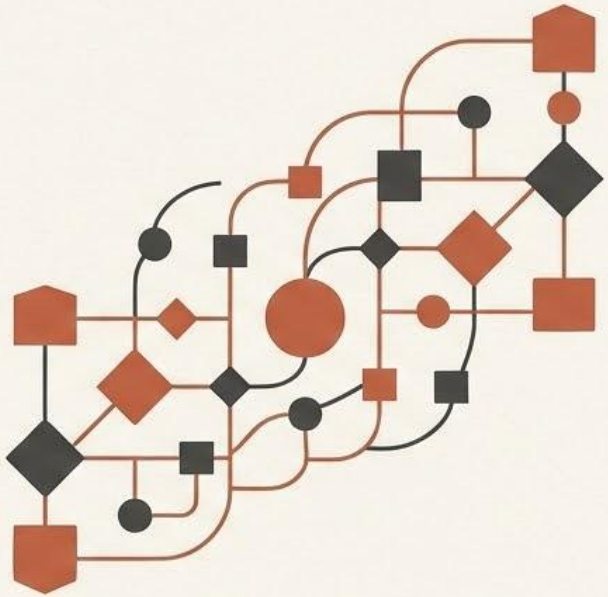
From Theory to Practice

Applications of Advanced Pattern Mining



Duration: 1 Hour

Target Audience: Data Science/Computer Science students

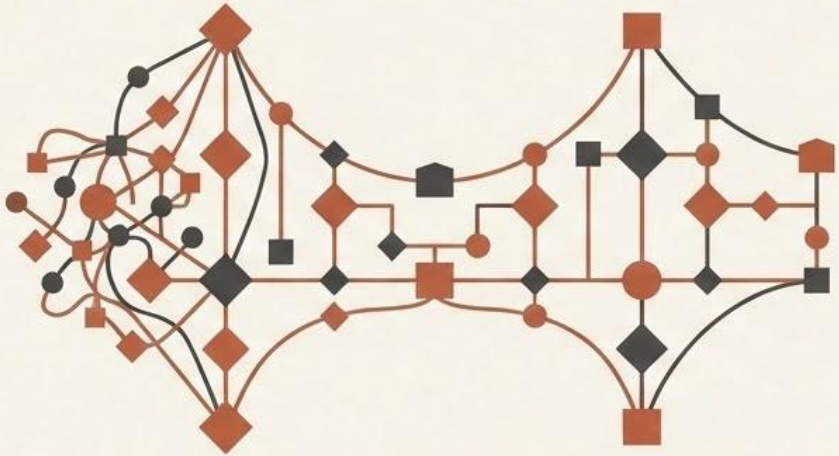
Prerequisites: Understanding of frequent pattern mining, sequential patterns, graph mining concepts

Part 1: Introduction & Learning Objectives

Pattern Mining in the Real World

Bridging Theory and Practice

From Algorithms to Applications



Opening Perspective

Beyond the Algorithm

The Journey So Far:

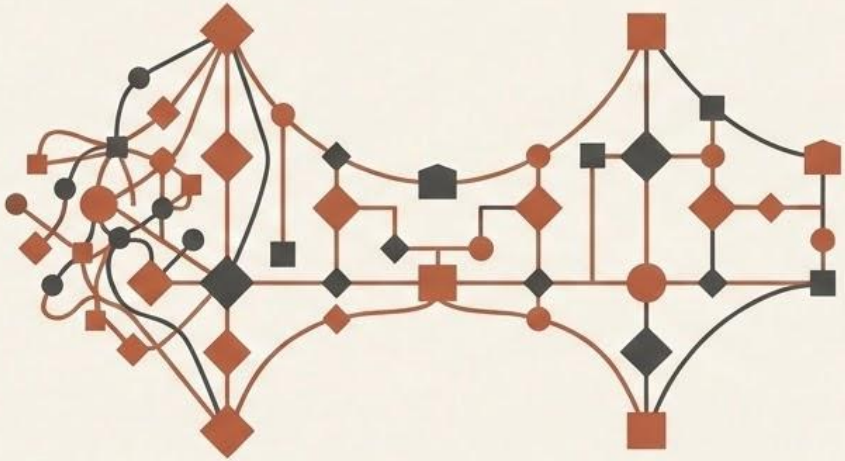
- Itemsets (Shopping Carts).
- Sequences (Clickstreams).
- Graphs (Chemical Compounds).

The Goal Today:

Bridging theory and practice by examining how these techniques solve real-world problems.

