SmellSweep - A tool to Detect and Refactor DataSmells

M.Siva Sai
cs21b032@iittp.ac.in
IIT TIRUPATI
Tirupati, Andhra Pradesh, India

G.Manoj Kumar
IIT TIRUPATI
Tirupati, India
cs21b020@iittp.ac.in

S.Tejeswara Reddy
IIT TIRUPATI
Tirupati, India
cs21b058@iittp.ac.in

N.Vikrama Simha Reddy
IIT TIRUPATI
Tirupati, India
cs21b036@iittp.ac.in

D.Shanmukh Sai
IIT TIRUPATI
Tirupati, India
cs21b018@iittp.ac.in

ABSTRACT

Data quality is paramount in the field of data science and analytics, as it directly influences the accuracy and reliability of insights and decision-making processes. This paper presents 'SmellSweep,' a comprehensive tool designed to address data quality issues by detecting and refactoring data smells within datasets. Leveraging methodologies and techniques such as data parsing with Pandas, backend development with Flask, and frontend development with React.js, SmellSweep offers a multifaceted approach to data quality management. Key features include 26 implemented data smell detection algorithms, metrics calculation, interactive data visualization with Recharts, and seamless integration into a user-friendly interface. The tool not only identifies data quality issues but also provides a solution through its refactoring functionality, enhancing the integrity and usability of datasets for analysis. This research contributes to the advancement of data quality management practices by providing a robust and accessible solution for data scientists and analysts.

CCS CONCEPTS

• **Information systems** → **Data management systems**; **Data quality**; • **Computing methodologies** → *Data cleaning*; *Data visualization*; • **Applied computing** → Data science; Analytics.

KEYWORDS

datasmells, datasets, Data quality management, Data refactoring, Data smell detection

ACM Reference Format:
M.Siva Sai, G.Manoj Kumar, S.Tejeswara Reddy, N.Vikrama Simha Reddy, and D.Shanmukh Sai. 2018. SmellSweep - A tool to Detect and Refactor DataSmells. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 6 pages. https://doi.org/XXXXXXX.XXXXXXX

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Mentor - Jahnavi K

In the realm of data science and analytics, the reliability and accuracy of insights derived from datasets are contingent upon the quality and integrity of the data itself. Data quality issues can manifest in various forms, leading to erroneous analyses, flawed models, and ultimately, unreliable decision-making. Recognizing the critical importance of data quality management, we introduce SmellSweep, a comprehensive tool meticulously crafted to detect and refactor data smells within datasets.

Data smells, akin to their software engineering counterpart code smells, serve as indicators of potential anomalies or issues within the dataset. These anomalies encompass a wide spectrum, including inconsistent formatting, missing values, outliers, redundant information, and mismatches in data types. Addressing these data smells is paramount to ensure the integrity and usability of datasets for analytical purposes.

SmellSweep encapsulates a suite of 26 meticulously implemented data smell detection algorithms, each meticulously designed to identify specific manifestations of data quality issues. Leveraging cutting-edge methodologies and techniques, SmellSweep offers a multifaceted approach to data quality management, encompassing data parsing, smell detection, metric calculation, interactive visualization, and seamless integration into a user-friendly interface.

By facilitating the detection and subsequent refactoring of data smells, SmellSweep empowers data scientists and analysts to enhance the quality, reliability, and trustworthiness of their datasets. This tool represents a significant advancement in the field of data quality management, providing practitioners with a robust and accessible solution to address the intricate challenges posed by data quality issues.

In this paper, we present an overview of the design, development, and functionalities of SmellSweep, elucidating its pivotal role in ensuring the integrity and accuracy of data-driven decision-making processes. Through a comprehensive exploration of its features and capabilities, we aim to underscore the transformative impact of SmellSweep on the landscape of data science and analytics.

2 RELATED WORK

https://arxiv.org/pdf/2203.10384.pdf.
https://dl.acm.org/doi/pdf/10.1145/3522664.3528621.

