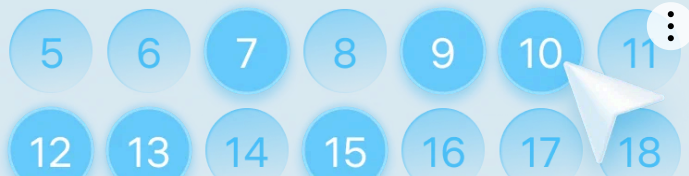


Все важнейшие
IT-события здесь



empenoso

20 янв в 03:25

Python и нечеткое сопоставление: решение проблемы разнобоя в адресах

🟢 Простой ⌚ 10 мин 👁 4.2K

Python*, Open source*, Геоинформационные сервисы*

Кейс

Иногда приходится заниматься сравнением больших списков адресов, в которых адреса записаны совершенно по разному без внятных идентификаторов вроде номера объекта - есть только адрес. Один и тот же адрес может фигурировать в различных списках следующим образом:

- "д. Малое Шилово, ул. Березовая, д. 7" и "Березовая 7_М Шилово".
- "п. Ласьва, ул. Весенняя, д. 5" и "Весенняя 5_Ласьва".
- "Луговой пер 5, Краснокамск г" и "г. Краснокамск, пер. Луговой, 5".
- "д. Новая Ивановка, ул. Солнечная, 18" и "д.Новая Ивановка, ул.Солнечная, 18".

Уже выделенные отдельно адреса могут выглядеть как на скриншоте Экселя ниже. А пример поставленной задачи может звучать так: **«В реестре поданных объектов отметить все согласованные объекты (из общего списка согласованных)»**.

Если отбросить вариант ручного исполнения и обратиться к скриптам, то мне видится всего два решения:

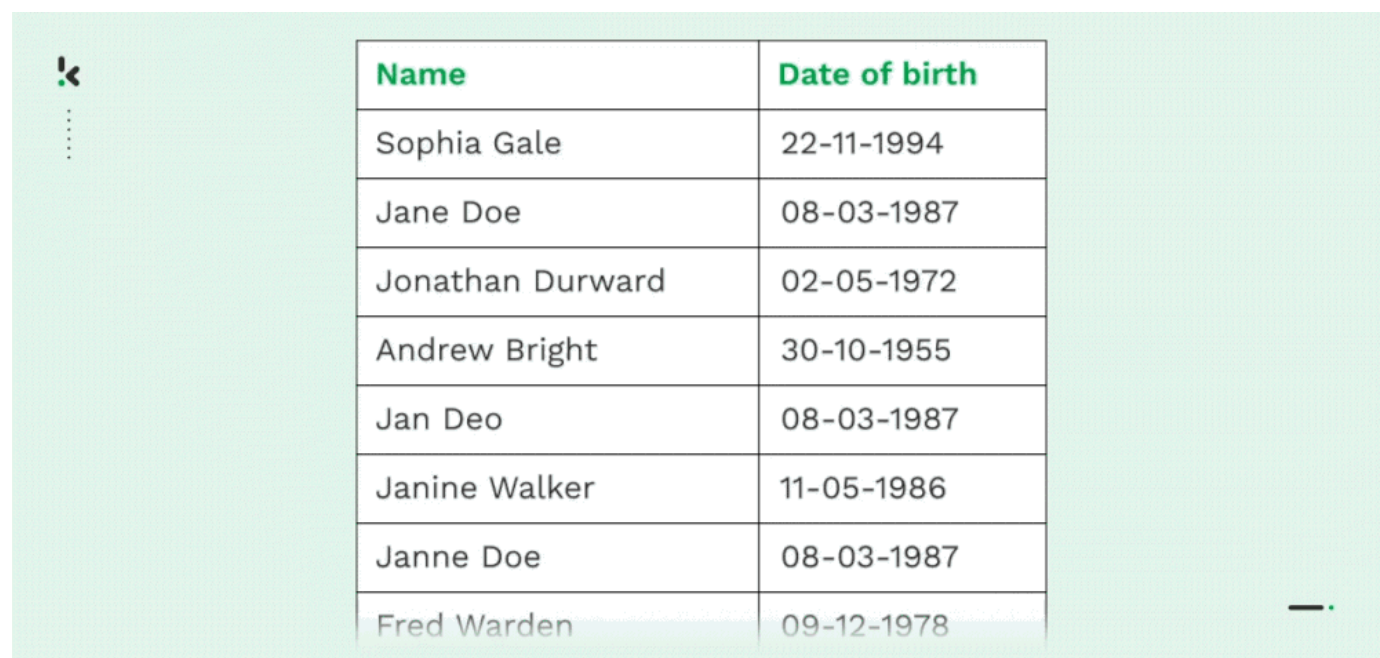
1. Использовать алгоритмы нечёткого сопоставления.
2. Использовать геокодирование адресов.

