Paspailleur

Python package for Pattern Structures

https://github.com/smartFCA/paspailleur

Egor Dudyrev, ConSoft workshop @ CONCEPTS'25, Cluj-Napoca, Romania

Outline

- Short intro to scaling and Pattern Structures
- Structure of Paspailleur package
- Demo
- Conclusion

How we treat the complex data

Running example: Titanic data

	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Fare	Embarked
PassengerId									
1	No	3	Braund, Mr. Owen Harris	male	22.0	1	0	7.2500	Southampton
2	Yes	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	71.2833	Cherbourg
3	Yes	3	Heikkinen, Miss. Laina	female	26.0	0	0	7.9250	Southampton
4	Yes	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	53.1000	Southampton
5	No	3	Allen, Mr. William Henry	male	35.0	0	0	8.0500	Southampton

Scaling →

Formal Context

	Survived = No	Survived = Yes	Pclass ≥ 1	Pclass ≥ 2	Pclass ≥ 3	Pclass ≤ 1	Pcla ≤
1	✓		✓	✓	✓		
2		✓	✓			✓	✓
3		✓	✓	✓	✓		
4		✓	✓			✓	✓
5	✓		✓	✓	✓		

Scaling problems

- One should write functions how to binarise and de-binarise the data
- Contexts with hundreds of attributes are hard to read
- Much slower computation time (Kaytoue et al., IJCAI 2011)
- Mining obvious implications:
  - “Pclass ≥ 3” => “Pclass ≥ 2”
  - “Pclass ≥ 2” => “Pclass ≥ 1”
  - etc

Scaling problems

Scaling does not scale well !

- One should write functions how to binarise and de-binarise the data
- Contexts with hundreds of attributes are hard to read
- Much slower computation time (Kaytoue et al., IJCAI 2011)
- Mining obvious implications:
  - “Pclass ≥ 3” => “Pclass ≥ 2”
  - “Pclass ≥ 2” => “Pclass ≥ 1”
  - etc

Solution

Pattern Structures

- Every object is described by a *pattern*
- What is pattern?
  Something that belongs to a complete meet-semilattice $(\mathbb{D}, \sqcap)$
- A dataset is modelled by a *pattern* structure $(G, (\mathbb{D}, \sqcap), \delta : G \to \mathbb{D})$

Pattern Structures and Their Projections

Bernhard Ganter[1] and Sergei O. Kuznetsov[2]

[1] Institut für Algebra, TU Dresden
D-01062 Dresden, Germany
ganter@math.tu-dresden.de
[2] All-Russia Institute for Scientific and Technical Information (VINITI)
Usievicha 20, 125219 Moscow, Russia
serge@viniti.ru

Abstract. Pattern structures consist of objects with descriptions (called patterns) that allow a semilattice operation on them. Pattern structures arise naturally from ordered data, e.g., from labeled graphs ordered by graph morphisms. It is shown that pattern structures can be reduced to formal contexts, however sometimes processing the former is often more efficient and obvious than processing the latter. Concepts, implications, plausible hypotheses, and classifications are defined for data given by pattern structures. Since computation in pattern structures may be intractable, approximations of patterns by means of projections are introduced. It is shown how concepts, implications, hypotheses, and classifications in projected pattern structures are related to those in original ones.

Introduction

Our investigation is motivated by a basic problem in pharmaceutical research. Suppose we are interested which chemical substances cause a certain effect, and which do not. A simple assumption would be that the effect is triggered by the presence of certain molecular substructures, and that the non-occurence of the effect may also depend on such substructures.

Suppose we have a number of observed cases, some in which the effect does occur and some where it does not; we then would like to form hypotheses on which substructures are responsible for the observed results. This seems to be a simple task, but if we allow for combinations of substructures, then this requires an effective strategy.

Molecular graphs are only one example where such an approach is natural. Another, perhaps even more promising domain is that of *Conceptual Graphs* (CGs) in the sense of Sowa [21] and hence, of logical formulas. CGs can be used to represent knowledge in a form that is close to language. It is therefore of interest to study how hypotheses can be derived from Conceptual Graphs.

A strategy of hypothesis formation has been developed under the name of JSM-method by V. Finn [8] and his co-workers. Recently, the present authors have demonstrated [11] that the approach can neatly be formulated in the language of another method of data analysis: Formal Concept Analysis (FCA) [12].

