Программные системы и вычислительные методы
Правильная ссылка на статью:
Алпатов А.Н., Богатырева А.А. Формат хранения данных для аналитических систем на основе метаданных и графов зависимостей между CSV и JSON // Программные системы и вычислительные методы. 2024. № 2. DOI: 10.7256/2454-0714.2024.2.70229 EDN: TVEPRE URL: https://nbpublish.com/library_read_article.php?id=70229

Формат хранения данных для аналитических систем на основе метаданных и графов зависимостей между CSV и JSON

Алпатов Алексей Николаевич

ORCID: 0000-0001-8624-1662

доцент, кафедра ИиППО, МИРЭА - Российский технологический университет

119454, Россия, г. Москва, ул. Проспекте Вернадского, 78

aleksej01-91@mail.ru

Богатырева Анна Алексеевна

студент, кафедра инструментального и прикладного программного обеспечения, МИРЭА-Российский технологический университет

111033, Россия, г. Москва, пр-д Танковый, 4А, кв. 24

pecherni@gmail.com

Статья из рубрики "Математическое и программное обеспечение новых информационных технологий"

DOI:

10.7256/2454-0714.2024.2.70229

EDN:

TVEPRE

Дата направления статьи в редакцию:

25-03-2024

Дата публикации:

01-04-2024

Аннотация: В современном информационном обществе объемы данных постоянно растут, и эффективная их обработка становится ключевой для предприятий. Передача и хранение этих данных также играет критическую роль. Большие данные, которые

используются в системах аналитики, чаще всего передаются в одном из двух популярных форматов: CSV для структурированных данных и JSON для неструктурированных данных. Однако существующие форматы файлов могут оказаться неэффективными или недостаточно гибкими для определенных задач анализа данных. Например, они могут не поддерживать сложные структуры данных или не предоставлять достаточного контроля над метаданными. Или же аналитические задачи могут требовать дополнительной информации о данных, такой как метаданные, схема данных и т.д. Исходя из вышеназванного, предметом данного исследования является формат данных, основанный на совместном использовании CSV и JSON для обработки и анализа больших объемов информации. Предлагается вариант совместного использования обозначенных типов данных для реализации нового формата данных. Для этого введены обозначения для структуры данных, включающей CSV-файлы, JSON-файлы, метаданные и граф зависимостей. Описаны различные типы функций, такие как агрегирующие, преобразующие, фильтрующие и т.д. Приведены примеры применения этих функций к данным. Предложенный подход представляет собой методику, которая может значительно облегчить процессы анализа и обработки информации. В её основе лежит формализованный подход, который позволяет установить четкие правила и процедуры для работы с данными, что способствует их более эффективной обработке. Другим аспектом предложенного подхода является определение критерия выбора наиболее подходящего формата хранения данных. Этот критерий основан на математических принципах теории информации и энтропии. Введение критерия выбора формата данных на основе энтропии позволяет оценить информационную содержательность и компактность данных. Этот подход основывается на расчете энтропии для выбранных форматов и весовых коэффициентов, отражающих важность каждого значения данных. Путем сравнения энтропий можно определить требуемый формат передачи данных. Такой подход учитывает не только компактность данных, но и контекст их использования, а также возможность включения дополнительной метаинформации в сами файлы и поддержку данных, готовых к анализу.

Ключевые слова:

Форматы хранения данных, JSON, CSV, Аналитически готовые данные, Метаданные, Обработка данных, Анализ данных, Интеграция форматов данных, Apache Parquet, Большие данные

Введение

Задача интеграции данных в различных научных областях является чрезвычайно важной. Развитие такого направления, как большие данные, привело к появлению большого числа разрозненных инструментов, используемых при проведении различных научных исследований, и написанных с использованием разных стеков технологий, а самое главное, форматы используемых данных разнятся от инструмента к инструменту, что приводит к необходимости переформатирования данных, что для больших данных может быть затруднительно.

