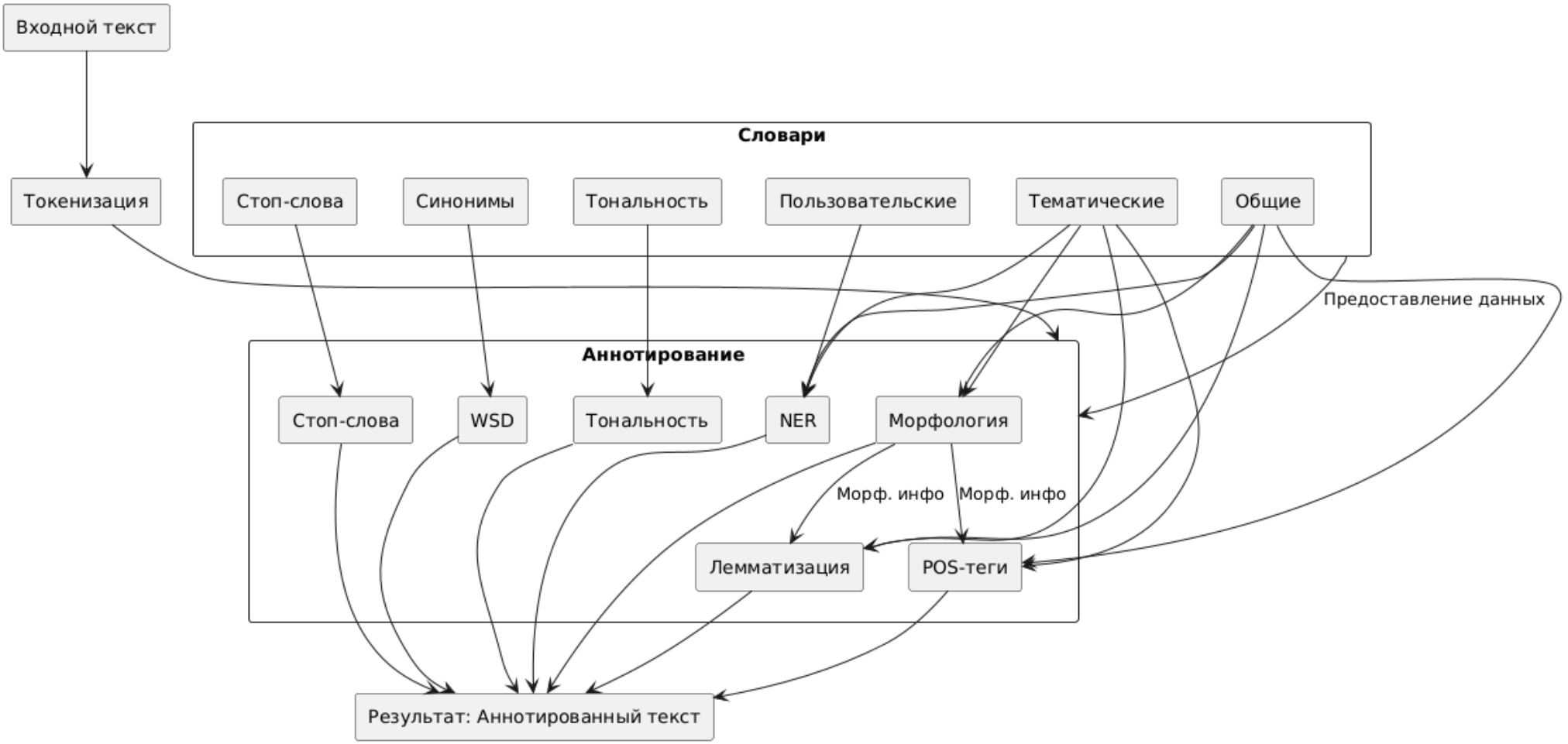

Рисунок 7 – Применение регулярных выражений

– коррекция орфографии – регулярные выражения могут быть использованы для выявления распространенных орфографических ошибок и их исправления;
– разметка сокращений и аббревиатур – регулярные выражения используются для поиска сокращений и аббревиатур в тексте и их расшифровки (например, "США" -> "Соединенные Штаты Америки");
– предварительная обработка текста – регулярные выражения используются для очистки и нормализации текста (например, для удаления лишних пробелов, приведения текста к единому формату).