H. Delugach and G. Stumme (Eds.): ICCS 2001, LNAI 2120, pp. 129–142, 2001.

What is Pattern

Practice

Pattern Name	When to use	Ex. Column	Example
ItemSet Pattern	tags and keywords	Scaled data	{Mr., Pclass ≤ 2} ⊓ {Miss., Pclass ≤ 2} = {Pclass ≤ 2}
CategorySet Pattern	categorical data	Embarkment	{Southampton} ⊓ {Cherbourg} = {South., Cherbourg}
SequenceSet Pattern	sequences	Name	{“Mr. Jack Smith”} ⊓ {“Mr. John Smith”} = {“Mr.”, “Smith”}
Interval Pattern	numerical data	Age	[20, 20] ⊓ [30, 30] = [20, 30]
Cartesian Pattern	tabular data	Embarkment x Age	({South.}, [20,20]) ⊓ ({Cherb.}, [30,30]) = ({South., Cherb.}, [20, 30])
GraphSet Pattern	graphs		{graph X} ⊓ {graph Y} = maximal common connected induced subgraphs of X and Y

“Worse than NP-complete” S.K.

What is Pattern

Practice

Pattern Name	When to use	Ex. Column	Example
ItemSet Pattern	tags and keywords	Scaled data	{Mr., Pclass ≤ 2} ⊓ {Miss, Pclass ≤ 2} = {Pclass ≤ 2}
CategorySet Pattern	categorical data	Embarkment	{Southampton} ⊓ {Cherbourg} = {South., Cherbourg}
SequenceSet Pattern	sequences	Name	{"Mr. Jack Smith"} ⊓ {"Mr. John Smith"} = {"Mr.", "Smith"}
Interval Pattern	numerical data	Age	[20, 20] ⊓ [30, 30] = [20, 30]
Cartesian Pattern	tabular data	Embarkment x Age	({South.}, [20,20]) ⊓ ({Cherb.}, [30,30]) = ({South., Cherb.}, [20, 30])
GraphSet Pattern	graphs		{graph X} ⊓ {graph Y} = maximal common connected induced subgraphs of X and Y

PS do not always scale well !

"Worse than NP-complete" S.K.

How to scale patterns

- Fix the datatype:
  - Graphs (S. Kuznetsov 2007, A. Buzmakov et al. 2017)
  - Numbers (M. Kaytoue et al. 2011)
  - Temporal sequences (S. Boukhetta et al., 2021)
- Fix the framework:
  - Predicates and strategies: GALACTIC
  - Atomic Patterns: Paspailleur

What are atomic patterns?

Magic !

More details on Friday at 12:30

What are Paspailleur?

Paspailleur Python Package

Pattern Mining with Pattern Structures.

🚢 Getting started

Learn the basics of Paspailleur in a single page.

Titanic Study

🌀 Patterns API

Read the documentation for Patterns and PatternStructure classes: the frontend of the package.

Patterns API

⚙️ Algorithms API

Read the documentation for functions that work behind Patterns API: the backend of the package.

Algorithms API

https://smartfca.github.io/paspailleur/

Patterns API
The frontend

	Pattern	Pattern Structure
Models	A data type	A dataset
Operations	Meets and joins of patterns	Mining patterns in a patterns-set
Base class	Pattern	PatternStructure
Specific classes	• `ItemSetPattern` • `CategorySetPattern` • `IntervalPattern` • `NgramSetPattern` • `CartesianPattern`	In the future releases
Custom class creation	In the future relasess	In the future releases

Algorithms API

The backend

Base functions.py

- `extension()`
- `intention()`
- `group_objects_by_patterns()`
- `order_patterns_via_extents()`
- `iter_patterns_ascending()`
- `iterate_antichains()`

Mine equivalence classes.py

- `iter_intents_via_ocbo()`
- `iter_intents_via_cboi()`
- `list_stable_extents_via_gsofia()`
- `iter_keys_of_pattern()`
- `iter_keys_of_patterns()`
- `iter_keys_of_patterns_via_atoms()`
- `iter_all_patterns_ascending()`