Для решения данной проблемы можно использовать разнообразные программные интерфейсы (API-Application programming interface)[1]. Данный подход имеет ряд несомненных преимуществ, но одним из существенных недостатков является высокая нагрузка на сериализацию и десериализацию данных, которая может стать серьезным узким местом для приложений. Кроме того, отсутствует стандартизация представления

данных, что приводит к тому, что разработчики выполняют настраиваемые интеграции с использованием существующих или настраиваемых систем. Для решения обозначенных выше проблем можно применить альтернативный метод обмена данными: обмен данными с использованием общих форматов файлов, оптимизированных для высокой производительности в аналитических системах. Фактически, способ преобразования формата данных является довольно распространённым способом интеграции данных [2]. Наиболее часто данные, без использования узкоспециализированных платформ, накапливаются в форматах CSV, JSON, а в случае таких систем, как Hadoop, может использоваться формат, например, avro.

CSV (Comma-Separated Values) – текстовый формат, предназначенный для представления табличных данных [3]. Строка таблицы соответствует строке текста, которая содержит одно или несколько полей, разделенных запятыми. В первой строке допускается указать наименования столбцов данных, но это не является обязательным требованием. Данный формат не предполагает типизацию данных, все значения данных считаются строками. CSV является наиболее часто используемым форматом для хранения структурированных данных, полученных из реляционной базы данных.

JSON (JavaScript Object Notation) – текстовый формат обмена данными, основанный на JavaScript, при этом формат независим от JS и может использоваться в любом языке программирования [4]. Формат предназначен для неструктурированных данных, обрел популярность в связи с распространением REST и NoSQL-подхода в хранении данных.

Формат Avro - это компактный, быстрый и сериализуемый формат данных, разработанный Apache Software Foundation [5]. Он обеспечивает схему данных в JSON формате, что делает его самоописывающим. Avro поддерживает различные типы данных, сжатие данных и является эффективным для передачи данных между различными системами. Он широко используется в Big Data и распределенных системах для обмена данными и хранения данных.

Основной сложностью использования таких форматов для хранения и представления больших данных является проблема гарантированной доступности метаданных для описания наборов данных с целью их использования в дальнейшем. Метаданные значительно упрощают процесс интеграции разрозненных данных, предоставляя пользователям дополнительную информацию о способах сбора или генерации набора данных, что позволяет оценить качество набора данных. Также в метаданных может заключаться информация об используемых методах при проведении исследований, которые привели к появлению данного датасета. К сожалению, представление метаданных не всегда является общепринятой практикой, что приводит к тому, что метаданные не поставляются или предоставляются в неполном объеме, что усложняет дальнейшее использование набора данных [6]. В ходе проведенного анализа, было также выявлено, что рассмотренные форматы также имеют не очень высокие показатели [7], но их преимуществом является полная платформонезависимость. Так, они могут не поддерживать сложные структуры данных или не предоставлять достаточного контроля над метаданными, или являются неоптимальными для обработки больших объемов данных из-за своей структуры или ограничений в производительности при чтении/записи.

Таким образом, разработка и использование форматов файлов больших данных, ориентированных на в оптимизацию хранения больших данных, является важной и актуальной задачей.

Разработка формата

Определим частный случай данных, при котором ни CSV, ни JSON не будут являться предпочтительным форматом передачи данных. Этот случай представлен ниже, на рисунке 1, как пример фрагмента диаграммы классов.

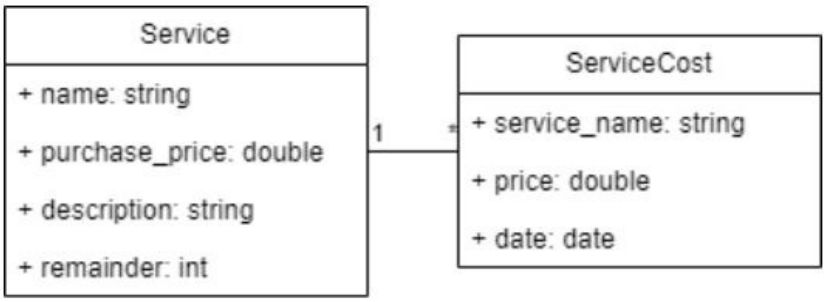

Рисунок 1 – Фрагмент диаграммы классов

На диаграмме представлены два класса: Service (список товаров / услуг) и ServiceCost (история цены товара). Service содержит наименование товара, его закупочную цену, описание характеристик и остаток товаров. ServiceCost содержит наименование товара, его стоимость и дату назначения этой стоимости.