The Spectrum:

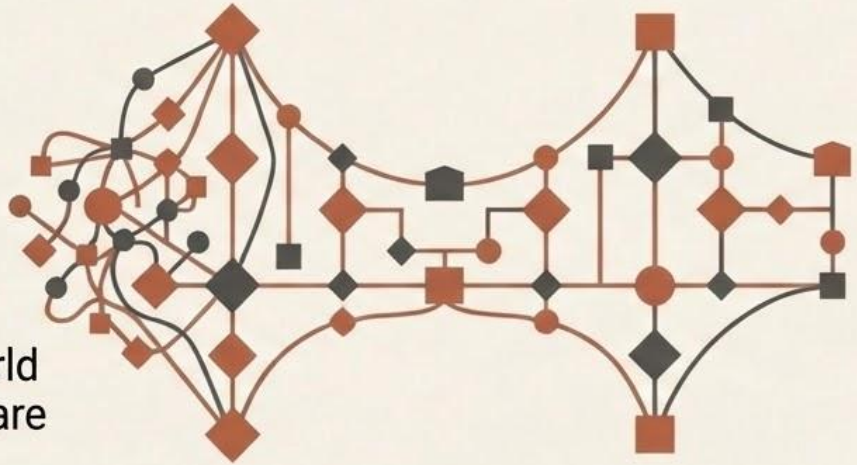
From understanding Genetic Diseases to finding Bugs in Operating Systems.



Learning Objectives

By the end of this module, you will be able to:

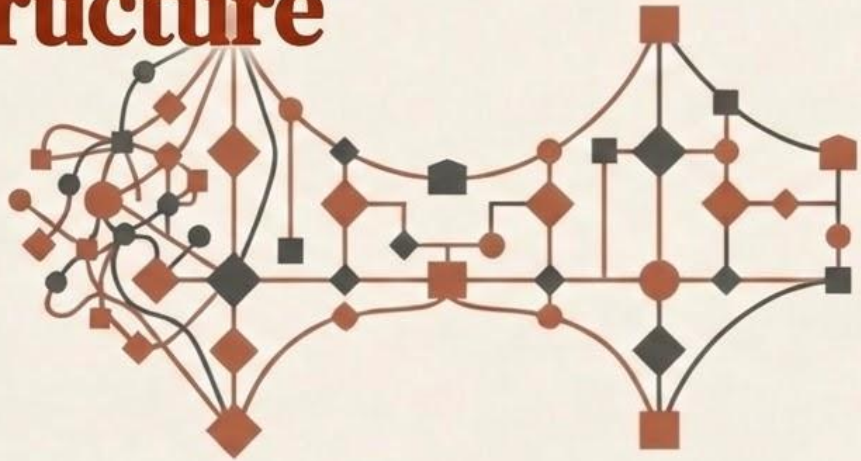
- **Bio-Mining**: Understand constrained substructure mining in biological contexts (e.g., Gene networks).
- **Text Mining**: Apply phrase mining techniques to massive text collections (e.g., Scientific literature).
- **Software Engineering**: Analyze real-world applications of pattern mining in software engineering (e.g., Bug detection).
- **Synthesis**: Connect theoretical pattern mining concepts to their practical implementations in industry.



Section 5.5.2

Part 2: Mining Variant and Constrained Substructure Patterns

Structure, Constraint, and Variation



Focus: Finding patterns in complex biological and chemical data.

Context: Moving beyond exact matches to biologically relevant variations.

Definition & Challenges

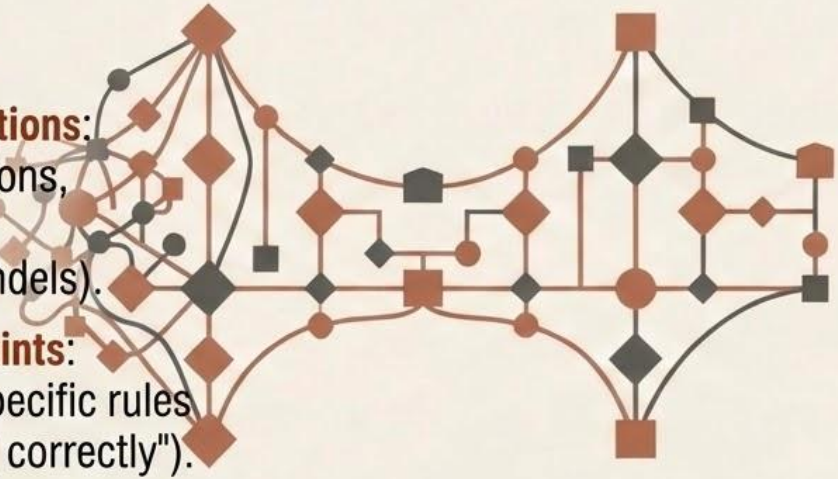
Mining with Flexibility

Definition: Mining patterns where structural variations exist within constraints.

Context: Common in biological sequences (DNA/Proteins), chemical compounds, and network data.

