Depth-first Traversal over a Mirrored Space for Non-redundant Discriminative Itemsets

Yoshitaka Kameya and Hiroki Asaoka
Meijo University

Outline

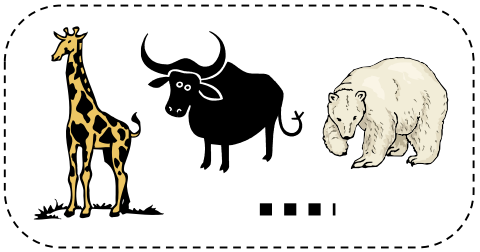
- Background
- Details of our proposed method
- Experiments

Outline

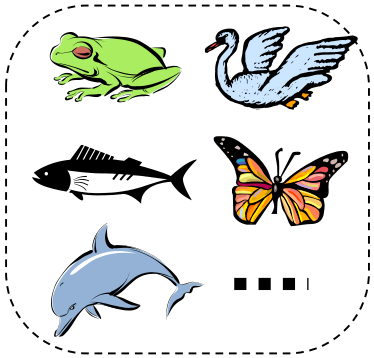
- Background
- Details of our proposed method
- Experiments

Background: Discriminative patterns

- Discriminative patterns:
  - Show differences between two groups (classes)
  - Used for:
    - Characterizing the class of interest
    - Building more precise classifiers



+:Positive class



−:Negative class

Discriminative pattern $\boldsymbol{X}$

milk=True $\wedge$ aquatic=False

➔ +

Class c of interest

- We focus on **top-k** mining

Background: Coping with redundancy (1)

Problem: Redundancy among patterns

Item i is significantly relevant to the target class

→ Patterns containing i tend to occupy the top-k list

Dataset

	Class	Transaction
Positive Transactions	+	{A, B, D, E}
	+	{A, B, C, D, E}
	+	{A, C, D, E}
	+	{A, B, C}
	+	{B}
Negative Transactions	−	{A, B, D, E}
	−	{B, C, D}
	−	{A, D, E}
	−	{B, D, E}
	−	{C}

Top-10 patterns (including ties)

Rank	Pattern	Positive Support	F-score
1	{A, C}	3	0.75
2	{A}	4	0.73
3	{B}	4	0.67
3	{A, B}	3	0.67
5	{A, D}	3	0.6
5	{A, E}	3	0.6
5	{A, E, D}	3	0.6
5	{C}	3	0.6
9	{A, B, C}	2	0.57
9	{A, C, D}	2	0.57
9	{A, C, E}	2	0.57
9	{A, C, E, D}	2	0.57
9	{C, E}	2	0.57
9	{C, E, D}	2	0.57

Support over the positive transactions

Relevance score to the positive class

Background: Coping with redundancy (2)

- Set-inclusion-based constraints
  - Closedness [Pasquier+ 99]
  - Productivity [Bayardo 00][Webb 07]

Class	Transaction
+	{A, B, D, E}
+	{A, B, C, D, E}
+	{A, C, D, E}
+	{A, B, C}
+	{B}
–	{A, B, D, E}
–	{B, C, D}
–	{A, D, E}
–	{B, D, E}
–	{C}

without closedness or productivity

Rank	Pattern	Positive Support	F-score
1	{A, C}	3	0.75
2	{A}	4	0.73
3	{B}	4	0.67
5	{A, E, D}	3	0.6
5	{C}	3	0.6
9	{A, B, C}	2	0.57
9	{A, C, D}	2	0.57
9	{A, C, E}	2	0.57
9	{A, C, E, D}	2	0.57
9	{A, B, C, E}	2	0.57
9	{A, B, C, D}	2	0.57

Closedness:
With the same positive support, pick the super-pattern

only with closedness

with closedness & productivity

Rank	Pattern	Positive Support	F-score
1	{A, C}	3	0.75
2	{A}	4	0.73
3	{B}	4	0.67

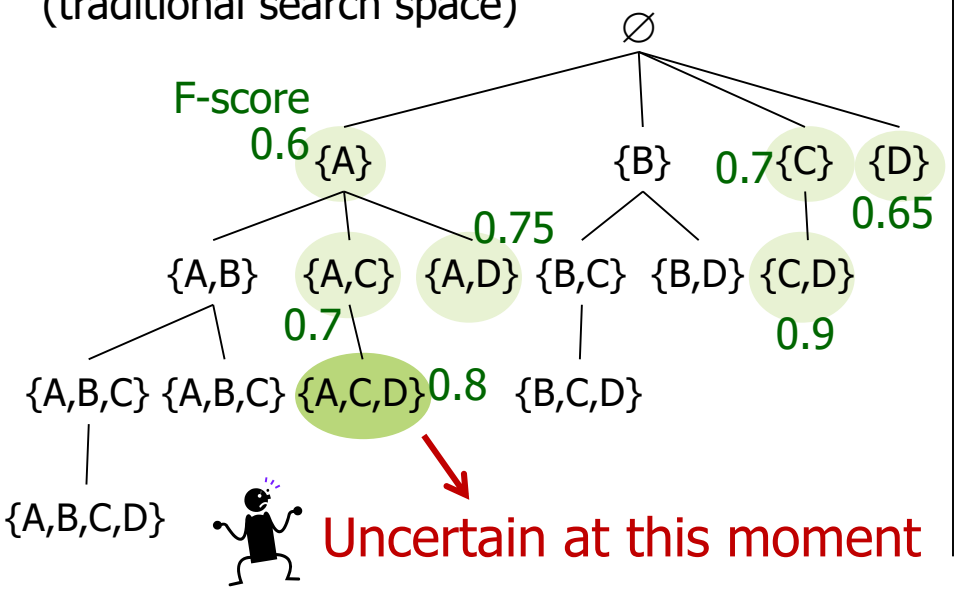
Rank	Pattern	Positive Support	F-score
1	{A, C}	3	0.75
2	{A}	4	0.73
3	{B}	4	0.67
3	{A, B}	3	0.67
5	{A, E, D}	3	0.6
6	{		
6	{		
8	{		
9	{A, B, C, E, D}	1	0.33

