Califano Adriano Emanuele

DSD+

Data Smell Detection +

A data analysis tool

D'Antuono Francesco Paolo

Fabiano Daniele

TABLE OF CONTENTS

01

INTRODUCTION

Some definitions to get started

What are Data Smells?

«Signals or clues about the presence of latent problems or anomalies in data»

What are Data Quality Dimensions (DQDs)?

«Characteristic or part of information for classifying information and data requirements»

Some papers about the topics

Data Smells: Categories, Causes and Consequences, and Detection of Suspicious Data in AI-based Systems

Harald Foidl
University of Innsbruck
Austria
harald.foidl@uibk.ac.at

Michael Felderer
University of Innsbruck
Austria
Blekinge Institute of Technology
Sweden
michael.felderer@uibk.ac.at

Rudolf Ramler
Software Competence Center
Hagenberg GmbH
Austria
rudolf.ramler@scch.at

ABSTRACT

High data quality is fundamental for today's AI-based systems. However, although data quality has been an object of research for decades, there is a clear lack of research on *potential* data quality issues (e.g., ambiguous, extraneous values). These kinds of issues are latent in nature and thus often not obvious. Nevertheless, they can be associated with an increased risk of future problems in AI-based systems (e.g., technical debt, data-induced faults). As a counterpart to code smells in software engineering, we refer to such issues as *Data Smells*. This article conceptualizes data smells and elaborates on their causes, consequences, detection, and use in the context of AI-based systems. In addition, a catalogue of 36 data smells divided into three categories (i.e., Believability Smells, Understandability Smells, Consistency Smells) is presented. Moreover, the article outlines tool support for detecting data smells and presents the result of an initial smell detection on more than 240 real-world datasets.

ACM Reference Format:
Harald Foidl, Michael Felderer, and Rudolf Ramler. 2022. Data Smells: Categories, Causes and Consequences, and Detection of Suspicious Data in AI-based Systems. In *1st Conference on AI Engineering - Software Engineering for AI (CAIN'22), May 16–24, 2022, Pittsburgh, PA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3522664.3528590

1 INTRODUCTION

Applications based on artificial intelligence (AI) (e.g., automated driving, predictive maintenance) have grown in popularity over the past decade. However, the resulting AI-based systems pose several challenges [7, 38]. One of these challenges is their strong data dependency [52]. This dependency is caused by data-hungry machine learning (ML) algorithms, typically used in AI-based systems to make intelligent decisions automatically. As a result, poor quality data can lead to abnormal behaviour and false decisions in such systems, resulting in huge monetary losses or, in the worst case, even harming people [8].

Recent research (e.g., [28, 49]), however, suggests that data quality problems are pervasive in AI-based systems. Data quality issues even became one of the main reasons why they suffer badly from technical debt [6].

To improve this situation and thus meet the demand for high data quality in the context of AI-based systems, research in the area of data validation has recently gained significant interest (e.g., [5, 9, 11, 37, 47]). To reliably detect data issues, data validation methods generally require some context-specific constraints, which are usually defined by schemes or rules [1]. However, this declarative and context-dependent nature of data validation combined with the constantly growing amount of data makes data validation a tedious and labor-intensive activity [17, 52].

To address this issue, we previously proposed using *potential* data issues as indicators of latent data quality problems to guide the data validation process in a risk-driven way [17]. These potential data problems are usually indicated by context-independent, suspicious data values, patterns, and representations, highlighting data that should be prioritised during the validation. We referred to these potential data issues as *data smells* by analogy with code smells.

In the field of software engineering, code smells have become established as indicators of bad design and programming practices that are not faults per se but increase the likelihood of introducing faults in the future [19, 60]. In that sense, code smells are *potential* faults or issues [50].

We assert that data smells share several characteristics with code smells. For example, they typically arise due to violated best practices in data handling (e.g., wrong sequence of operations) or data management (e.g., missing data catalogue). Further, they can lead to interpretation issues of software components and impede the evolution and maintenance of AI-based systems. Therefore, we claim that data smells contribute massively to the emergence of technical debt and data-induced bugs within AI-based systems and are thus an increasingly important area of research.