Key Challenges:

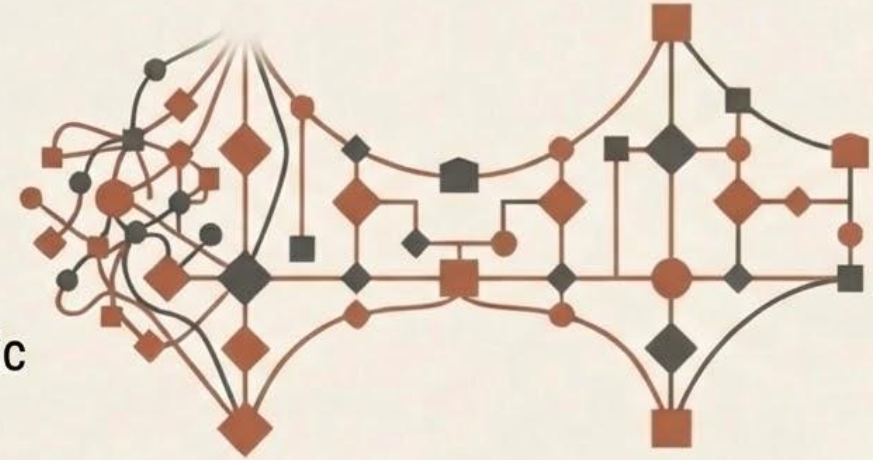
- **Structural Variations**: Handling Insertions, Deletions, and Substitutions (Indels).
- **Domain Constraints**: Incorporating specific rules (e.g., "Must fold correctly").
- **Search Space**: Managing the exponential explosion of possibilities.



Case Study: Protein Structures

Sivley et al. (2017)

Background: A comprehensive study analyzing spatial patterns of genetic variation in 4,568 solved human protein structures.



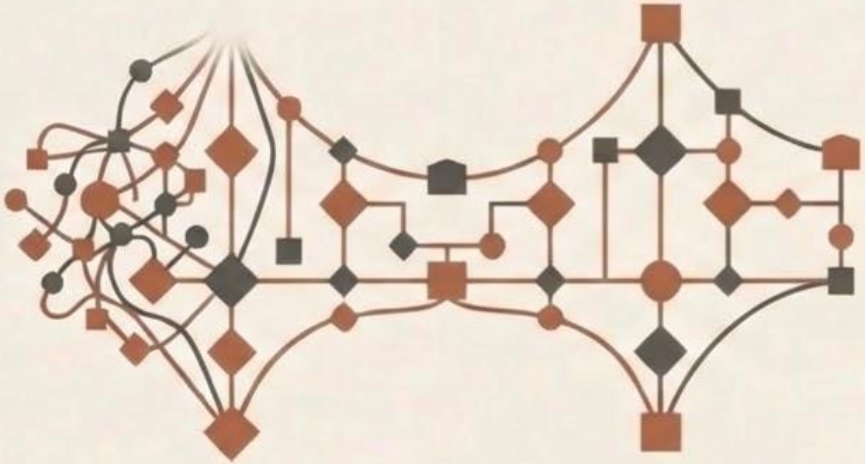
The Problem: Understanding how genetic variants are distributed in 3D Protein Space (not just 1D sequence) to:

- Identify functionally important regions.
- Classify disease-causing mutations.

Datasets Analyzed

Scale of the Study

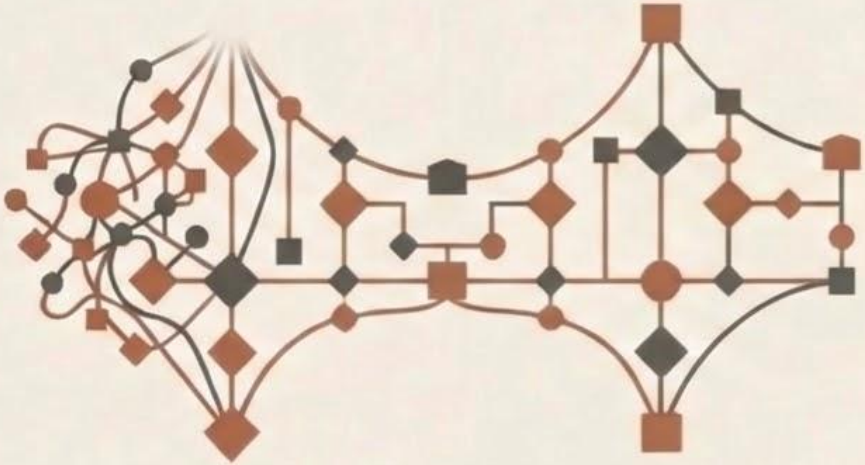
- **ExAC**: 137,352 synonymous + 210,007 missense variants. From 60,706 individuals.
- **ClinVar**: 4,888 missense Mendelian disease variants (Known Pathogenic).
- **COSMIC**: 12,230 recurrent somatic missense variants from tumors (Cancer).