3 DESIGN AND DEVELOPMENT

The design and development of SmellSweep involved a meticulous process aimed at creating a robust and user-friendly tool for data quality management. This section provides insights into the methodologies, techniques, and technologies employed during the development of SmellSweep.

3.1 Methodologies

SmellSweep was developed using an iterative and collaborative approach, drawing upon established methodologies in software engineering and data science. The development process consisted of the following key stages:

- **Requirements Gathering:** In our requirements gathering process, we extensively reviewed research papers relevant to our project domain. We held meetings with mentors to discuss project goals, scope, and potential challenges, incorporating their advice into our requirements.Brainstorming sessions within the team facilitated idea generation and requirement identification. Feedback from faculty,mentors, coupled with prototyping efforts, further refined our project requirements.
- **Design Phase:** Based on the gathered requirements, the project team formulated a comprehensive design blueprint for SmellSweep. This phase included the creation of wireframes, user interface mockups, and architecture diagrams to guide the development process.
- **Implementation:** The implementation phase involved the actual coding and development of SmellSweep. Leveraging a combination of programming languages and frameworks, including Python (for backend development), React.js (for frontend development), and Flask (for web server development), the project team translated the design specifications into functional software components.
- **Testing and Validation:** We conducted internal testing of our website with our team members to evaluate its functionality, usability, and overall user experience. This testing, encompassing User Acceptance Testing (UAT) and Beta Testing, provided valuable feedback from a representative sample of our target audience. Through our team's involvement, we gained insights into real user interactions, identifying any issues or areas for improvement. This collaborative testing approach ensured that our website met user expectations and delivered a positive experience before its official release.

3.2 Technologies

SmellSweep leveraged a range of technologies and tools to facilitate its development and deployment. Key technologies utilized in the implementation of SmellSweep include:

- **Pandas:** Pandas, a powerful data manipulation library in Python, was utilized for parsing and processing uploaded CSV files. Its extensive functionality enabled efficient handling of large datasets and facilitated the implementation of data smell detection algorithms.
- **Flask:** Flask, a lightweight web framework for Python, served as the backbone of SmellSweep's backend infrastructure. Flask provided the necessary tools and utilities for handling HTTP requests, managing sessions, and communicating with the frontend interface.
- **React.js:** React.js, a popular JavaScript library for building user interfaces, was employed for developing the frontend components of SmellSweep. Its component-based architecture, virtual DOM rendering, and state management capabilities facilitated the creation of interactive and responsive user interfaces.
- **Recharts:** Recharts, a charting library for React.js, was integrated into SmellSweep to enable interactive data visualization. With a variety of chart types and customization options, Recharts allowed users to visualize the distribution and severity of data smells across different columns in the dataset.

3.3 User Interface Integration

A key focus during the development of SmellSweep was ensuring seamless integration with the user interface. The frontend components were designed with usability and accessibility in mind, offering intuitive navigation, responsive design, and interactive features. By integrating Recharts graphs, tables, and other visualization elements seamlessly into the user interface, SmellSweep provides users with a cohesive and streamlined experience.

3.4 DataSmells Implemented

The following is a list of the datasmells implemented in SmellSweep along with their descriptions:

(1) **Missing Value:**
- Real-life example: In a customer database, some records might have missing values for the "phone number" field.
- Detection: Use descriptive statistics, data visualization, or data profiling tools to identify columns with missing values.
- Rectification: Depending on the situation, you can either remove records with missing values, impute missing values using statistical methods, or collect the missing data if possible.

(2) **Duplicated Value:**
- Real-life example: In a product inventory database, there might be duplicate entries for the same product due to data entry errors or system glitches.
- Detection: Use SQL queries or data profiling tools to identify duplicate records based on unique identifiers.
- Rectification: Remove duplicate records or merge them if necessary, ensuring data consistency and accuracy.