Словари используются для проверки и уточнения результатов, полученных с помощью регулярных выражений. Например, можно проверить, является ли распознанная именованная сущность реальным именем, или является ли предложенное исправление орфографической ошибки реальным словом.

Выводы по разделу.

Во втором разделе выполнен подробный анализ и разработка алгоритмов аннотирования текста.

Разработан алгоритм аннотирования текста с использованием библиотеки spaCy.

Разработан алгоритм аннотирования текста и применение интегрированных универсальных средств обработки естественного языка.

Исследованы закономерности и разработаны алгоритмы применения словарей и регулярных выражений для аннотирования текста.

3 Программная реализация и экспериментальное исследование приложений аннотирования текста

3.1 Проектирование архитектуры и выбор средств разработки

Программа аннотирования текста с использованием библиотеки SpaCy (рисунок 8) начинает свою работу с загрузки модели ru_core_news_sm.

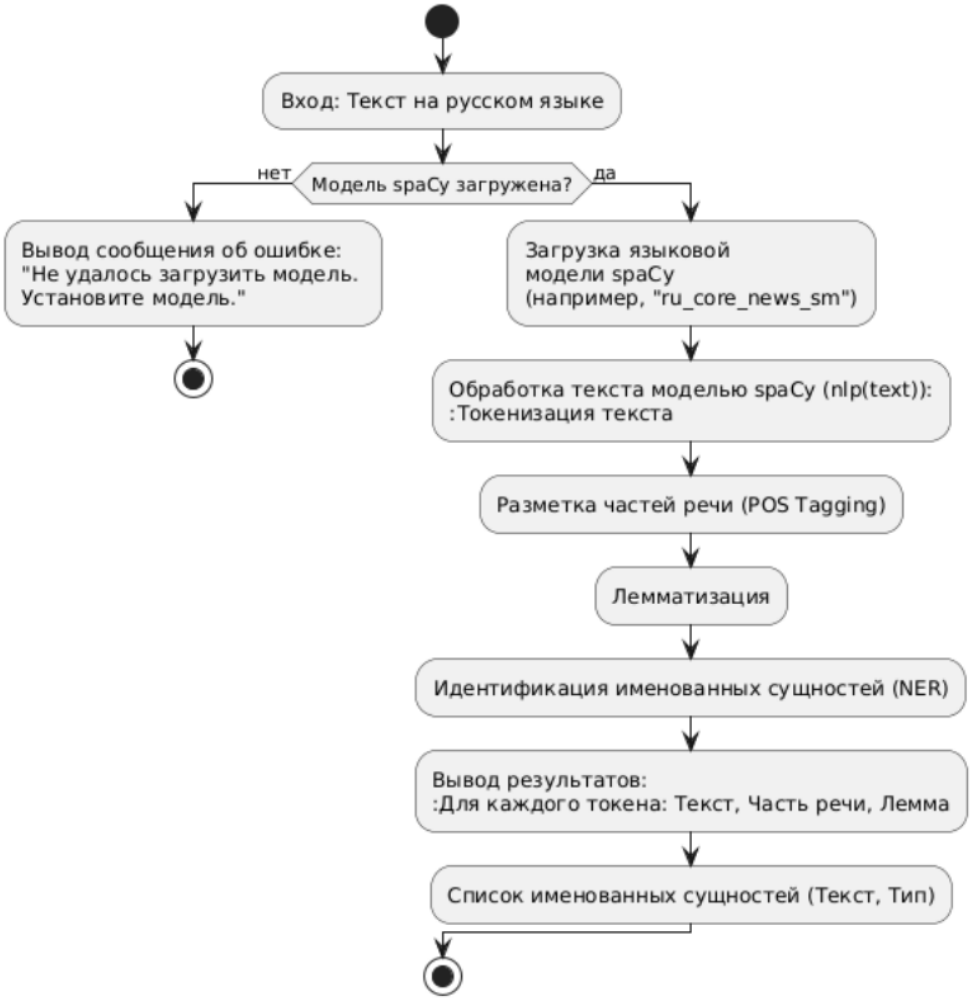

Рисунок 8 – Блок-схема программы SpaCy

Блок схема программы аннотирования на основе универсальных библиотек показана на рисунке 9.