Для Service оптимальным форматом передачи данных является CSV-таблица, а ServiceCost удобнее передавать в JSON, так как если добавить эти данные в тот же файл CSV, все данные товара будет дублироваться для каждой его цены. На данный момент предлагается передавать данные в виде двух заархивированных файлов. В листингах 1 и листинге 2 представлена структура этих файлов.

Листинг 1 – Передача Service в файле service.csv

```
sofa_low,10000.00,description,8
armchair_rocking,8000.00,description,14
bed_double,18000.00,description,6
chair_computer, 12000.00,description,2
bed_children,16000.00,description,5
```

Листинг 2 – Передача ServiceCost в файле service_cost.json

```
{
"sofa_low": [
{
"price": "21000.00",
"date": "2021.01.16"
},
{
"price": "19000.00",
```

"date": "2021.02.21"

}

],

"armchair_rocking": [

{

"price": "15000.00",

"date": "2021.01.16"

}

]

}

Данный формат все еще будет обладать лишней избыточностью, так как присутствует много лишних символов, но также и обладать недостаточностью информации, так как отсутствует дополнительная информация о типе данных в файлах.

Введем критерий выбора эффективного способа передачи данных для классов Service и ServiceCost, воспользовавшись для этого теорией информации и энтропии. Энтропия представляет собой меру информационной неопределенности данных [8]. Она вычисляется путем суммирования вероятностей появления каждого значения данных и их логарифмов по основанию 2 (формула Шеннона) [9]. Пусть H_{CSV} и H_{JSON} будут энтропиями данных в файлах service.csv и service_cost.json. Введём дополнительные весовые коэффициенты для каждой вероятности появления значения данных. Пусть w_i и v_i будут весовыми коэффициентами для вероятностей появления значений данных в CSV и JSON. Эти коэффициенты могут отражать важность каждого значения или контекст, в котором оно используется. Тогда можно использовать энтропию для оценки информационной содержательности и в качестве критерия выбора форматов данных, как показано в (1) и (2).

$H_{CSV} = -\sum_{i=1}^{n} w_i \cdot p_i \log_2 p_i$, (1)

где p_i - вероятности появления данных в записях CSV,

n - количество различных значений в CSV файле.

$H_{JSON} = -\sum_{i=1}^{m} v_i \cdot q_i \log_2 q_i$ (2)

где q_i - вероятности появления данных в записях JSON,

m - количество различных значений в CSV файле.

Сравним энтропии данных, на примере CSV и JSON форматов, для определения оптимального формата передачи данных. Меньшая энтропия может указывать на более компактное представление данных. Иными словами, в случае, если $H_{CSV} < H_{JSON}$, то CSV может быть более оптимальным форматом для передачи данных. В противном случае, если $H_{CSV} > H_{JSON}$, то JSON может быть предпочтительнее. Или для определения предпочтительного из двух форматов можно использовать немного иной подход

учитывающий усложненную энтропию и весовые коэффициенты, как показано в (3).

$$K = \left(\alpha \cdot H_{CSV} + \beta \cdot H_{JSON}\right), \quad (3)$$

где α, β - коэффициенты, отражающие важность энтропии в каждом формате.

Если $K < 0$, то предпочтительным форматом будет CSV. Если же $K > 0$, то предпочтительным форматом будет JSON. В противном случае, при $K = 0$, форматы хранения данных будут одинаково предпочтительными.

Такой подход позволяет учитывать не только компактность данных, но и возможность включения ключевой информации в самих файлах. Для задачи разработки нового формата такой ключевой информацией может являться дополнительная метаинформация, или, например, наличие шума в данных. В случае, если данные содержат много шума или случайных значений, высокая энтропия может указывать на неопределенность и сложность анализа этих данных. Предложенный критерий оценки учитывает контекст использования данных, их ценность, степень сложности и возможных последствий ошибок в анализе данных из-за высокой энтропии.

