INDIAN INSTITUTE OF INFORMATION TECHNOLOGY

System Evolution Analytics based on Network Data Science

Dr. Animesh Chaturvedi

Assistant Professor: IIIT Dharwad

Young Researcher: Heidelberg Laureate Forum

Postdoc: King's College London & The Alan Turing Institute

PhD: IIT Indore MTech: IIITDM Jabalpur

HEIDELBERG LAUREATE FORUM

KING'S College LONDON

Indian Institute of Technology Indore
भारतीय प्रौद्योगिकी संस्थान इंदौर

The Alan Turing Institute

PDPM
Indian Institute of Information Technology, Design and Manufacturing, Jabalpur

Network Data Science Research

System Evolution Analytics of a State Series

1. **System Evolution Analytics of a State Series**
2. System Network Analytics: Stable Network Evolution Rules of a State Series
3. Network Evolution Subgraph Mining for System Stability, Changeability, and Complexity
4. System Neural Network for Evolution Learning

Evolving System Examples

Apache Hadoop HDFS

Apache Hadoop HDFS

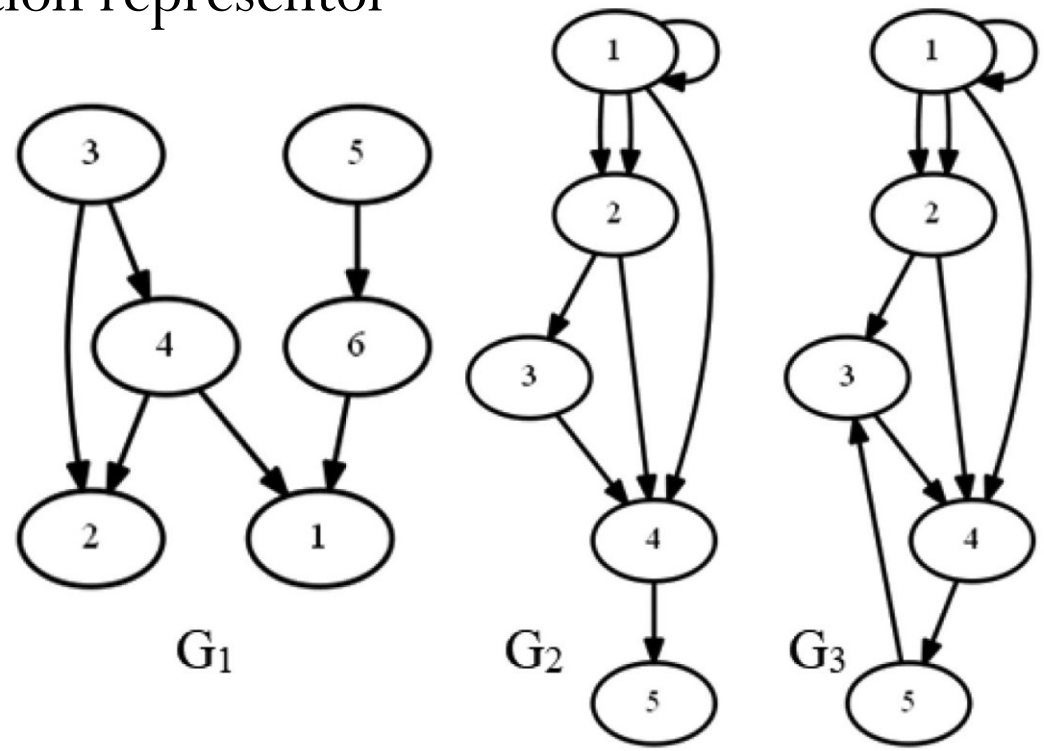
License	Apache 2.0
Categories	Distributed File Systems
Tags	file-system, distributed, hadoop, apache, storage
Ranking	**#358 in MvnRepository** (See Top Artifacts) **#1 in** Distributed File Systems
Used By	**1,169 artifacts**

	Version	Vulnerabilities	Repository	Usages	Date
3.3.x	3.3.4		Central	36	Aug 05, 2022
	3.3.3		Central	53	May 17, 2022
	3.3.2		Central	47	Mar 03, 2022
	3.3.1		Central	70	Jun 15, 2021
	3.3.0		Central	63	Jul 15, 2020
3.2.x	3.2.4		Central	31	Jul 22, 2022
	3.2.3		Central	31	Mar 28, 2022
	3.2.2		Central	43	Jan 09, 2021
	3.2.1		Central	78	Sep 10, 2019
	3.2.0		Central	52	Jan 21, 2019

https://mvnrepository.com/artifact/org.apache.hadoop/hadoop-hdfs

Defining State series

- We define (S_i, ER_i, t_i) for an evolving system
  - represented as a **State Series (SS)**, a collection of states (or data points)
  - $SS = \{S_1, S_2 \ldots S_N\}$ at various time points $\{t_1, t_2, t_3 \ldots t_N\}$.
- Each state is represented as a series of evolution representor
  - $ER = \{ER_1, ER_2 \ldots ER_N\}$.
- Evolving state series:
  - "software version series",
  - "natural-language document versions"



Evolution of a system graph from G_1 to G_2 to G_3 when a system evolves from state S_1 to S_2 to S_3.

Problem definition

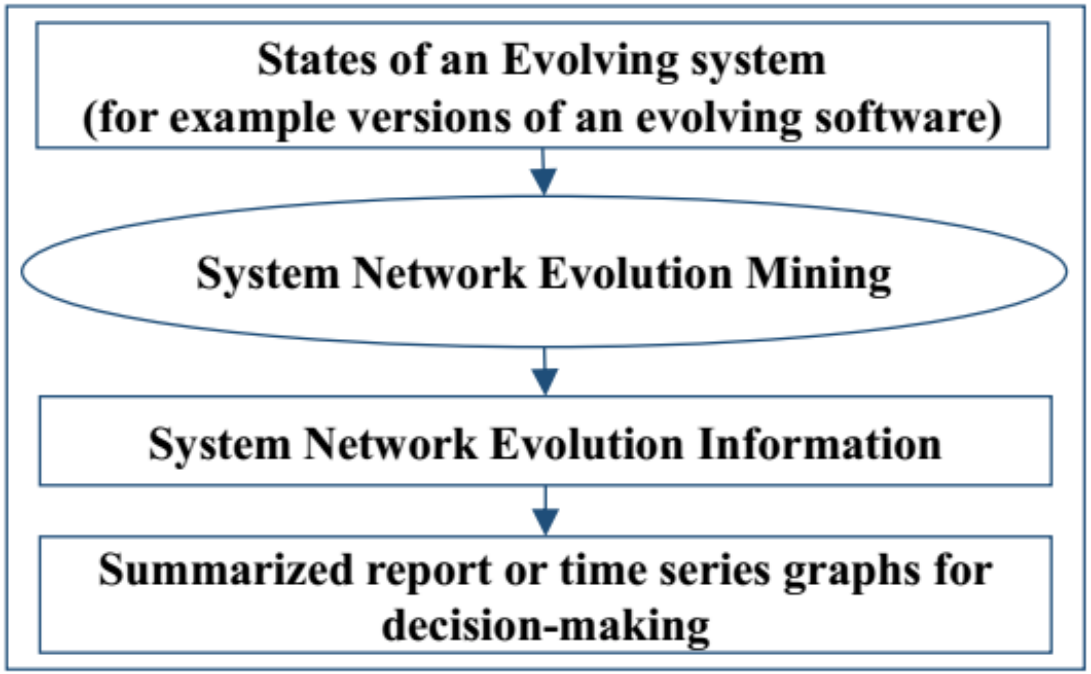
- Aim to study the change and evolution in an evolving system.

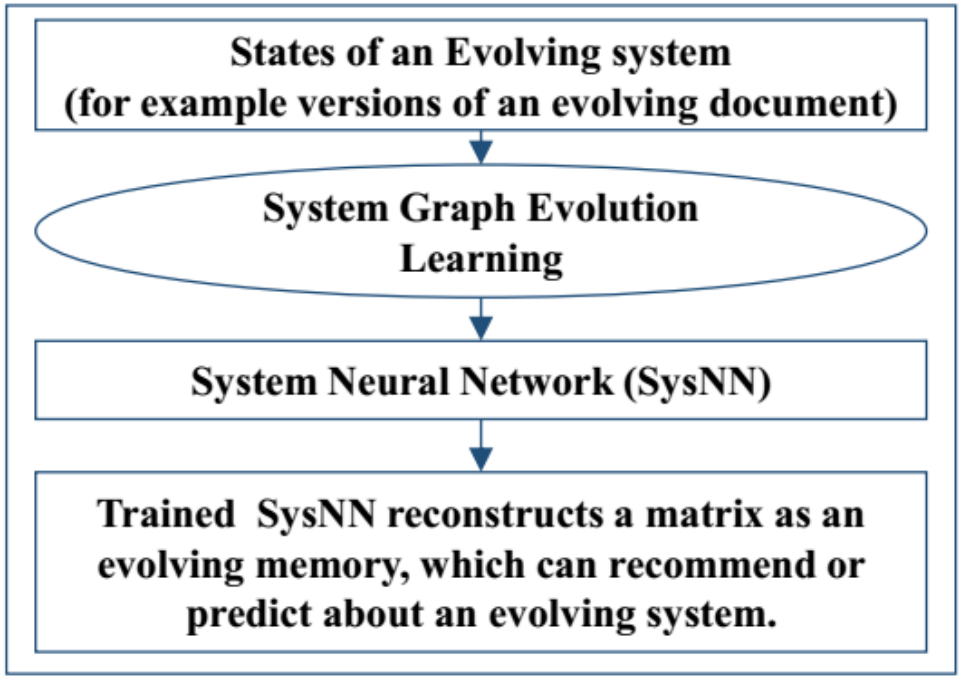
Two challenges:

1. Generally, evolving systems are complex in nature, thus pre-processing of an evolving system is a challenging task and requires a system domain expert.

2. The data analytics on a single state is a straightforward technique. However, it is challenging to process multiple states of an evolving system with the same technique.

System Evolution Analytics





Evolving Systems and their Networks

Domains of Evolving System	Evolving Systems	"Source" and "Target" Entities	Type of Connections	Type of network	Applications of System Evolution Analytics
Evolving Software System	Hadoop HDFS-Core[1]	"Caller" and "Callee" Procedures	Procedural calls	Call graph	Software Evolution Analytics
Evolving Natural-language systems	Bible Translation[2]	Words in "Source biblical language" and "English variant" languages	Translations	Words Network	Natural-language Evolution Analytics
	Multi-sport Events[3]	Words in "Titles" (name) and "Scopes" (region) of events	Regional names	Words network	
Evolving Retail Market System	Frequent Market Basket[4]	Words in "Product description" and Words in "Product description"	Purchases	Purchase network	Market Evolution Analytics
Evolving IMDb movie genre systems[5]	Positive sentiment[6] of movie genres[5]	"Positive words in names" and "genres" of movies	Sentiments	Positive sentiment network	Movie Evolution Analytics
	Negative sentiment[6] of movie genres[5]	"Negative words in names" and "genres" of movies	Sentiments	Negative sentiment network	

1. https://mvnrepository.com/artifact/org.apache.hadoop/hadoop-hdfs 2016.
2. https://en.wikipedia.org/wiki/List_of_English_Bible_translations Oct 2016.
3. https://en.wikipedia.org/wiki/List_of_multi-sport_events Oct 2016.
4. https://archive.ics.uci.edu/ml/datasets/Online+Retail Oct 2016.
5. http://www.imdb.com/interfaces/ Oct 2016.
6. https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html June 2017

Network Data Science Research

System Network Analytics: Stable Network Evolution Rules of a State Series

1. System Evolution Analytics of a State Series
2. **System Network Analytics: Stable Network Evolution Rules of a State Series**
3. Network Evolution Subgraph Mining for System Stability, Changeability, and Complexity
4. System Neural Network for Evolution Learning

Apriori algorithm: Association Rule Mining (1993)

- Let $I = \{i_1, i_2, \ldots . i_m\}$ be a set of literals, called items.
- *Support* of a rule $X \rightarrow Y$ is the percentage of transactions that contain both X and Y.
- *Confidence* of a rule is percentage the if-then statements ($X \rightarrow Y$) are found true
- Find all rules that satisfy a user-specified *minimum support* and *minimum confidence*

TID	Transaction Items
1	Bread, Jelly, PeanutButter
2	Bread, PeanutButter
3	Bread, Milk, PeanutButter
4	Beer, Bread
5	Beer, Milk

{Bread} → {PeanutButter} (Sup = 60%, Conf = 75%)

{PeanutButter} → {Bread} (Sup = 60%, Conf = 100%)

{Beer} → {Bread} (Sup = 20%, Conf = 50%)

{PeanutButter} → {Jelly} (Sup = 20%, Conf = 33.33%)

{Jelly} → {PeanutButter} (Sup = 20%, Conf = 100%)

{Jelly} → {Milk} (Sup = 0%, Conf = 0%)

Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. "Mining association rules between sets of items in large databases." SIGMOD. 1993.

Ramakrishnan Srikant, and Rakesh Agrawal. "Mining Generalized Association Rules." VLDB 1995.

Apriori algorithm: Association Rule Mining (1993)

- Let I = $\{i_1, i_2, \ldots . i_m\}$ be a set of literals, called items.
- *Support* of a rule X $\rightarrow$Y is the percentage of transactions that contain both X and Y.
- *Confidence* of a rule is percentage the if-then statements (X $\rightarrow$Y) are found true
- Find all rules that satisfy a user-specified *minimum support* and *minimum confidence*
  - 75% of transactions that purchase *PeanutButter* (antecedent) also purchase *Bread* (consequent). The number 75% is the confidence factor of the rule
    - [PeanutButter] $\rightarrow$ [Bread] (Sup = 60%, Conf = 75%)
  - 98% of customers who purchase *Tires* and *Auto accessories* also buy some *Automotive services*; here 98% is called the confidence of the rule.
    - [*Auto Accessories*], [*Tires*] $\rightarrow$ [*Automotive Services*] 98%

Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. "Mining association rules between sets of items in large databases." SIGMOD. 1993.
Ramakrishnan Srikant, and Rakesh Agrawal. "Mining Generalized Association Rules." VLDB 1995.

Evolution Rule and Stable Rule

Suppose a state series SS makes a database series $DS = \{D_1, D_2 \ldots D_N\}$, assume a rule ($X \rightarrow Y$) in database D_i for state S_i. The rule is *interesting* in D_i, if its support and confidence exceeds given thresholds.

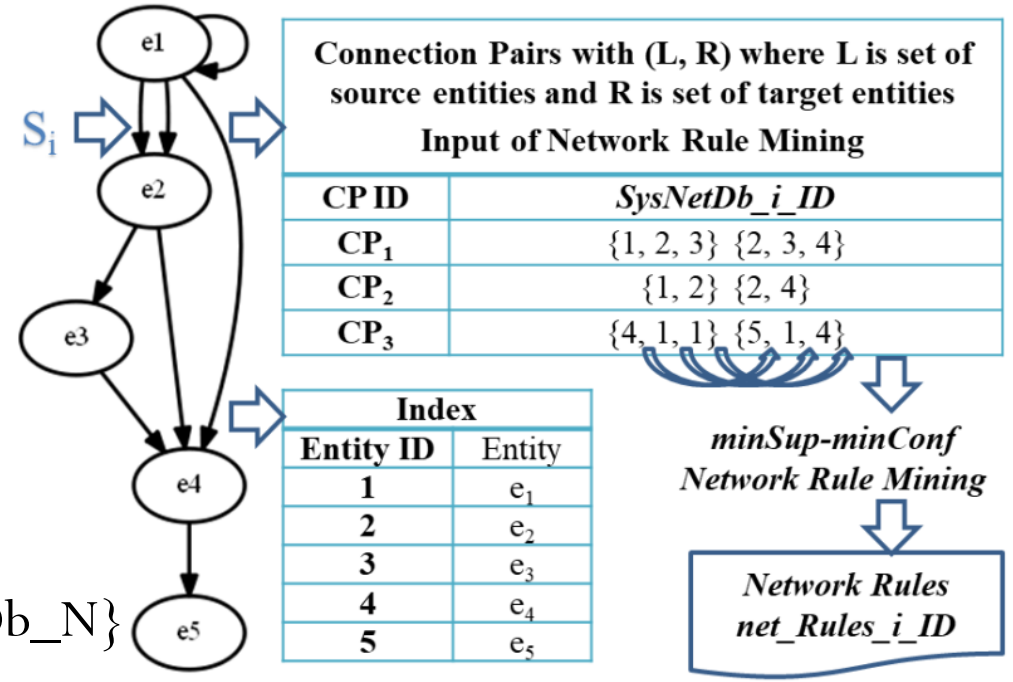
- A distinct rule occurring in multiple states is said to be an *Evolution Rule* that has some *stability* in the state series, where stability is the fraction of states in which the rule is interesting.