(3) **Extreme Value:**
- Real-life example: In a dataset of employee salaries, there might be extremely high or low values due to outliers or data entry errors.
- Detection: Use data visualization techniques such as box plots or scatter plots to identify extreme values.
- Rectification: Evaluate the extreme values to determine if they are valid data points or outliers. If they are outliers, consider removing them or transforming the data if appropriate.

(4) **Misspelling:**

- Real-life example: In a customer database, names might be misspelled due to data entry errors or inconsistencies.
- Detection: Use string matching algorithms or manual inspection to identify misspelled or inconsistent values.
- Rectification: Standardize the data by correcting misspellings using fuzzy matching algorithms or manual editing.

(5) **Suspect Class Value:**
- Real-life example: In a dataset of medical records, there might be invalid or suspicious values in the "diagnosis" field.
- Detection: Use domain knowledge or data validation rules to identify suspect class values.
- Rectification: Review and verify suspect values with domain experts, and either correct or remove them as necessary.

(6) **Special Characters:**
- Description: Data values containing special characters beyond alphanumeric characters.

(7) **Spacing Smell:**
- Description: Unusual patterns of spaces (leading, trailing, multiple, or missing) within data values.

(8) **Missing Value Inconsistency:**
- Description: Inconsistent ways of handling missing values within the same dataset.

(9) **Separating Smell:**
- Description: Data values contain thousands separators (such as commas or dots), creating ambiguity in how the data is interpreted.

(10) **Contracting (CSV Data):**
- Description: Contractions in text data (e.g., "don't", "can't") can introduce inconsistencies and affect analysis if not handled appropriately.

(11) **Suspect Sign (CSV Data):**
- Description: Presence of data points with unexpected signs (+, -) associated with their values in a CSV file, potentially indicating errors or anomalies.

(12) **Suspect Distribution (CSV Data):**
- Description: Data distribution deviates significantly from the expected pattern.

(13) **Date as DateTime:**
- Description: This data smell occurs when date information is stored in a format that combines both date and time components, even though only the date is relevant for analysis or further processing.

(14) **Integer As Floating Point Number:**
- Description: This data smell occurs when integer values (whole numbers) are incorrectly stored as floating-point numbers, which can represent both whole numbers and decimals.

(15) **Integer As String:**
- Description: This data smell occurs when integer values (whole numbers) are stored as text strings instead of dedicated integer data types.

(16) **Empty String:**
- Description: When addressing empty strings (missing values, null values) in data analysis, the choice between rule-based and machine learning (ML)-based approaches depends on various factors.

(17) **Unnecessary Character:**
- Description: This refers to the presence of extraneous characters within the data that don't contribute meaningful information.

(18) **Outlier Skew:**
- Description: This data smell refers to the presence of data points that fall significantly outside the expected range of the majority of data (i.e., outliers).

(19) **Dummy Value:**
- Description: Placeholder values used to represent missing data without proper indication.

(20) **Unit Inconsistency:**
- Description: Different units used to represent the same measurement.

(21) **Ambiguous Value:**
- Real-life example: In a dataset of customer feedback, some entries might contain ambiguous or vague comments.
- Detection: Review the dataset manually or with the help of domain experts to identify ambiguous values.
- Rectification: Clarify ambiguous values by adding more context or detail, or consider removing them if they cannot be interpreted accurately.

(22) **Casing:**
- Real-life example: In a dataset of customer names, some names might be inconsistently capitalized.
- Detection: Compare values within the same column to identify inconsistent casing.
- Rectification: Standardize casing by converting all values to lowercase or uppercase, ensuring consistency across the dataset.

(23) **Long Data Value:**
- Real-life example: In a product description field, some entries might be excessively long due to data entry errors or formatting issues.
- Detection: Use descriptive statistics or data profiling tools to identify unusually long values.
- Rectification: Trim or truncate long values to an appropriate length, ensuring readability and consistency.