sub...	Вход	sub...	Вход
Файл	Глав	Встав	Рисое
Буфер обмена	Шрифт	Выравнивание	Число
A378	A	D86	A
322 г. Краснокамск, ул. Раздольная, 11		118 с. Стряпунята, ул. Набережная, 4а	
323 д. Семичи, ул. Золотая, 2		119 г. Краснокамск, пер. Луговой, 5А	
324 д. Никитино, ул. Полевая, 20		120 г. Краснокамск, пер. Свободный, 7	
325 с. Мысы, ул. Радужная, 8		121 г. Краснокамск, пер. Черный, 3	
326 п. Ласьва, ул. Снежная, 5		122 г. Краснокамск, ул. Осинская, 8	
327 п. Ласьва, ул. Снежная, 12		123 д. Карабаи, ул. Полевая, 37	
328 ДНТ Никитино, ул. Крайняя, 9а		124 д. Большое Шилово, ул. Мирная, 16	
329 ст. Шабуничи, пер. Полевой, 1Б		125 д. Большое Шилово, ул. Сюзьвенская, 19/2	
330 ст. Шабуничи, ул. Тракторная, 3		126 д. Большое Шилово, ул. Сюзьвенская, 27	
331 ст. Шабуничи, ул. 3-я Тракторная, 21а		127 д. Волеги, ул. Дорожная, 1	
332 д. Семичи, ул. Вишневая, 10		128 д. Конец-Бор, ул. Конец-Борская, 2Б	
333 г. Краснокамск, ул. Камская, 62		129 д. Конец-Бор, ул. Некрасова, 18А	
334 с. Мысы, ул. Рублевская, 45		130 д. Конец-Бор, ул. Трудовая, 51	
335 п. Оверята, пер. Сосновый, 8		131 д. Малое Шилово, ул. Дачная, 4	
336 д. Большое Шилово, ул. Сюзьвенская, 11		132 д. Нижние Симонята, ул. Набережная, 14а	
337 д. Новая Ивановна, ул. Тракторная, 15		133 д. Новая Ивановка ул. Железнодорожная, 5	
338 д. Мошни, ул. Запрудная, 15		134 д. Новая Ивановка ул. Зеленая, 27-2	
339 д. Хухрята, ул. Изумрудная, 2		135 д. Семичи, ул. 1-я Подгорная, 2	
340 г. Краснокамск, ул. Камская, 60		136 д. Семичи, ул. 1-я Подгорная, 22	
341 с. Мысы, проезд Рыбацкий, 13		137 д. Семичи, ул. Виноградная, 6	
342 г. Краснокамск, пер. Нагорный, 3а		138 д. Семичи, ул. Земляничная, 2	
343 д. Конец-Бор, пер. Технический, 1Б		139 д. Семичи, ул. Молодежная, 11	
344 с. Мысы, ул. Попова, 6		140 д. Семичи, ул. Молодежная, 20	
345 п. Ласьва, кв-л Восточный, 15а		141 д. Семичи, ул. Светлая, 21	
346 д. Брагино, ул. Лесная, 5		142 тер. ДНП Южные Мысы, ул. Лучистая, 37	
Лист1	Параметры отображения	Лист1	Параметры отображения

Варианты решения этой задачи

Первый вариант – использование алгоритмов нечёткого сопоставления (fuzzy matching). Эти алгоритмы позволяют сравнивать строки, учитывая возможные опечатки, разные порядок слов и сокращения. В нашем случае, алгоритм сможет распознать "д. Малое Шилово, ул. Березовая, д. 7" и "Березовая 7_М Шилово" как варианты одного и того

же адреса, несмотря на различия в формате и сокращения. Fuzzy matching оценивает «схожесть» строк, выдавая число от 0 до 1, что позволяет гибко настраивать порог совпадения и находить соответствия даже при значительных расхождениях в написании. Это делает данный метод весьма эффективным для обработки больших списков адресов с вариативностью написания.

A screenshot of a web application interface. On the left, there is a vertical sidebar with a green 'k' logo and a list of items. The main area displays a table with two columns: 'Name' and 'Date of birth'. The table contains eight rows of data. The background is a light green color.

Name	Date of birth
Sophia Gale	22-11-1994
Jane Doe	08-03-1987
Jonathan Durward	02-05-1972
Andrew Bright	30-10-1955
Jan Deo	08-03-1987
Janine Walker	11-05-1986
Janne Doe	08-03-1987
Fred Warden	09-12-1978

Не прямо в тему, но наглядно. Источник: pub.aimind.so

Второй подход – геокодинг. Этот метод преобразует текстовое описание адреса в географические координаты. Получив координаты для каждого адреса в обоих списках, можно сравнивать их близость и таким образом находить соответствия. Геокодинг полезен для проверки корректности адресов и выявления дубликатов, записанных по-разному. Однако, этот метод имеет существенные ограничения в контексте данной задачи. Во-первых, не все адреса могут быть найдены на картах. Если объект ещё строится, то адрес еще не внесен в картографические сервисы. Во-вторых, геокодинг может быть неточным, особенно в сельской местности. Таким образом, полагаться исключительно на геокодинг в данном случае рискованно.