Productivity:
Remove super-patterns with smaller relevance scores

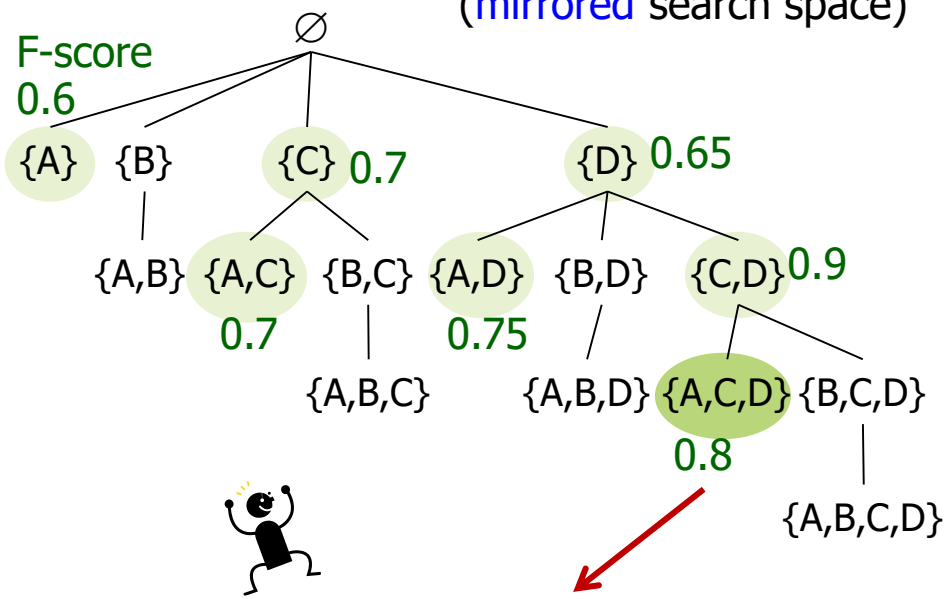
Background: Suffix enumeration trees

- We test:
  - Closedness by "on-the-fly" closure check
  - Productivity over suffix enumeration trees [Kameya+ SDM12]

Prefix enumeration tree
(traditional search space)



Suffix enumeration tree
(mirrored search space)



Memory-efficient (depth-first) search is possible with safe productivity tests

Our goal

- To propose an efficient, exact method for finding top-k productive "closed-on-the-positives"

  Closed patterns over the positive transactions

- **Contributions**:
  - Dual-monotonicy
    - A generalized condition on relevance scores
    - Gives a theoretical basis
  - Suffix-preserving closure extension
    - A mirrored operation of the one used in LCM [Uno+ DS04]
    - Can work with closedness and productivity smoothly at the same time

Outline

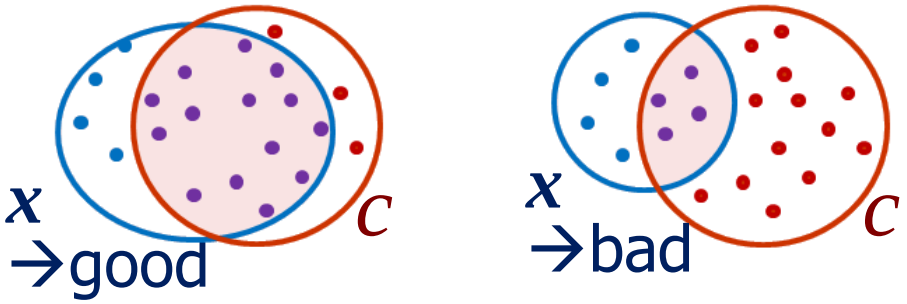
- ✓ Background
- Details of our proposed method
  - Dual-monotonicity
  - Suffix-preserving closure extension
- Experiments

Outline

- ✓ Background
- Details of our proposed method
  - Dual-monotonicity
  - Suffix-preserving closure extension
- Experiments

Dual-monotonicity: Preliminaries (1)

- Discriminative pattern x is often evaluated under a relevance score to the class c of interest
  - Confidence/PMI
  - Support Difference/WRA/Leverage
  - χ^2
  - F-score/Dice/Jaccard
  - …



These scores measure the distributional overlap between x and c

- Computational difficulty:
  Most of popular relevance scores do *not* satisfy anti-monotonicity (the Apriori property)

  → Standard technique: Branch-and-bound search
  [Morishita+ 00][Zimmermann+ 09][Nijssen+ 09]