- An evolution rule is said to be a *Stable Rule* if its stability exceeds a given threshold named **minimum stability** (minStab).

System Network Database

A *system network database* (SysNetDb) is

- a set of *connection pairs* $\{CP_1, CP_2 \ldots CP_M\}$



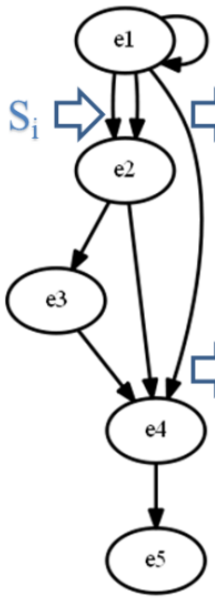
Dynamic databases for a state series is denoted as

- ***SysNetDbs*** = {SysNetDb_1, SysNetDb_2… SysNetDb_N}

Let a SysNetDbs contains a set of entities X∪Y to form a ***Network Rule*** X→Y in state S_i

where X is a subset of source entities and Y is a subset of target entities

- *support* sup(X→Y, S_i) = minSupCount ÷ M, where minSupCount is the support count of connection pairs in which X∪Y occurs and M is the total number of connection pairs in SysNetDb
- *confidence* conf(X→Y, S_i) = sup(X∪Y, S_i) ÷ sup(X, S_i)
- *interestingness*, if X→Y support and confidence are greater than thresholds minSupCount and minimum confidence (minConf)



S_i

Connection Pairs with (L, R) where L is set of source entities and R is set of target entities

Input of Network Rule Mining

CP ID	*SysNetDb_i_ID*
CP_1	{1, 2, 3} {2, 3, 4}
CP_2	{1, 2} {2, 4}
CP_3	{4, 1, 1} {5, 1, 4}

Index

Entity ID	Entity
1	e_1
2	e_2
3	e_3
4	e_4
5	e_5

minSup-minConf Network Rule Mining

Network Rules net_Rules_i_ID

Input of NRM

CP ID	*SysNetDb_1_ID*
CP_1	{1, 2, 3} {2, 3, 4}
CP_2	{1, 2} {2, 4}
CP_3	{4, 1, 1} {5, 1, 4}

Input of NRM

CP ID	*SysNetDb_2_ID*
CP_1	{1, 2, 3} {2, 3, 4}
CP_2	{1, 2} {2, 4}
CP_3	{4, 1, 2} {5, 1, 4}

Input of NRM

CP ID	*SysNetDb_3_ID*
CP_1	{1, 2, 3} {2, 3, 4}
CP_2	{1, 3} {2, 4}
CP_3	{4, 1, 2} {5, 1, 4}

Output of NRM for SysNetDb_1_ID

Network Rules	Sup Count	Conf
1 → 4	3	1.0
1, 2 → 4	2	1.0
2 → 4	2	1.0
1 → 2	2	0.6
1 → 2, 4	2	0.6

Output of NRM for SysNetDb_2_ID

Network Rules	Sup Count	Conf
1 → 4	3	1.0
1, 2 → 4	3	1.0
2 → 4	3	1.0
1 → 2	2	0.6
1 → 2, 4	2	0.6

Output of NRM for SysNetDb_3_ID

Network Rules	Sup Count	Conf
1 → 4	3	1.0
1, 3 → 2	2	1.0
1, 3 → 2, 4	2	1.0
1, 3 → 4	2	1.0
3 → 2	2	1.0
3 → 2, 4	2	1.0
3 → 4	2	1.0
1, 2 → 4	2	0.6
2 → 4	2	0.6
1 → 2	2	0.6
1 → 2, 4	2	0.6

Collections of Network Rules

1 → 4	1 → 4	1 → 4
--	--	1, 3 → 2
--	--	1, 3 → 2, 4
--	--	1, 3 → 4
--	--	3 → 2
--	--	3 → 2, 4
--	--	3 → 4
1, 2 → 4	1, 2 → 4	1, 2 → 4
2 → 4	2 → 4	2 → 4
1 → 2	1 → 2	1 → 2
1 → 2, 4	1 → 2, 4	1 → 2, 4

Network Evolution Rules (NERs)	Stability Count
1 ==> 4	3
1, 2 ==> 4	3
2 ==> 4	3
1 ==> 2	3
1 ==> 2, 4	3
1, 3 ==> 2	1
1, 3 ==> 2, 4	1
1, 3 ==> 4	1
3 ==> 2	1
3 ==> 2, 4	1
3 ==> 4	1

Stable Network Evolution Rules (SNERs)
1 → 4
1, 2 → 4
2 → 4
1 → 2
1 → 2, 4

System Network Persistence metric SNP

$$= minStab \times \frac{\text{SNER\_Count}}{\text{NER\_Count}} \times 100$$

$$= \{(2 \div 3) \times (5 \div 11)\} \times 100$$

Stable Network Evolution Rule

A ***Network Evolution Rule*** **(NER)** X→Y of a state series SS

- if it is a distinct network rule present in multiple states
- here, distinct means considering identical rules only once
- *stability* stab(X→Y, SS) = *stabilityCount* ÷ N, where stabilityCount is the number of states in which X→Y is interesting and N is the cardinality (number) of states in SS.

Stable NER in SS

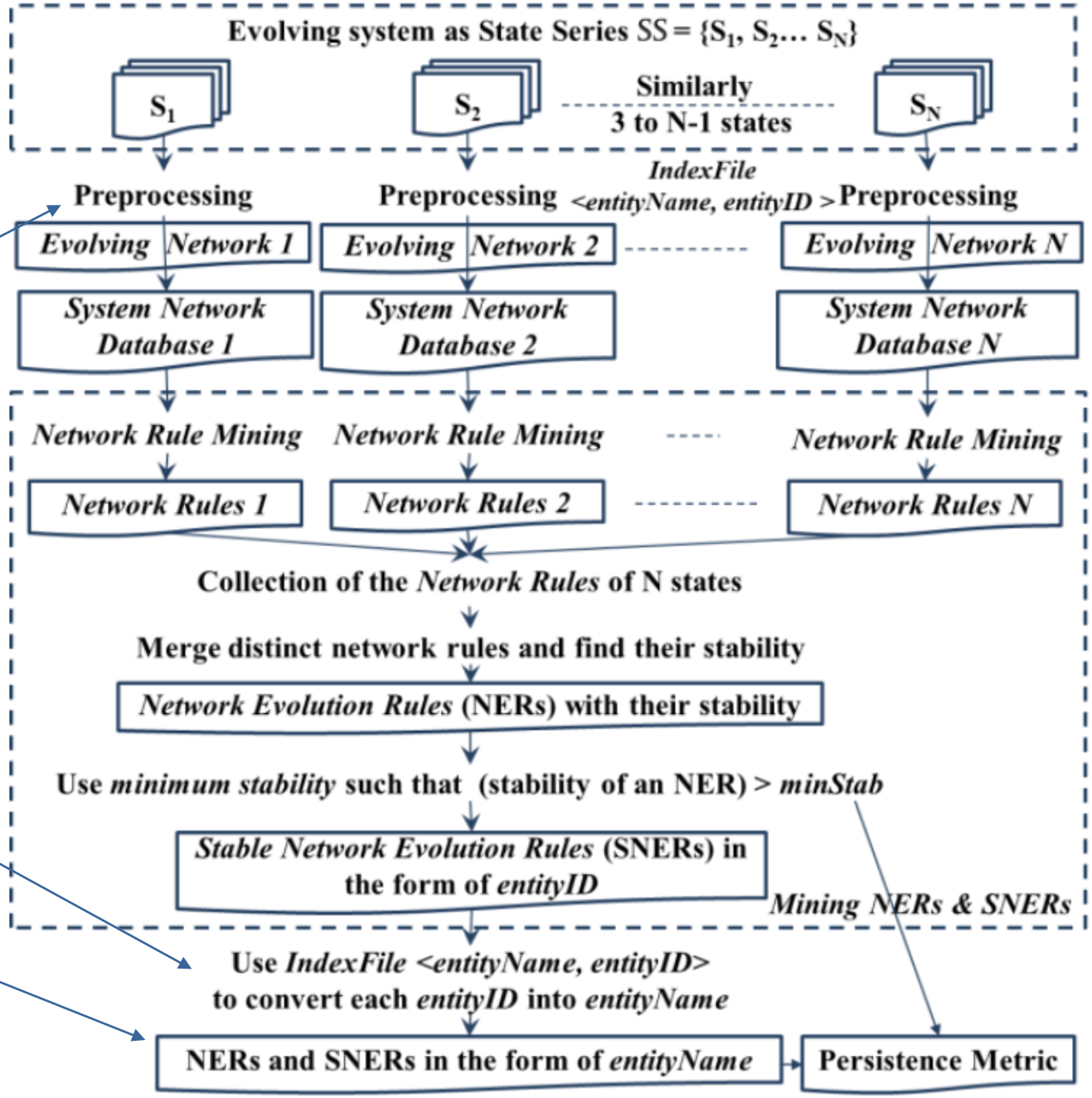
- if its stability count is greater than (≥) a threshold *minimum stability* (minStab)
- minStab = minStabCount ÷ N.

SysNet-Analytics

Algorithm **SysNet-Analytics**(*repository*)

Retrieve N system states and store them in a *repository*.

1. *SysNetDbs* & *IndexFile* = **Pre-process**(*repository*)
2. *NERs_ID* & *SNERs_ID* = **Mining_NERs_SNERs** (*SysNetDbs*, *minSupCount*, *minConf*, *minStabCount*)
3. *NERs_Name* & *SNERs_Name* = **Indexing**(*NERs_ID*, *SNERs_ID*, *IndexFile*)
4. *SNP_metric* = **SNP_Metric**(*minStab*, *SNER_Count*, *NER_Count*)



Algorithm 2 **Mining_NERs_SNERs(***SysNetDbs*, *minSupCount*, *minConf*, *minStabCount***)**

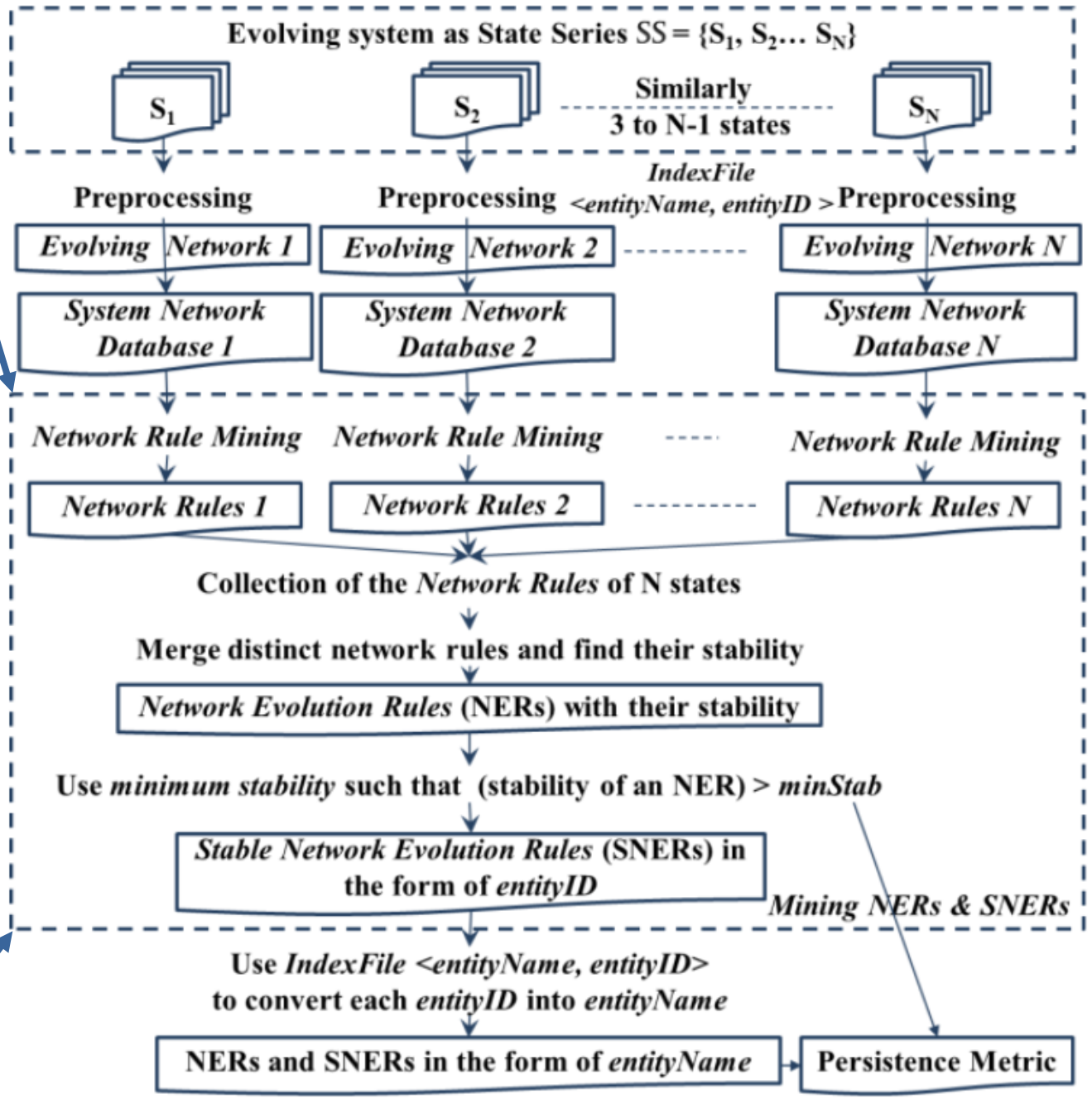
Initialize File *net_Rules_i_ID NERs_ID*, *SNERs_ID*
Initialize Array *Collect_NRs_ID*,
Initialize HashMap *NERs_HM* < NER_ID, stability >
Initialize i ∈ integer

For each *SysNetDb_i_ID* in *SysNetDbs*, where i varies from 1 to N
  net_Rules_i_ID= **NRM(***SysNetDb_i_ID*, *minSupCount*, *minConf***)**
  Store *net_Rules_i_ID* file in directory *netRules*
End For

For each state *net_Rules_i_ID* in *netRules*, where i varies 1 to N
  Collect_NRs_ID = **Merge(***Collect_NRs_ID*, *net_Rules_i_ID***)**
End For

For each distinct rule (as *NER_ID*) in *Collect_NRs_ID*
  Initialize int *stabilityCount* = 0
  For each *rule_x* in *Collect_NRs_ID*
    if(*NER_ID* equal to *rule_x***)**
      then *stabilityCount*++
    end if
  End for
  Initialize float *stability* = *stabilityCount* ÷ N
  Initialize float *minStab* = *minStabCount* ÷ N
  if(*NER_ID* is not in *NERs_HM***)**
    then Add(<*NER_ID*, *stability*> to *NERs_HM***)**
  end if
  if(*stability* > *minStab***)**
    if(*NER_ID* is not in *SNERs_ID***)**
      then Add(*NER_ID* to *SNERs_ID***)**
    end if
  end if
End For