Geocoding

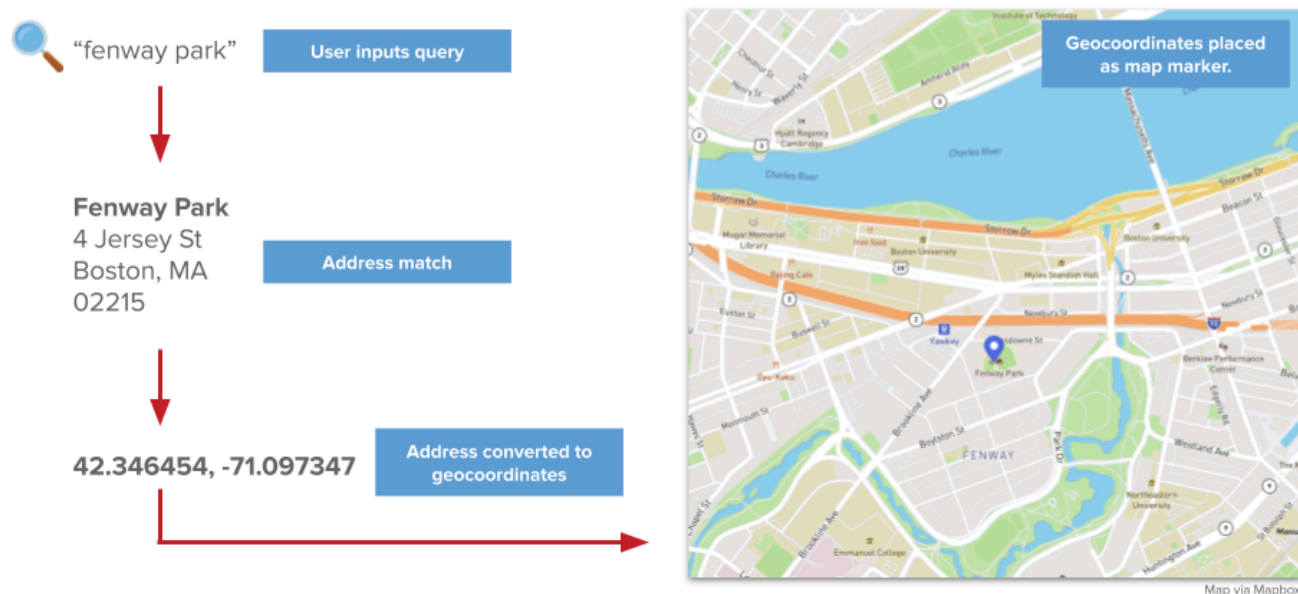


Иллюстрация геокодинга. Источник: pubnub.com

Для нашей задачи, где требуется сравнить большие списки адресов с высокой вариативностью написания и наличием потенциально «несуществующих» адресов, алгоритмы нечёткого сопоставления представляются более подходящим решением. Они не требуют наличия адреса на карте и способны эффективно обрабатывать различные варианты написания одного и того же адреса. Гибкость настройки позволяет подобрать оптимальный баланс между точностью и полнотой поиска соответствий, минимизируя как ложноположительные, так и ложноотрицательные результаты. В то время как геокодинг может служить дополнительным инструментом для верификации результатов, основным методом сравнения адресов в данном случае следует выбрать fuzzy matching.

Подготовка данных

Прежде чем приступить к сравнению адресов, необходимо привести их к единому формату. Это значительно повысит точность алгоритмов нечёткого сопоставления. Различия в регистре, сокращениях, пунктуации и лишние пробелы могут помешать алгоритму правильно идентифицировать одинаковые адреса. Например, "д. Малое Шилово" и "малое шилово" будут рассматриваться как разные адреса, если не провести предварительную обработку.

Для обработки списков адресов используем Python с библиотеками pandas, openpyxl и fuzzywuzzy. pandas предоставляет удобные инструменты для работы с табличными

данными, `openpyxl` позволяет читать и записывать файлы Excel, а `fuzzywuzzy` реализует алгоритмы нечёткого сопоставления.

```
def clean_address(address):
    print(f"Очистка адреса: {address}") # Вывод текущего адреса для очистки
    if pd.isnull(address): # Проверяем, является ли адрес пустым значением
        return None

    # Приведение к нижнему регистру
    address = address.lower()

    # Список замен с сохранением структуры
    replacements = [
        (r"\бп/ст\b", ""), # Убираем "п/ст"
        (r"\бднт\b", ""), # Убираем "ДНТ"
        (r"\бснт\b", ""), # Убираем "СНТ"
        (r"\бднп\b", ""), # Убираем "ДНП"
        (r"\бкв-л\b", ""), # Убираем "кв-л"
        (r"\бпоезд\b", ""), # Убираем "поезд"
        (r"\бквартал\b", ""), # Убираем "квартал"
        (r"\бд\. \s?", ""), # Убираем "д." с пробелом
        (r"\бг\. \s?", ""), # Убираем "г." с пробелом
        (r"\бпер\. \s?", ""), # Убираем "пер." с пробелом
        (r"\бул\s?", ""), # Убираем "ул" с пробелом
        (r"\бп\. \s?", ""), # Убираем "п." с пробелом
        (r"\бс\. \s?", ""), # Убираем "с." с пробелом
        (r"\бст\. \s?", ""), # Убираем "ст." с пробелом
        (r"\бпр-д\b", "") # Убираем "пр-д"
    ]

    # Применение замен
    for pattern, replacement in replacements:
        address = re.sub(pattern, replacement, address)

    # Удаление текста в скобках
    address = re.sub(r"\([^)]*\)", "", address) # Убираем текст в скобках

    # Удаление лишних символов, но с сохранением структуры
    address = re.sub(r"[.,]", "", address) # Убираем точки и запятые
    address = re.sub(r"\s{2,}", " ", address) # Убираем множественные пробелы
    address = re.sub(r"[\"]", "", address) # Убираем кавычки
    address = address.strip() # Убираем пробелы по краям
```

```
print(f"Очищенный адрес: {address}") # Вывод очищенного адреса
return address
```