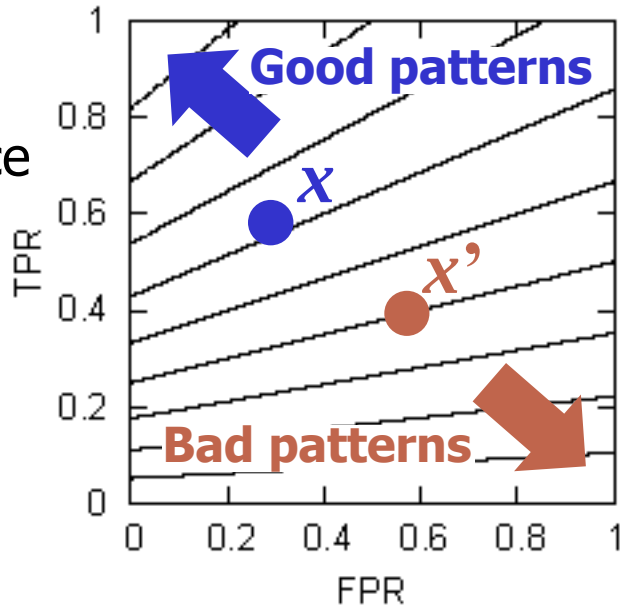
Dual-monotonicity: Preliminaries (2)

- ROC analysis of a relevance score R_c
  - Confusion matrix for a rule "$\boldsymbol{x} \rightarrow c$":

	c	$\neg c$
$\boldsymbol{x}$	True positive: $p(c, \boldsymbol{x})$	False positive: $p(\neg c, \boldsymbol{x})$
$\neg \boldsymbol{x}$	False negative: $p(c, \neg \boldsymbol{x})$	True negative: $p(\neg c, \neg \boldsymbol{x})$

  - Any relevance score R_c can be seen as a function of true positive rate (TPR) $p(\boldsymbol{x} \mid c)$ and false positive rate (FPR) $p(\boldsymbol{x} \mid \neg c)$

F-score's ROC space

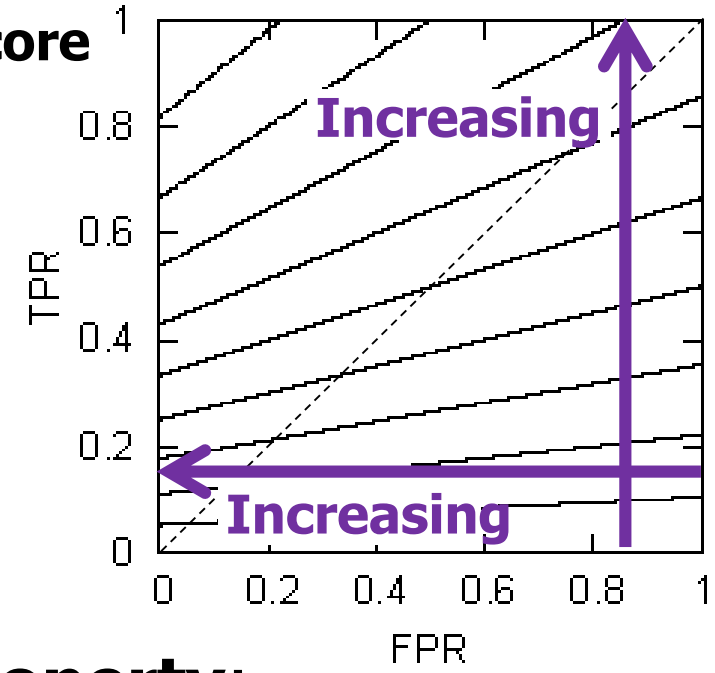


Dual-monotonicity: Definition

Relevance score R_c is dual-monotonic

$\Leftrightarrow$ $R_c(\boldsymbol{x})$ is monotonically increasing w.r.t. TPR $p(\boldsymbol{x} \mid c)$ and $R_c(\boldsymbol{x})$ is monotonically decreasing w.r.t. FPR $p(\boldsymbol{x} \mid \neg c)$ (wherever TPR $\geq$ FPR)

F-score



Dual-monotonicity is more general than convexity [Morishita+ 00][Nijssen+ 09]

(e.g. F-score does not satisfy convexity but dual-monotonicity)

Property:

Branch-and-bound (B&B) pruning is safe under dual-monotonicity

→ The applicablility of B&B pruning is enlarged

Dual-monotonicity: Closed patterns

- We focus only on "closed-on-the-positives"
  - Closed patterns over the positive transactions
- Such closed patterns are beneficial in:
  - **Efficiency**:
    - Some set of patterns ("generators") are compressed into a closed pattern
    - Search space is (possibly exponentially) reduced
  - **Relevance**:
    Under a dual-monotonic score, closed-on-the-positives are no less relevant than their generators [Soulet+ PAKDD04]

Pattern	Positive Support	F-score	Closed on the positives?
{A, C}	3	0.75	Yes
{A}	4	0.73	Yes
{B}	4	0.67	Yes
{A, B}	3	0.67	Yes
{A, D}	3	0.6	No
{A, E}	3	0.6	No
{A, E, D}	3	0.6	Yes
{C}	3	0.6	No
{A, B, C}	2	0.57	Yes
{A, C, D}	2	0.57	No
{A, C, E}	2	0.57	No
{A, C, E, D}	2	0.57	Yes
{C, E}	2	0.57	No
{C, E, D}	2	0.57	No

Outline

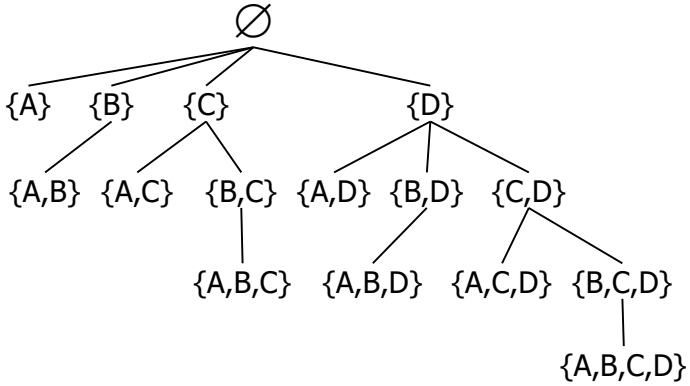
- ✓ Background
- Details of our proposed method
  - ✓ Dual-monotonicity
  - – Suffix-preserving closure extension
- Experiments