NERs_ID = *NERs_HM*
Return *NERs_ID* and *SNERs_ID*



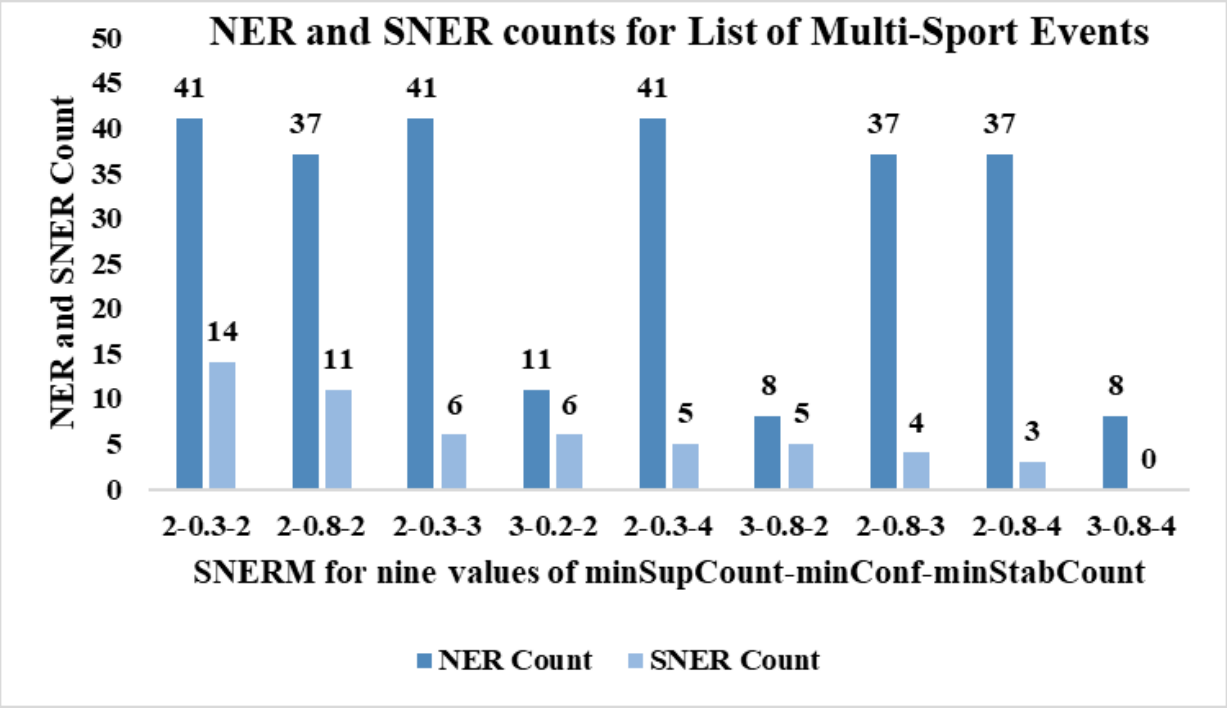
Experiments on Evolving Systems

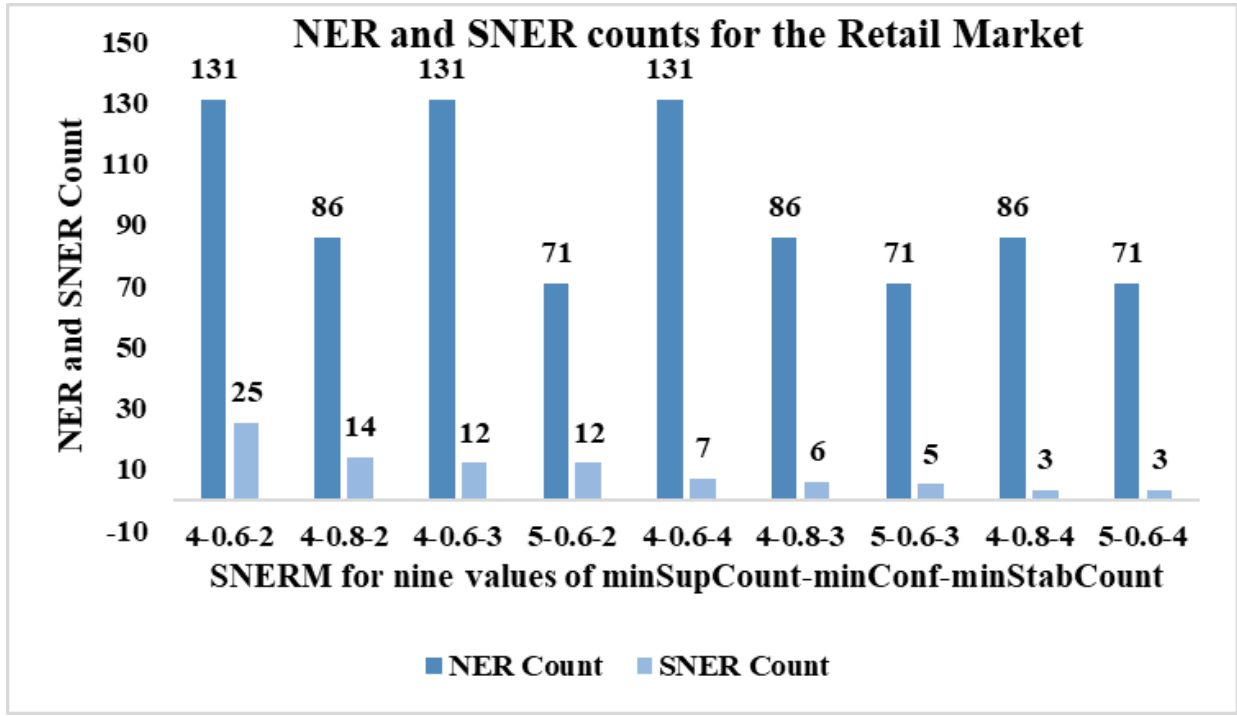
Hadoop-HDFS[1]	List of Multi-sport Events[3]	Positive[6] sentiment in IMDb[5]
create ==> convert readAll ==> readFully checkAccess ==> getDelegationToken close ==> getRemoteAddressString	World ==> International Games ==> Regional Games ==> International Games, World ==> International Games, Asian ==> Regional Asian ==> Regional	premier ==> Short fond ==> Short fine ==> Short humor ==> Comedy grand ==> Music
List of Bible Translation[2]	**Retail Market[4]**	**Neagative[6] sentiment in IMDb[5]**
English ==> Vulgate English ==> Masoretic English, Modern ==> Masoretic English ==> Masoretic, Text English, Modern ==> Masoretic, Text English ==> Text English, Modern ==> Text English ==> Text, Greek English, Modern ==> Text, Greek English ==> Greek English, Modern ==> Greek Modern ==> Masoretic Modern ==> Masoretic, Text Modern ==> Text Modern ==> Text, Greek Modern ==> Greek	SUKI SHOULDER BAG ==> 17841 ASSORTED MONKEY SUCTION CUP HOOK ==> 17841 CARRIAGE ==> 14911 SKULL DESIGN TV DINNER TRAY ==> 17841 ASSORTED COLOUR LIZARD SUCTION HOOK ==> 17841 SMALL YELLOW BABUSHKA NOTEBOOK ==> 17841 LIPSTICK PEN RED ==> 17841 UNION STRIPE CUSHION COVER ==> 17841 DISCO BALL CHRISTMAS DECORATION ==> 17841 BLUE/CREAM STRIPE CUSHION COVER ==> 17841 CAKE PLATE LOVEBIRD WHITE ==> 17841 KITCHEN METAL SIGN ==> 17841	rue ==> Short terrible ==> Short rue ==> Documentary pain ==> Short vent ==> Short passe ==> Short sin ==> Drama brat ==> Drama terror ==> Horror perverse ==> Adult

1. https://mvnrepository.com/artifact/org.apache.hadoop/hadoop-hdfs
2. https://en.wikipedia.org/wiki/List_of_English_Bible_translations
3. https://en.wikipedia.org/wiki/List_of_multi-sport_events
4. https://archive.ics.uci.edu/ml/datasets/Online+Retail
5. http://www.imdb.com/interfaces/
6. https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html

NER and Stable NER counts comparisons

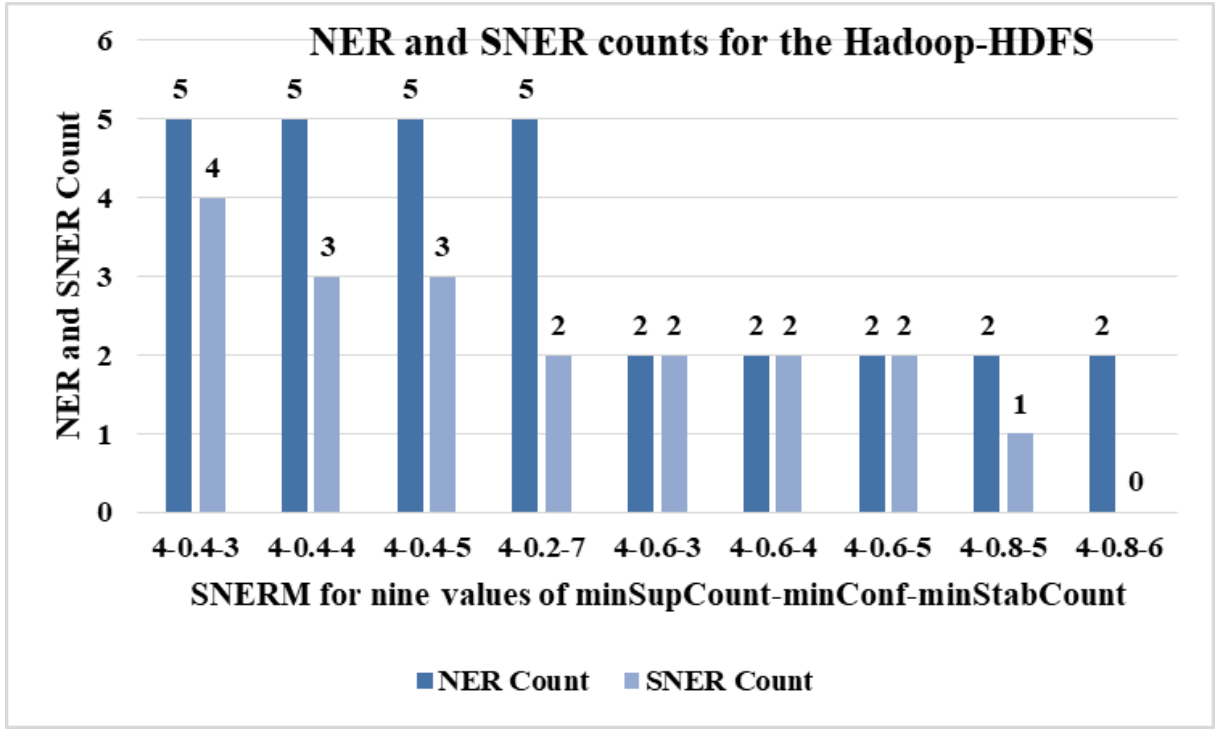
- Natural-Language Evolution Analytics: List of Multi-sport events system available on Wikipedia
- Market Evolution Analytics: Frequent Market Basket system made from retail-market baskets available on UCI Repository





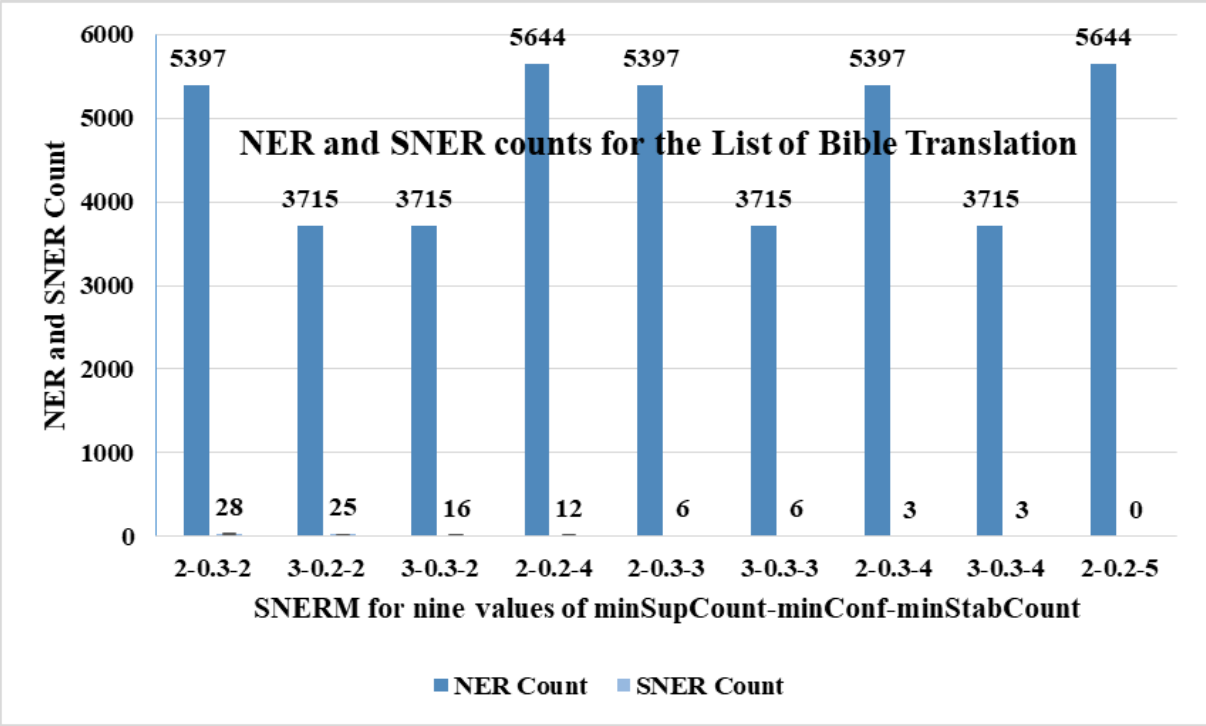
Software Evolution Analytics

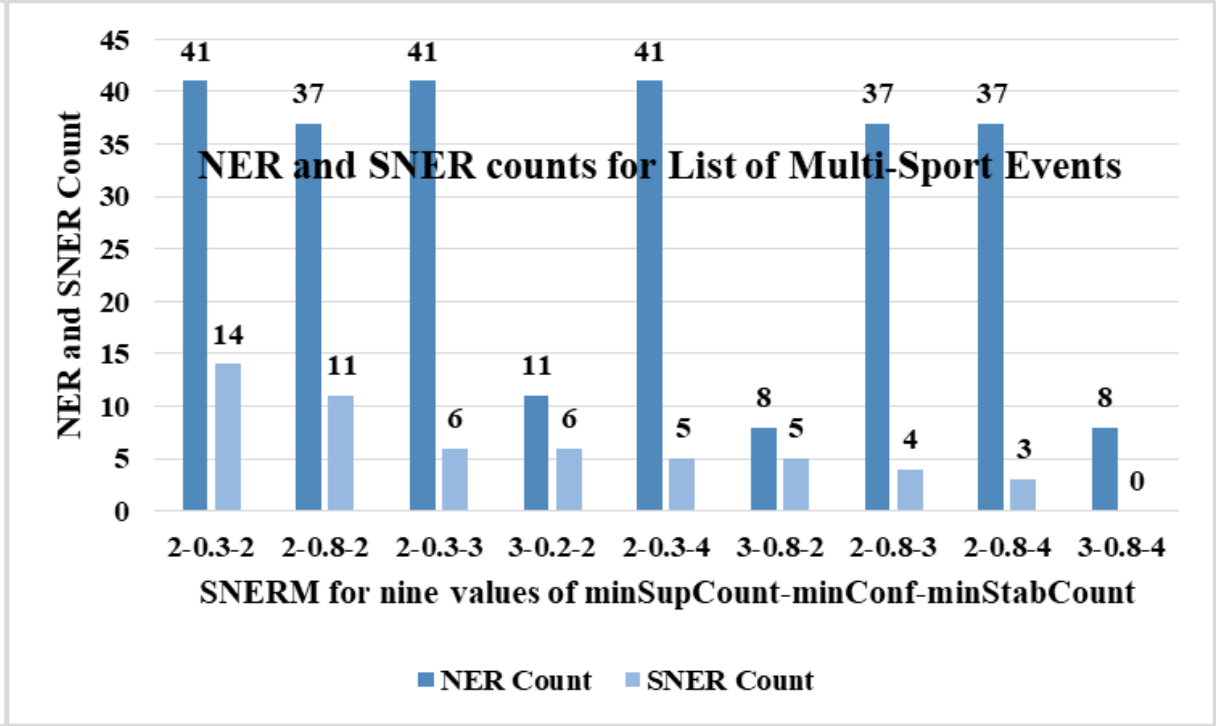
- Evolving Software Systems:
  - Hadoop – HDFS available on Software Repository