Для приведения адресов к единому формату используем функцию `clean_address`, представленную в коде выше. Она приводит адрес к нижнему регистру, удаляет сокращения (например, "д.", "ул.", "г."), текст в скобках, лишние пробелы и знаки препинания. Применение регулярных выражений обеспечивает гибкость и эффективность очистки. Функция также включает вывод исходного и очищенного адресов для контроля процесса обработки.

Перед началом работы необходимо установить упомянутые библиотеки. Это можно сделать с помощью `pip`:

```
pip install pandas openpyxl fuzzywuzzy
```

После установки библиотек и подготовки данных можно переходить к реализации алгоритма нечёткого сопоставления.

Основы работы с fuzzywuzzy

Библиотека `fuzzywuzzy` предоставляет несколько функций для сравнения строк, основанных на алгоритме Левенштейна. Этот алгоритм вычисляет минимальное количество операций (вставка, удаление, замена символов), необходимых для преобразования одной строки в другую. Чем меньше операций требуется, тем больше сходство между строками.

`fuzzywuzzy` предлагает три основные функции:

- **fuzz.ratio**: Сравнивает строки целиком, учитывая порядок слов. Например, `fuzz.ratio("ул. Ленина 10", "Ленина ул 10")` вернёт относительно низкий балл, несмотря на то, что слова одинаковые, но расположены в разном порядке.
- **fuzz.partial_ratio**: Ищет наиболее похожую подстроку. Полезно, когда одна строка является частью другой. Например, `fuzz.partial_ratio("ул. Ленина 10", "г. Москва, ул. Ленина 10, кв 5")` вернёт высокий балл, так как первая строка полностью содержится во второй.

- **fuzz.token_sort_ratio**: Сначала сортирует слова в строках по алфавиту, а затем сравнивает их с помощью `fuzz.ratio`. Это позволяет игнорировать порядок слов. В нашем примере `fuzz.token_sort_ratio("ул. Ленина 10", "Ленина ул 10")` выдаст высокий балл, поскольку после сортировки строки станут идентичными.

```
# Функция для поиска совпадений с помощью fuzzy matching
def match_address(row, approved_addresses):
    cleaned_address = row["cleaned_address"]
    if not cleaned_address: # Проверка, если адрес пустой (None или пустая строка)
        print("Пропущен пустой адрес")
        return None

    # Извлекаем цифры из текущего адреса
    current_digits = set(re.findall(r'\d+', cleaned_address))
    if not current_digits:
        print(f"Адрес без цифр пропущен: {cleaned_address}")
        return None

    # Отфильтровываем список одобренных адресов, оставляя только те, где есть совпадающ
    filtered_addresses = [
        addr for addr in approved_addresses
        if current_digits & set(re.findall(r'\d+', addr))
    ]

    if not filtered_addresses:
        print(f"Совпадений по цифрам не найдено для адреса: {cleaned_address}")
        return None

    print(f"Поиск совпадения для адреса: {cleaned_address}") # Лог текущего адреса
    result = process.extractOne(cleaned_address, filtered_addresses, scorer=fuzz.token_

    if result: # Если совпадение найдено
        match, score = result
        print(f"Найдено совпадение: {match} с оценкой {score}") # Вывод найденного сое
        return match if score > 70 else None # Возвращаем совпадение только при достат
    else:
        print("Совпадений не найдено")
        return None
```

Используя `fuzz.token_sort_ratio` в сочетании с предварительной фильтрацией по совпадающим цифрам в адресах. Это позволяет существенно ускорить процесс и повысить точность сопоставления, так как сравниваются только те адреса, номера которых потенциально могут совпадать.

Порог сходства установлен на 70, что означает, что совпадение считается найденным, только если оценка `fuzz.token_sort_ratio` превышает это значение. Это позволяет отсеять ложные совпадения.

Скрипт для сопоставления списков разных адресов

Скрипт вначале загружает данные из файлов Excel с помощью библиотеки `pandas`, после загрузки скрипт очищает адреса в обоих списках, используя функцию `clean_address`, приводя их к единому формату.

Затем начинается процесс сопоставления. Для каждого адреса из реестра поданных объектов скрипт ищет соответствие в реестре согласованных объектов с помощью библиотеки `fuzzywuzzy`. Функция `process.extractOne`, используемая в коде, позволяет эффективно находить совпадения в большом списке, применяя алгоритм `token_sort_ratio`. Предварительная фильтрация по совпадающим цифрам в адресах значительно ускоряет обработку больших списков.

Результаты сопоставления, включая найденный адрес и отметку о согласованности "+ " или нет "X", добавляются в исходный реестр поданных объектов. Окончательный результат сохраняется в новый файл Excel.

