

Разработка текстового анализатора для поиска автомобильных номеров в произвольном тексте

Гордейчук Владислав Александрович

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

В статье описан процесс создания системы для поиска автомобильных номеров в текстовых данных на языке Python. В основе решения лежит использование регулярных выражений для описания формата номерного знака. Приведены примеры кода и результаты работы программы на тестовых текстах. Разработанный скрипт представляет собой простой и эффективный инструмент для извлечения структурированной информации из произвольного текста.

Ключевые слова: регулярные выражения, текстовый анализ, обработка естественного языка, Python, поиск шаблонов, извлечение данных, автомобильные номера, ANPR

Developing a text analyzer for searching for license plates in arbitrary text

Gordeychuk Vladislav Aleksandrovich

Sholom-Aleichem Priamursky State University

Student

Abstract

The article describes the development of a system for detecting automotive license plates in textual data using the Python programming language. The core of the solution is based on regular expressions that define the format of a Russian license plate number. The paper details the algorithm's implementation stages: pattern definition, text processing, match searching, and result validation. Key code fragments and examples of the system's operation on test texts are provided. The developed script serves as a simple yet effective tool for extracting structured information from unstructured text, demonstrating the practical application of pattern-matching techniques for data retrieval. The work also discusses the current limitations of the approach and outlines potential directions for its development, such as expanding pattern libraries and integrating with data processing pipelines.

Keywords: regular expressions, text analysis, natural language processing, Python, pattern matching, data extraction, license plate recognition, ANPR.

1 Введение

1.1 Актуальность

В условиях цифровизации и роста объемов неструктурированных текстовых данных — от сообщений в мессенджерах и социальных сетях до автоматических логов систем видеонаблюдения и документооборота — задача автоматического извлечения структурированной информации приобретает критическое значение. Автомобильные номерные знаки, как строго формализованные идентификаторы, часто встречаются в таких текстах, но их ручной поиск трудоемок и подвержен ошибкам. Эффективное автоматическое обнаружение номеров в тексте является ключевым этапом для последующего анализа в системах безопасности, логистики, управления доступом и бизнес-аналитики. Разработка легковесных и точных алгоритмов на основе шаблонов, таких как регулярные выражения, позволяет интегрировать функционал поиска в разнообразные приложения без необходимости использования ресурсоемких систем компьютерного зрения, делая решение универсальным и доступным.

1.2 Обзор исследований

Исследование Ф. Зандамелаю, Ф. Селетани, П. Малатджи, Д. Кунене, М. А. Нана и Т. Секопа выявляют ограничения существующих решений при обработке неструктурированных текстовых данных, что обосновывает необходимость разработки дополнительных модулей постобработки — в частности, инструментов валидации распознанных номеров с использованием регулярных выражений [1]. В работе М. Саммерфильда рассмотрены принципы модульного проектирования, использование стандартных библиотек и паттернов, что легло в основу архитектуры разрабатываемого скрипта и выбора встроенного модуля `re` для реализации поиска по шаблону [2]. В статье Ф. Фуджитти и Т. Чакраборти представлен подход к парсингу неструктурированного текста демонстрирует общие принципы, применимые для извлечения шаблонных данных, включая автомобильные номера [3]. Чж. Чжун и др. подчеркивают важность использования стандартных, хорошо поддерживаемых модулей, что подтверждает обоснованность выбора встроенного инструментария для обеспечения долгосрочной работоспособности приложения [4]. Н. Р. Галимуллина демонстрирует эффективность простых скриптов для обработки текстовых массивов, что соответствует парадигме «простого и эффективного инструмента», реализованной в данной работе [5]. Н. С. Юлкина, М. С. Шмелёва и И. К. Фомина в статье выделяют регулярные выражения как один из наиболее быстрых и ресурсоэффективных методов для работы с текстовыми шаблонами фиксированной структуры, что служит теоретическим обоснованием выбора данного подхода в настоящем исследовании [6].

1.3 Цель исследования

Целью исследования является разработка, реализация и тестирование алгоритма автоматического поиска и извлечения российских автомобильных

номеров стандартного формата из произвольного неструктурированного текста с использованием языка программирования Python и механизма регулярных выражений.

2 Материалы и методы

В рамках работы разработано консольное приложение на языке Python 3, использовалась среда разработки Visual Studio Code (VS Code). В качестве основного инструмента использован встроенный модуль `re` для работы с регулярными выражениями.

Алгоритм работы системы состоит из последовательных этапов:

1. Определение формального шаблона. На основе ГОСТ Р 50577-93 был составлен паттерн, описывающий формат типичного российского номерного знака для легковых автомобилей: одна буква, три цифры, две буквы, пробел, две или три цифры кода региона. В регулярном выражении это записывается как `[А-Яа-я]\d{3}[А-Яа-я]{2}\s\d{2,3}`.