SPC extension: Background

- Suffix-preserving closure (SPC) extension
  - A mirrored operation of the one used in LCM [Uno+ DS04]
  - Only generates closed patterns from closed patterns

    → We need not maintain the top-k list for closedness
  - Ensures the depth-first traversal over a space like a suffix enumeration tree

    → This makes it easy to maintain the top-k list for productivity



SPC extension: Illustrated example (1)

Preparation:

Get the item order and reorder items in the transactions

Original dataset:

Class	Transaction
+	{A, B, D, E}
+	{A, B, C, D, E}
+	{A, C, D, E}
+	{A, B, C}
+	{B}
–	{A, B, D, E}
–	{B, C, D}
–	{A, D, E}
–	{B, D, E}
–	{C}

Item	F-score
A	0.78
B	0.63
C	0.57
D	0.46
E	0.51

Item order:

(young) A < B < C < E < D (old)

Modified dataset:

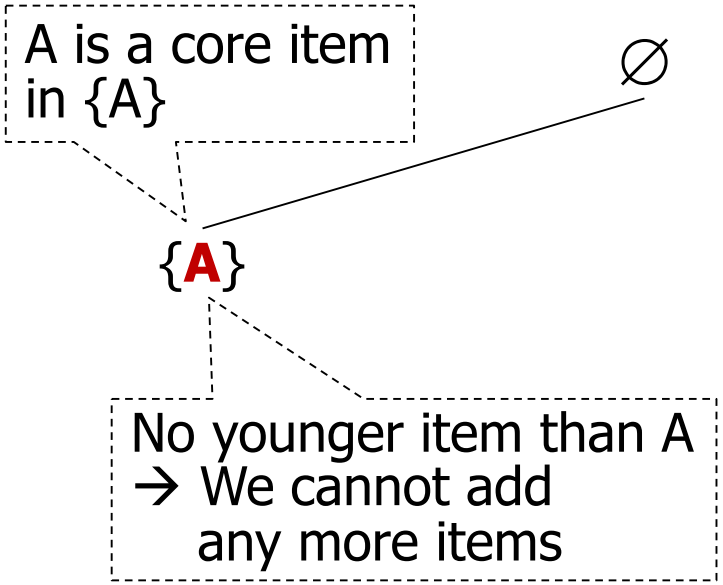
Class	Transaction
+	{A, B, E, D}
+	{A, B, C, E, D}
+	{A, C, E, D}
+	{A, B, C}
+	{B}
–	{A, B, E, D}
–	{B, C, D}
–	{A, E, D}
–	{B, E, D}
–	{C}

We use negative transactions only when computing relevance scores (Details are omitted)

SPC extension: Illustrated example (2)

Iteration:

Add *younger* items one by one to the parent pattern
(such added items are called "core items")



Item order:

A < B < C < E < D
(young) (old)

The intersection of the transactions including {A} is {A}

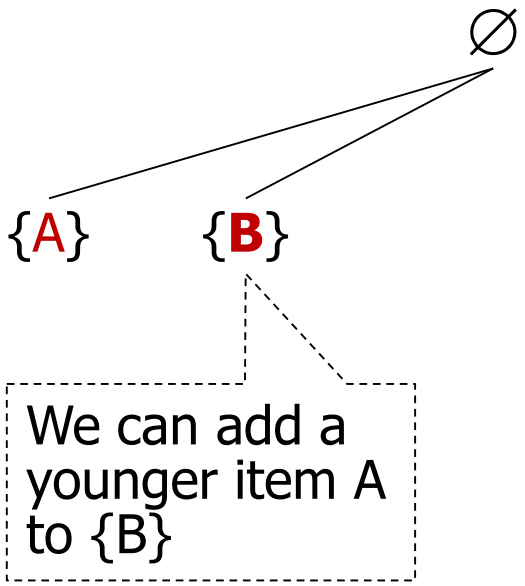
→ {A} is closed on the positives

Class	Transaction
+	{**A**,B,E,D}
+	{**A**,B,C,E,D}
+	{**A**,C,E,D}
+	{**A**,B,C}
+	{B}
–	{A,B,E,D}
–	{B,C,D}
–	{A,E,D}
–	{B,E,D}
–	{C}

SPC extension: Illustrated example (3)

Iteration:

Add *younger* items one by one to the parent pattern
(such added items are called "core items")



Item order:

A < B < C < E < D
(young) (old)