Полный код:

```
# pip install pandas openpyxl fuzzywuzzy

# Подробнее: https://habr.com/ru/articles/873242/

"""
Иногда приходится заниматься сравнением больших списков адресов, в которых адреса записаны по-разному.

• "д. Малое Шилово, ул. Березовая, д. 7" и "Березовая 7_М Шилово".
• "п. Ласьва, ул. Весенняя, д. 5" и "Весенняя 5_Ласьва".
• "Луговой пер 5, Краснокамск г" и "г. Краснокамск, пер. Луговой, 5".
• "д. Новая Ивановка, ул. Солнечная, 18" и "д.Новая Ивановка, ул.Солнечная, 18".
```


Уже выделенные отдельно адреса могут выглядеть как на скриншоте Экселя. А пример постав

Если отбросить вариант ручного исполнения и обратиться к скриптам, то мне видится всего

✓ Использовать алгоритмы нечёткого сопоставления.

✓ Использовать геокодинг адресов.

"""

```
import sys
sys.stdout.reconfigure(encoding='utf-8')

import re
import pandas as pd
from fuzzywuzzy import fuzz, process

def clean_address(address):
    print(f"Очистка адреса: {address}") # Вывод текущего адреса для очистки
    if pd.isnull(address): # Проверяем, является ли адрес пустым значением
        return None

    # Приведение к нижнему регистру
    address = address.lower()

    # Список замен с сохранением структуры
    replacements = [
        (r"\бп\ст\b", ""), # Убираем "п/ст"
        (r"\бднт\b", ""), # Убираем "ДНТ"
        (r"\бснт\b", ""), # Убираем "СНТ"
        (r"\бднп\b", ""), # Убираем "ДНП"
        (r"\бкв-л\b", ""), # Убираем "кв-л"
        (r"\бпоезд\b", ""), # Убираем "поезд"
        (r"\бквартал\b", ""), # Убираем "квартал"
        (r"\бд\. \s?", ""), # Убираем "д." с пробелом
        (r"\бг\. \s?", ""), # Убираем "г." с пробелом
        (r"\бпер\. \s?", ""), # Убираем "пер." с пробелом
        (r"\бул\s?", ""), # Убираем "ул" с пробелом
        (r"\бп\. \s?", ""), # Убираем "п." с пробелом
        (r"\бс\. \s?", ""), # Убираем "с." с пробелом
        (r"\бст\. \s?", ""), # Убираем "ст." с пробелом
        (r"\бпр-д\b", "") # Убираем "пр-д"
    ]
```

]

```

# Применение замен
for pattern, replacement in replacements:
    address = re.sub(pattern, replacement, address)

# Удаление текста в скобках
address = re.sub(r"\([^()]*\)", "", address) # Убираем текст в скобках

# Удаление лишних символов, но с сохранением структуры
address = re.sub(r"[.,]", "", address) # Убираем точки и запятые
address = re.sub(r"\s{2,}", " ", address) # Убираем множественные пробелы
address = re.sub(r"\"", "", address) # Убираем кавычки
address = address.strip() # Убираем пробелы по краям

print(f"Очищенный адрес: {address}") # Вывод очищенного адреса
return address

# Функция для поиска совпадений с помощью fuzzy matching
def match_address(row, approved_addresses):
    cleaned_address = row["cleaned_address"]
    if not cleaned_address: # Проверка, если адрес пустой (None или пустая строка)
        print("Пропущен пустой адрес")
        return None

    # Извлекаем цифры из текущего адреса
    current_digits = set(re.findall(r'\d+', cleaned_address))
    if not current_digits:
        print(f"Адрес без цифр пропущен: {cleaned_address}")
        return None

    # Отфильтровываем список одобренных адресов, оставляя только те, где есть совпадающ
    filtered_addresses = [
        addr for addr in approved_addresses
        if current_digits & set(re.findall(r'\d+', addr))
    ]

    if not filtered_addresses:
        print(f"Совпадений по цифрам не найдено для адреса: {cleaned_address}")
        return None

    print(f"Поиск совпадения для адреса: {cleaned_address}") # Лог текущего адреса
    result = process.extractOne(cleaned_address, filtered_addresses, scorer=fuzz.token_

    if result: # Если совпадение найдено

```

```

    match, score = result
    print(f"Найдено совпадение: {match} с оценкой {score}") # Вывод найденного совпадения
    return match if score > 70 else None # Возвращаем совпадение только при достаточной уверенности
else:
    print("Совпадений не найдено")
    return None

# Загружаем данные из Excel-файлов
print("Загрузка данных...")
submitted_df = pd.read_excel("submitted.xlsx") # Реестр поданных объектов
approved_df = pd.read_excel("approved.xlsx") # Реестр согласованных объектов

# Очистка адресов в обоих реестрах
print("Очистка адресов в таблицах...")
submitted_df["cleaned_address"] = submitted_df["address"].apply(clean_address)
approved_df["cleaned_address"] = approved_df["address"].apply(clean_address)

# Формируем список очищенных адресов из реестра согласованных объектов
approved_addresses = approved_df["cleaned_address"].dropna().tolist()

# Ищем совпадения и добавляем их в реестр поданных объектов
print("Сопоставление адресов...")
submitted_df["matched_address"] = submitted_df.apply(
    lambda x: x["address"] if x["address"] in approved_addresses else None, axis=1
)

# Добавляем отметку о согласованности
print("Добавление отметки о согласованности...")
# Проверяем наличие совпадения и добавляем соответствующий символ
submitted_df["is_approved"] = submitted_df["matched_address"].notnull().apply(
    lambda x: "+" if x else "X"
)

# Сохраняем результат в новый Excel-файл
print("Сохранение результатов...")
submitted_df.to_excel("submitted_with_matches_v2.xlsx", index=False)

print("Готово! Результаты сохранены в 'submitted_with_matches_v2.xlsx'.")