Natural-Language Evolution Analytics

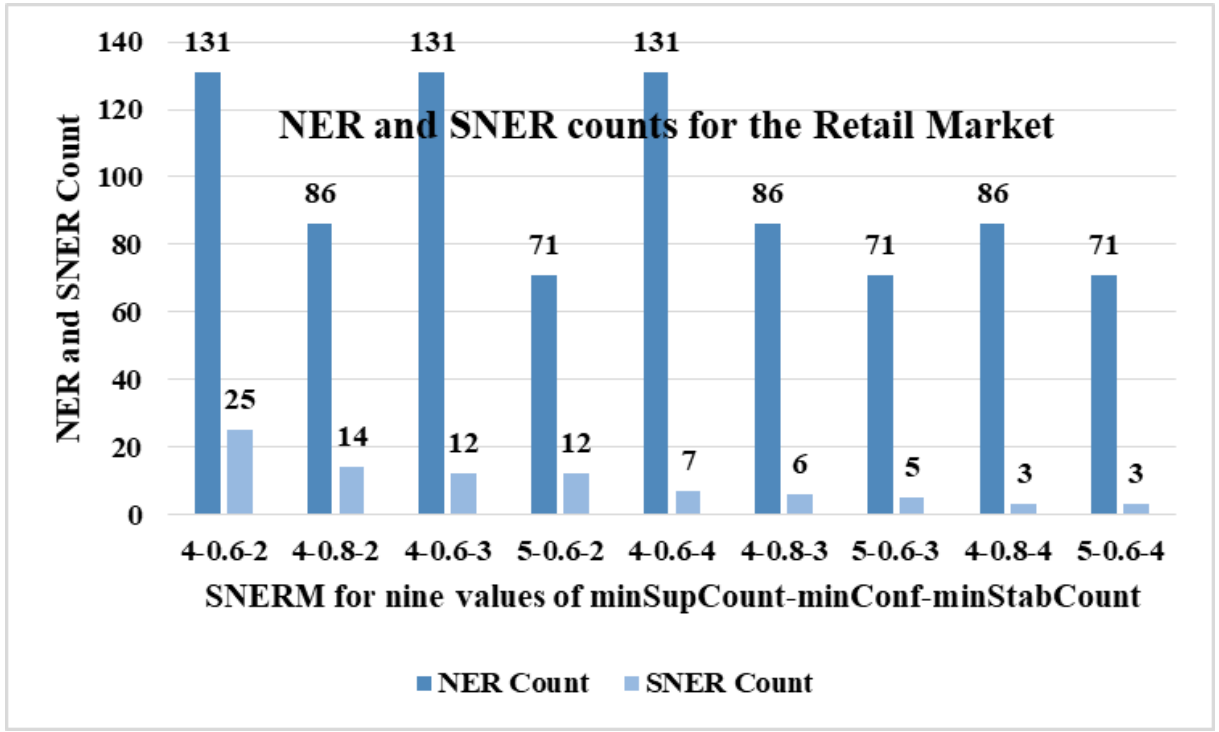
- Two Evolving Natural-Language systems:
  - List of Bible Translation system available on Wikipedia
  - List of Multi-sport events system available on Wikipedia





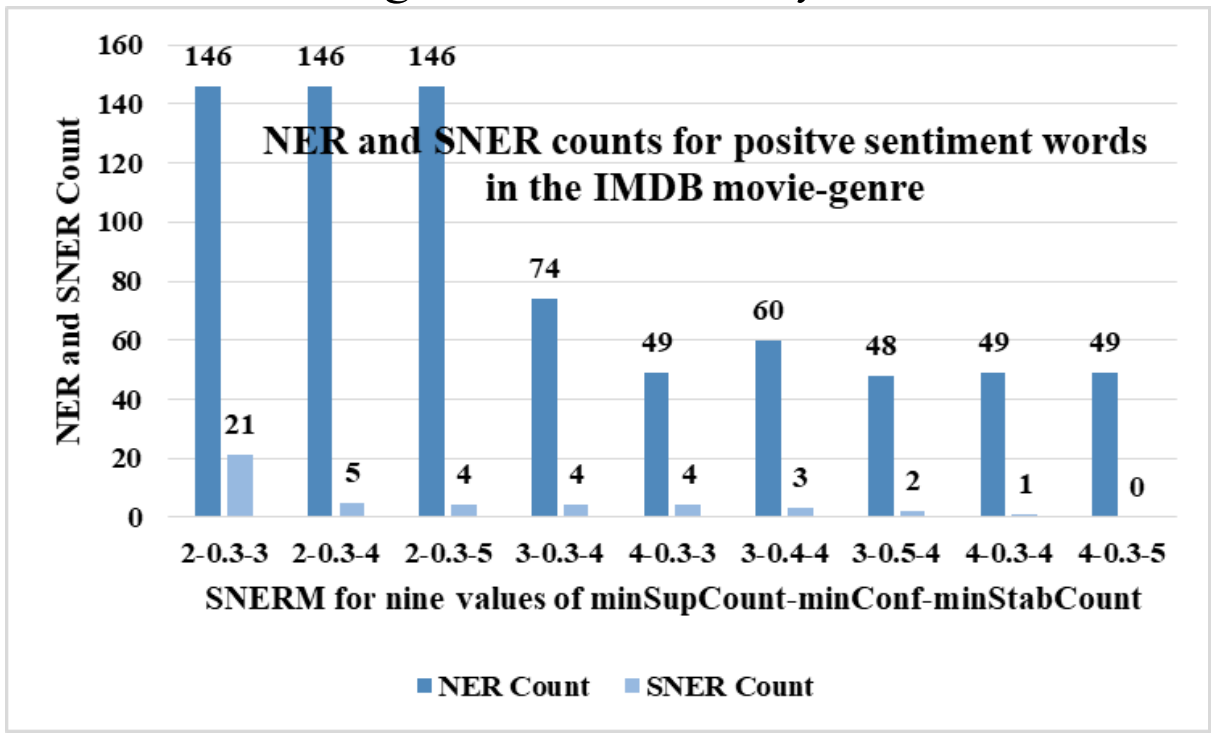
Market Evolution Analytics

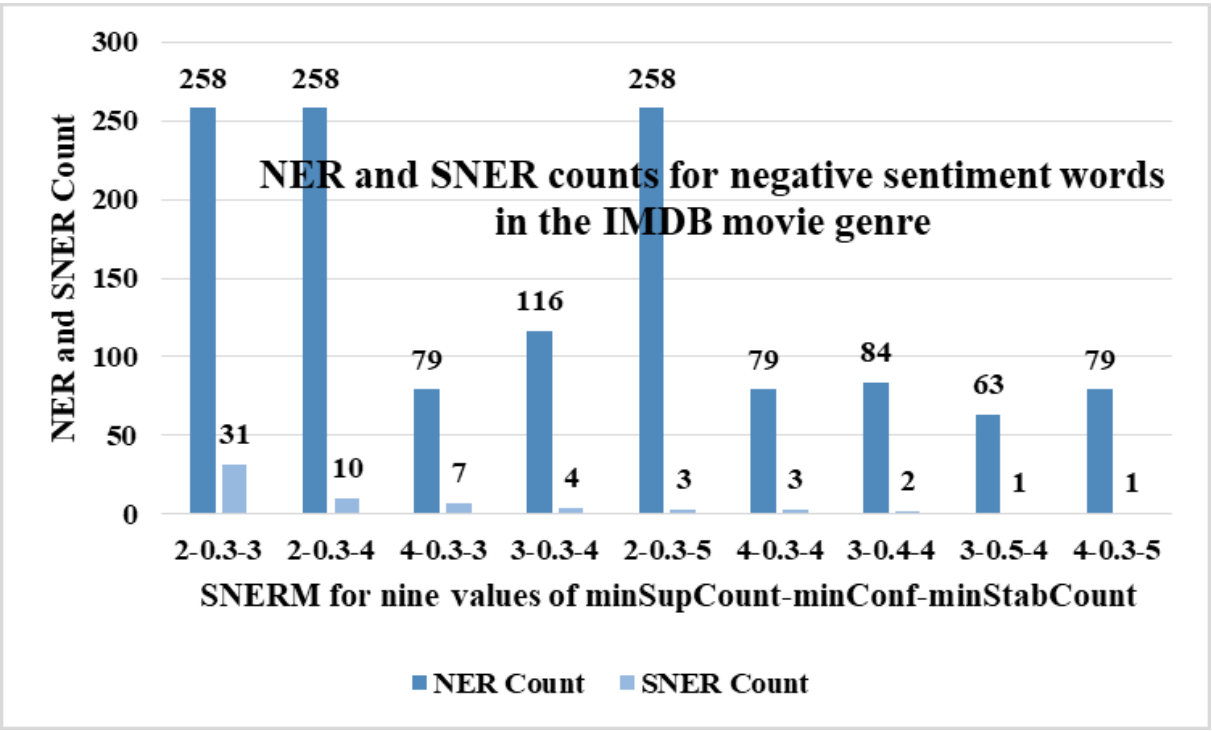
- Evolving Retail Market system:
  - Frequent Market Basket system made from retail-market baskets available on UCI Repository



Movie Evolution Analytics

- Two systems made from IMDB movie-genre data available on IMDb Repository :
  - Positive sentiment system
  - Negative sentiment system



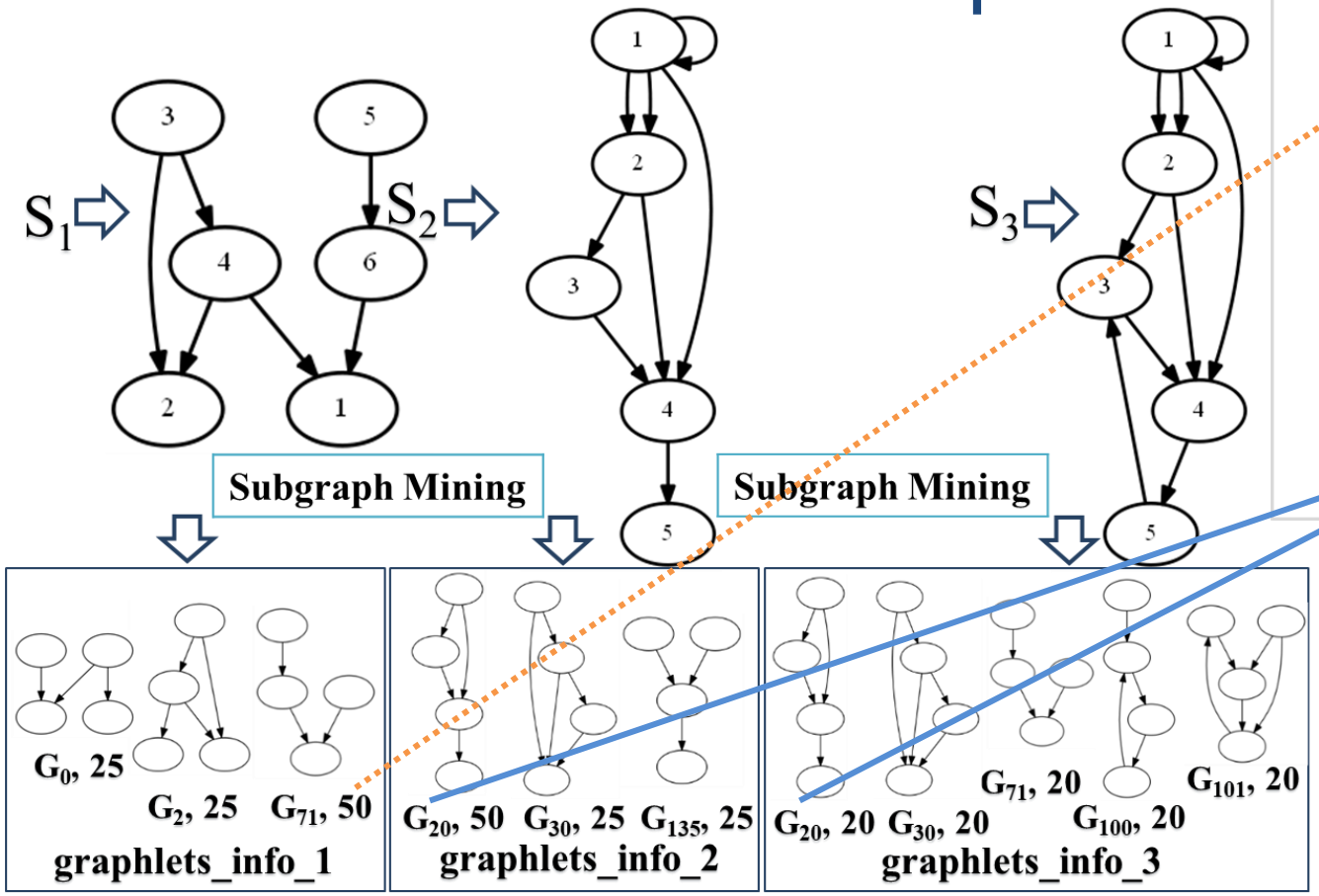


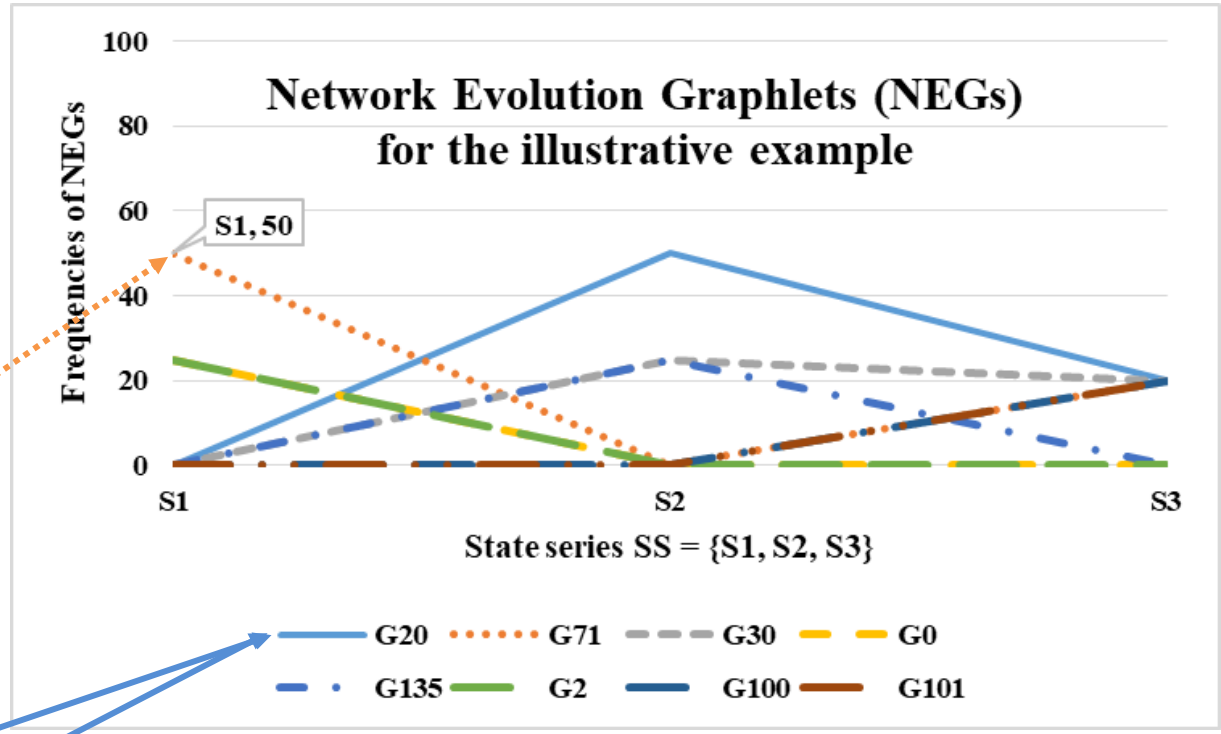
Network Data Science Research

Network Evolution Subgraph Mining for System Stability, Changeability, and Complexity

1. System Evolution Analytics of a State Series
2. System Network Analytics: Stable Network Evolution Rules of a State Series
3. **Network Evolution Subgraph Mining for System Stability, Changeability, and Complexity**
4. System Neural Network for Evolution Learning

Illustrative example





NEGs_info is $Aggregate\_freq_j = \frac{\sum_{i=1}^{N} freq_{ji}}{N}$

[<G_{20}, 23.33>, <G_{71}, 23.33>, <G_{30}, 15>, <G_0, 8.33>, <G_{135}, 8.33>, <G_2, 8.33>, <G_{100}, 6.66>, <G_{101}, 6.66>]

NEMs [<G_{20}, 23.33>, <G_{71}, 23.33>] with the threshold frequency is 20%.