Исходя из вышеобозначенного можно также сформулировать базовые требования к формату данных для области BigDate. К ним можно отнести:

1. Стандартизация и распространённость. Стандартизация означает установление общих и унифицированных стандартов для представления и обработки данных. Например, к стандартизированным форматам данных можно отнести XML (eXtensible Markup Language), который используется для структурирования и обмена данными в различных областях, включая веб-службы. В контексте форматов хранения больших данных, стандартизация помогает создать общепринятые и унифицированные способы представления информации. Это облегчает обмен данными между различными системами и приложениями, упрощает разработку программного обеспечения и обеспечивает совместимость данных между различными инструментами. Распространенность же формата данных связана с тем, насколько широко данный формат используется в индустрии и среди различных приложений. Чем более распространен формат данных, тем более вероятно, что множество инструментов и систем его поддерживают, что очень важно для области анализа больших данных, где имеется развитая экосистема для работы с этим форматом, включая языки программирования и инструменты анализа данных и системы хранения.

2. Компактность. Формат должен занимать минимальное количество места для хранения данных. Это важно для эффективного использования хранилища и передачи данных по сети, особенно в случае больших объемов информации. Компактные форматы могут сэкономить ресурсы хранения, ускорить передачу данных и уменьшить нагрузку на сеть.

3. Возможность сжатия. Поддержка механизмов сжатия данных может значительно уменьшить объем хранимых данных и ускорить их передачу по сети.

4. Хранение метаданных. Хранение метаданных в формате данных играет важную роль в обеспечении полной и правильной интерпретации информации, сохраненной в этом формате. Метаданные представляют собой данные об данных и содержат информацию о структуре, типах, формате, и других характеристиках данных.

5. Поддержка данных, готовых к анализу. Analysis-Ready Data (ARD) - это концепция в области анализа больших данных, которая подразумевает предварительную обработку и

подготовку данных перед анализом. Она призвана упростить и ускорить процесс анализа данных, обеспечивая готовые к использованию наборы данных. В настоящее время данная концепция находит широкое развитие в области анализа спутниковых изображений [10].

Предложенный формат хранения данных будет базироваться на синтезе двух распространённых форматах хранения данных, таких как CSV и JSON. Такое решение имеет ряд преимуществ, в силу того, что CSV позволяет хранить данные в табличной структуре с явно определенными столбцами, в то время как JSON обладает более гибкой структурой. Объединение их может обеспечить удобное хранение табличных данных с сохранением иерархии и вложенных объектов. Также использование JSON для хранения данных предоставляет типизацию данных, что облегчает восприятие и работу с данными, а дополнительный метафайл может содержать информацию о типах данных для CSV-части, что обеспечивает полную типизацию.

Модификация формата хранения данных с метаданными

В CSV-файле есть возможность добавить названия столбцов и, так как CSV не может хранить информацию о типе данных, можно добавить отдельным meta-файлом информацию о типах. Модифицированный CSV-файл и метафайл с информацией о типах представлены в листингах 3 и 4.

Листинг 3 – Модификация service.csv.

```
name,purchase_price,description,remainder
sofa_low,10000.00,description,8
armchair_rocking,8000.00,description,14
bed_double,18000.00,description,6
chair_computer, 12000.00,description,2
bed_children,16000.00,description,5
```

Листинг 4 – Мета-файл service.meta.

```
{
"name": "string",
"purchase_price": "double",
"description": "string",
"remainder": "int"
}
```

Формат JSON при десериализации в различных языках программирования предусматривает типизацию данных, но для унификации формата типы данных будут храниться в отдельном метафайле (листинг 5).

Листинг 5 – Метафайл service_cost.meta.

```
{
```

{

"price": "double",

"date": "date"

}

Таким образом, разработанный формат представляет собой архив, состоящий из четырех вышеописанных файлов.

Формат может быть расширен дополнительными данными в формате CSV или JSON с соответствующими метафайлами при необходимости.

В качестве названия формата предлагается «PandasKeepFormat», расширение формата соответственно имеет вид PDKF.

Математическое описание формата и применимых функций над структурой данных

Далее рассмотрим математическое описание для предложенного в данной работе формата данных, который объединяет CSV и JSON, определив, таким образом множество допустимых операций над данными.