```

Результат работы скрипта:

	A	B	C	D
1	address	cleaned_address	matched_address	is_approved
205	п. Ласьва, ул. Весенняя, 10а	ласьва весенняя 10а	ласьва весенняя 10а	+
206	с. Мысы, ул. Линейная, 19а	мысы линейная 19а	мысы линейная 19а	+
207	СНТ Никитино, ул. Парковая, 20	никитино парковая 20		×
208	д. Семичи, ул. Яблочная, 4	семичи яблочная 4	яблочная 4 семичи д	+
209	ДНТ Никитино, ул. Лесная, 15	никитино лесная 15	лесная 15 никитино тер	+

Заключение

Автоматизация процесса сопоставления адресов с помощью Python позволяет значительно сэкономить время и исключить ошибки, связанные с человеческим фактором. Вместо утомительной ручной проверки скрипт быстро и точно обрабатывает большие объемы данных. Более того, представленный скрипт легко адаптируется под похожие задачи, требующие сравнения текстовых строк, например, сопоставление наименований товаров или данных клиентов.

Для повышения точности сопоставления можно рассмотреть комбинирование fuzzy matching с геокодингом. Если адрес можно успешно геокодировать, то координаты служат дополнительным критерием для подтверждения совпадения.

Буду рад обсудить возможные улучшения и ответы на ваши вопросы в комментариях.

Автор: Михаил Шардин

 [Моя онлайн-визитка](#)

 [Telegram «Умный Дом Инвестора»](#)

20 января 2025 г.

Теги: [алгоритм](#), [Алгоритм нечёткого сопоставления](#), [карта](#), [fuzzywuzzy](#)

Хабы: [Python](#), [Open source](#), [Геоинформационные сервисы](#)