However, although research on data issues is quite mature, it only partly considered such potential data issues. In fact, previous studies (e.g., [31, 33, 35]) mainly focused on actual data errors, lacking a precise definition or categorization of such potential issues.

This paper aims to address this gap by providing a solid foundation of data smells. In detail, we describe the characteristics of data smells and outline their potential causes, consequences, and use in the context of AI-based systems. We further present a catalogue comprising 36 data smells divided into three categories (Believability, Understandability, and Consistency Data Smells). Moreover, the detection of data smells is discussed and corresponding tool support is presented. Although we consider data smells in the context of AI-based systems in this article, the concept presented is

ACM ISBN 978-1-4503-9275-4/22/05...$15.00
https://doi.org/10.1145/3522664.3528590

Unmasking Data Secrets: An Empirical Investigation into Data Smells and Their Impact on Data Quality

Gilberto Recupito
Sesa Lab - University of Salerno
Salerno, Italy
grecupito@unisa.it

Raimondo Rapacciuolo
University of Salerno
Salerno, Italy
r.rapacciuolo1@studenti.unisa.it

Dario Di Nucci
Sesa Lab - University of Salerno
Salerno, Italy
ddinucci@unisa.it

Fabio Palomba
Sesa Lab - University of Salerno
Salerno, Italy
fpalomba@unisa.it

ABSTRACT

Artificial Intelligence (AI) is rapidly advancing with a data-centered approach suitable for various domains. Nevertheless, AI faces significant challenges, particularly in data quality. Data collection from diverse sources can introduce quality issues that may threaten the development of AI-enabled systems. A growing concern in this context is the emergence of *data smells* – issues specific to the data used in building AI models, which can have long-term consequences. In this paper, we aim at enlarging the current body of knowledge on data smells, by proposing a two-step investigation into the matter. First, we updated an existing literature review in an effort of cataloguing the currently existing data smells and the tools to detect them. Afterward, we assess the prevalence of data smells and their correlation with data quality metrics. We identify a novel set composed of 12 data smells distributed across three additional categories. Secondly, we observe that the correlation between data smells and data quality is notably impactful, exhibiting a pronounced and substantial effect, especially in highly diffused data smell instances. This research sheds light on the complex relationship between data smells and data quality, providing valuable insights into the challenges of maintaining AI-enabled systems.

CCS CONCEPTS

• **Software and its engineering → Software maintenance tools.**

KEYWORDS

AI Technical Debt; Data Smells; Data Quality; Software Engineering for Artificial Intelligence, Empirical Software Engineering.

ACM Reference Format:
Gilberto Recupito, Raimondo Rapacciuolo, Dario Di Nucci, and Fabio Palomba. 2023. Unmasking Data Secrets: An Empirical Investigation into Data Smells and Their Impact on Data Quality. In *CAIN '24: 3rd International Conference on AI Engineering – Software Engineering for AI, April 14–15, 2024, Lisbon, ES.* ACM, New York, NY, USA, 11 pages. https://doi.org/XXXXXXX.XXXXXXX

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

1 INTRODUCTION

Artificial Intelligence (AI) is more and more diffused nowadays, being used by individuals and companies to make informed decisions [54] and automate tasks that would typically done by humans [38]. Indeed, AI-intensive systems, i.e., systems that embed artificial intelligence models and algorithms, have been recently deployed in multiple domains, with some recent applications showing highly efficient and accurate performance [31, 34].

However, the development of artificial intelligence-enabled systems differs from other types of software because the program and its effectiveness in solving a specific task heavily rely on the data and observations used to train models [3].

More specifically, AI-enabled systems are defined as *a system consisting of various software components, out of which at least one is an AI-specific component* [30]. In this context, data represents the primary source of producing business-oriented AI-enabled systems. Performing data analysis and validation is a crucial initial step for designers of machine learning components. Failure to properly analyze the training data may lead to model degradation [27]. Therefore, it is crucial to prioritize data quality to build a reliable and effective AI-enabled system. Data quality issues can arise for various reasons, *e.g.*, data entry errors, inadequate data cleaning, or bias in the data. Addressing these issues requires a well-defined data quality management strategy that includes profiling, cleansing, and data enrichment [33]. While different tools and practices are available to support feature engineering and data transformation for managing AI pipelines [39], the need to improve the practices related to quality assurance is continuously increasing [10].