Proposed Definitions

- A *network evolution subgraph* is a subgraph with changing frequencies in the state series SS
  - *network evolution frequent subgraph*: frequently occurring in a network;
  - *network evolution motif*: subgraph is statistically recurrent in a network; and
  - *network evolution graphlet*: subgraph is induced subgraph in a network.
- The $Aggregate\_freq_j$ denotes *aggregate frequency* of a NEG G_j.
- The arithmetic mean of frequency over N states for a NEG G_j is given by

$$Aggregate\_freq_j = \frac{\sum_{i=1}^{N} freq_{ji}}{N}$$

where, $freq_{ji}$ is the *frequency* of NEG G_j in state S_i with i varying from integer 1 to N and j is constant.

Proposed Definitions

- *Network Evolution Graphlets information* is a doubleton set of retrieved NEGs (as a subgraph) and their *aggregate frequencies* $Aggregate\_freq_j$.

$$NEGs\_info = < G_j, Aggregate\_freq_j > \mid 0 \leq j, m' \leq M$$

where,

- G_j is j^{th} NEG with aggregate frequency as $Aggregate\_freq_j$ and j is the enumeration of the NEG;
- m' is the number of retrieved NEGs (distinctly non-redundant graphlets) over all the states in SS.

System Network Complexity

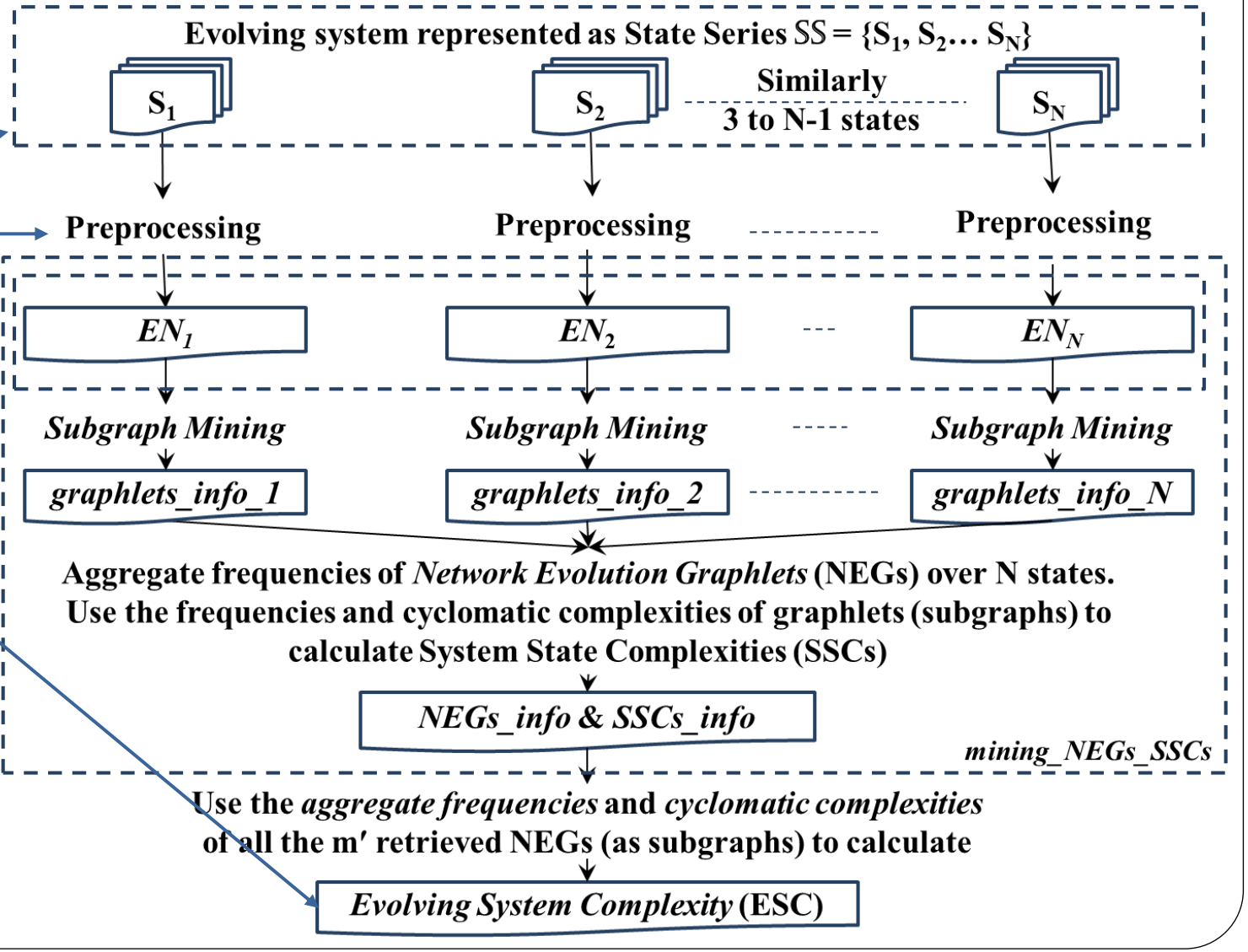
An overview of flow-chart

Algorithm SNC(*repository*)

Initialize $i \in$ integer 1 to N

Retrieve N states of a *state series* SS $= \{S_1, S_2 \ldots S_N\}$ stored in *repository*

1. *evolvingNetworks* = **Preprocess**(*repository*)
2. *NEGs_info* & *SSCs_info* = **mining_NEGs_SSCs**(*evolvingNetworks*)
3. *ESC* = **Calculate_ESC**(*NEGs_info*)



$$\text{SSC of } S_i \ (\text{SSC}_i) = \frac{\sum_{j=0}^{m} (freq_{ji} \times C_j)}{\sum_{j=0}^{m} freq_{ji}}$$

$$\text{ESC} = \frac{\sum_{j=0}^{m'} (Aggregate\_freq_j \times C_j)}{\sum_{j=0}^{m'} Aggregate\_freq_j}$$

Algorithm SNC(*repository*)

Initialize $i \in$ integer 1 to N
Retrieve N states of a *state series* SS $= \{S_1, S_2 \ldots S_N\}$ stored in *repository*
1. *evolvingNetworks* = ***Preprocess***(*repository*)
2. *NEGs_info* & *SSCs_info* = ***mining_NEGs_SSCs***(*evolvingNetworks*)
3. *ESC* = ***Calculate_ESC***(*NEGs_info*)

Algorithm 2: mining_NEGs_SSCs(*evolvingNetworks*)

Initialize $j \in$ integer for graphlet G_j such that $0 \leq j \leq M$
Initialize $i \in$ integer varying from 1 to N states
Initialize HashMap *graphlets_info*$< G_j, freq_{ji} >$
Initialize HashMap *NEGs_info*$< G_j, Aggregate\_freq_j >$
Initialize HashMap *SSCs_info*$< S_i, SSC_i >$

Where, G_j is j^{th} *graphlet*, $freq_{ji}$ is *frequency* of G_j in state S_i, and $Aggregate\_freq_j$ is aggregate frequency of NEG G_j over a state series.
For each EN_i in *evolvingNetworks* where *i* varies from 1 to N
$graphlets\_info\_i < G_j, freq_{ji} > =$ ***subgraphMining***(EN_i)
End For
Let m′ is count of retrieved graphlets over all states
For each G_j in m′ graphlets, where $1 \leq j, m' \leq M$
Initialize float $frequencySum = 0$
Initialize integer $Aggregate\_freq_j = 0$
  For each *graphlets_info_i* where *i* varies from 1 to N
  $frequencySum = frequencySum + freq_{ji}$
  End For
  $Aggregate\_freq_j = frequencySum \div$ N
  Add tuple $< G_j, Aggregate\_freq_j >$ to *NEGs_info*
End For
Initialize cyclomatic complexity array C[M] for all graphlets
For each *graphlets_info_i* of S_i where *i* varies from 1 to N
Initialize float $frequencySum = 0$
Initialize float $sumOfProducts = 0$
  For each G_j in *graphlets_info_i*
  $frequencySum = frequencySum + freq_{ji}$
  $sumOfProducts = sumOfProducts + \{freq_{ji} \times C_j\}$
  //where *Cyclomatic complexity* C_j for graphlet G_j at C[j]
  End For
  Float $SSC_i = sumOfProducts \div frequencySum$
  Add tuple $< S_i, SSC_i >$ to *SSCs_info*
End For
Return *NEGs_info* & *SSCs_info*

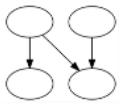
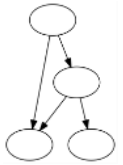
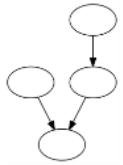
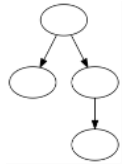
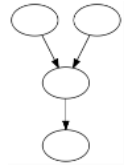
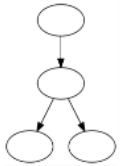
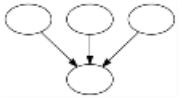
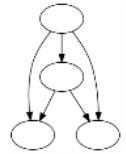
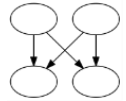
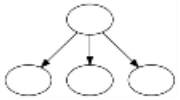
Algorithm 3: Calculate_ESC(*NEGs_info*)

Let m′ is the number of retrieved graphlets in *NEGs_info*
Initialize $j \in$ integer for graphlets such that $0 \leq j \leq M$
Initialize float $sumOfProducts = 0$
Initialize cyclomatic complexity array C[M] for all graphlets
$G_j \in j^{th}$ NEG in *NEGs_info*
Initialize $Aggregate\_freq_j = 0$
$Aggregate\_freq_j \in$ *aggregate frequency* of G_j in *NEGs_info*
For each NEG G_j in *NEGs_info*
  $sumOfProducts = sumOfProducts + \{Aggregate\_freq_j \times C_j\}$
  //where *Cyclomatic complexity* C_j for graphlet G_j at C[j]
  $frequencySum = frequencySum + Aggregate\_freq_j$
End For
Float $ESC = sumOfProducts \div frequencySum$
Return *ESC*

Experiments on Evolving Systems

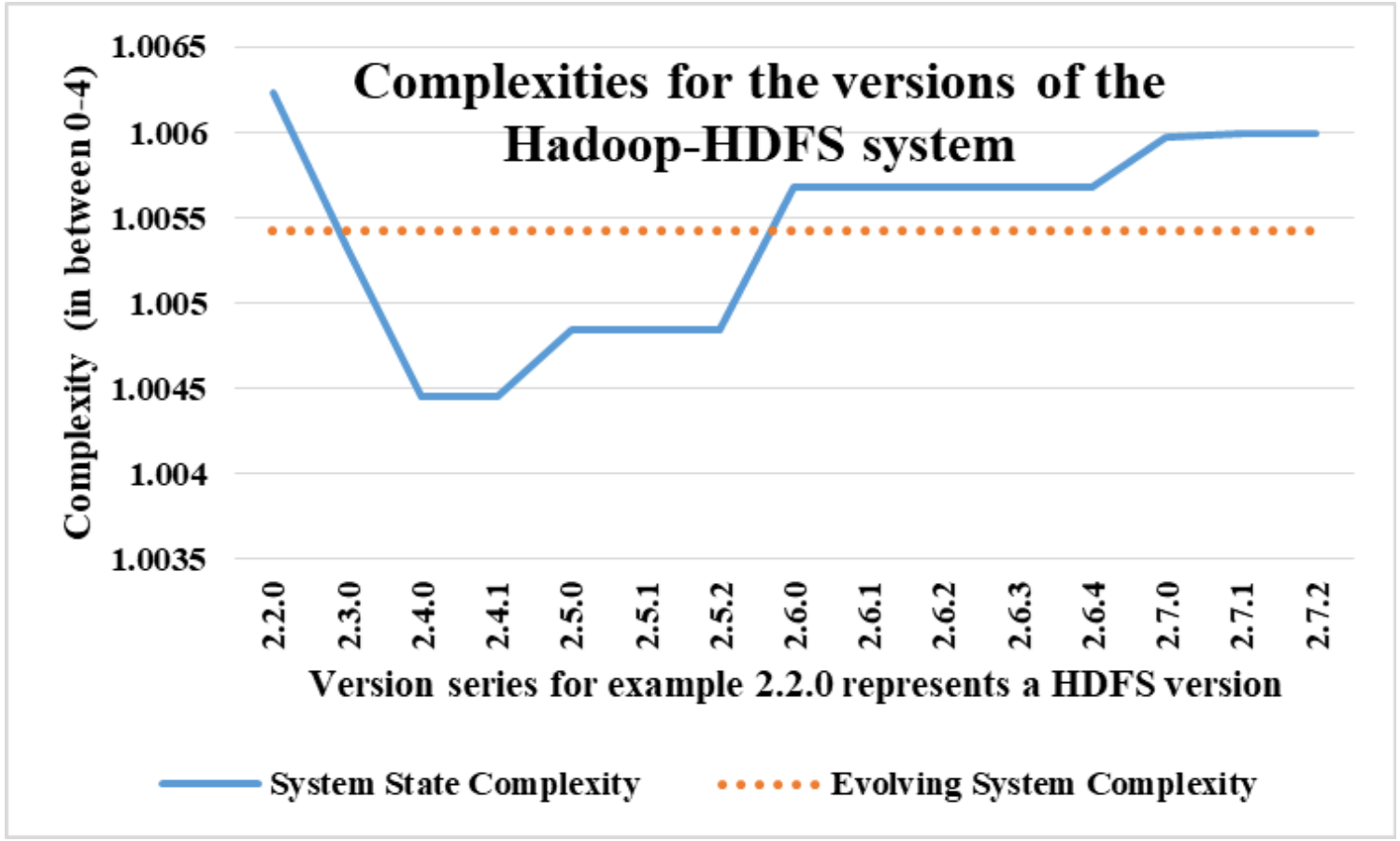
Evolving Systems	N	# entities	Average # neighbour	# NEGs	ESC*
HDFS-Core	15	3129	2.166	24	1.0054
Bible Translation	13	246	1.456	17	1.456
Multi-sport Events	13	141	1.786	10	1.00487
Frequent Market Basket[4]	13	118	8.002	34	1.33888
Positive sentiment of movie genres	16	284	2.661	4	1.03050
Negative sentiment of movie genres	16	510	3.303	20	1.03683