Пусть:

– C - множество CSV-файлов, представляющих собой набор строк и столбцов.

– J - множество JSON-файлов, содержащих структурированные данные в формате JSON.

– M - множество метаданных, описывающих информацию о структуре данных в CSV или JSON, такую как типы данных, названия столбцов, и другие характеристики.

Тогда, новый формат данных D_{pdkf} можно определить следующим образом, как показано в формуле 4.

$D_{pdkf} = (C \cup J) \times M$ (4) .

В вышеобозначенном равенстве $C \cup J$ является множеством, объединяющее данные из CSV и JSON. Элементы этого множества могут быть как табличными (CSV), так и иерархическими (JSON) данными. А $(C \cup J) \times M$ является множеством пар, где первый элемент представляет собой данные из , а второй элемент - соответствующие метаданные M. Таким образом, каждая запись данных из $C \cup J$ сопровождается своими метаданными.

Данный подход позволит объединить данные и метаданные, однако, он не учитывает наличие более сложных зависимостей в данных. Например, одни данные могут зависеть от других, но для некоторых задач может быть важным ещё и понимание контекста и последовательности использования этих данных. Или, в некоторых аналитических сценариях может быть необходимость в проведении сложных операций, которые включают в себя связанные элементы данных. Например, вычисления, которые требуют комбинации данных из нескольких источников, могут использовать информацию о связях. Особенностью предложенной выше формата хранения данных является то, что он может быть относительно легко расширен, за счёт возможности добавления дополнительных данных, о чем было указано выше. Например, функции, определенные при реализации данного формата, добавляются в словарь функций и могут быть

применены к данным, хранящимся в структуре.

Для расширения данной модели в работе предлагается внедрить элементы из теории графов и теории категорий. Для этого дополнительно введём в модель ориентированный граф G, где вершины V представляют элементы из $C \cup J$ и ребра E представляют связи или зависимости между этими элементами, иными словами G=(V,E).

Для работы с данными, определенными предложенным форматом, необходимо дополнительно определить набор применимых функций над данными. Здесь, в данной работе, под применимыми функциями, в контексте структур данных понимаются функции или операции, которые могут быть корректно применены или выполнены на данных предложенной структуры. То есть, это функции, которые соответствуют типу или структуре данных и могут быть использованы для обработки или изменения этих данных. Соответственно, пусть имеется множество функций F, которые могут быть применены к данным из $C \cup J$ с учетом их структуры и метаданных. Предположим, что каждая функция имеет определенные характеристики, такие как агрегация, фильтрация, преобразование, комбинация и т.д. Определим некоторые из этих функций.

Пусть D - структура данных, C - множество CSV-файлов, J - множество JSON-файлов, M - множество метаданных, а f - функция из множества применимых функций $F = \{f_1, f_2, \ldots\}$, где f_i - функции, которые могут выполнять различные операции, такие как фильтрация, сортировка, агрегация и т.д.

Обозначим элемент данных в D как d_{ij}, где i и j - координаты элемента в координатной сетке. Иными словами $D = \{d_{ij}\}$, $d_{ij} \in \mathbb{R}^{n \times m}$, где n - количество строк, m - количество столбцов.

Тогда, с учетом вышеизложенного, ***агрегирующая функция*** может быть выражена $f_{\text{агрегация}}(D, M) \rightarrow \text{Result\_Aggregation}$.

Преобразующая функция $f_{\text{преобразование}}$ преобразует данные из структуры D в новую структуру $f_{\text{преобразование}}(D, M) \rightarrow D'$.

Функция извлечения данных столбца извлекает определенный столбец из CSV или JSON данных (при соответствующей организации данных). Пусть D_i представляет собой элемент из *D*, а M_i - соответствующий элемент из *M*. Тогда функция $f_{\text{извлечение}}(D_i, M_i, columnName) \rightarrow ColumnDate$. Например, если D_i - CSV файл, M_i - метаданные, и $\text{columnName} = \text{"Age"}$, то $f_{\text{извлечение}}$ вернет столбец возрастов.