The intersection of the transactions including {B} is {B}

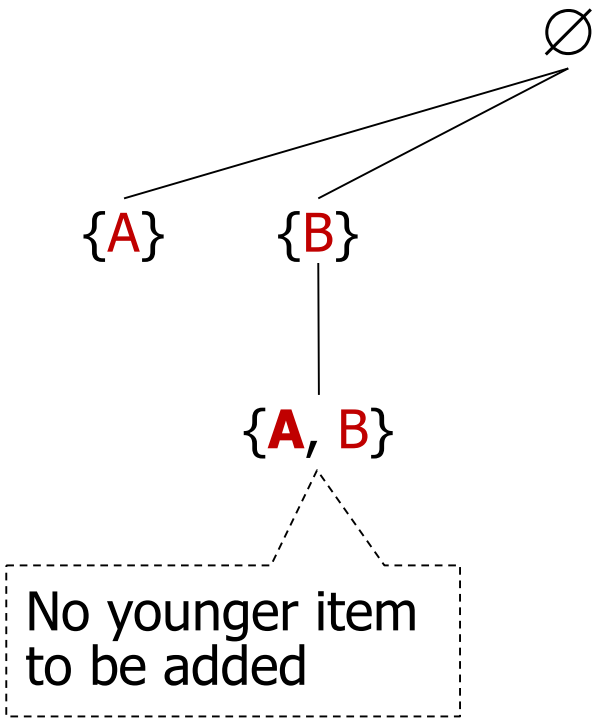
→ {B} is closed on the positives

Class	Transaction
+	{A,**B**,E,D}
+	{A,**B**,C,E,D}
+	{A,C,E,D}
+	{A,**B**,C}
+	{**B**}
–	{A,B,E,D}
–	{B,C,D}
–	{A,E,D}
–	{B,E,D}
–	{C}

SPC extension: Illustrated example (4)

Iteration:

Add *younger* items one by one to the parent pattern (such added items are called "core items")



Item order:

A < B < C < E < D

(young) (old)

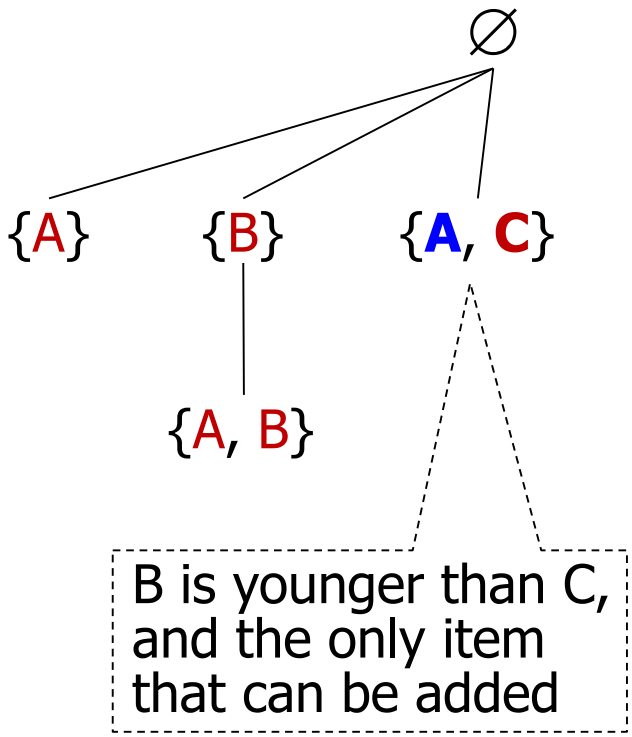
{A, B} is closed on the positives

Class	Transaction
+	{A,B,E,D}
+	{A,B,C,E,D}
+	{A,C,E,D}
+	{A,B,C}
+	{B}
–	{A,B,E,D}
–	{B,C,D}
–	{A,E,D}
–	{B,E,D}
–	{C}

SPC extension: Illustrated example (5)

Iteration:

Add *younger* items one by one to the parent pattern (such added items are called "core items")



A always accompanies the added item C
→ {A, C} is closed on the positives

Item order:

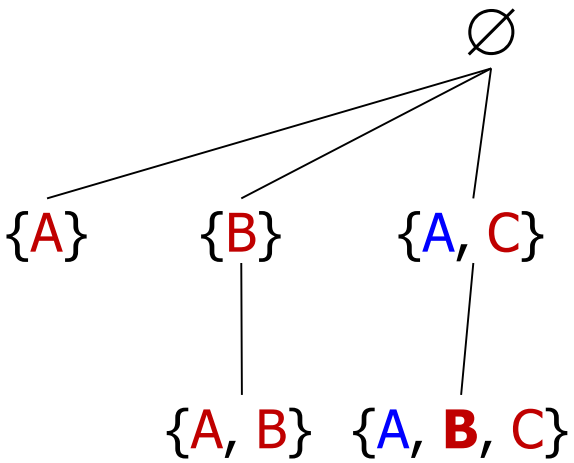
A < B < C < E < D
(young) (old)

Class	Transaction
+	{A,B,E,D}
+	{**A**,B,**C**,E,D}
+	{**A**,**C**,E,D}
+	{**A**,B,**C**}
+	{B}
–	{A,B,E,D}
–	{B,C,D}
–	{A,E,D}
–	{B,E,D}
–	{C}

SPC extension: Illustrated example (6)

Iteration:

Add *younger* items one by one to the parent pattern (such added items are called "core items")



Item order:

A < B < C < E < D

(young) (old)

{A, B, C} is closed on the positives

Class	Transaction
+	{A,B,E,D}
+	{A,**B**,C,E,D}
+	{A,C,E,D}
+	{A,**B**,C}
+	{B}
–	{A,B,E,D}
–	{B,C,D}
–	{A,E,D}
–	{B,E,D}
–	{C}

SPC extension: Illustrated example (7)