Network Evolution Motifs (NEMs) and their complexities according to their subgraph patterns

$M_0, C_0 = 1$	$M_2, C_2 = 2$	$M_{71}, C_{71} = 1$	$M_{87}, C_{87} = 1$	$M_{135}, C_{135} = 1$
				
$M_{156}, C_{156} = 1$	$M_{178}, C_{178} = 1$	$M_{185}, C_{185} = 3$	$M_{192}, C_{192} = 2$	$M_{198}, C_{198} = 1$
				

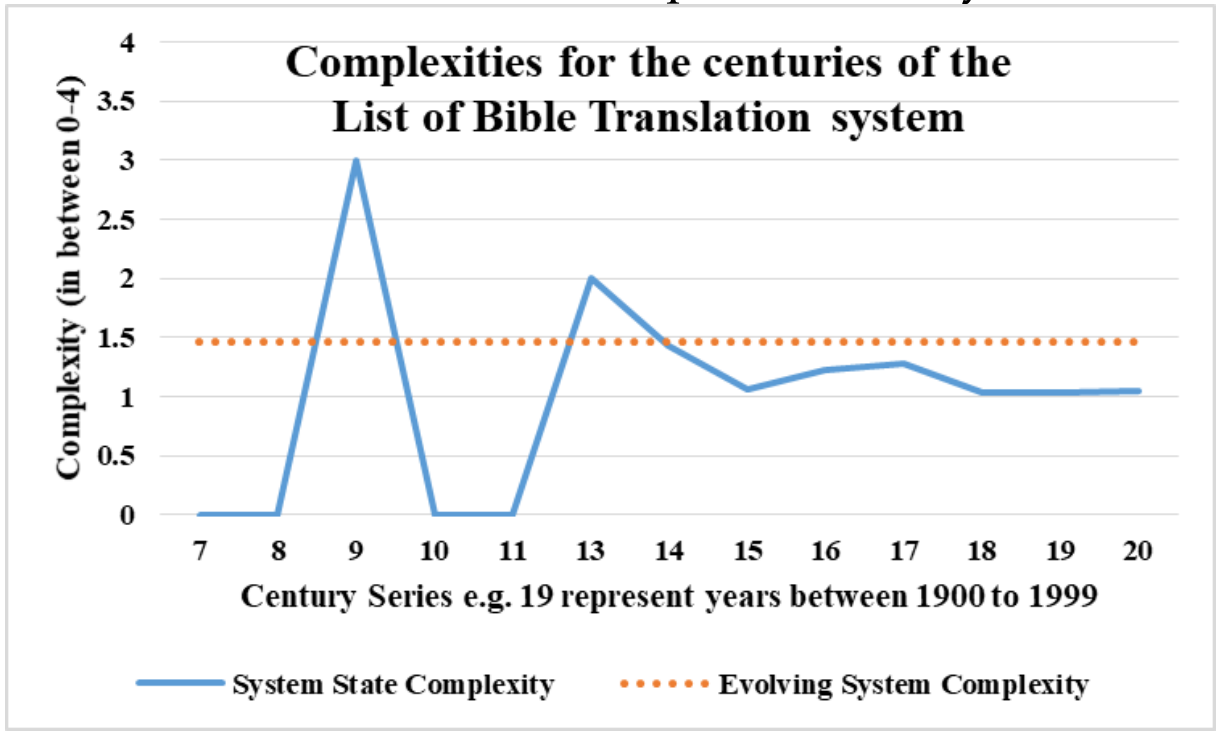
Software Evolution Analytics

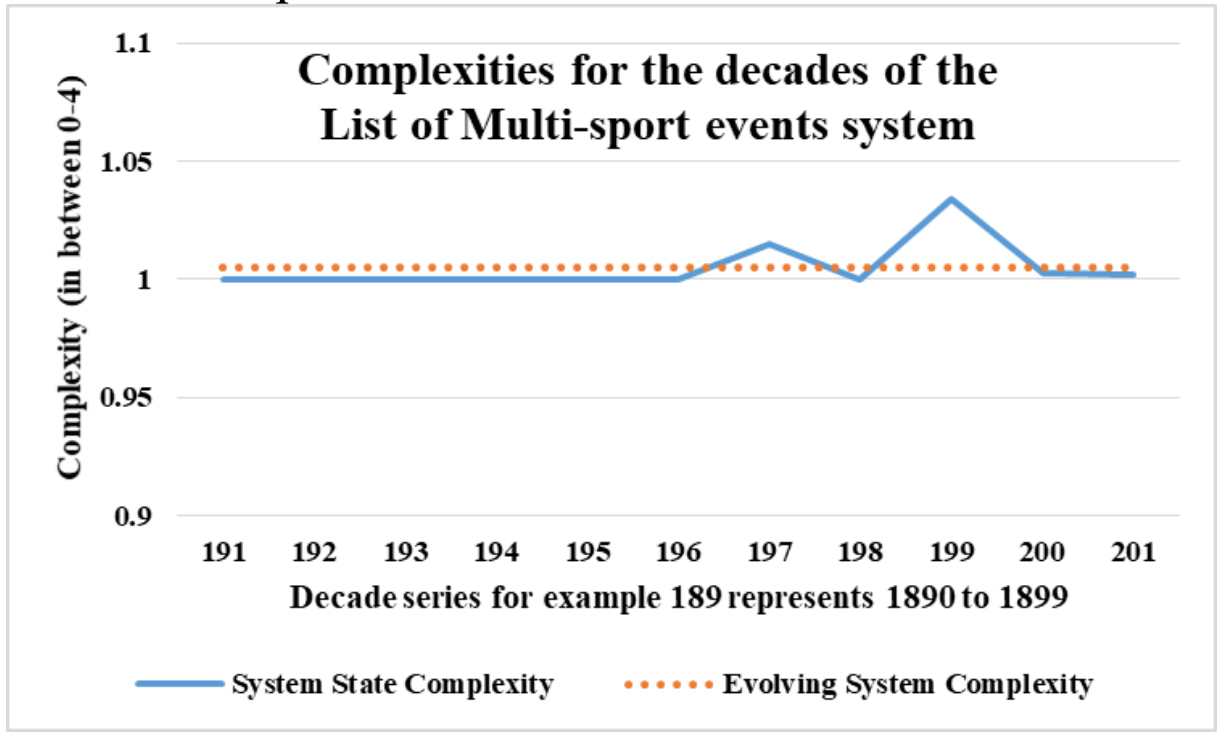
- Evolving Software Systems
  - Hadoop – HDFS available on Software Repository



Natural-Language Evolution Analytics

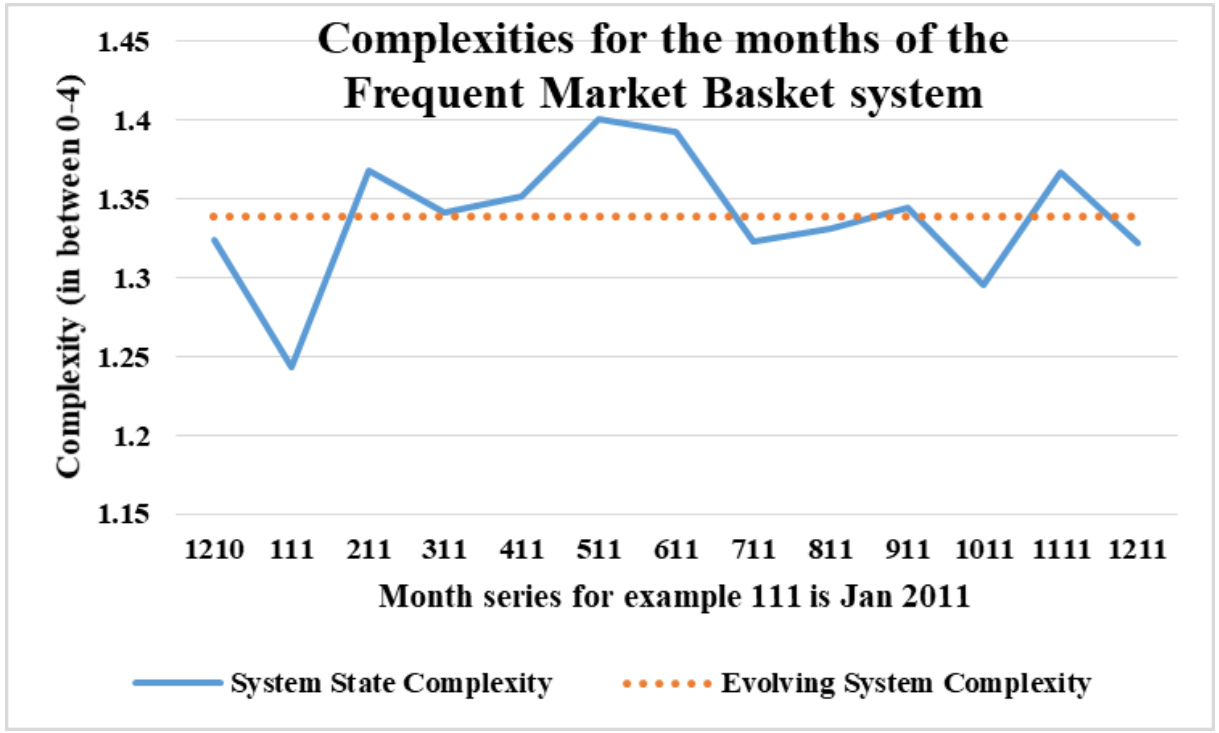
- Two Evolving Natural-Language systems:
  - List of Bible Translation system available on Wikipedia
  - List of Multi-sport events system available on Wikipedia





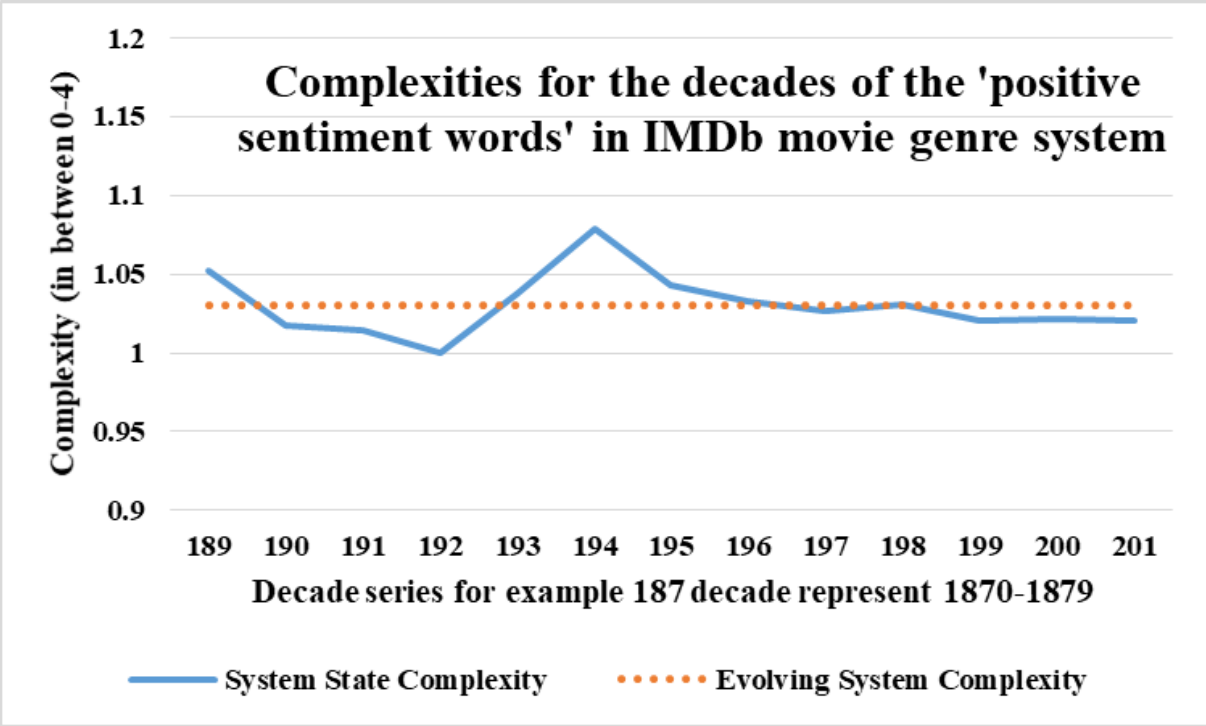
Market Evolution Analytics

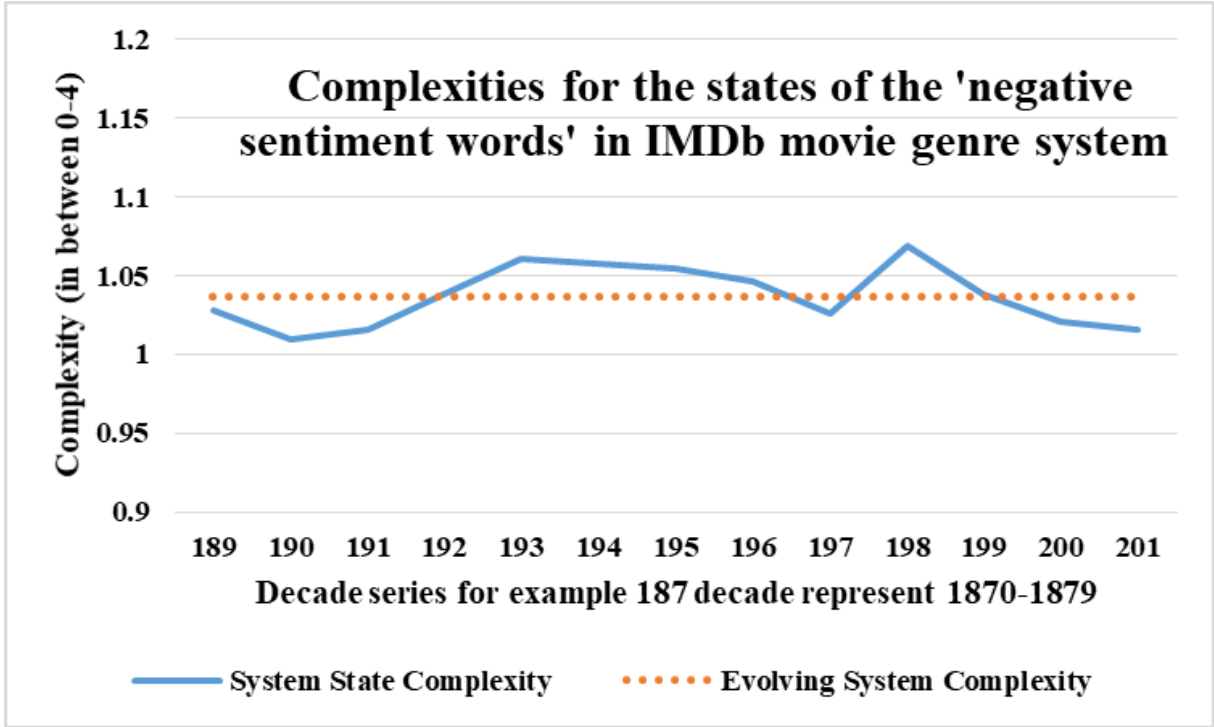
- Evolving Retail Market system:
  - Frequent Market Basket system made from retail-market baskets available on UCI Repository