2. Ввод и предобработка текста. Текст подается на вход алгоритму либо через интерактивный ввод пользователя, либо используется встроенный демонстрационный пример.

3. Поиск совпадений (Matching). Ядро алгоритма — вызов функции `re.findall(pattern, text)`, которая сканирует весь текст и возвращает список всех подстрок, в точности соответствующих заданному регулярному выражению.

3 Результаты и обсуждения

В результате работы разработан консольный скрипт на Python, реализующий алгоритм поиска автомобильных номеров в тексте. Функция `find_car_numbers(text: str)`: Основная функция поиска автомобильных номеров в тексте, возвращает список найденных номеров (рис. 1).

```
import re
from typing import List

def find_car_numbers(text: str) -> List[str]:
    """
    Ищет автомобильные номера в тексте
    Работает с заглавными и строчными буквами
    Формат: буква-3 цифры-2 буквы-пробел-цифры (регион)
    """
    # ИСПРАВЛЕННЫЙ паттерн - без \D* в конце
    pattern = r'[А-Яа-я]\d{3}[А-Яа-я]{2}\s\d{2,3}'
    numbers = re.findall(pattern, text)
    return numbers
```

Рисунок 1 – Функция поиска автомобильных номеров с использованием регулярного выражения

Регулярное выражение `[А-Яа-я]\d{3}[А-Яа-я]{2}\s\d{2,3}\D*` описывает российский номерной знак: одна буква (`[А-Яа-я]`), три цифры (`\d{3}`), две буквы (`[А-Яа-я]{2}`), пробел (`\s`), код региона из 2-3 цифр (`\d{2,3}`).

Функция `main()`: Основной цикл программы с обработкой пользовательского ввода и демонстрационными примерами.

Код программы:

```
def main():
    print("=" * 60)
    print("ПРОСТОЙ ПОИСК АВТОМОБИЛЬНЫХ НОМЕРОВ")
    print("=" * 60)
    # Пример 1: Разные регистры букв
    test_case_1 = """Привет! Я видел машины с номерами:
A123БВ 777, потом еще е001ке 102,
и одна была М777ММ 96, а там вообще р863ох 27рус"""
    # Пример 2: Номера в контексте предложений
    test_case_2 = """Сегодня утром по дороге на работу я заметил
несколько интересных машин. Первой была черная Toyota Camry
с номером A123БВ 777, она ехала очень быстро.
Потом видел синий BMW с госномером K999KK 199,
который нарушил правила. А на парковке стояла
старая Лада с номером B555AA 134."""
    # Пример 3: Смешанные случаи (корректные и некорректные номера)
    test_case_3 = """Отчет о нарушениях:
Корректные номера: A123БВ 777, K999KK 199, М001МР 799
Некорректные: A12БВ 77 (мало цифр), АВ123CD 99 (латиница)
Еще номера: E111EE 77, P456OP 98, T789TT 134
Словом номер: АБВГДЕ (не номер)
Номер без пробела: A123БВ777 (не найдется)"""
    print("\nВыберите тестовый пример:")
    print("1 - Разные регистры букв (заглавные и строчные)")
    print("2 - Номера в контексте предложений")
    print("3 - Смешанные случаи (корректные и некорректные)")
    print("0 - Ввести свой текст")
    print("Или нажмите Enter для примера 1")
    choice = input("\nВаш выбор: ").strip()
    if choice == "1":
        user_text = test_case_1
        print("\n" + "=" * 60)
        print("ТЕСТ 1: Разные регистры букв")
        print("=" * 60)
    elif choice == "2":
        user_text = test_case_2
        print("\n" + "=" * 60)
        print("ТЕСТ 2: Номера в контексте предложений")
        print("=" * 60)
    elif choice == "3":
        user_text = test_case_3
        print("\n" + "=" * 60)
        print("ТЕСТ 3: Смешанные случаи")
        print("=" * 60)
    elif choice == "0":
        print("\nВведите свой текст:")
        user_text = input("> ")
    else:
        user_text = test_case_1
        print("\n" + "=" * 60)
        print("ТЕСТ 1: Разные регистры букв (по умолчанию)")
```

```
print("=" * 60)
if choice != "0":
    print(f"\nТекст для анализа:\n{user_text}")
print("\n" + "-" * 60)
print("РЕЗУЛЬТАТЫ ПОИСКА:")
print("-" * 60)
found_numbers = find_car_numbers(user_text)
if found_numbers:
    print(f"\nНайдено номеров: {len(found_numbers)}")
    print("\nСписок найденных номеров:")
    for i, number in enumerate(found_numbers, 1):
        print(f" {i}. {number.strip()}")
else:
    print("\nНомера не найдены")
print("\n" + "=" * 60)
print("\nПроверьте формат номера:")
print("✓ Буква (русская, большая или маленькая)")
print("✓ Три цифры (123)")
print("✓ Две буквы (АБ)")
print("✓ Пробел")
print("✓ 2-3 цифры региона (777)")
print("\nПример: A123БВ 777 или a123бв 777")
print("=" * 60)
```

Логика работы кода программы

Выбор режима: Пользователь выбирает один из трех тестовых примеров или вводит свой текст.