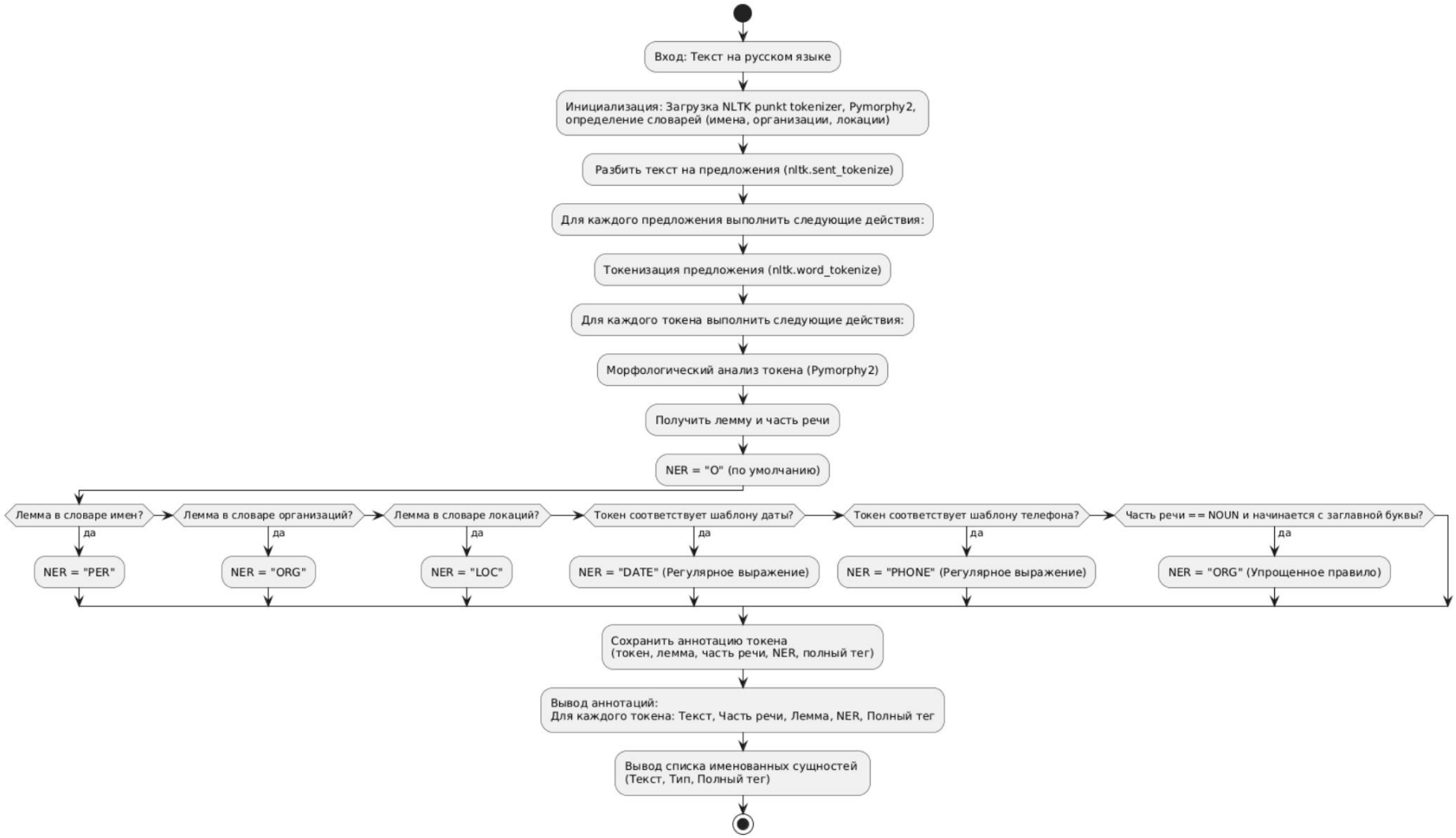

Рисунок 9 – Блок-схема программы NLTK

Если модель успешно загружена, она применяется к входному тексту – spaCy и выполняет токенизацию, частеречную разметку, лемматизацию и идентификацию именованных сущностей.

Программа выводит результаты аннотирования в консоль для каждого токена выводятся его текст, часть речи и лемма. Также выводится список найденных именованных сущностей с указанием их типа.

Программа использует при выделении именованных сущностей словари, регулярные выражения и правила.