Data quality degradation could also lead to technical debt for the whole system [13]. Data debt, primarily when introduced by data quality issues or data anomalies, can strongly impact AI-enabled systems, degrading model performance and causing problems to all the subsequent phases involved in the pipeline [2].

In analyzing technical debt specific to data, *data smells* are represented using the analogy of code smells. As code smells are defined as *symptoms of poor design and implementation choices* [15], data smells are *data value-based indications of latent data quality issues caused by poor practices that may lead to problems in the future* [14]. While other types of data quality issues are investigated in research

What is DSD?

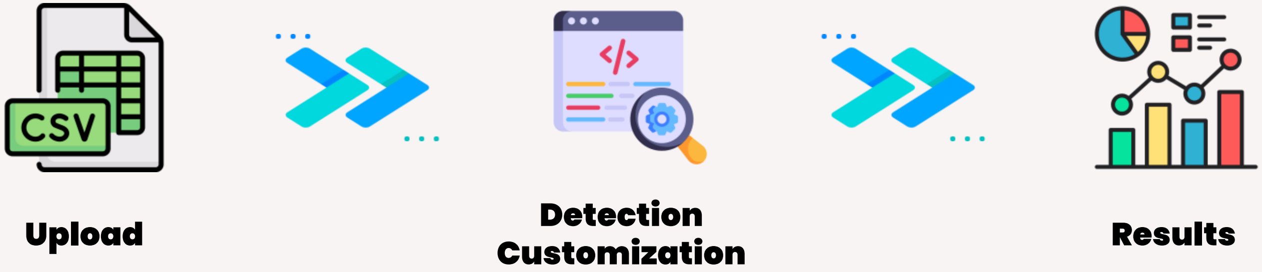
Data Smell Detection (DSD) is solution to enable automated data smell detection.

Rule-based approach

Machine Learning approach

Detection procedure

Data Smell Detection (DSD) is solution to enable automated data smell detection.



Goals of the Project

Addition of new data smell detectors

Calculate the data quality dimensions

Improvement on the reporting system

02

METHODOLOGY

DQDs Choice

Completeness

Using results of the implemented **Missing Value** smell

$$\frac{NumberOfNonEmptyValues}{(TotalValues)} * 100$$

Uniqueness

Using results of the implemented **Duplicated Value** smell

$$\frac{NumberOfUniqueRows}{(TotalRows)} * 100$$

Validity

Using results of the implemented **Integer as String**, **Floating-Point Number as String**, **Integer as Floating-Point Number** smells

$$\frac{NumberOfValidValues}{(TotalValues)} * 100$$

Data Smell Choice

2 new regex-based data smell

Spacing Smell

"^\s|\s\s + |\s$"

Special Character Smell

"[^a − zA − z0 − 9\s]"

Data Collection

A subset of **60** datasets used on earlier studies

These datasets cover most application domains, to have multiple **different** data types

Detector Testing

Spacing Smell Test Cases

```
"data": {
        "begin_space": [" test", "test", "test"],
        "multiple_begin_space": ["       test", "test", "test"],
        "inner_space": ["test     test", "test", "test"],
        "end_space": ["test ", "test", "test"],
        "multiple_end_space": ["test       ", "test", "test"],
        "no_spacing_smell": ["test", "test test", "test"],
}
```

Special Character Smell Test Cases

```
"data": {
        "punctuation_special_character": ["te§t", "t&st", "¿test?", "test ¶", "~test"],
        "stressed_letter_special_character": ["tæst", "tëst", "tÉst", "test Å", "çtest"],
        "no_special_character_smell": ["22", "hello", "hello22", "HELLO", "HELLO 22"]
}
```

Test Cases Examples

```
"tests": [
            {
                "title": "begin_space_test",
                "exact_match_out": False,
                "include_in_gallery": True,
                "in": {"column": "begin_space", "mostly": 1},
                "out": {"success": False}
            },

            {
                "title": "punctuation_special_character_test",
                "exact_match_out": False,
                "include_in_gallery": True,
                "in": {"column": "punctuation_special_character", "mostly": 1},
                "out": {"success": False}
            }
]
```