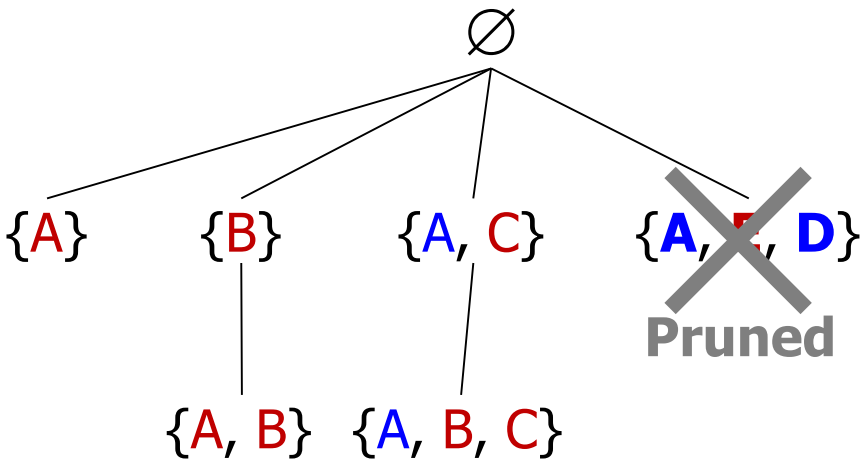
Iteration:

Add *younger* items one by one to the parent pattern (such added items are called "core items")

Item order:

A < B < C < E < D

(young) (old)



Class	Transaction
+	{**A**,B,**E**,**D**}
+	{**A**,B,C,**E**,**D**}
+	{**A**,C,**E**,**D**}
+	{A,B,C}
+	{B}
–	{A,B,E,D}
–	{B,C,D}
–	{A,E,D}
–	{B,E,D}
–	{C}

Older item D accompanies the added item E

→ Such extension is *not* an SPC extension!

SPC extension: Illustrated example (8)

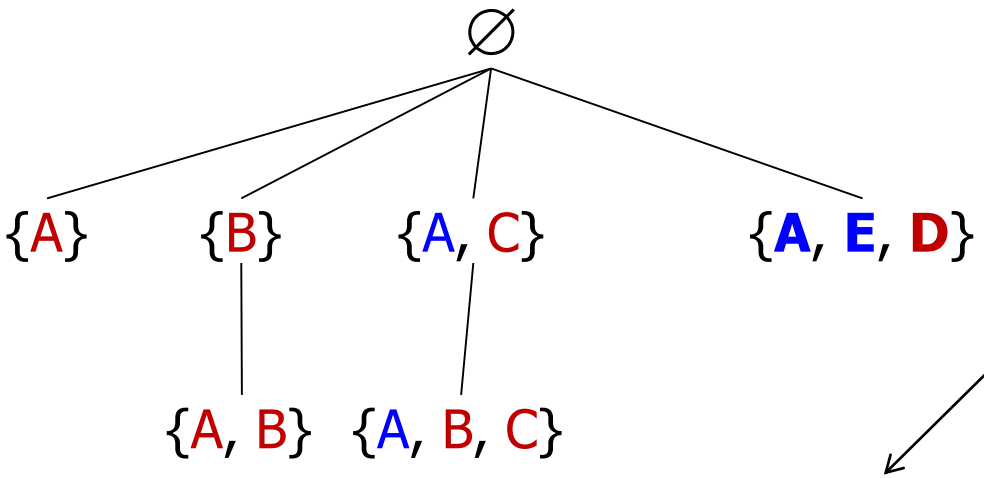
Iteration:

Add *younger* items one by one to the parent pattern (such added items are called "core items")

Item order:

A < B < C < E < D

(young) (old)



Class	Transaction
+	{**A**,B,**E**,**D**}
+	{**A**,B,C,**E**,**D**}
+	{**A**,C,**E**,**D**}
+	{A,B,C}
+	{B}
–	{A,B,E,D}
–	{B,C,D}
–	{A,E,D}
–	{B,E,D}
–	{C}

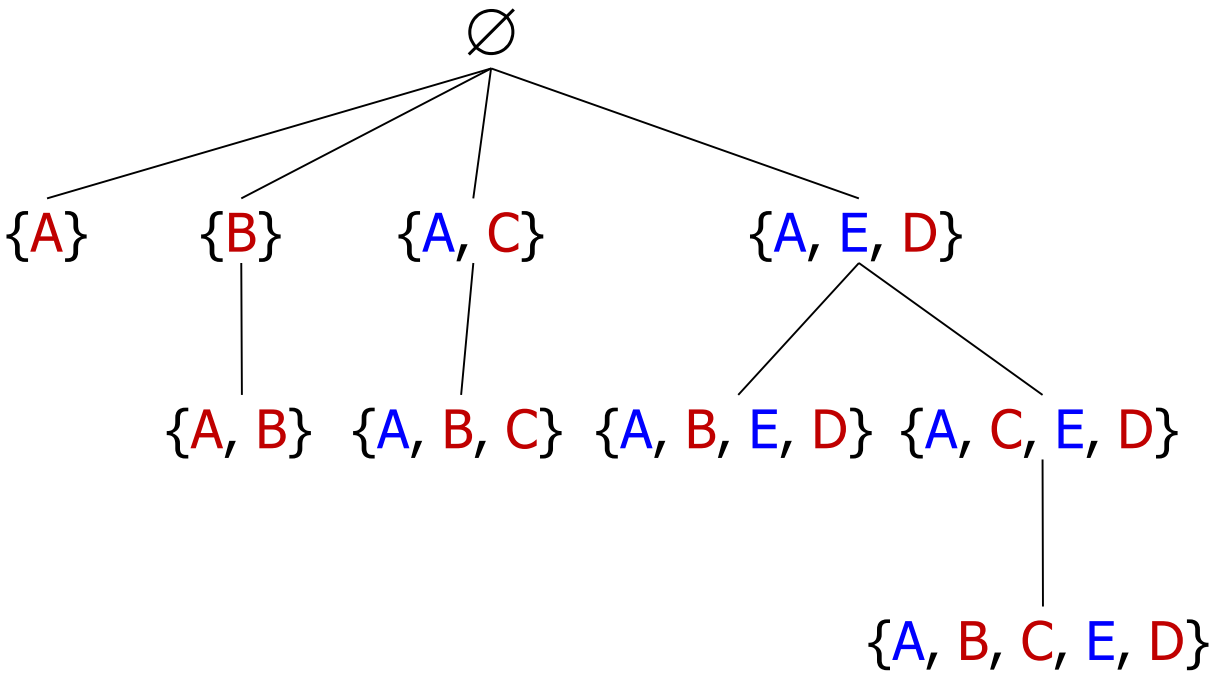
Accompanying items are all younger than the added item D