Редакторский дайджест

Присылаем лучшие статьи раз в месяц



**183**

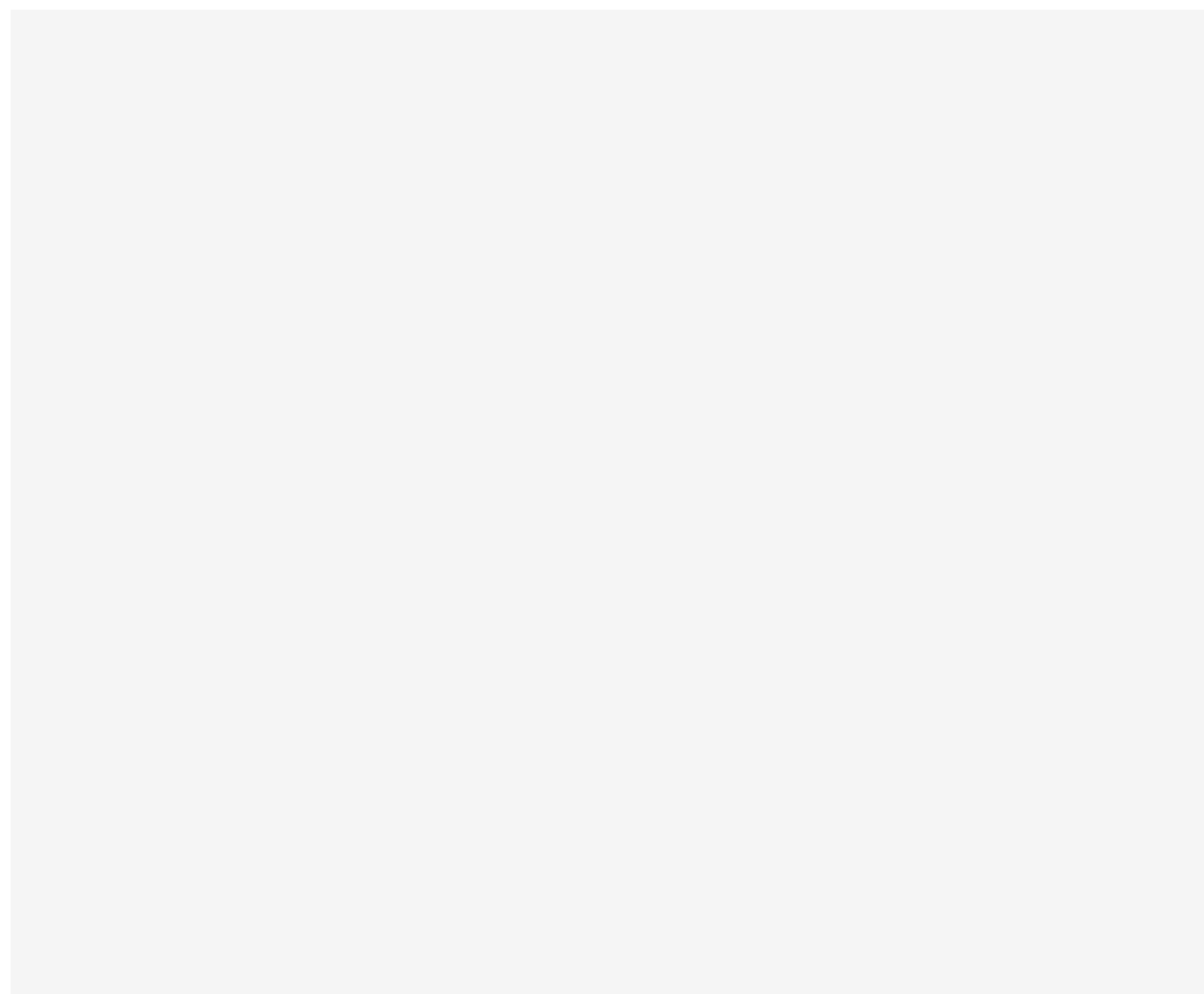
Карма

87.1

Рейтинг

Михаил Шардин @empenoso

Автоматизация / Данные / Финансы / Умные дома

[Подписаться](#)[Сайт](#) [Сайт](#) [Github](#) [Комментарии 31](#)

Публикации

ЛУЧШИЕ ЗА СУТКИ

ПОХОЖИЕ



rssdev10

22 часа назад

Почему въехав по «визе талантов» в США я с радостью вернулся в Россию



Средний



32 мин



34K

Мнение

+169

108

492



melnik909

18 часов назад

Вы не знаете CSS. Мои вопросы о CSS с ответами. Часть 2



Средний



7 мин



2K

Обзор

+38

31

1



DAN_SEA

16 часов назад

Генерация случайных чисел



Средний



10 мин



2.3K

Обзор

+31

23

33



OrkBiotechnologist

22 часа назад

VPS за 139 рублей — дом для вашего резюме на основе Hugo



Простой



7 мин



8.2K

Тutorial

+28

45

12

**nebykoff_anton**

23 часа назад

До и после: оптимизация изображений для Lighthouse и не только

**Простой**

8 мин



943

Тutorial

**+28**

24



2

**Корcheniy**

23 часа назад

Имитатор касаний. Ч2: Железная часть

**Средний**

9 мин



1.6K

Case

**+24**

20



12

**PatientZero**

1 час назад

Пишем стек TCP/IP с нуля: Ethernet, ARP, IPv4 и ICMPv4

**Простой**

13 мин



762

Tutorial

Перевод

**+16**

22



1

**tertiumnon**

17 часов назад

Минимум книг, которые нужно прочитать начинающему или продолжающему свою кривую обучения программисту

**Простой**

3 мин



8.3K

Обзор

**+16**

182



17

**lbkanter**

49 минут назад

Бэкдор Auto-color: разбор угрозы, технический анализ и способы защиты

 Средний

 4 мин

 228

Обзор

 +13

 5

 2


alexander-shustanov

18 часов назад

В поисках идеального Database-клиента для IDE: Amplicode выбирает DBeaver

 Простой

 6 мин

 2.5K

 +13

 13

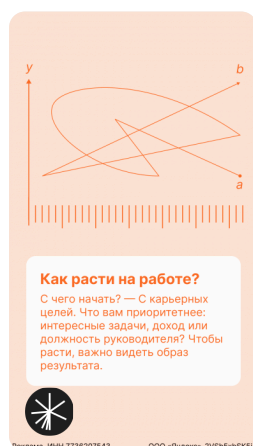
 7

Паглядзіце ўважліва: наш першы рейтинг ІТ-работодателів РБ

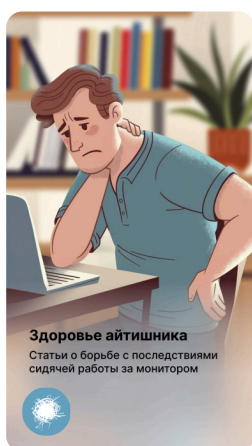
Турбо

Показать еще

ИСТОРИИ



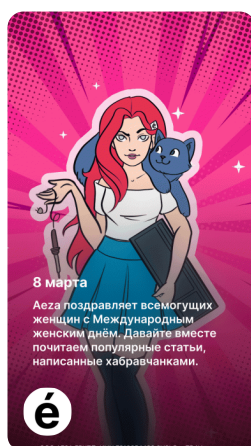
Как расти на работе?



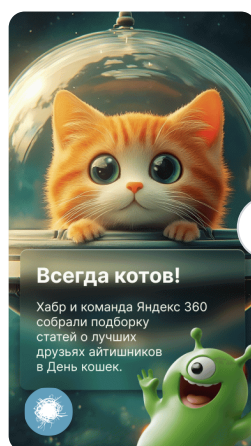
Здоровье айтишника



Угадайте будущее в новом сезоне



С праздником весны!



Всегда котов!