Expected output: False, if test data contains the smell, otherwise True

DQDs Testing

Randomly generation of **3** new datasets of **3** columns to evaluate metrics calculation

Dataset	Metric to evaluate	Columns: first_name	last_name	street_number
Dataset #1	Completeness Uniqueness Validity	5 empty 4 duplicates all valid	9 empty 6 duplicates all valid	10 empty 10 duplicates 7 non-valid

Dataset	Metric to evaluate	Columns: car_maker	color	model_year
Dataset #2	Completeness Uniqueness Validity	10 empty 20 duplicates all valid	10 empty 22 duplicates all valid	10 empty 19 duplicates 5 non-valid

Dataset	Metric to evaluate	Columns: ssn	currency	number
Dataset #3	Completeness Uniqueness Validity	5 empty 12 duplicates all valid	7 empty 19 duplicates all valid	5 empty 16 duplicates 1 non-valid

Table 2.2: Datasets informations

03

RESULTS

DQDs Testing

The testing phase on DQDs implementations led to the following results

Completeness				
Dataset	**Expected global value**	**Actual global value**	**Expected column values**	**Actual colum values**
Dataset #1	77.33%	77.3%	83.33% / 70% / 66.67%	83.33% / 70% / 66.67%
Dataset #2	66.67%	66.67%	66.67% / 66.67% / 66.67%	66.67% / 66.67% / 66.67%
Dataset #3	81.11%	81.11%	83.33% / 76.67% / 83.33%	83.33% / 76.67% / 83.33%

Uniqueness				
Dataset	**Expected global value**	**Actual global value**	**Expected column values**	**Actual colum values**
Dataset #1	77.78%	77.78%	86.67% / 80% / 66.67%	86.67% / 80% / 66.67%
Dataset #2	33.22%	33.22%	33.33% / 26.67% / 36.67%	33.33% / 26.67% / 36.67%
Dataset #3	47.78%	47.78%	60% / 36.67% / 46.67%	60% / 36.67% / 46.67%

Validity				
Dataset	**Expected global value**	**Actual global value**	**Expected column values**	**Actual colum values**
Dataset #1	92.22%	70%	100% / 100% / 76.66%	100% / 100% / 86.67%
Dataset #2	94.44%	70%	100% / 100% / 83.34%	100% / 100% / 93.33%
Dataset #3	98.89%	98.89%	100% / 100% / 96.67%	100% / 100% / 96.67%

No differences between oracles and actual results

Assumptions and implementations are both right!

DQDs Testing

The testing phase on DQDs implementations led to the following results

Completeness				
Dataset	**Expected global value**	**Actual global value**	**Expected column values**	**Actual colum values**
Dataset #1	77.33%	77.3%	83.33% / 70% / 66.67%	83.33% / 70% / 66.67%
Dataset #2	66.67%	66.67%	66.67% / 66.67% / 66.67%	66.67% / 66.67% / 66.67%
Dataset #3	81.11%	81.11%	83.33% / 76.67% / 83.33%	83.33% / 76.67% / 83.33%

Uniqueness				
Dataset	**Expected global value**	**Actual global value**	**Expected column values**	**Actual colum values**
Dataset #1	77.78%	77.78%	86.67% / 80% / 66.67%	86.67% / 80% / 66.67%
Dataset #2	33.22%	33.22%	33.33% / 26.67% / 36.67%	33.33% / 26.67% / 36.67%
Dataset #3	47.78%	47.78%	60% / 36.67% / 46.67%	60% / 36.67% / 46.67%

Validity				
Dataset	**Expected global value**	**Actual global value**	**Expected column values**	**Actual colum values**
Dataset #1	92.22%	70%	100% / 100% / 76.66%	100% / 100% / 86.67%
Dataset #2	94.44%	70%	100% / 100% / 83.34%	100% / 100% / 93.33%
Dataset #3	98.89%	98.89%	100% / 100% / 96.67%	100% / 100% / 96.67%

Small differences between oracles and actual results

Wrong assumptions on how the smells involved work!

The implementation is right!

Real-World Simulation

Test procedures were also conducted on the new detectors with complete success!