Для реализации программ выбран язык программирования Python 3.10. Выбраны библиотеки NLTK 3.9.1, numpy 1.23.5, pymorphy2 0.9.1, regex 2024.11.6, spacy 3.5.4. Выбраны модели pymorphy2-dicts-ru 2.4.417127.4579844 и ru_core_news_sm 3.5.0.

Среда разработки Visual Studio Code 1.97.2

Аппаратное обеспечение – процессор AMD A10-5800K APU with Radeon(tm) HD Graphics 3.80 GHz, оперативная память 8,00 ГБ.

3.2 Разработка и оценка качества работы приложений

Результат запуска программы показан на рисунке 10.

В качестве текста для аннотирования использован следующий образец «Президент России Владимир Путин посетил Москву. Компания Яндекс разрабатывает поисковую систему. Дата: 15.03.2023, телефон: +79161234567. Иван пришел в Сбербанк.»

В ходе оценки качества аннотирования выполнялось сравнение вывода программ с ручным аннотированием. В ручном режиме в тексте выделены следующие именованные сущности:

- России – географическое название GPE,
- Владимир Путин – персона PER,
- Москву – локация LOC,
- Яндекс – организация ORG,

– 15.03.2023 – дата DATE,

– +79161234567 – телефон PHONE,

– Иван – персона PER,

– Сбербанк – организация ORG.

Всего выделено 8 сущностей.

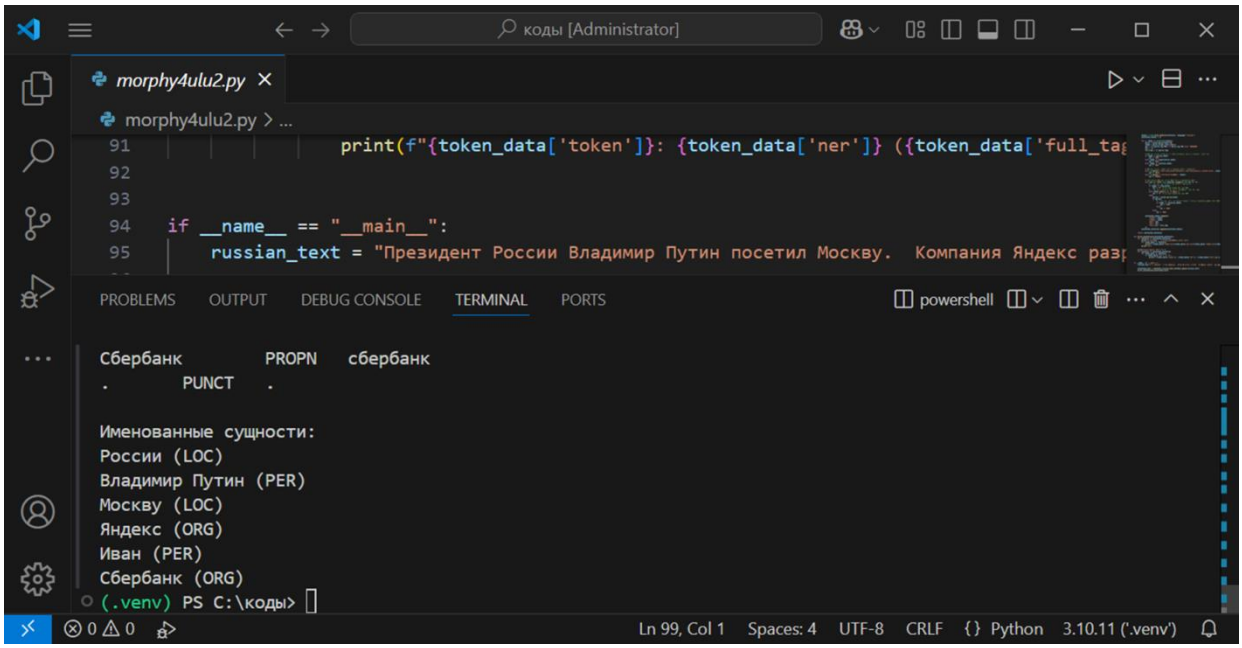

Рисунок 10 – Пример запуска программы в Visual Studio Code

Программа, основанная на специализированной библиотеке, выделила следующие именованные сущности:

– России LOC,

– Владимир Путин PER,

– Москву LOC,

– Яндекс ORG,

– Иван PER,

– Сбербанк ORG.