КУРСЫ

Аналитик 2.0

По факту набора · SF Education

Бизнес-аналитик

По факту набора · SF Education

Бэкенд-разработчик на Python

По факту набора · SF Education

Python-разработчик: расширенный курс

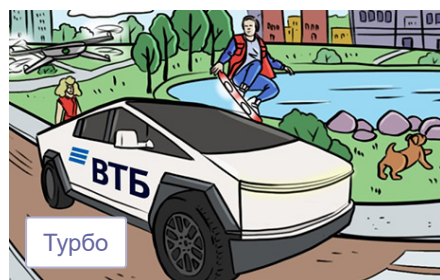
По факту набора · Нетология

DevOps-инженер

По факту набора · Skillbox

[Больше курсов на Хабр Карьере](#)

МИНУТОЧКУ ВНИМАНИЯ



Сезон футурологии на Хабре:
какой будет жизнь 3.0



Экономим деньги со скидками в
Промокодусе



«Люк, я твой фактор!» —
защищаем подключения с MFA

РАБОТА

[Django разработчик](#)

20 вакансий

[Data Scientist](#)

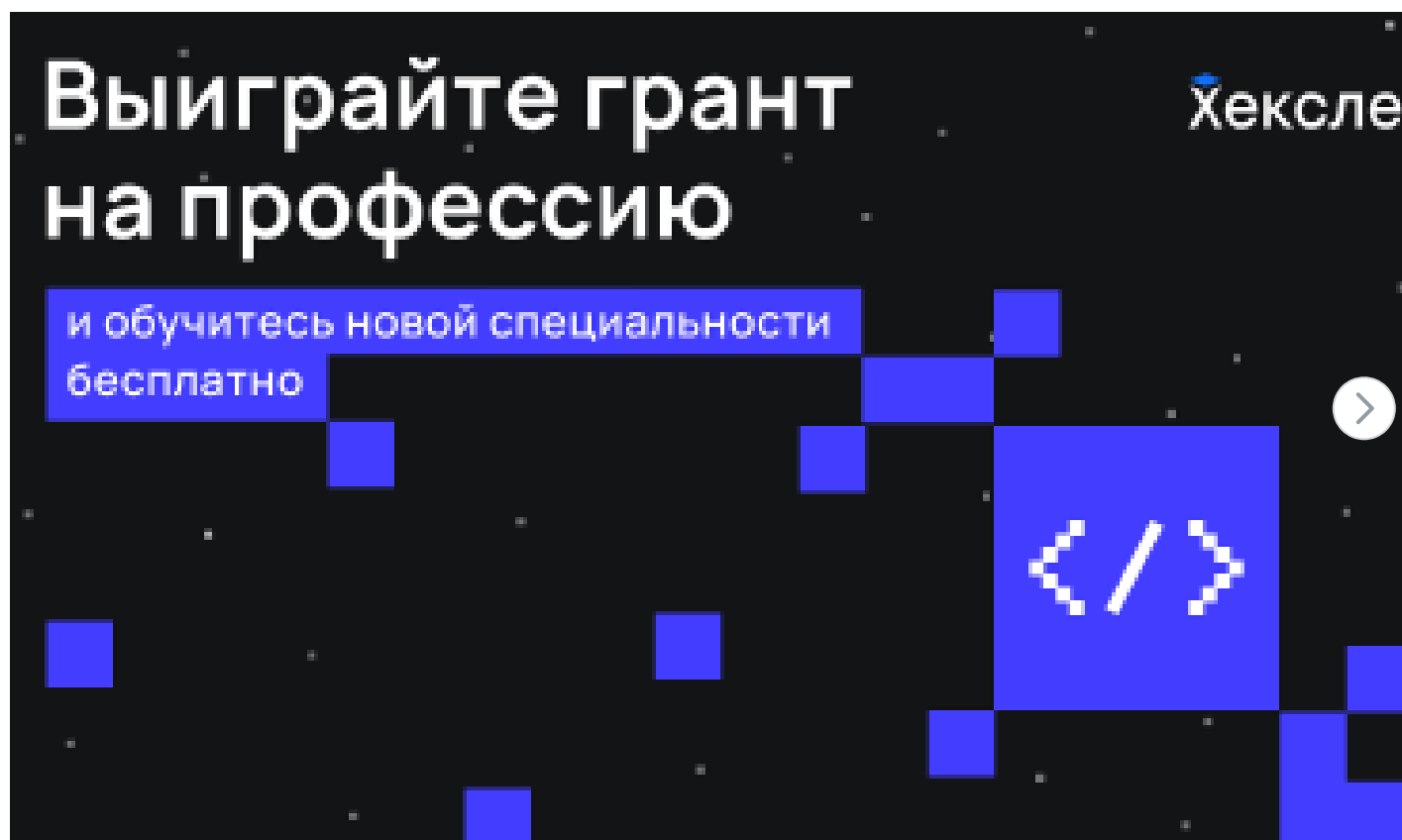
55 вакансий

[Python разработчик](#)

65 вакансий

[Все вакансии](#)

БЛИЖАЙШИЕ СОБЫТИЯ



17 февраля – 24 марта

Конкурс «Снежный код» от Хекслета. Три гранта на бесплатное 10-месячное обучение

Онлайн

Разработка

[Больше событий в календаре](#)

Хабр



 [Настройка языка](#)

Техническая поддержка

© 2006–2025, Habr