(24) **Floating Point Number As String**
- Description: This data smell occurs when numerical data representing floating-point numbers (numbers with decimal points) are stored as text strings instead of dedicated numeric data types. This can lead to several issues:

In conclusion, the design and development of SmellSweep encompassed the implementation of diverse data smell detection algorithms, visualized effectively through Recharts bar graphs. Additionally, SmellSweep offers a comprehensive solution by refactoring the provided CSV data, ensuring enhanced data quality and usability for subsequent analysis.

4 USER SCENARIO

This section contains the flow of how a user can interact with our tool via demo images.

Figure 1: Detecting DataSmell - Step 1 (interface to upload the .csv file to proceed for detecting or customized refactoring)

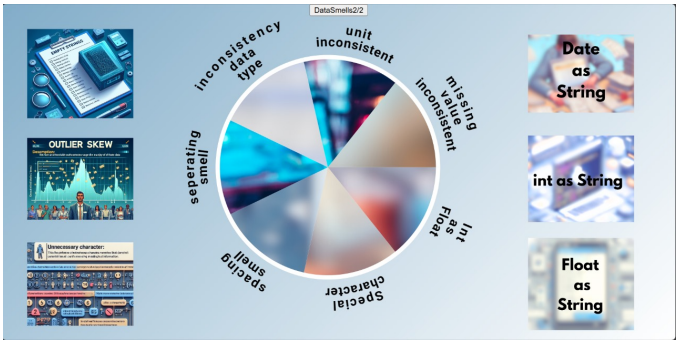

Figure 2: Detecting DataSmell - Step 2 (interface in which we can select any datasmell to check its metrics via graphs)

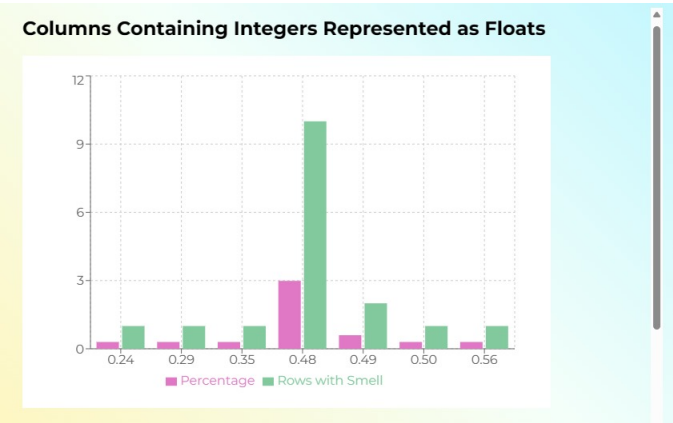

Figure 3: Detecting DataSmell - Step 3 (interface which is showing chart of a particular datasmell via charts)

Figure 4: Refactoring - Step 1 (interface to upload the .csv file to proceed for detecting or customized refactoring)

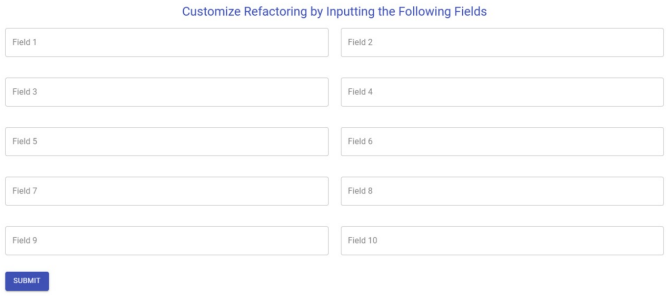

Figure 5: Refactoring - Step 2 (interface which is for taking inputs from user for customized refactoring)

Figure 6: Refactoring - Step 3 (interface which is showing the status of the refactoring)

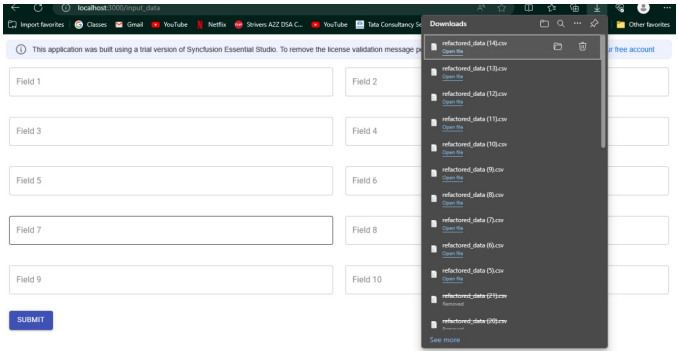

Figure 7: Refactoring - Step 4, (The Refactored .csv file is downloaded)