Функция фильтрации по значению оставляет только те строки данных, где определенный столбец удовлетворяет заданному условию. Аналогично функции извлечения данных, функция $f_{\text{фильтрация}}$ может быть выражена через $f_{\text{фильтрация}}(D_i, M_i, columnName, condition) \rightarrow FiltredDate$, где condition означает условие, которое определяет, какие строки данных следует оставить после применения функции фильтрации. Это условие проверяется для определенного столбца в данных, и только те строки, которые удовлетворяют этому условию, остаются в результирующем наборе данных. Например, если D_i - JSON файл, M_i - метаданные, $columnName = \text{"Price"}$ и $\text{condition} = \text{" > \$100"}$, то $f_{\text{фильтрация}}$ оставит только записи с ценой выше $100.

С учётом вышесказанного, окончательно, новый формат данных D можно определить как показано в формуле 5.

$$D = (C \cup I) \times (G, F) \quad (5)$$

Реализация обработчика формата

В данном разделе представлена реализация обработчика формата PDKF на языке программирования Python, который преобразует данные из файла в список датафреймов Pandas. Фрагмент исходного кода обработчика представлен в листинге 6.

Листинг 6 – Реализация обработчика формата.

```
import pandas

import json

from zipfile import ZipFile

from io import BytesIO

from io import StringIO

def read_file(file_name, file_content):

if file_name[-3:] == 'csv':

data = pandas.read_csv(BytesIO(file_content))

return data

elif file_name[-4:] == 'json':

json_list = json.loads(file_content.decode())

data = pandas.DataFrame()

for item in json_list:

if len(data) == 0:

data = pandas.json_normalize(list(item.values())[0])

else:

data = pandas.concat([data, pandas.json_normalize(list(item.values())
[0])], ignore_index = True)

return data

else:

raise Exception(f'Wrong file format: {file_name}')

def read_pdbc(file):

if file[-4:] != 'pdkf':

raise Exception(f'Wrong file format')

dataframe_dict = {}

file_zip = ZipFile(file, 'r', allowZip64=True)
```

```
try:
for file_name in file_zip.namelist():
if file_name[-4:] != 'meta':
file_content = file_zip.read(file_name)
dataframe_dict[file_name[:file_name.find('.')]] = read_file(file_name,
file_content)
for file_name in file_zip.namelist():
if file_name[-4:] == 'meta':
file_content = file_zip.read(file_name)
data = json.loads(file_content.decode())
for column, type in data.items():
dataframe_dict[file_name[:-5]][column] = dataframe_dict[file_name[:-5]]
[column].astype(type)
finally:
file_zip.close()
return dataframe_dict
```

Для реализации обработчика необходимы следующие зависимости:

· Pandas – популярная библиотека, необходимая для аналитики больших данных, именно в ней содержится тип данных DataFrame.

· Zipfile – встроенная библиотека, необходимая для работы с архивами ZIP.

Представленный выше код позволяет читать файлы в форматах CSV и JSON, а также из ZIP-архива с определенной структурой, и создавать соответствующие pandas DataFrame для дальнейшей обработки данных. Кроме того, он обеспечивает обработку метаданных и преобразование типов данных в DataFrame в соответствии с указанными типами в метаданных. Пример использования вышеописанного модуля и некоторые манипуляции с полученными данными представлены в листинге 7.

Листинг 7 – Пример использования модуля.

```
import pdkf
dataframe_dict = pdkf.read_pdkf('file.pdkf')
for key, dataframe in dataframe_dict.items():
print(key)
print(dataframe.info())
```

В данном случае, выводится информация о каждом полученном из файла датафрейме. В

дальнейшем можно производить с данными любые манипуляции, доступные в модуле Pandas, такие как аналитика, прогнозирование на основе этих данных и т.д.

Разработанный модуль выложен в каталог пакетов Python PYPI [https://pypi.org/project/pdkf/] под лицензией MIT, что позволяет установить его консольной командой «pip install pdkf». На рисунке 2 показан скриншот размещения разработанного решения в официальном репозитории программного обеспечения для Python - PyPI, аналоге CRAN для языка R.