Поиск номеров: Функция `find_car_numbers()` применяет регулярное выражение к тексту.

Вывод результатов: Найденные номера отображаются в виде нумерованного списка.

Подсказка: При отсутствии результатов выводится информация о правильном формате номера.

Для каждого тестового примера предусмотрен конкретный ожидаемый вывод, основанный на исправленном регулярном выражении `r'[А-Яа-я]\d{3}[А-Яа-я]{2}\s\d{2,3}'`.

Пример 1: Разные регистры букв для проверки гибкости паттерна (рис.2).

```
C:\Users\ui\Desktop> cmd /C "c:\Python313\python.exe c:\Users\ui\.vscode\extensions\ms-python.d
ebugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher 61770 -- "C:\Users\ui\Desktop\Номера м
ашин.py" "
```

ПРОСТОЙ ПОИСК АВТОМОБИЛЬНЫХ НОМЕРОВ

Выберите тестовый пример:

- 1 - Разные регистры букв (заглавные и строчные)
- 2 - Номера в контексте предложений
- 3 - Смешанные случаи (корректные и некорректные)
- 0 - Ввести свой текст

Или нажмите Enter для примера 1

Ваш выбор:

ТЕСТ 1: Разные регистры букв (по умолчанию)

Текст для анализа:

Привет! Я видел машины с номерами:
A123БВ 777, потом еще е001ке 102,
и одна была М777ММ 96, а там вообще р863ох 27рус

РЕЗУЛЬТАТЫ ПОИСКА:

Найдено номеров: 4

Список найденных номеров:

1. A123БВ 777
2. е001ке 102
3. М777ММ 96
4. р863ох 27

Проверьте формат номера:

- ✓ Буква (русская, большая или маленькая)
- ✓ Три цифры (123)
- ✓ Две буквы (АБ)
- ✓ Пробел
- ✓ 2-3 цифры региона (777)

Пример: A123БВ 777 или a123бв 777

Рисунок 2 – Результаты первого примера

Регулярное выражение [А-Яа-я] допускает любые буквы кириллицы независимо от регистра. Символы рус после номера 27 не захватываются, так как в паттерне нет \D*. В результате программа не чувствительна к регистру букв. В реальных данных текст может вводиться пользователем произвольно (Caps Lock, случайный регистр).

Пример 2: Номера в контексте предложений для проверки точности извлечения (рис.3).

```
i\.\vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher 61835
-- "C:\Users\ui\Desktop\Номера машин.py" " "
=====
ПРОСТОЙ ПОИСК АВТОМОБИЛЬНЫХ НОМЕРОВ
=====

Выберите тестовый пример:
1 - Разные регистры букв (заглавные и строчные)
2 - Номера в контексте предложений
3 - Смешанные случаи (корректные и некорректные)
0 - Ввести свой текст
Или нажмите Enter для примера 1

Ваш выбор: 2

=====
ТЕСТ 2: Номера в контексте предложений
=====

Текст для анализа:
Сегодня утром по дороге на работу я заметил
    несколько интересных машин. Первой была черная Toyota Camry
    с номером A123БВ 777, она ехала очень быстро.
    Потом видел синий BMW с госномером K999KK 199,
    который нарушил правила. А на парковке стояла
    старая Лада с номером B555AA 134.

-----
РЕЗУЛЬТАТЫ ПОИСКА:
-----

Найдено номеров: 3

Список найденных номеров:
1. A123БВ 777
2. K999KK 199
3. B555AA 134
```

Рисунок 3 – Результаты второго примера

Алгоритм игнорирует окружающий текст («Toyota Camry», «госномером», точки, запятые) и извлекает только последовательности, строго соответствующие шаблону номера.