The Spatial Analysis Method

Ripley's K Statistic

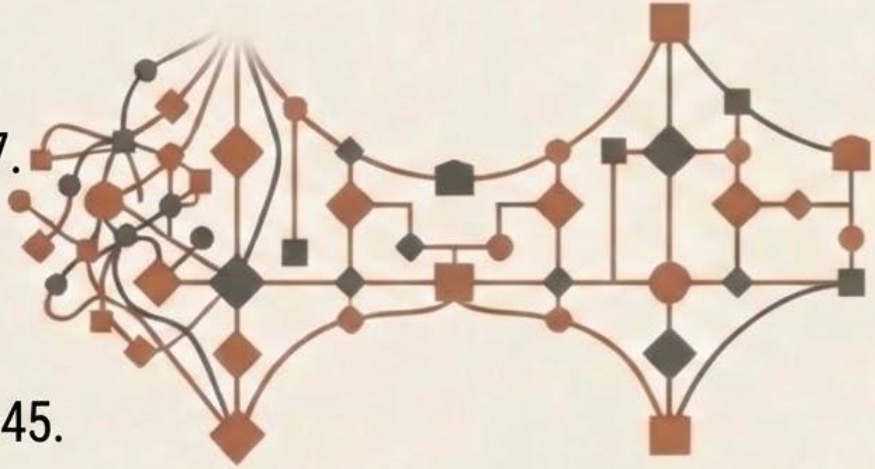
Method: Developed a statistic based on Ripley's K to evaluate deviations from random spatial distribution.



Two Types of Deviation:

- Clustered (Z-score > 0): Variants concentrated in specific regions (Hotspots).
- Dispersed (Z-score < 0): Variants spread more than expected (Constraint).

Key Findings: Synonymous vs. Missense
Random vs. Constrained



Synonymous Variants (Silent):

- Random Distribution: Median Z = 0.07.
- Insight: Generally doesn't affect protein structure.

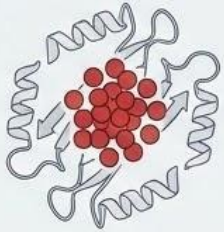
Missense Variants (Healthy Pop):

- Significant Dispersion: Median Z = -0.45.
- Insight: Driven by constraint against substitutions in protein cores; residues tolerant of variation had high solvent accessibility.

Key Findings: Pathogenic & Cancer

Clustering Indicates Disease

Pathogenic Missense Variants:

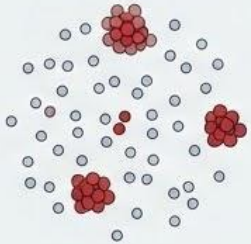


Dramatic Clustering: Median Z = 1.01.

Stat: 28% of proteins exhibited significant clustering.

Insight: Conserved amino acids cluster spatially.

Somatic Cancer Variants:



Modest Overall Clustering: But 16 proteins (0.8%) showed significant clustering.

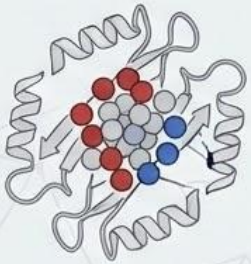
Key Players: Included known cancer drivers like TP53 and KRAS.

Clinical Application: PathProx

From Data to Diagnosis

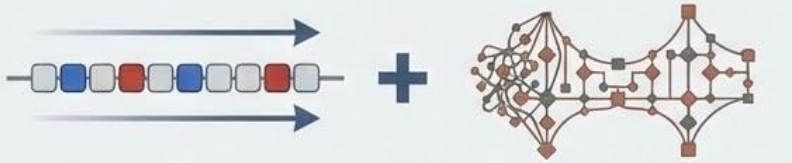
The Algorithm & Mechanism

The Algorithm: Researchers developed PathProx to classify Variants of Unknown Significance (VUS).



Mechanism: Classifies based on “Relative to known Pathogenic and Benign variants.”

Value & Approach



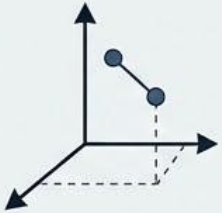
1D Sequence-Based 3D Spatial Approach

Value: A 3D spatial approach that is complementary to traditional 1D sequence-based prediction algorithms.

Constraint Types in Substructure Mining

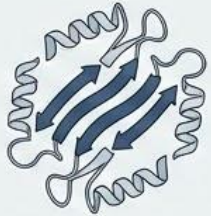
Defining Validity

Spatial Constraints



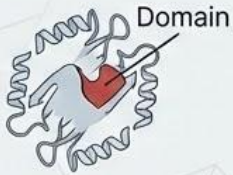
Variants must be within a certain distance in 3D space.

Structural Constraints



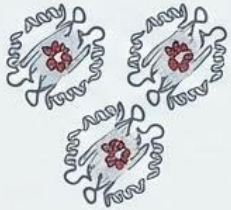
Patterns must preserve protein folding properties.

Functional Constraints



Substructures must correspond to known functional domains.

Frequency Constraints

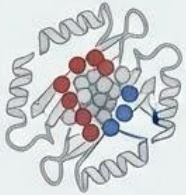


Patterns must appear in multiple protein structures.