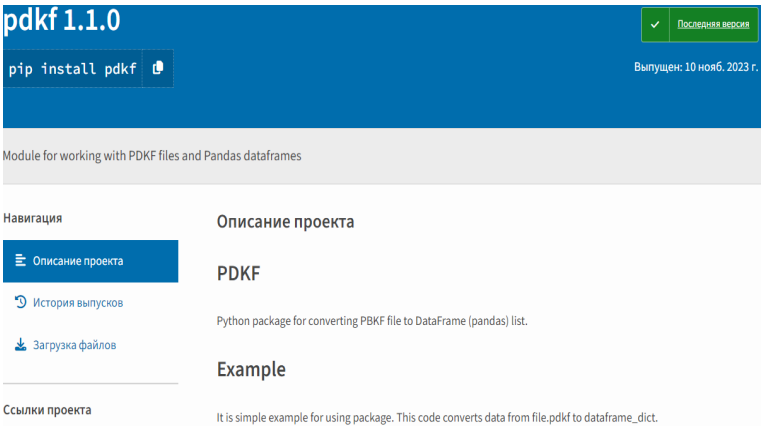

Рисунок 2 – Размещение модуля-обработчика формата pdkf в репозитории PYPI

Заключение

Предложенное решение предлагает использовать комбинацию форматов CSV и JSON для оптимального хранения и передачи данных. Для удобства восприятия данных и обеспечения полной типизации, предлагается дополнительно использовать метафайлы, которые содержат информацию о типах данных для соответствующих CSV и JSON файлов. Модификация формата CSV включает добавление названий столбцов и хранение информации о типах данных в отдельном метафайле. JSON формат уже обеспечивает типизацию данных при десериализации, но также предлагается хранить информацию о типах в метафайле для унификации формата.

Таким образом, предложенное решение представляет собой архив, состоящий из CSV и JSON файлов, а также соответствующих метафайлов, которые обеспечивают структурированное хранение данных с сохранением типов и метаданных. Формат может быть дополнен дополнительными данными в CSV или JSON формате с соответствующими метафайлами при необходимости.

Библиография

1. Malcolm R., Morrison C., Grandison T., Thorpe S., Christie K., Wallace A., Green D., Jarrett J., Campbell A. Increasing the accessibility to big data systems via a common services api // IEEE International Conference on Big Data. 2014. Pp. 883–892.
2. Wu T. System of teaching quality analyzing and evaluating based on data warehouse // Computer Engineering and Design. 2009. No. 6(2). Pp. 1545-1547.
3. Vitagliano G. et al. Pollock: A Data Loading Benchmark //Proceedings of the VLDB Endowment. 2023. No. 8(16). Pp. 1870-1882.
4. Xiaojuan L., Yu Z. A data integration tool for the integrated modeling and analysis for east //Fusion Engineering and Design. 2023. No. 195. Pp. 113933. https://doi.org/10.1016/j.fusengdes.2023.113933
5. Lemzin A. Streaming Data Processing //Asian Journal of Research in Computer

Science. 2023. No. 1(15). Pp. 11-21.

6. Hughes LD, Tsueng G, DiGiovanna J, Horvath TD, Rasmussen LV, Savidge TC, Stoeger T, Turkarslan S, Wu Q, Wu C, Su AI, Pache L. Addressing barriers in FAIR data practices for biomedical data //Scientific Data. 2023. No. 1(10). P. 98. DOI:https://doi.org/10.1038/s41597-023-01969-8
7. Gohil A., Shroff A., Garg A., Kumar S. A Compendious Research on Big Data File Formats. *2022 6th International Conference on Intelligent Computing and Control Systems (ICICCS).* IEEE Press, Madurai, India. 2022. Pp. 905-913. DOI: https://doi.org/10.1109/ICICCS53718.2022.9788141
8. Елсуков П. Ю. Информационная асимметрия и информационная неопределенность //ИТНОУ: Информационные технологии в науке, образовании и управлении. 2017. No. 4 (4). С. 69-76.
9. Bromiley P. A., Thacker N. A., Bouhova-Thacker E. Shannon entropy, Renyi entropy, and information //Statistics and Inf. Series (2004-004). 2004. No. 9. Pp. 2-8.
10. Dwyer, J. L. Roy, D. P., Sauer B., Jenkerson C. B., Zhang H. K., Lymburner L. Analysis ready data: enabling analysis of the Landsat archive //Remote Sensing. 2018. №. 9(10). 1363.