Пример 3: Смешанные случаи (валидация) для проверки устойчивости к ошибкам (рис.4). Текст содержит 6 корректных номеров и 4 некорректных формата.

```
C:\Users\ui\Desktop> c: && cd c:\Users\ui\Desktop && cmd /C "c:\Python313\python.exe c:\Users\ui\
i\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher 61959
-- "C:\Users\ui\Desktop\Номера машин.py" "

=====
ПРОСТОЙ ПОИСК АВТОМОБИЛЬНЫХ НОМЕРОВ
=====

Выберите тестовый пример:
1 - Разные регистры букв (заглавные и строчные)
2 - Номера в контексте предложений
3 - Смешанные случаи (корректные и некорректные)
0 - Ввести свой текст
Или нажмите Enter для примера 1

Ваш выбор: 3

=====
ТЕСТ 3: Смешанные случаи
=====

Текст для анализа:
Отчет о нарушениях:
  Корректные номера: A123БВ 777, K999КК 199, М001МР 799
  Некорректные: A12БВ 77 (мало цифр), АВ123СD 99 (латиница)
  Еще номера: E111ЕЕ 77, P456ОР 98, Т789ТТ 134
  Словом номер: АБВГДЕ (не номер)
  Номер без пробела: A123БВ777 (не найдется)

-----
РЕЗУЛЬТАТЫ ПОИСКА:
-----

Найдено номеров: 6

Список найденных номеров:
1. A123БВ 777
2. K999КК 199
3. М001МР 799
4. E111ЕЕ 77
5. P456ОР 98
6. Т789ТТ 134
```

Рисунок 4 – Результаты третьего примера

Критически важный тест на ложные срабатывания. Программа не только находит правильное, но и игнорировать неправильное.

Неправильные номера:

- 1) A12БВ 77 — не найден (только 2 цифры вместо 3).
- 2) АВ123СD 99 — не найден (латинские буквы вместо кириллицы).
- 3) АБВГДЕ — не найден (отсутствуют цифры и пробел).
- 4) A123БВ777 — не найден (отсутствует обязательный пробел \s).

Комбинация этих трех тестов позволяет подтвердить, что разработанная система поиска является гибкой (работает с разным регистром), точной (выделяет номера из текста) и надежной (отсеивает некорректные форматы).

4 Выводы

В данной статье рассмотрен метод поиска автомобильных номеров в текстовых данных с использованием регулярных выражений на Python. На примере создания консольного приложения продемонстрированы ключевые преимущества подхода: возможность описывать сложные текстовые паттерны в компактной форме, что упрощает обработку больших объемов текста и делает код более читаемым и поддерживаемым.

Разработанная программа представляет собой простой и эффективный инструмент для извлечения структурированной информации из

неструктурированного текста. Поддержка различных регистров букв и интерактивный режим ввода делают приложение удобным для тестирования и демонстрации.

Использование стандартного модуля обеспечило кроссплатформенность решения, что позволяет запускать приложение на различных операционных системах без дополнительных изменений в коде. Приведенные примеры кода и описание работы регулярного выражения могут служить основой для дальнейшего изучения и применения в более сложных системах, включая интеграцию с ANPR и базами данных.

Библиографический список

1. Zandamela F., Seletani F., Malatjie P., Kunene D., Nana M. A., Sekopa T. Uncovering the ANPR Performance Gap: A Commercial Systems Analysis // Journal of Transportation Technologies. 2024. Т. 14. № 2. С. 112–128. URL: https://link.springer.com/chapter/10.1007/978-3-031-78255-8_11
2. Саммерфилд М. Python на практике. М.: ДМК Пресс, 2016. 338 с. URL: https://dmkpress.com/catalog/computer/programming/python/978-5-97060-095-5/?srsltid=AfmBOopJxqfrCXhS40FRaRmziR-_bpwsje4_vA8QSyTvQGw4QZ-dWK0E
3. Fuggitti F., Chakraborti T. NL2LTL—a python package for converting natural language (NL) instructions to linear temporal logic (LTL) formulas // Proceedings of the AAAI Conference on Artificial Intelligence. 2023. Т. 37. № 13. С. 16428–16430. URL: <https://dl.acm.org/doi/10.1609/aaai.v37i13.27068>
4. Zhong Z. et al. An empirical study on package-level deprecation in Python ecosystem // 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE). IEEE, 2025. С. 66–77. URL: <https://dl.acm.org/doi/abs/10.1109/ICSE55347.2025.00046>
5. Галимуллин Н. Р. Python: использование Python для автоматизации повседневных задач // Экономика и управление: проблемы, решения. 2024. Т. 9. № 9. С. 150. URL: <https://cyberleninka.ru/article/n/sravnenie-i-analiz-metodov-i-instrumentov-primenyaemyh-dlya-sbora-dannyh>
6. Юлкина Н. С., Шмелев М. С., Фомина И. К. Сравнение и анализ методов и инструментов, применяемых для сбора данных // Молодой исследователь Дона. 2023. Т. 8. № 6. С. 49–54. URL: https://mid-journal.ru/upload/mid/iblock/70a/9_YUlkina_49_54.pdf