General Principle & Takeaway

Why This Matters

Impact

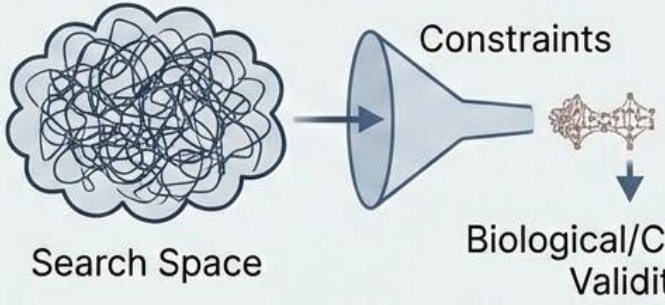


Identifies functional regions.

Helps interpret genetic tests.

Guides drug development.

General Principle



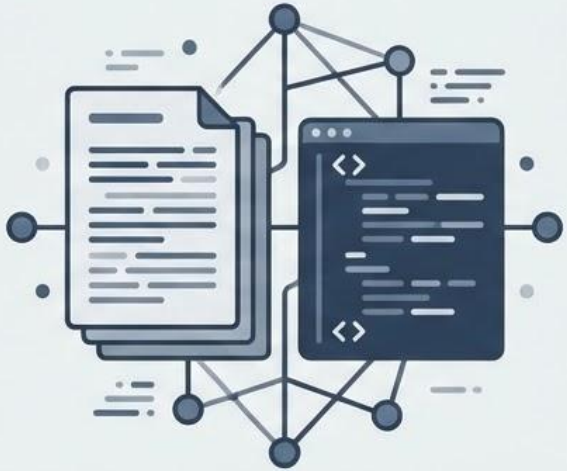
Constraints reduce the search space while ensuring biological/chemical validity.

Part 3: Pattern Mining Applications

Section 5.6

Mining Text & Code

Focus: From Massive Text Collections to Software Engineering.
Context: Applying pattern mining to unstructured data.



Phrase Mining in Massive Text Data

Section 5.6.1: Beyond Single Words

Definition & Concept

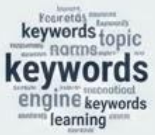
Definition: Automated extraction of meaningful multi-word phrases from large text collections without relying on predefined dictionaries.

Why It Matters: Single words are often insufficient for semantic understanding.