Решение SpaCy имеет всего одну неточность – Россия распознана как локация, а не как географическое название. Но при этом пропущены телефон и дата.

Программное решение на базе библиотеки NLTK показало существенно более низкое качество аннотирования:

- Президент: ORG,
- России: ORG,
- Владимир: PER,
- Путин: ORG,
- Москву: LOC,
- Компания: ORG,
- Яндекс: ORG,
- Дата: ORG,
- 15.03.2023: DATE,
- +79161234567: PHONE,
- Иван: PER,
- Сбербанк: ORG.

Программа NLTK построена на универсальных инструментах и имеет более широкие возможности для доработки по сравнению с программой SpaCy. Новая версия программы NLTK2 описана в следующем подразделе.

3.3 Совершенствование приложения аннотирования

Для повышения качества аннотирования необходимо улучшить словари, ужесточить регулярные выражения и добавить проверку на известность слова для POS-тегов.

Структура программы NLTK2 (рисунок 11) также изменена – добавлена дополнительная логика в выделение именованных сущностей, связанная с обработкой стоп-слов.

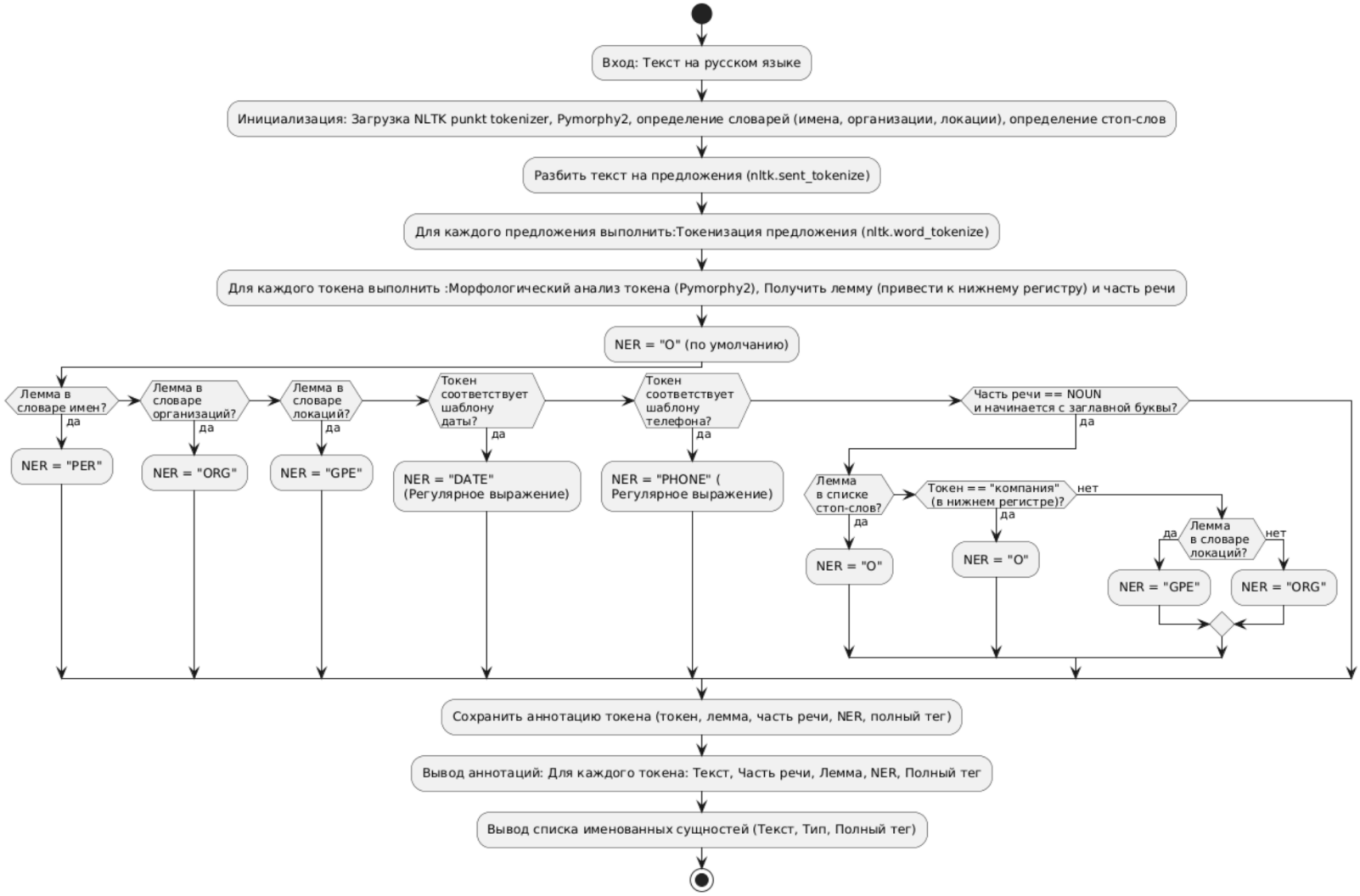

Рисунок 11 – Блок-схема программы NLTK2

Программный код словарей:

```
person_names = ["владимир", "иван", "петр", "мария", "анна", "путин"]
person_names = [name.lower() for name in person_names]
organization_names = ["яндекс", "сбербанк", "газпром", "компания"]
organization_names = [name.lower() for name in organization_names]
location_names = ["москва", "санкт-петербург", "лондон"]
location_names = [name.lower() for name in location_names]
```

Переработанные регулярные выражения:

```
# регулярное выражение для даты
elif re.match(r"^(0[1-9]|[12][0-9]|3[01])\.(0[1-9]|1[012])\.(19|20)\d\d$", token):
# регулярное выражение для телефона
elif re.match(r"^\+?[78][0-9]{10}$", token):
```

Правило проверки на известность слова для POS-тегов:

```
# Улучшенный NER на основе POS-тегов и заглавных букв
        elif pos == "NOUN" and token[0].isupper() and ner == "O":
            if morph.is_known(token):
                ner = "O"
            else:
                ner = "ORG"
```

Список стоп-слов, которые не должны классифицироваться как именованные сущности:

```
stop_words = ["президент", "дата", "телефон"]
stop_words = [word.lower() for word in stop_words]
```