The next step was to test them on some real data to see the results.

- **sf-salaries**
- **aws-cyberattacks**
- **indicators-by-company**
- **red-light-camera-violations**
- **food-inspections**
- **metropolitan-objects**

The choice of this subset of datasets was simply related to the fact that they represent different domains and fields.

Real-World Simulation

Dataset & Columns		Total Elements Count	Faulty Elements
sf-salaries	EmployeeName	148654	10507
	JobTitle	148654	16
aws-cyberattacks	localeabbr	451581	83
metropolitan-objects	City	448203	7
	Country	448203	8
	Culture	448203	71
	Dimensions	448203	91856
	Dinasty	448203	3
	Excavation	448203	13
	Locale	448203	24
	Locus	448203	5
	Medium	448203	5753
	Portfolio	448203	981
	Region	448203	408
	Reign	448203	3
	River	448203	8
	State	448203	8
	Subregion	448203	530
	Title	448203	4735
food-inspections	Address	153810	153366
	Violations	153810	86170

Spacing smells detected

Dataset & Columns		Total Elements Count	Faulty Elements
sf-salaries	EmployeeName	148654	4614
	JobTitle	148654	20703
aws-cyberattacks	localeabbr	451581	83
	country	451581	872
	datetime	451581	451581
	host	451581	451581
	srcstr	451581	451581
	locale	451581	11229
red-light-camera-violations	INTERSECTION	521533	11188
	LOCATION	521533	494422
metropolitan-objects	City	448203	1656
	Classification	448203	152157
	Country	448203	2681
	County	448203	1696
	Culture	448203	34459
	Department	448203	12427
	Dimensions	448203	381726
	Dinasty	448203	9013
	Excavation	448203	12608
	Locale	448203	5372
	Locus	448203	2453
	Medium	448203	153559
	Portfolio	448203	12307
	Region	448203	12532
	Reign	448203	2062
	Repository	448203	448203
	River	448203	299
	State	448203	1181
	Subregion	448203	8006
	Title	448203	191502
food-inspections	Address	153810	17136
	Location	153810	153266
	Results	153810	14530
	Risk	153810	153725
	Violations	153810	123012

Special-character smells detected

User Interface Update

Improvements on the UI are **clearly** visible.