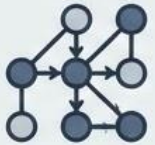
Concept Capture

Benefits

Enables better Information Retrieval



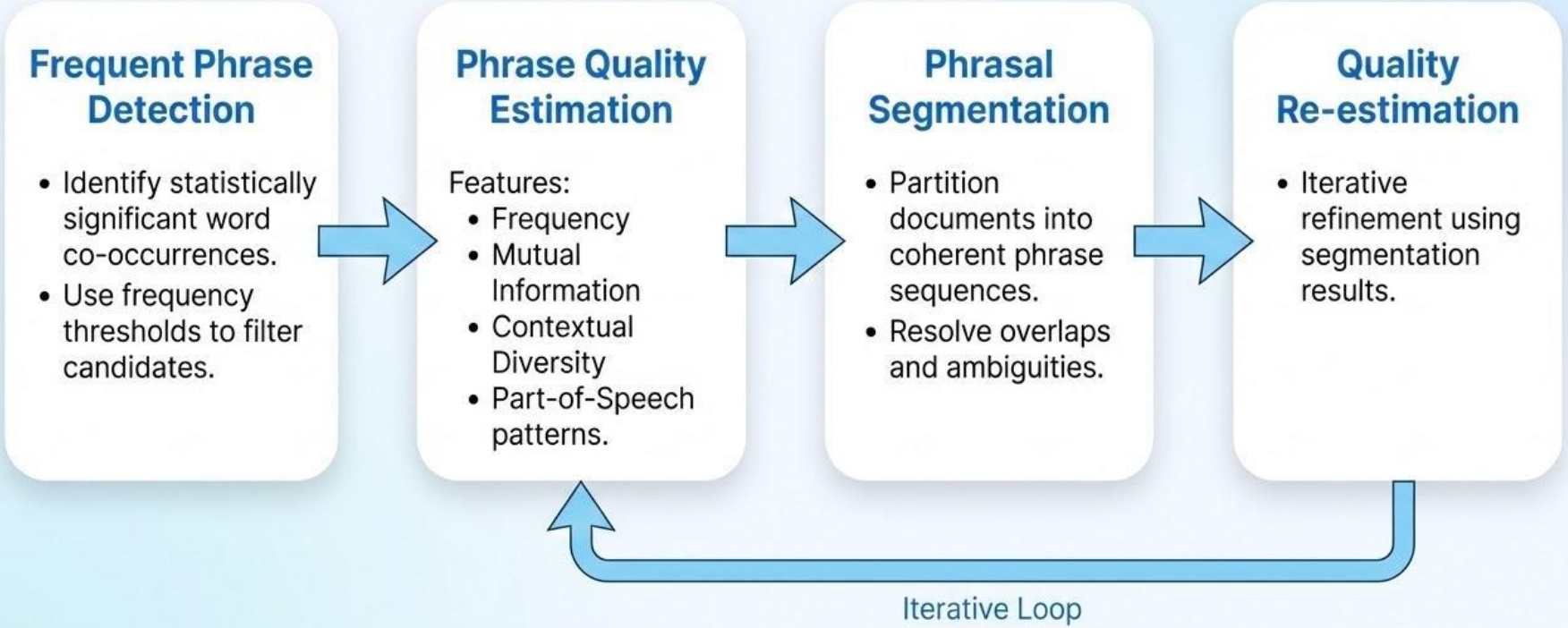
Topic Modeling



Knowledge Base Construction

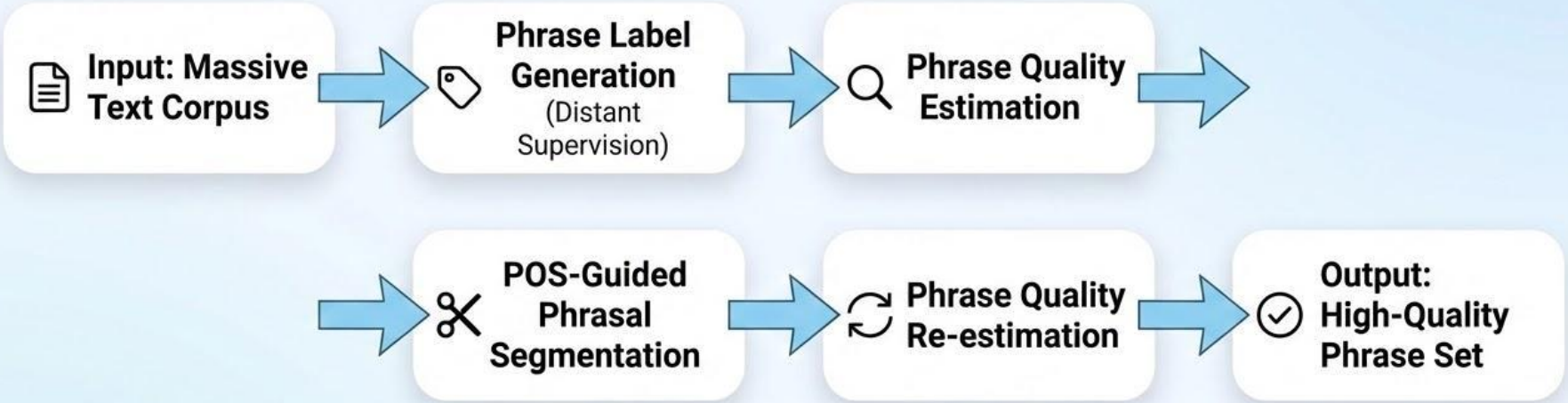
The Phrase Mining Framework (Liu et al.)

A Systematic Pipeline



Automated Phrase Mining Workflow

Reducing Manual Effort



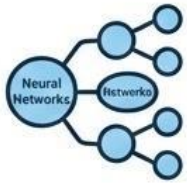
Key Innovation: Uses Distant Supervision to generate training data automatically, avoiding manual annotation.

Applications of Phrase Mining

Turning Text into Knowledge

Latent Keyphrase Inference

Automatically extract keyphrases to improve document indexing.



Topic Exploration

Use phrases ("Neural Networks") instead of single words ("Networks") for clearer topic models.

Knowledge Base Construction

Extract concept phrases to build ontologies and knowledge graphs.

Research Frontier Analysis

Track terminology evolution in scientific literature.

Practical Tools: R Package 'phm'

Small & Cabrera (2025)

Purpose & Performance

Extract commonly occurring principal phrases from text collections.

Core functions rewritten in C++ for speed.

Key Functions

- **phraseDoc()**: Create phrase-document matrix.
- **freqPhrases()**: Display most frequent phrases.
- **getDocs()**: Find documents containing specific phrases.
- **textDist()**: Calculate distance between texts.

Example Usage (R)

```
tst = c("This is a test text", "This is another test text")
pd = phraseDoc(tst)
freqPhrases(pd, 2) # Shows top 2 frequent phrases
```

Mining Copy and Paste Bugs

Section 5.6.2: The Software Engineering Challenge

The Problem

Copy-pasted code is common but prone to introducing bugs (e.g., forgetting to rename a variable in the pasted block).

Scale of the Problem

- **Linux Kernel**: ~190,000 copy-pasted segments.
- **FreeBSD**: ~150,000 copy-pasted segments.

Insight

A significant portion of operating system bugs concentrate in these Clone regions.