Вывод программы NLTK2 не содержит ошибок и содержит 2 неточности. Сущность Владимир Путин распознана как две сущности и тэг Москвы не в полной мере соответствует ручному разбору, но близок по смыслу:

– России: GPE,

– Владимир: PER,

– Путин: PER,

– Москву: GPE,

– Яндекс: ORG,

– 15.03.2023: DATE,

– +79161234567: PHONE,

– Иван: PER,

– Сбербанк: ORG.

Результаты экспериментального исследования (таблица 7) показывают, что решение на базе универсальных инструментов после доработки показывает лучший результат по сравнению со специализированным решением.

Таблица 7 – Результаты экспериментального исследования

Параметр	Ручной разбор	SpaCy	NLTK	NLTK 2
Число именованных сущностей	8	6	12	9
Ошибочно отобрано	0	0	3	0
Не выявлено	0	2	0	0
Ошибок	0	0	4	0
Неточностей	0	1	5	2

Выводы по разделу.

Для проведения экспериментального исследования разработаны 2 программы NLP аннотирования текста – на основе специализированной библиотеки SpaCy и на основе универсальной библиотеки NLTK.

В результате экспериментов показано, что решение на основе универсальной библиотеки за счет целенаправленного совершенствования и настройки своих элементов может обеспечить результат лучше, чем специализированное решение.

Заключение

В ходе выполнения ВКР рассмотрены и проанализированы современные подходы к решению задачи аннотирования текстов методами NLP. Проведен обзор современных методов и инструментов, применяемых на каждом этапе аннотирования

В результате анализа определены наборы методов и инструментов реализующие два подхода к организации приложений для аннотирования – с использованием специализированных инструментов и с использованием набора интегрированных универсальных инструментов обработки естественного языка. Сформулированы два решения для дальнейшей разработки – специализированное и универсальное.

Выполнено исследование моделей и алгоритмов выбранных инструментов NLP аннотирования.

Разработан алгоритм аннотирования текста с использованием библиотеки spaCy.

Разработан алгоритм аннотирования текста и применение интегрированных универсальных средств обработки естественного языка.

Исследованы закономерности и разработаны алгоритмы применения словарей и регулярных выражений для аннотирования текста.