Результаты процедуры рецензирования статьи

В связи с политикой двойного слепого рецензирования личность рецензента не раскрывается.
Со списком рецензентов издательства можно ознакомиться здесь.

Статья посвящена актуальной задаче интеграции и оптимизации хранения данных в разнообразных форматах, особенно акцентируя внимание на форматах CSV и JSON. Исследование затрагивает вопросы эффективного представления больших объемов данных, с целью упрощения процесса аналитики и обработки. Авторы предлагают новаторский подход к объединению форматов хранения данных с использованием метаданных и графовых структур для оптимизации и улучшения доступности данных. Исследование включает теоретический анализ, разработку нового формата данных, математическое описание предложенных методов, а также практическую реализацию на языке программирования Python. В эпоху "больших данных", актуальность исследования несомненна. Проблематика интеграции и эффективного хранения данных остается важной задачей, особенно в контексте растущего объема неструктурированных данных и необходимости их анализа. Научная новизна работы заключается в предложении оригинального формата хранения данных "PandasKeepFormat", который синтезирует преимущества CSV и JSON форматов с дополнением метаданных и графов зависимостей. Это позволяет значительно упростить работу с большими объемами данных и обеспечивает новые возможности для аналитики. Статья имеет логичное и последовательное изложение материала, начиная с обоснования актуальности проблемы и заканчивая практической реализацией предложенного решения. Стиль изложения ясный и доступный, материал структурирован, а содержание информативно и насыщено. Выводы статьи подчеркивают значимость разработанного формата для упрощения процессов аналитики и интеграции данных. Работа будет интересна широкому кругу читателей, включая специалистов в области информационных технологий, аналитики данных, а также разработчиков программного обеспечения. Статья представляет значительный вклад в развитие методов работы с большими данными и заслуживает публикации. Было бы полезно рассмотреть возможность дальнейшего развития исследования, например, путем проведения сравнительного

анализа производительности нового формата с существующими решениями на реальных наборах данных. Рекомендую принять к публикации, учитывая актуальность темы, научную новизну и качество представленного материала.

Для дальнейшего развития представленной в статье работы по новому формату хранения данных можно предложить несколько направлений, которые включают проведение экспериментального исследования для сравнения производительности предложенного формата "PandasKeepFormat" с другими существующими форматами данных, такими как Parquet, ORC или Avro. Важно рассмотреть различные аспекты, включая скорость чтения и записи данных, эффективность сжатия данных и потребление ресурсов памяти. Также предлагаю исследовать возможность дополнения формата поддержкой более широкого спектра типов данных, включая сложные структуры данных и пользовательские типы, что может повысить его применимость для различных специализированных областей анализа данных. Одним из важных направлений развития является разработка плагинов или модулей для интеграции с популярными библиотеками и фреймворками для работы с данными, чтобы облегчить использование нового формата в существующих экосистемах. Оптимизация формата и алгоритмов сериализации и десериализации для эффективного использования в распределенных вычислительных системах также является ключевым аспектом, включая возможности эффективного разбиения и параллельной обработки данных. Важно уделить внимание безопасности данных, разрабатывая механизмы шифрования и обеспечения безопасности данных, встроенные в формат, для повышения защиты конфиденциальности информации при хранении и передаче. Улучшение управления метаданными, включая продвинутые подходы к управлению и версионированию метаданных, обеспечит лучшую совместимость и возможности для эволюции формата данных без потери информации о предыдущих версиях. Создание инструментов для визуализации структуры данных и метаданных, а также для мониторинга производительности работы с данными в новом формате, позволит пользователям более эффективно управлять и анализировать свои данные. Наконец, организация платформы для взаимодействия с пользователями и разработчиками, заинтересованными в использовании и развитии нового формата, а также сбор и анализ обратной связи помогут выявить потребности пользователей и определить направления для дальнейших улучшений. Эти шаги способствуют улучшению эффективности предложенного формата хранения данных, расширению его применяемости и популярности в научном и профессиональном сообществе.