Slide 1: CP-Miner: A Data Mining Approach
Section 5.6.2

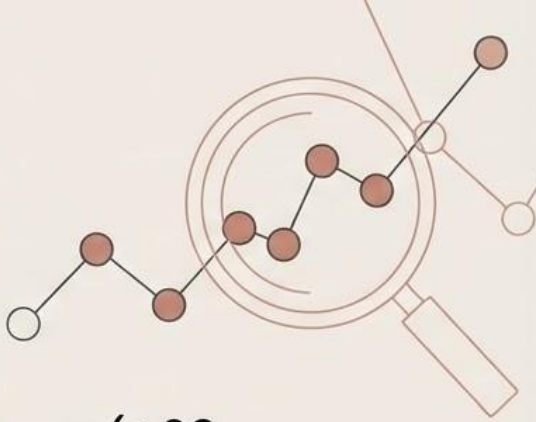
MINING BUGS IN OPERATING SYSTEMS

Li, Lu, Myagmar, and Zhou
(USENIX OSDI 2004)

Context: Applying sequential pattern mining to find copy-paste errors in massive codebases.

Slide 2: Background & Capabilities

WHY CP-MINER?



Key Capabilities:

- **Efficiency**: Identifies copy-pasted code in large software (< 20 minutes for Linux/FreeBSD).
- **Bug Detection**: Detects copy-paste related bugs that other tools miss.
- **Robustness**: Handles small modifications (gaps/insertions), not just exact copies.

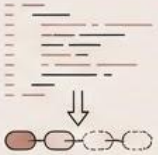
Performance Achievements:

- **Linux**: Analyzed 190,000 segments in < 20 mins -> Found 28 Bugs.
- **FreeBSD**: Analyzed 150,000 segments in < 20 mins -> Found 23 Bugs.

Slide 3: How CP-Miner Works

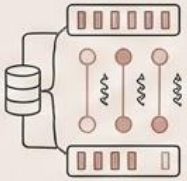
THE MINING PIPELINE

Token Sequence Extraction



- Convert source code to token sequences.
- **Normalize** identifiers (ignore specific variable names like x vs y).

Frequent Subsequence Mining



- Apply sequential pattern mining (similar to PrefixSpan).
- Find long, frequent token sequences that indicate copied code.

Copy-Paste Detection

- Identify segments with high similarity.
- Handle modifications by allowing gaps and substitutions.

Slide 4: Bug Patterns Detected

WHEN COPY-PASTE GOES WRONG

Type 1: Inconsistent Changes

Original: if (x > 0) {
  do_a(); do_b();
}
Copied: if (y > 0) {
  do_a(); do_b();
}
(Forgot to change do_a for variable y)

Type 2: Missing Updates

Variable names or function calls not updated to the new context.

Type 3: Improper Modifications

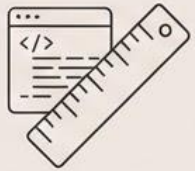
Partial updates create inconsistent behavior (e.g., updating the if condition but not the loop body).

Slide 5: Analysis of Copy-Paste Characteristics

UNDERSTANDING THE ECOSYSTEM

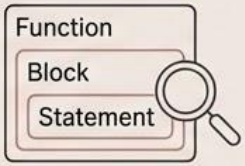
CP-Miner enabled the first large-scale analysis of:

Length



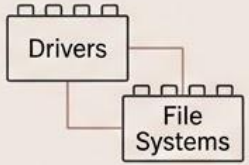
How long are copied segments?

Granularity



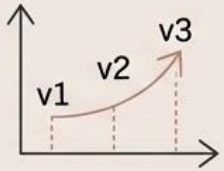
Function-level vs. Block-level vs. Statement-level.

Modules



Which subsystems (e.g., Drivers vs. File Systems) have the most redundancy?

Evolution

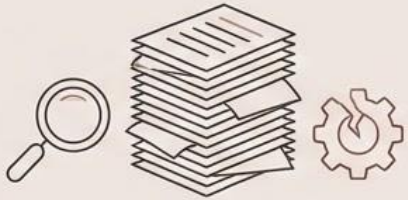


How does copy-paste evolve across software versions?

Slide 6: Real-World Impact

BEFORE VS. AFTER

Before CP-Miner:



- Tools couldn’t scale to Linux-sized kernels.
- Manual review missed subtle copy-paste bugs.
- OS shipped with known but undetected bugs.

After CP-Miner:

- 28+ Bugs Fixed in Linux.
- 23+ Bugs Fixed in FreeBSD.
- Established ‘Data Mining’ as a viable approach for Software Engineering (mining software repositories).