5 DISCUSSION & LIMITATIONS

- **Efficiency of Detection**:
  - The detection process might not be 100% efficient, especially for large datasets.
  - In cases where a data smell is not present, the tool will still check for it, potentially leading to false positives or unnecessary processing.
- **Rule-based Approaches**:
  - Limitations arise when dealing with certain data smells, such as special characters.
  - While it's possible to remove special characters, some columns may legitimately contain them, such as email addresses.
  - This necessitates careful consideration and customization of rules to avoid unintended alterations.
- **Incomplete Smell Coverage**:
  - Only used rule based approach and didn't use ml based approach for detecting data smells.
  - Despite detecting and refactoring 23 out of 26 identified data smells, it's important to recognize that the list may not be exhaustive.
  - There could be other undetected smells that require different approaches for identification and rectification.
- **Collaborative Pipeline**:
  - The approach involved sharing detected smells among team members for detection and refactoring.
  - This collaborative pipeline facilitated efficient handling of various data smells by distributing tasks and responsibilities.
- **Adjustments and Refinements**:
  - The process involved continuous adjustments and refinements to address specific data smells effectively.
  - This iterative approach allowed for adaptation to diverse datasets and nuanced smell characteristics.

6 EXPERIMENTS AND RESULTS

We tested our detection method step by step. First, we checked each data smell on its own in separate CSV files to make sure our algorithm could find them accurately. Then, we grouped these smells into batches and tested them again to ensure our method worked consistently across different groups. Once we confirmed each batch, we put all 26 smells together in one CSV file for a final test. Even with all the smells mixed, our method still found them accurately every time.

7 FUTURE WORK

Our current approach utilizes rule-based methods, employing regular expressions to identify data smells across all columns within the dataset. However, for future iterations of the project, the team may consider transitioning to a machine learning-based approach for more targeted detection of specific columns exhibiting data smells. This ML approach would enable the team to enhance project efficiency by accurately identifying problematic columns without flagging unnecessary ones. Furthermore, it offers the flexibility to incorporate additional data smells as they are discovered, allowing for continuous improvement and refinement of the project

8 CONCLUSION

We've done a lot with our Smell Sweep Tool to clean up messy data. Here's a closer look at what we've accomplished and how our tool works:

- Finding and Fixing Problems:
  - We found 26 different types of problems in our data, like missing info or mistakes.
  - Our tool fixed 23 of these problems, making our data cleaner and more reliable.
  - We made sure our fixes fit the data perfectly by using information from the CSV file.
- Easy to Use:
  - Our tool is simple to use, even if you're not a data expert.
  - If you want to learn more about the problems we found, you can check out our website. Just click the "analysis" button.
- Personalized inputs:
  - For some dataset user know what data should be not removed like suspect sign which we cant remove - for every column like - is allowed in some columns like temperature so user can give that column name to not to remove - .
  - This makes the whole process quicker and ensures we don't mess up any of the good data.

In short, our Smell Sweep Tool is a big help in keeping data tidy. With it, we make sure data is accurate and reliable, so organizations can do their best work.

REFERENCES

[1] *arXiv.* https://arxiv.org/pdf/2203.10384.pdf
[2] *Conference research paper* . https://fpalomba.github.io/pdf/Conferencs/C82.pdf
[3] *Report.* https://dl.acm.org/doi/pdf/10.1145/3522664.3528621
[4] *Rule based.* https://github.com/mkerschbaumer/rb-data-smell-detection
[5] *ML based.* https://github.com/georg-wenzel/ml-data-smell-detection