→ Such extension is an SPC extension

SPC extension: Illustrated example (9)

Iteration:

Add *younger* items one by one to the parent pattern (such added items are called "core items")



Item order:

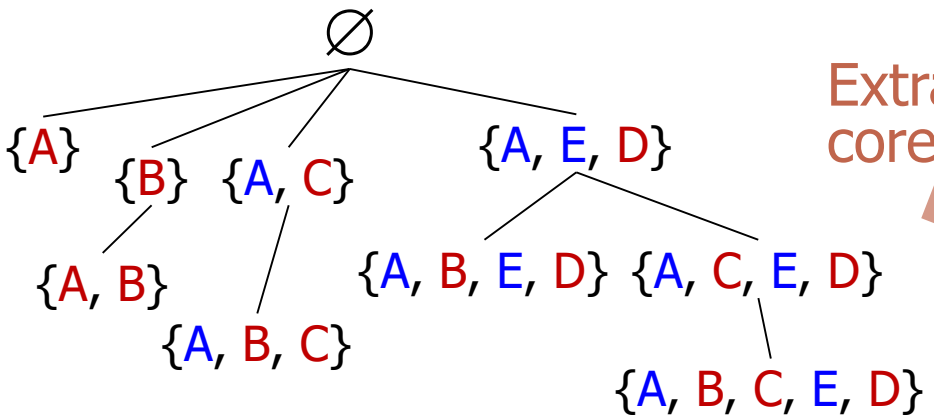
A < B < C < E < D

(young) (old)

Class	Transaction
+	{A,B,E,D}
+	{A,B,C,E,D}
+	{A,C,E,D}
+	{A,B,C}
+	{B}
–	{A,B,E,D}
–	{B,C,D}
–	{A,E,D}
–	{B,E,D}
–	{C}

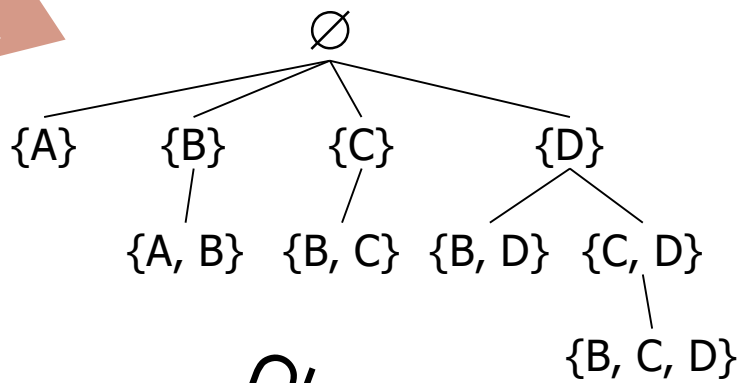
SPC extension: Justification

SPC enumeration tree: Search tree formed by SPC extension



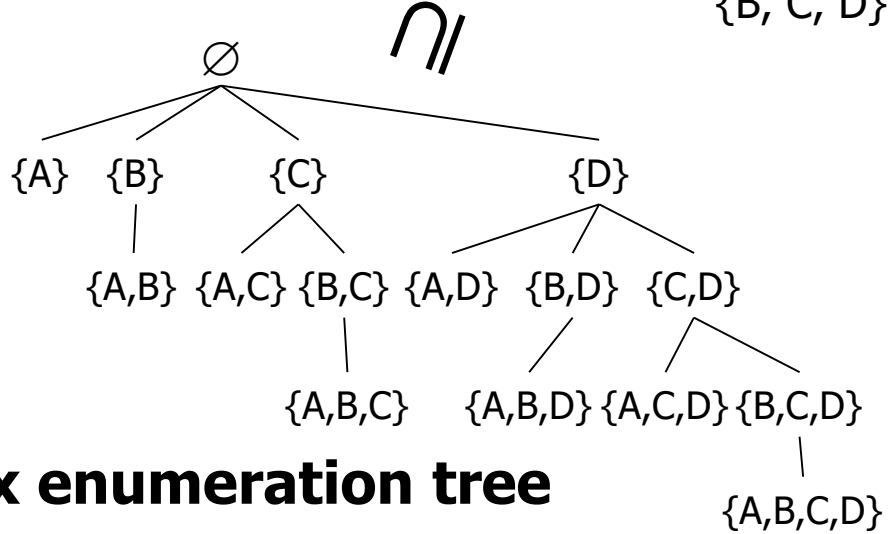
Extract core items

Core enumeration tree



We can inherit
the property on visiting order
from suffix enumeration trees

→ Safe productivity tests



Suffix enumeration tree

Outline

- ✓ Background
- ✓ Details of our proposed method
  - ✓ Dual-monotonicity
  - ✓ Suffix-preserving closure extension
- Experiments