Movie Evolution Analytics

- Two Evolving IMDb movie genre systems made from IMDb Repository
  - Positive sentiment system
  - Negative sentiment system





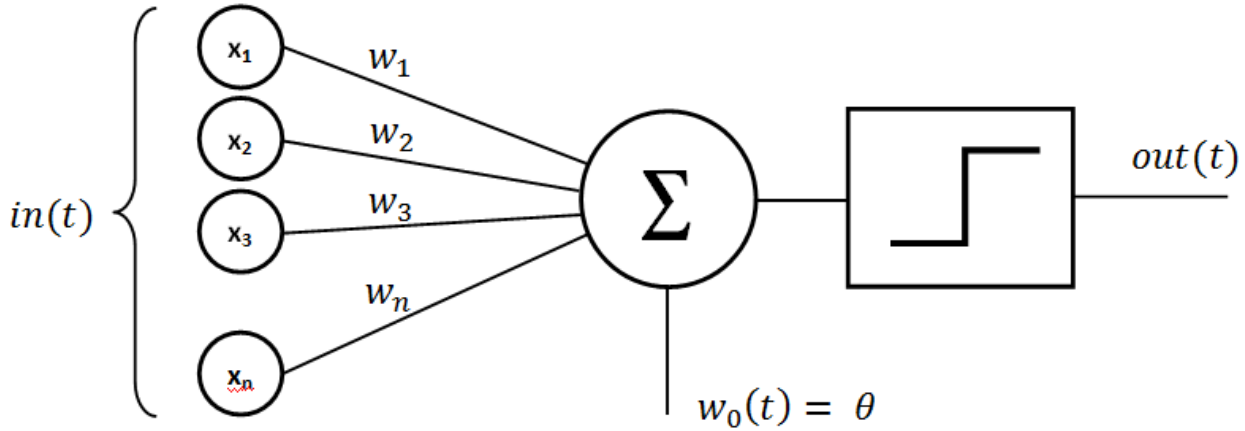
Network Data Science Research

System Neural Network for Evolution Learning

1. System Evolution Analytics of a State Series
2. System Network Analytics: Stable Network Evolution Rules of a State Series
3. Network Evolution Subgraph Mining for System Stability, Changeability, and Complexity
4. **System Neural Network for Evolution Learning**

Perceptron

- **1943 Perceptron** was invented by **McCulloch and Pitts**
- **1958 Mark I Perceptron** hardware developed and constructed by **Frank Rosenblatt**



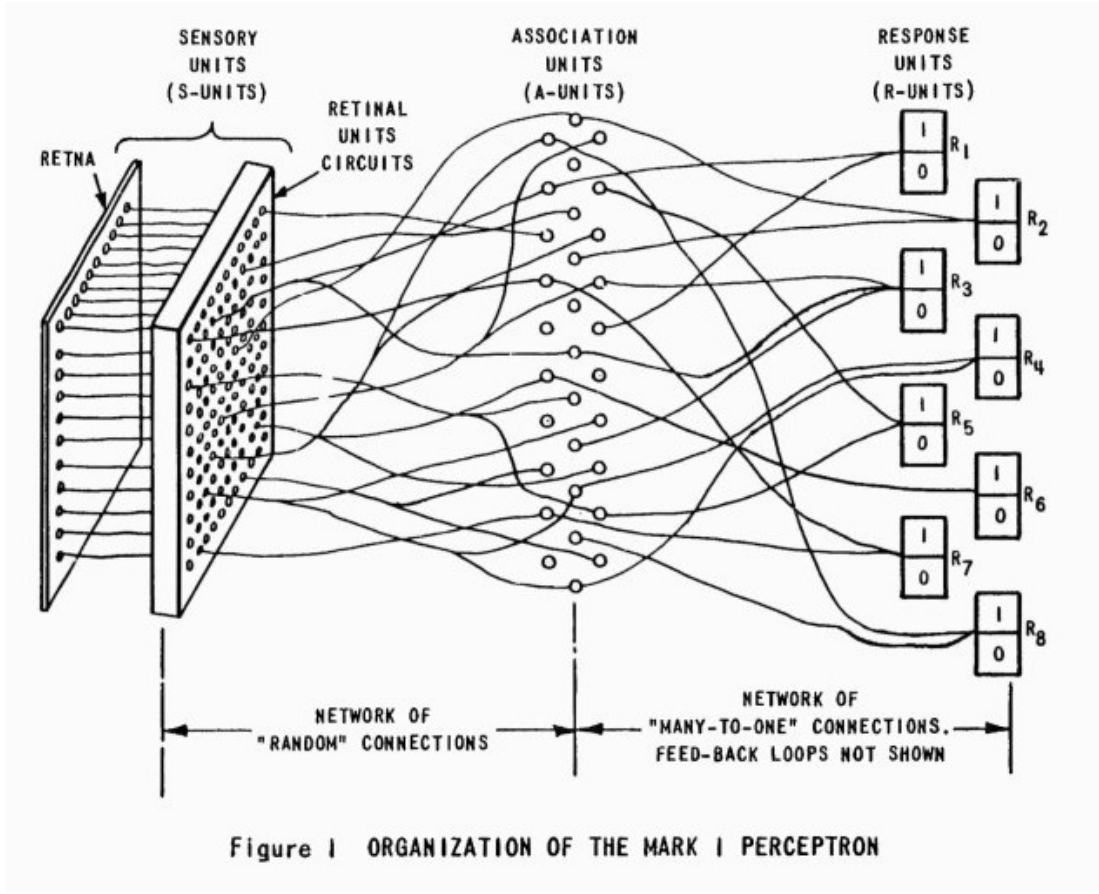
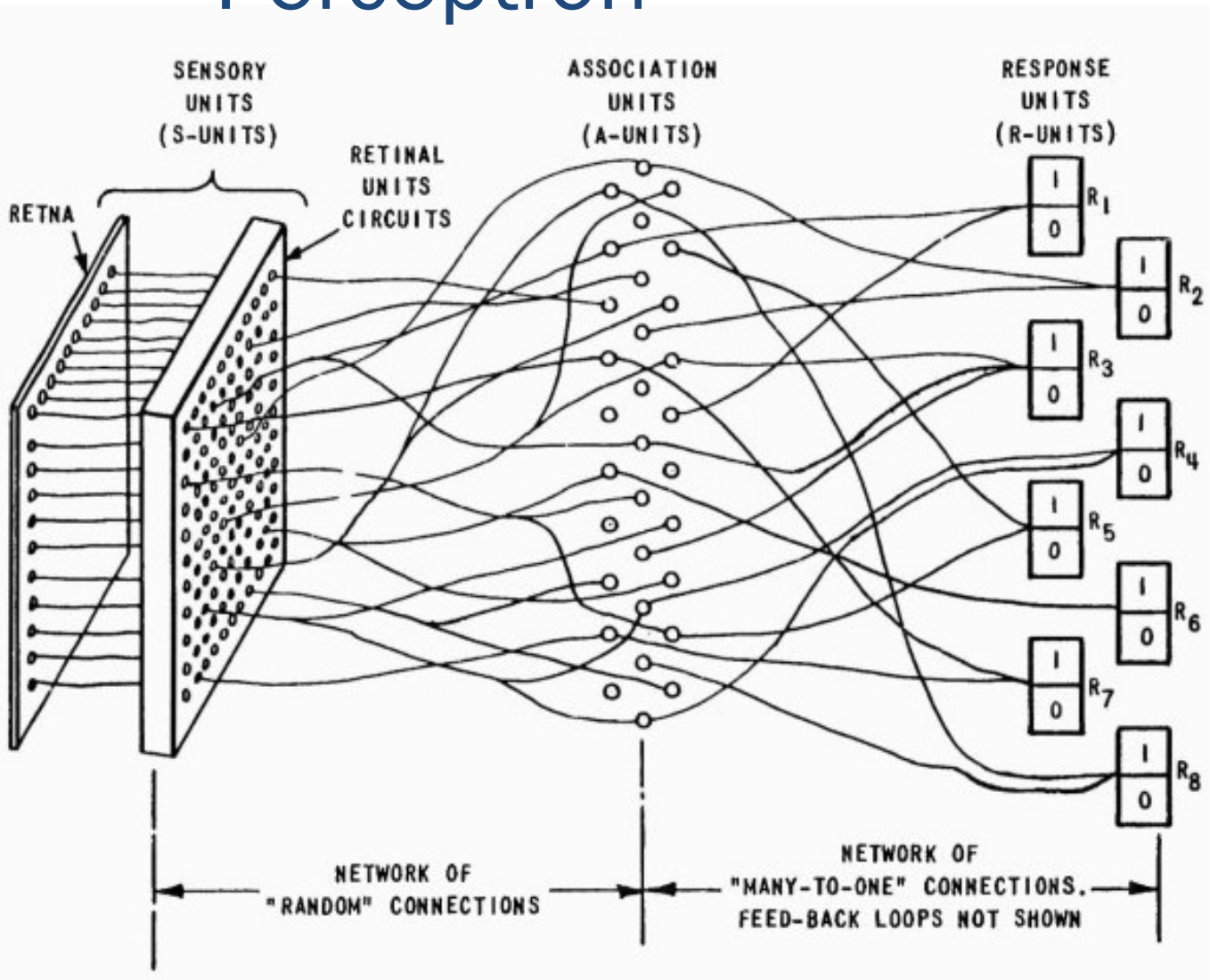
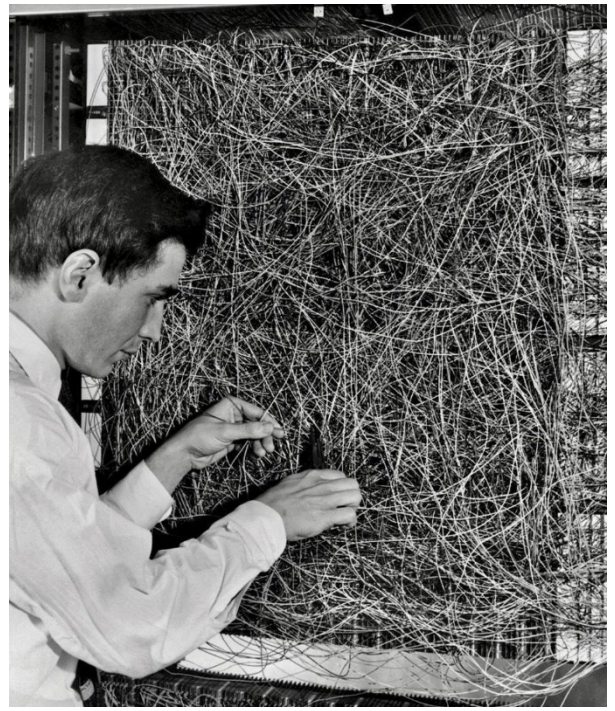
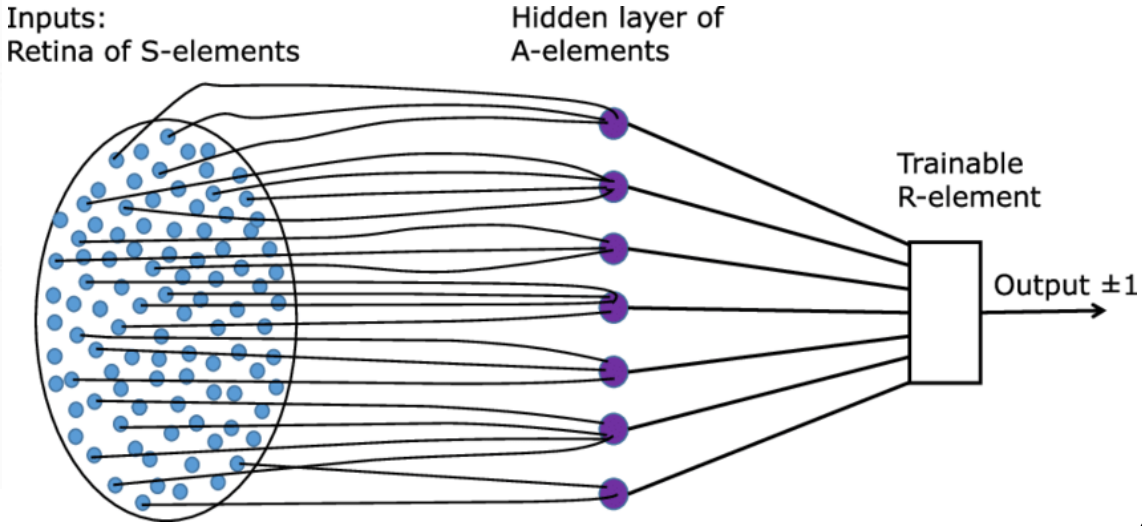


Figure 1 ORGANIZATION OF THE MARK I PERCEPTRON

Perceptron

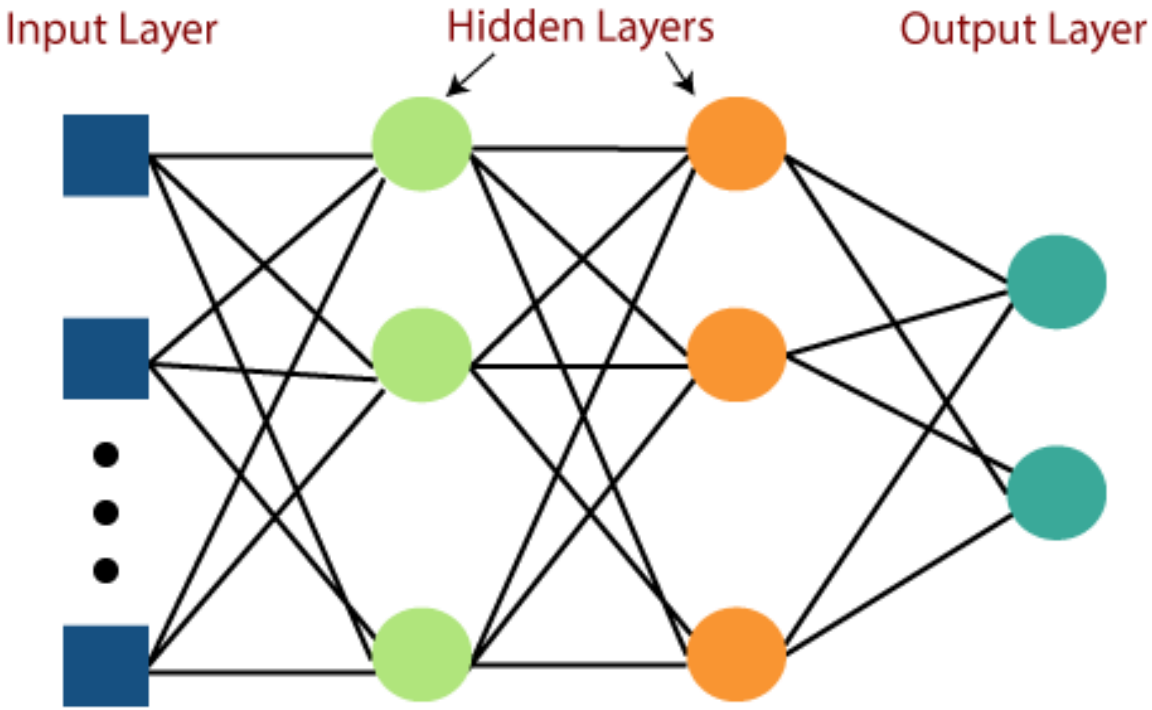

SENSORY UNITS (S-UNITS)
ASSOCIATION UNITS (A-UNITS)
RESPONSE UNITS (R-UNITS)
RETINAL UNITS CIRCUITS
RETNA
R1
R2
R3
R4
R5
R6
R7
R8
NETWORK OF "RANDOM" CONNECTIONS
NETWORK OF "MANY-TO-ONE" CONNECTIONS. FEED-BACK LOOPS NOT SHOWN