data_smell_testset_EoWvU7K.csv

Selected parameters for detected data smells

DUPLICATED VALUE SMELL
mostly: 0.9

CASING SMELL
mostly: 0.9 same_case_wordcount_threshold: 2.0

Columns with data smells

int2

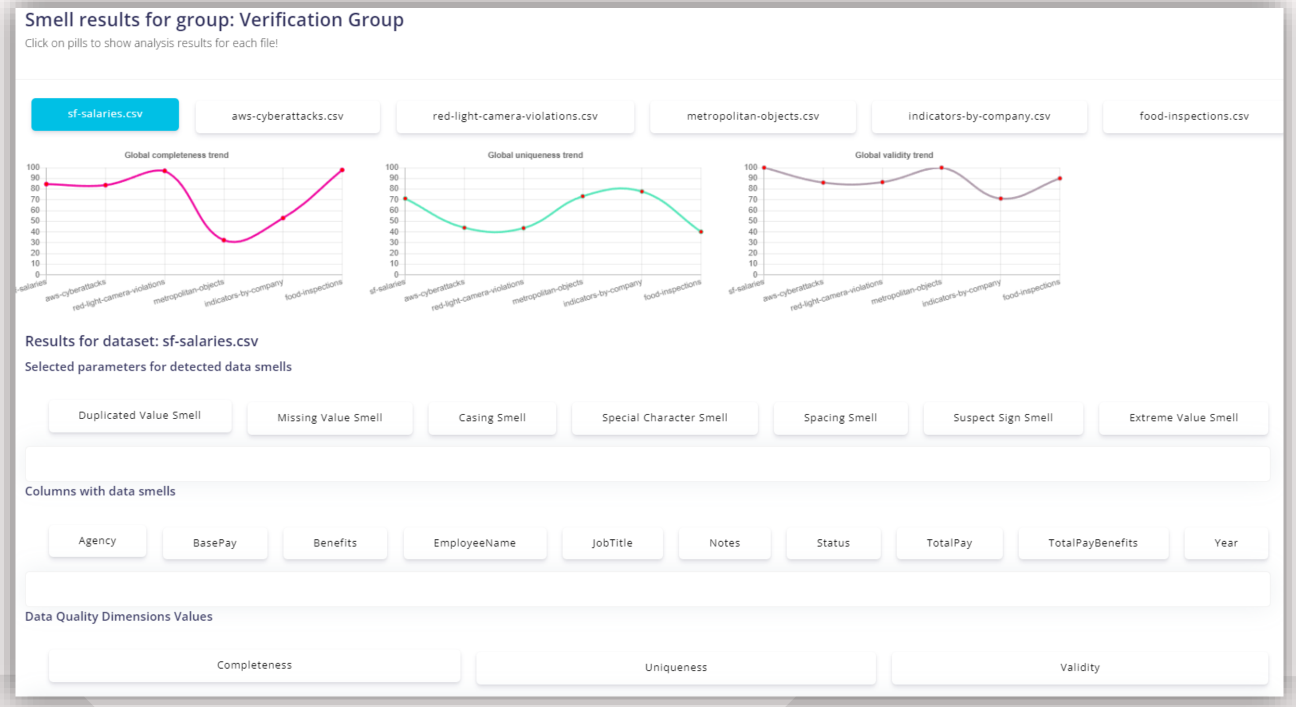
DATA SMELL TYPE	TOTAL ELEMENT COUNT	FAULTY ELEMENT COUNT	FAULTY ELEMENT OVERVIEW
Duplicated Value Smell	11	2	[8, 8]

string1

DATA SMELL TYPE	TOTAL ELEMENT COUNT	FAULTY ELEMENT COUNT	FAULTY ELEMENT OVERVIEW
Casing Smell	11	6	['abc def ghi', 'abc def ghi', 'cAsing 1', 'CaSing 2', 'all lowercase', 'ALL UPPERCASE']
Duplicated Value Smell	11	2	['abc def ghi', 'abc def ghi']

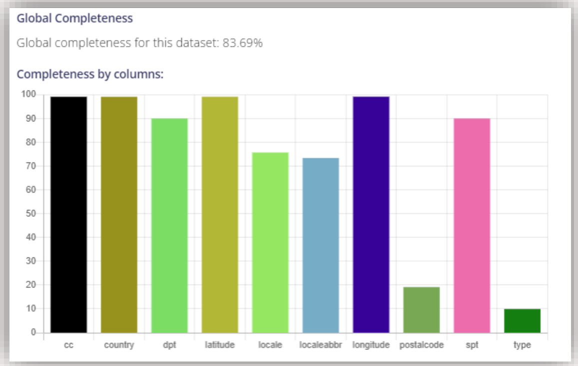
User Interface Update

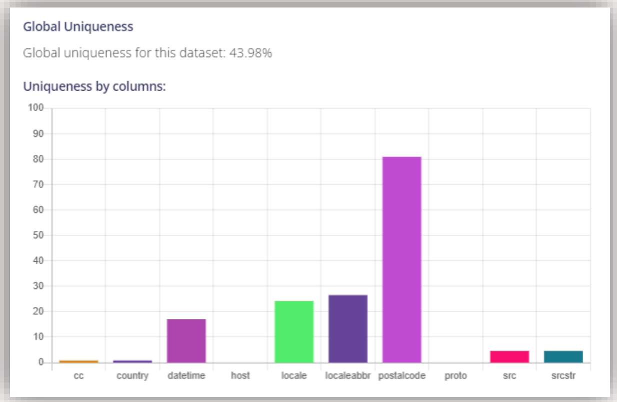
Improvements on the UI are **clearly** visible.