Experiments: Settings

- 16 datasets from UCI ML Repository
- We used a preprocessed version available from http://dtai.cs.kuleuven.be/CP4IM/datasets/

Dataset	#Trans.	#Items
anneal	812	93
audiology	216	148
australian-credit	653	125
german-credit	1,000	112
heart-cleveland	296	95
hepatitis	137	68
hypothyroid	3,247	88
kr-vs-kp	3,196	73

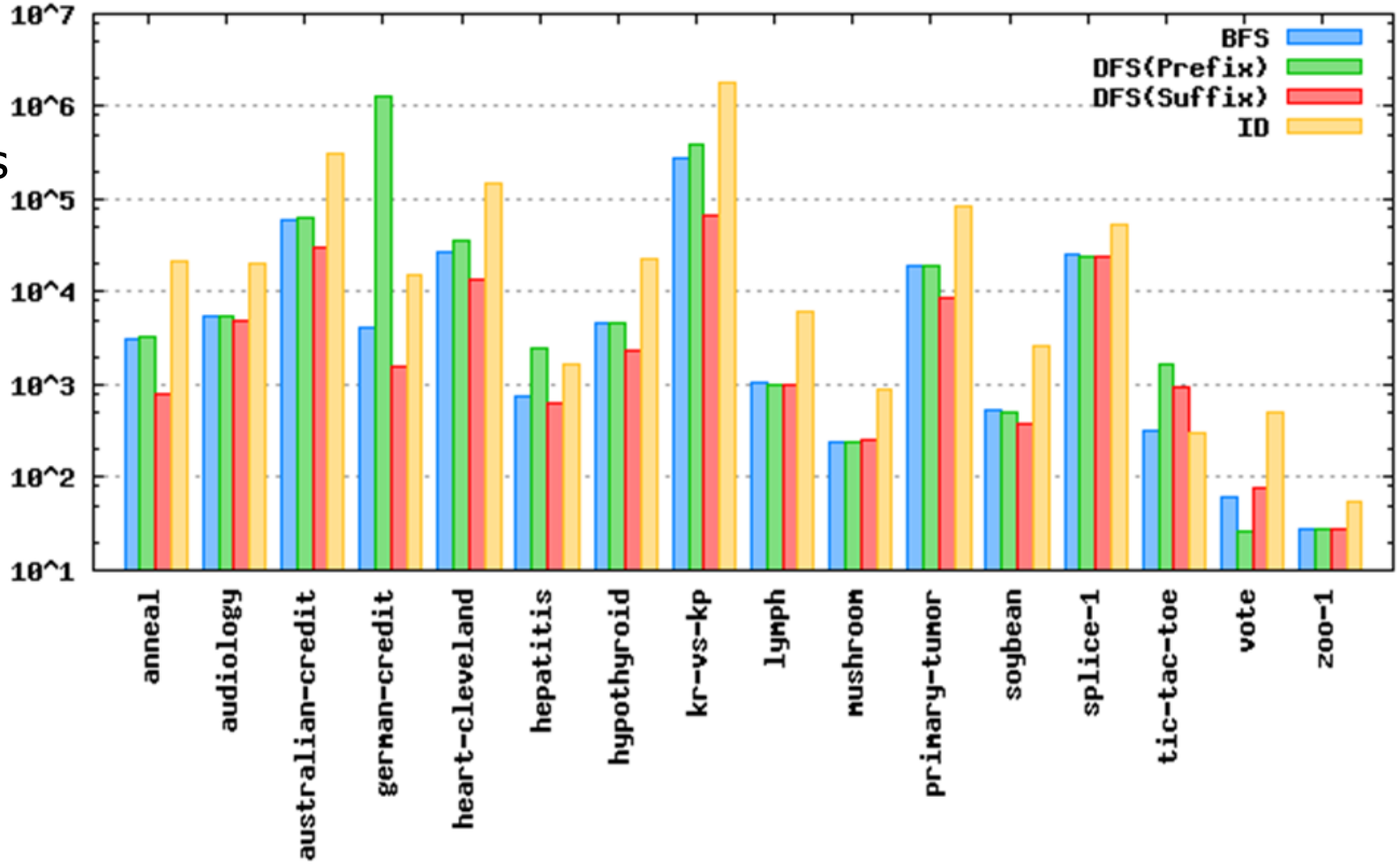
Dataset	#Trans.	Items
lymph	148	68
mushroom	8,124	110
primary-tumor	336	31
soybean	630	50
splice-1	3,190	287
tic-tac-toe	958	28
vote	435	48
zoo-1	101	36

- Comparison among four search strategies:
  - Breadth-first
  - Depth-first over *prefix* enumeration trees
  - Depth-first over *suffix* enumeration trees (Our proposal)
  - Iterative deepening [Grosskreutz+ ECML/PKDD11]

Experiments: Search Strategy

#visited patterns (in log scale)

More efficient ↓



- DFS (suffix) runs fast on average, compared to BFS and DFS (prefix)
- The overhead of iterative deepening is not ignorable

Summary

- We proposed an efficent and exact method for finding top-k productive "closed-on-the-positives"
  - Dual-monotonicity
  - Suffix-preserving closure extension
- Experimental results show the efficiency of the proposed method

Future work

- More sophisticated implementation (e.g. FP-trees)
- Extension to more complex patterns (e.g. sequences)