Для проведения экспериментального исследования разработаны 2 программы NLP аннотирования текста – на основе специализированной библиотеки SpaCy и на основе универсальной библиотеки NLTK.

В результате экспериментов показано, что решение на основе универсальной библиотеки за счет целенаправленного совершенствования и настройки своих элементов может обеспечить результат лучше, чем специализированное решение.

Цель работы - создание системы автоматического аннотирования текстов методами обработки естественного языка – достигнута.

Список используемой литературы и используемых источников

нализ текстовых данных с помощью NLTK и Python [Электронный ресурс]. – URL: https://habr.com/ru/companies/otus/articles/774498/ (дата обращения:

раткий обзор NLP библиотеки SpaCy [Электронный ресурс]. – URL: https://habr.com/ru/articles/504680/ (дата обращения: 06.03.2025).

одели BERT для машинного обучения: гайд для начинающих [Электронный ресурс]. – URL: https://blog.skillfactory.ru/modeli-bert-dlya-mashinnogo-

орфологический анализатор pymorphy2 [Электронный ресурс]. – URL: https://pymorphy2.readthedocs.io/en/stable/ (дата обращения: 06.03.2025).

окенизация на подслова [Электронный ресурс]. – URL: подслова-subword-tokenization/ (дата обращения: 06.03.2025).

i

L

S 7. Complete Guide to Subword Tokenization Methods in the Neural Era [T Электронный ресурс]. – URL: https://blog.octanove.org/guide-to-subword-M tokenization/ (дата обращения: 06.03.2025).

P 8. Conditional Random Fields (CRFs) for POS tagging in NLP [O Электронный ресурс]. – URL: https://www.geeksforgeeks.org/conditional-S random-fields-crfs-for-pos-tagging-in-nlp/?ysclid=m7uvr9sbco603017797 (дата T обращения: 06.03.2025).

a 9. Part of Speech (POS) tagging with Hidden Markov Model [g Электронный ресурс]. – URL: https://www.mygreatlearning.com/blog/pos-ta gging/ (дата обращения: 06.03.2025).

i 10. POS Tagging and Hidden Markov Model [Электронный ресурс]. – h URL: https://www.h2kinfosys.com/blog/pos-tagging-and-hidden-markov-m model/ (дата обращения: 06.03.2025).

Issue [Электронный ресурс]. – URL: https://discuss.pytorch.org/t/bilstm-pos-

11. POS Tagging using conditiional random field [Электронный ресурс]. – URL: https://ijsrem.com/download/pos-tagging-using-conditiional-random-field/ (дата обращения: 06.03.2025).

ymorphy2 [Электронный ресурс]. – URL: https://habr.com/ru/articles/176575/ (дата обращения: 06.03.2025).

13. RoBERTa: A Robustly Optimized BERT Pretraining Approach [Электронный ресурс]. – URL: https://www.labellerr.com/blog/roberta-a-robustly-optimized-bert-pretraining-approach/ (дата обращения: 06.03.2025).

14. Russian spaCy Models Documentation [Электронный ресурс]. – URL: https://spacy.io/models/ru (дата обращения: 06.03.2025).

15. spaCy. Industrial-strength Natural Language Processing [Электронный ресурс]. – URL: https://spacy.io/ (дата обращения: 06.03.2025).

16. Text Annotation for Natural Language Processing – A Comprehensive Guide [Электронный ресурс]. – URL: https://www.habiledata.com/blog/text-annotation-for-nlp/ (дата обращения: 06.03.2025).

17. The Evolution of Tokenization – Byte Pair Encoding in NLP [Электронный ресурс]. – URL: https://www.freecodecamp.org/news/evolution-of-tokenization/ (дата обращения: 06.03.2025).

18. The Natural Language Toolkit (NLTK) is an open source Python library for Natural Language Processing [Электронный ресурс]. – URL: https://www.nltk.org/api/nltk.html (дата обращения: 06.03.2025).

19. UDPipe Natural Language Processing - Text Annotation [Электронный ресурс]. – URL: https://cran.um.ac.ir/web/packages/udpipe/vignettes/udpipe-annotation.html (дата обращения: 06.03.2025).

20. UDPipe: Tokenization, Parts of Speech Tagging [Электронный ресурс]. – URL: https://cran.rstudio.com/web/packages/udpipe/udpipe.pdf (дата

обращения: 06.03.2025).