Slide 7: Tool Comparison

Choosing the Right Detector

CP-Miner	CCFinder	DECKARD	Sim
Approach Sequential Pattern Mining	**Approach** Token-based Detection	**Approach** Abstract Syntax Tree (AST)	**Approach** Similarity Scoring
Strengths Scalable, Handles modifications	**Strengths** Language independent	**Strengths** Finds structural clones	**Strengths** Fast for small codebases

Slide 8: Connection to Pattern Mining Concepts

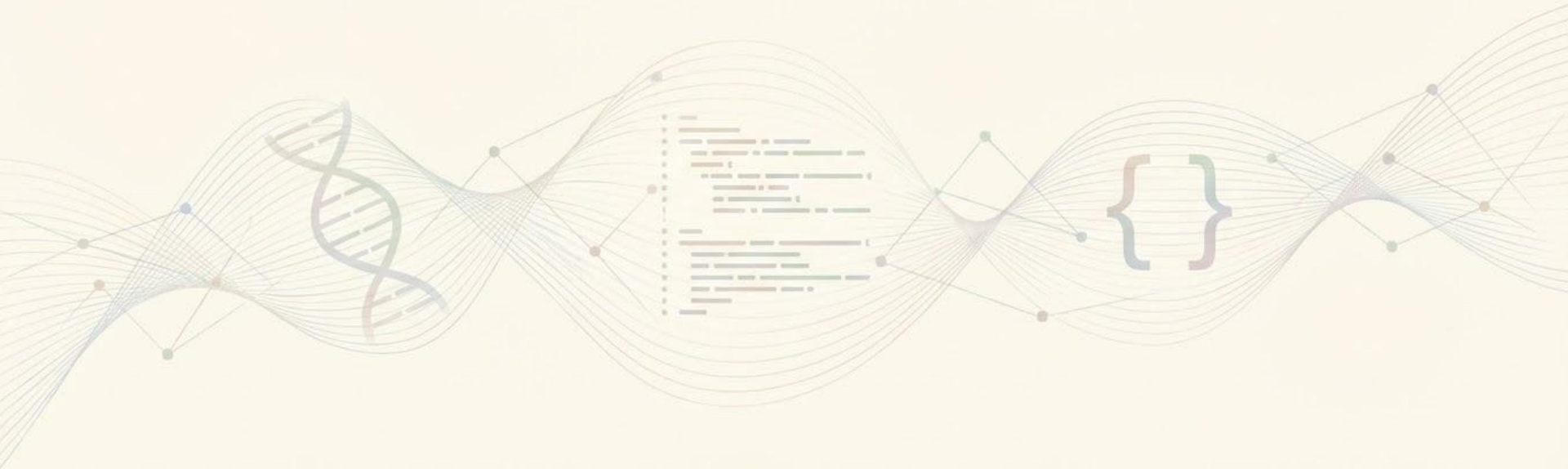
Translating Terms

CP-Miner Component	Pattern Mining Concept
Token Sequences	Transaction Database
Frequent Subsequences	Sequential Patterns
Similarity Threshold	Minimum Support
Modification Handling	Gap Constraints
Bug Detection	Post-processing + Constraints

The Power of Pattern Mining

Unifying Concepts across Biology, Text, and Code

Final synthesis of Real-World Applications.



Slide 2: Unified Framework for Applications

Connecting the Dots

Application	Pattern Type	Mining Method	Key Constraint
Protein Variant Analysis	Spatial Clusters	Ripley's K + Permutation	Distance Thresholds
Phrase Mining	Sequential Word Patterns	Frequent Pattern + POS	Linguistic Validity
Copy-Paste Bug Detection	Code Token Sequences	Sequential Pattern Mining	Similarity Thresholds

Slide 3: Real-World Impact

Measuring Success

Bioinformatics

PathProx classifies disease variants using 3D spatial patterns.

- **Result:** Better diagnosis of genetic diseases.

Text Analytics

Phrase Mining enables better topic models and knowledge bases.

- **Result:** Smarter search engines and AI assistants.

Software Engineering

CP-Miner found 50+ bugs in Linux and FreeBSD.

- **Result:** More stable operating systems.

Slide 4: Key Insights (1/2)

Scale & Domain

Scalability

Pattern mining scales to real-world problems (e.g., millions of genetic variants, 190,000 code segments).

Constraints are Essential

Without domain constraints, results are noise.

- **Biology:** Must account for 3D space.
- **Text:** Must account for Linguistic Rules.
- **Code:** Must account for Variable Renaming.

Slide 5: Key Insights (2/2)

Data Types & Performance

3. Different Data = Different Methods:

- **Biological Data**: Spatial Statistics.
- **Text Data**: Phrase Frequency + NLP features.
- **Code Data**: Token Sequences + Tolerance for modification.

4. Performance Matters:

- **CP-Miner**: Processes Linux Kernel in < 20 minutes.
- **Phrase Mining**: Tools optimized in C++ for massive corpora.

Slide 6: Future Directions

Where We Go From Here

Key Areas for Future Mining

Multi-Modal Mining: Combining text, code, and structural data (e.g., mining comments and code together).

Real-Time Mining: Streaming applications for instant insights.

Deep Learning Integration: Using Neural Networks to enhance pattern extraction accuracy.

Interactive Mining: Keeping the Human-in-the-loop to refine constraints dynamically.