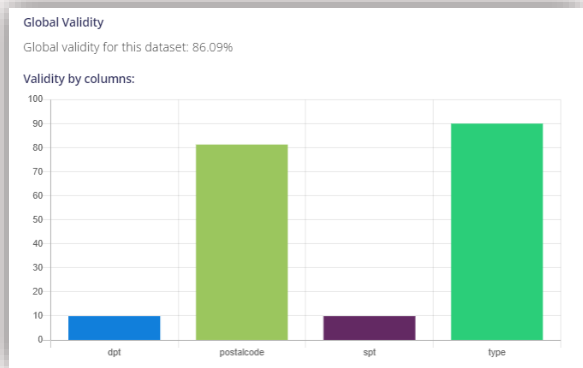


User Interface Update

Improvements on the UI are **clearly** visible.







04

CONCLUSIONS

Conclusions

Report Improvements

Improvements made on this tool are clearly visible and help users to visualize in a better way the results of the tool

Detector Extension

Enrichment of the detector suite with two new smells that let users detect new type of smells and visualize faulty elements

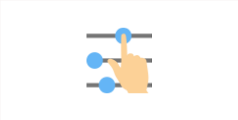
DQDs Implementation

New DQDs implemented which help users understanding some problems on it and verify if he was able to solve the problems checking how they evolve over time

Future Works

Export Results

View the results in a convenient file format, without opening every time the tool



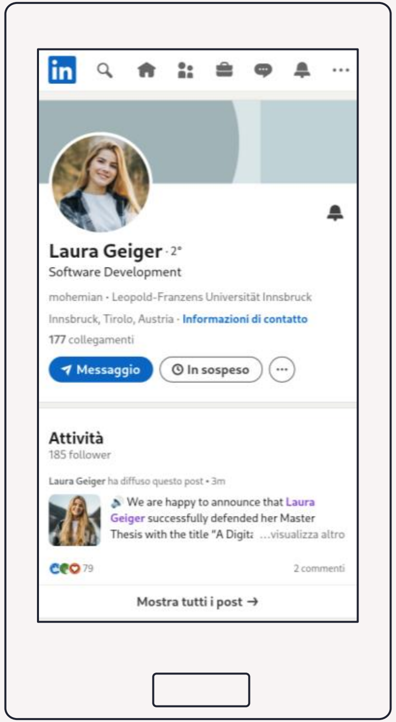
Custom Profiles

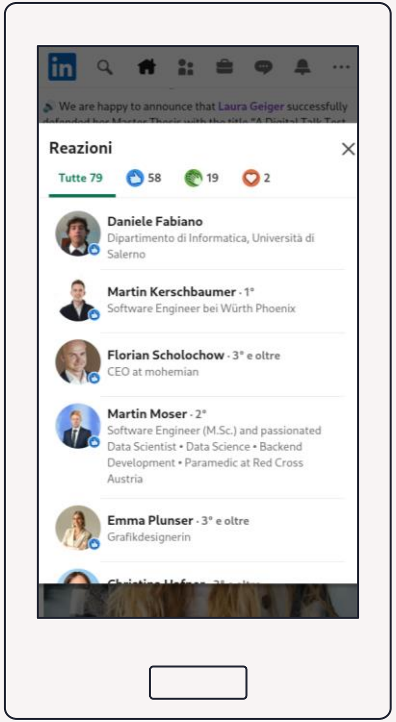
Saving a specific customization that could be used at multiple times

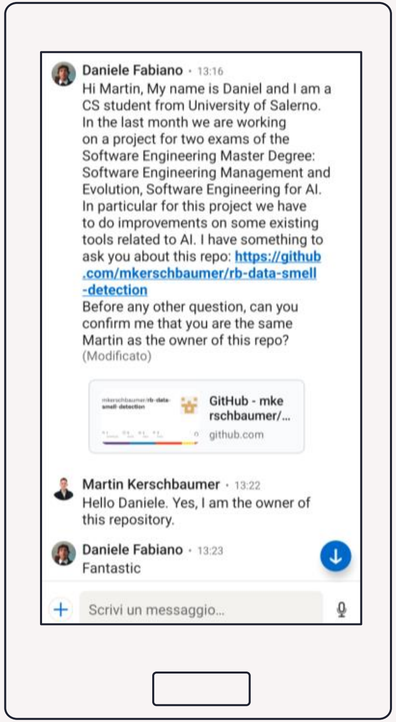
Extend DSD+ Suite

There are (still) some other unimplemented detectors that can be inserted into the detector suite

Contact with Mantainers







Thank you for your attention