Mine subgroups.py

- `iter_subgroups_bruteforce()`
- `iter_subgroups_via_atoms()`

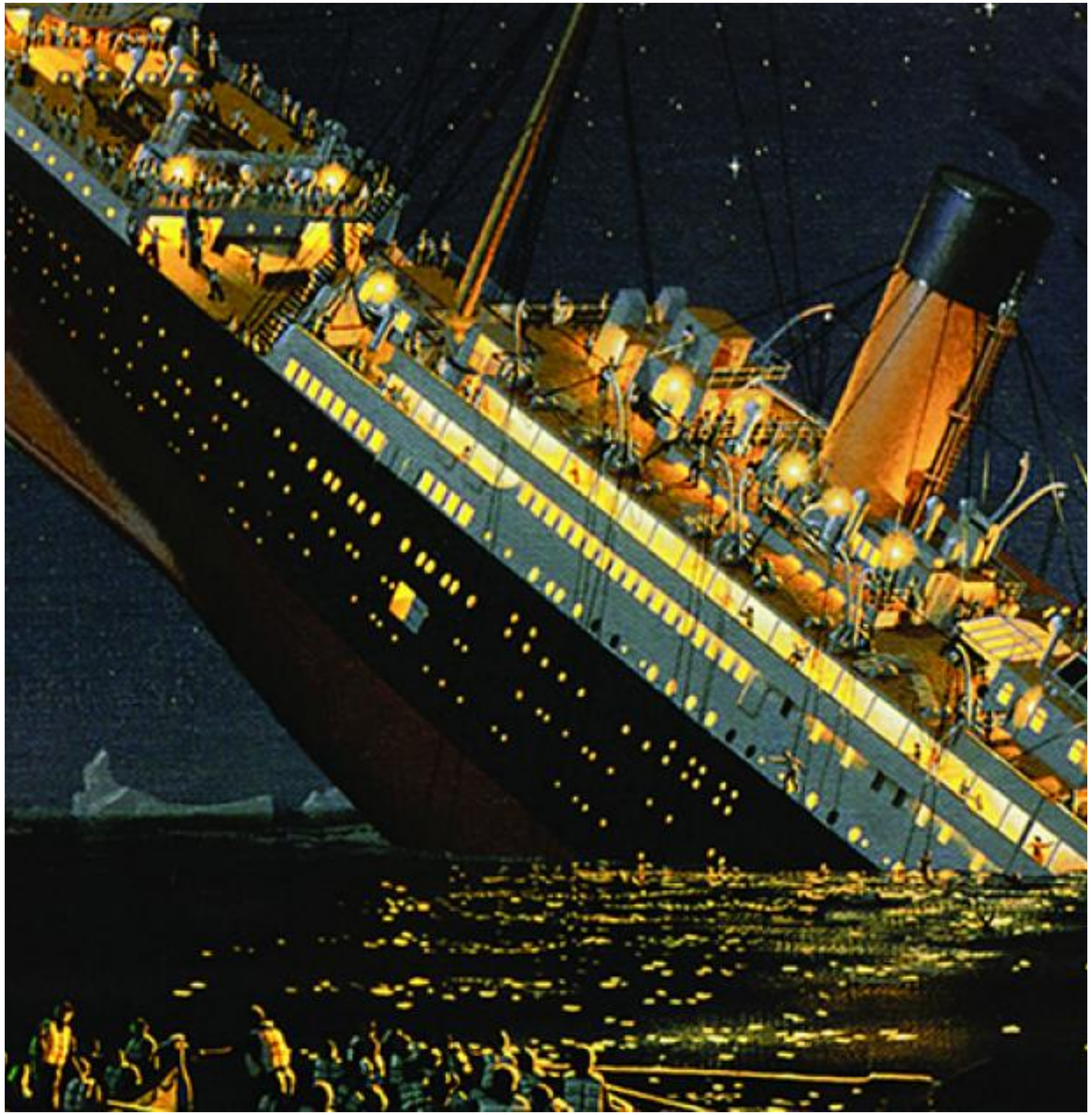
Mine implications.py

- `iter_proper_premises_from_atomised_premises()`
- `iter_pseudo_intents_from_atomised_premises()`

Titanic Example

https://smartfca.github.io/paspailleur/example_from_titanic.html

(All code there is run by GitHub in about a minute)



Conclusions

- Paspailleur is a Python package for Pattern Structures started in 2023
- It got rewritten from scratch in the first 2 weeks of January 2025
- Is based on the idea of atomic patterns (to be introduced on Friday)
- It mines concepts, implications, and subgroups in complex data without the need for manual binarization
- Supported data formats: itemsets, categories, numbers and intervals, sequences of words (ngrams), cartesian products of everything above.
- Works (surprisingly) fast on the data of thousands objects.

Future work

- More data types:
  - Graphs,
  - Convex polygons,
  - Images (???).
- More specialised pattern structures:
  - For Intervals (via Numpy),
  - For ItemSets and Categories (via bitarrays).