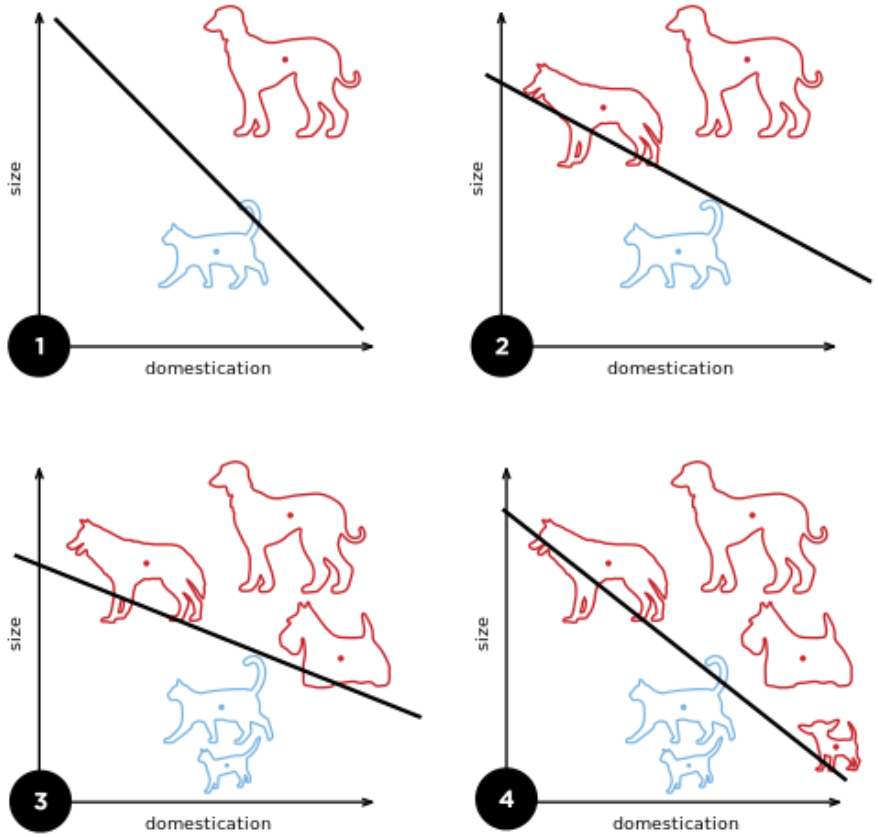



Inputs:
Retina of S-elements
Hidden layer of
A-elements
Trainable
R-element
Output ±1

Multi-layer Perceptron

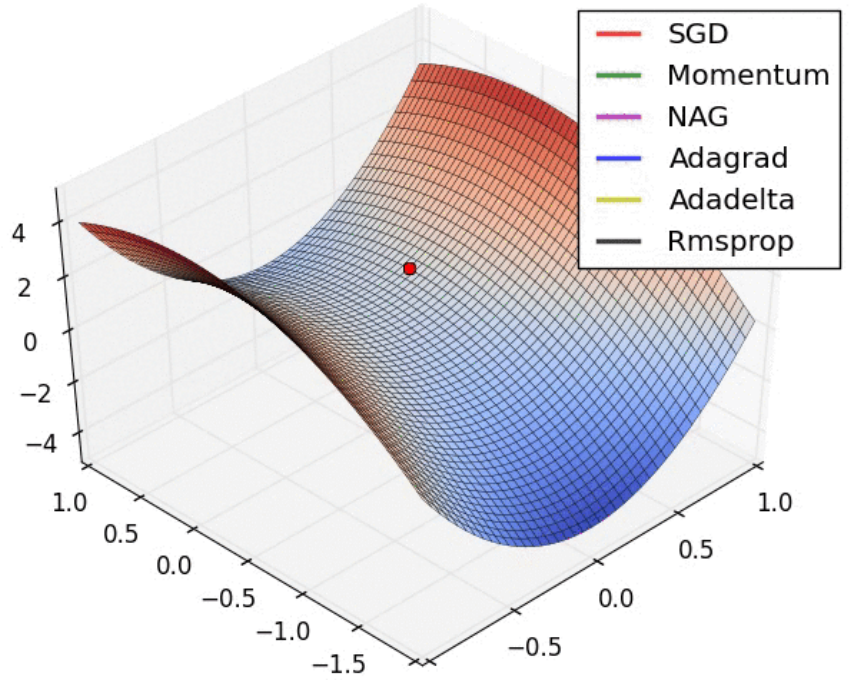
- **1960s** Limitation of Perceptron: Failed to classify basic problems
- Ivakhnenko et. al. introduced Multilayer Perceptrons

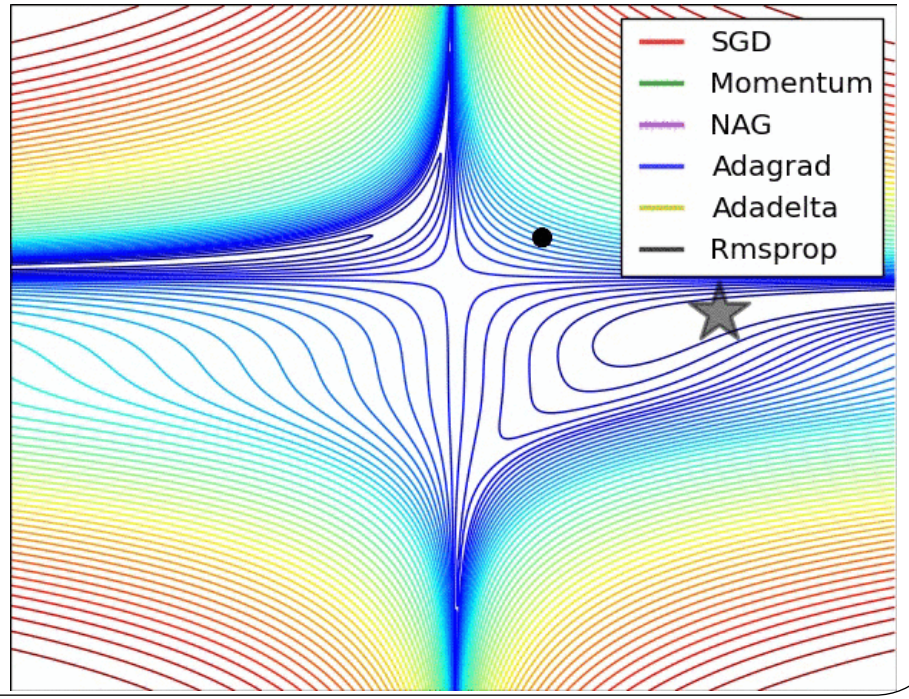


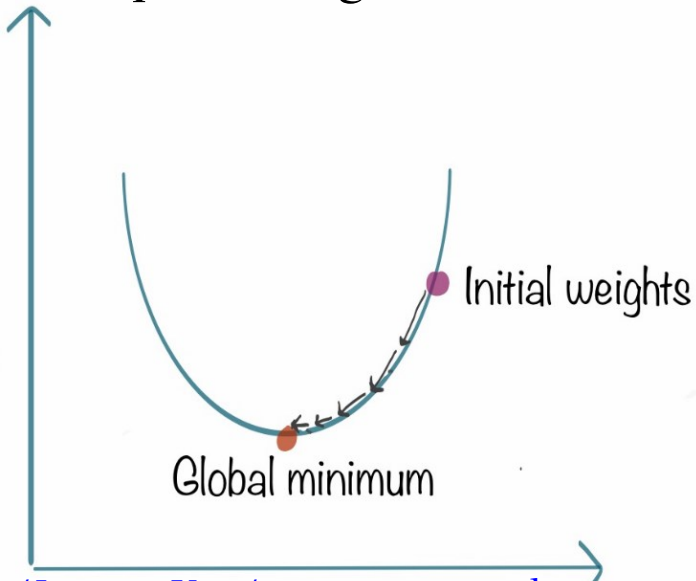


Deep Learning

- For the Probabilities and Expectation equations
  - an optimization algorithm finds the value of the parameters (weights) that minimize the error when mapping inputs to outputs.
  - Optimization algorithms affect the accuracy and the speed of the training a deep learning model.







https://awesomeopensource.com/project/Jaewan-Yun/optimizer-visualization

Deep Learning

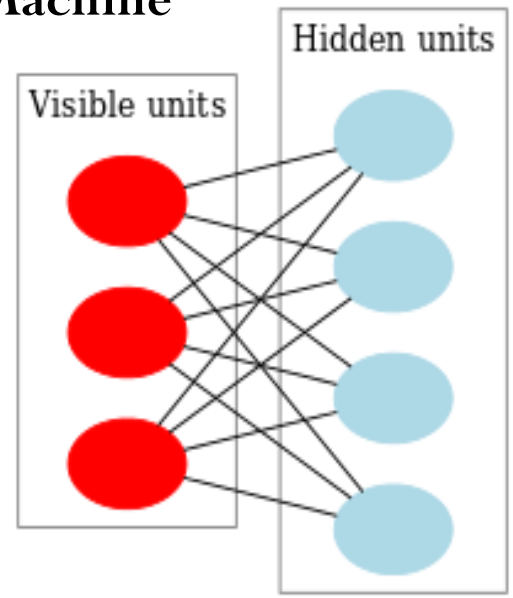
- Deep learning form a Deep Neural Network (DNN)
- DNN is good for deeper level learning based on classification of data with small error rate.
- We used three famous techniques:
  - *Restricted Boltzmann Machines* (*RBM*)
  - *Deep Belief Networks* (*DBN*)
  - *denoising Autoencoders* (*dA*)

These techniques learn from the input matrices to reconstruct an output matrix.

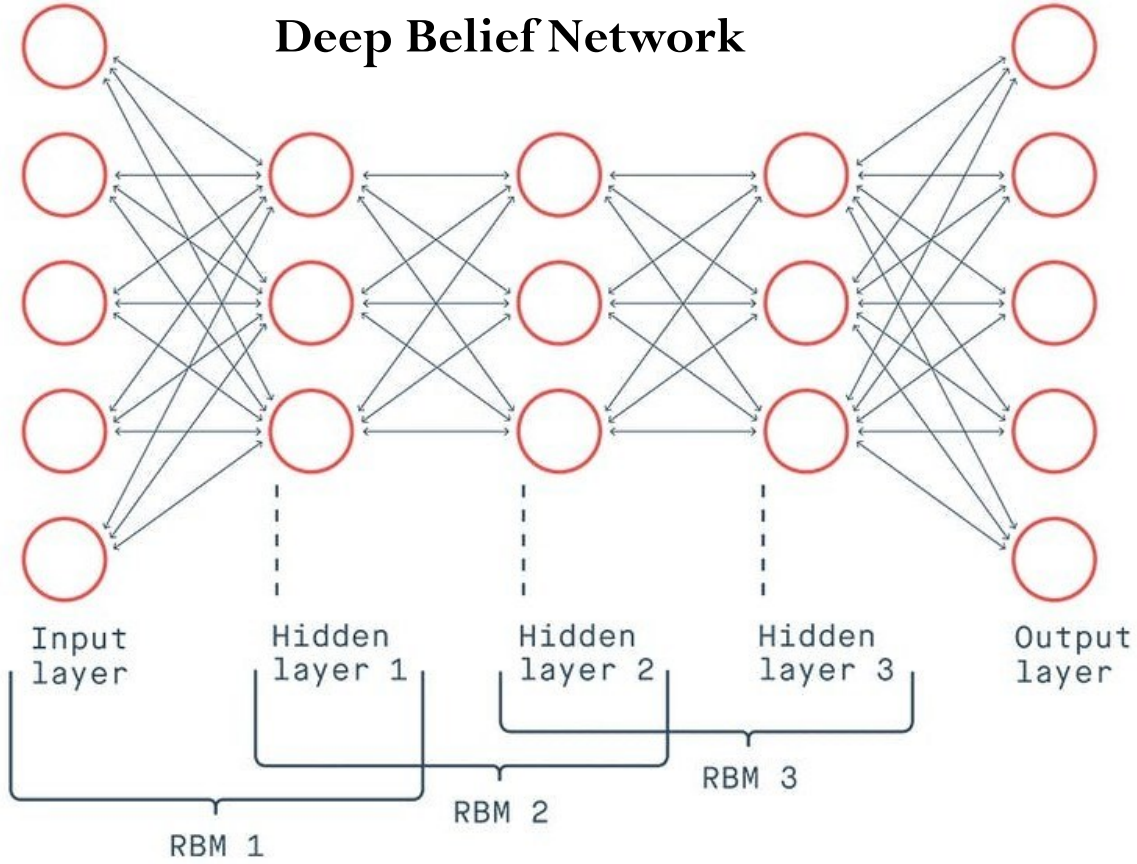
Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." *Nature* 521.7553 (2015): 436-444.

Restricted Boltzmann Machine and Deep Belief Network

Restricted Boltzmann Machine



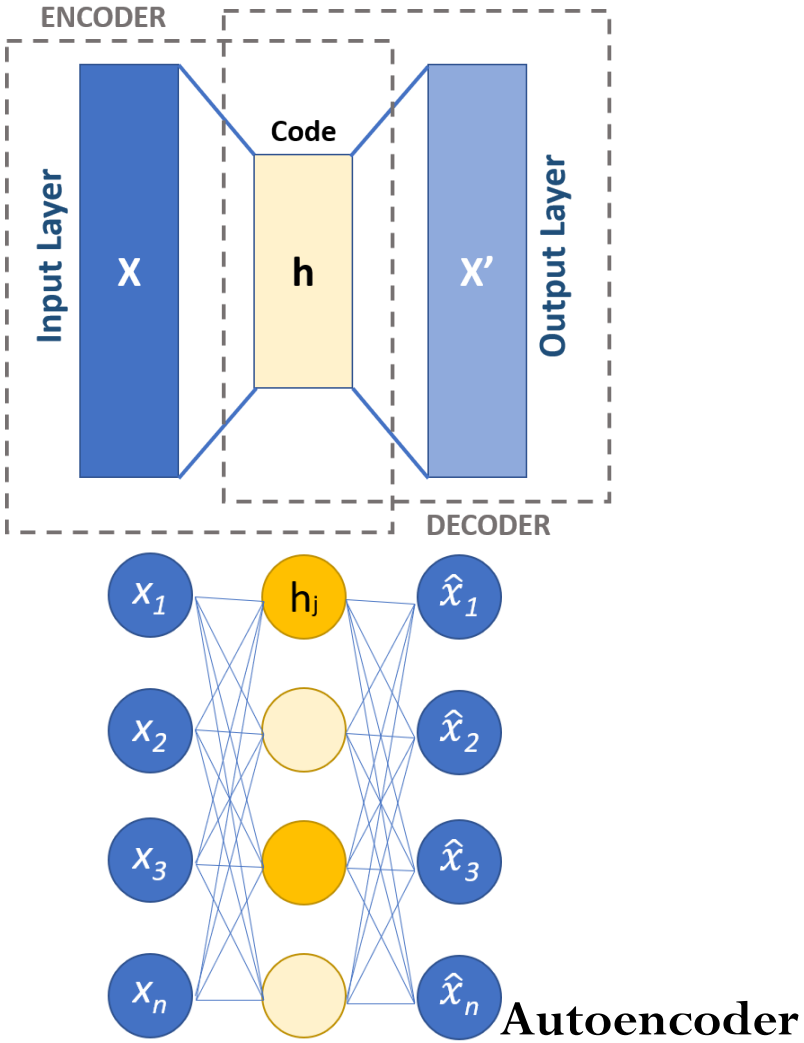
Deep Belief Network



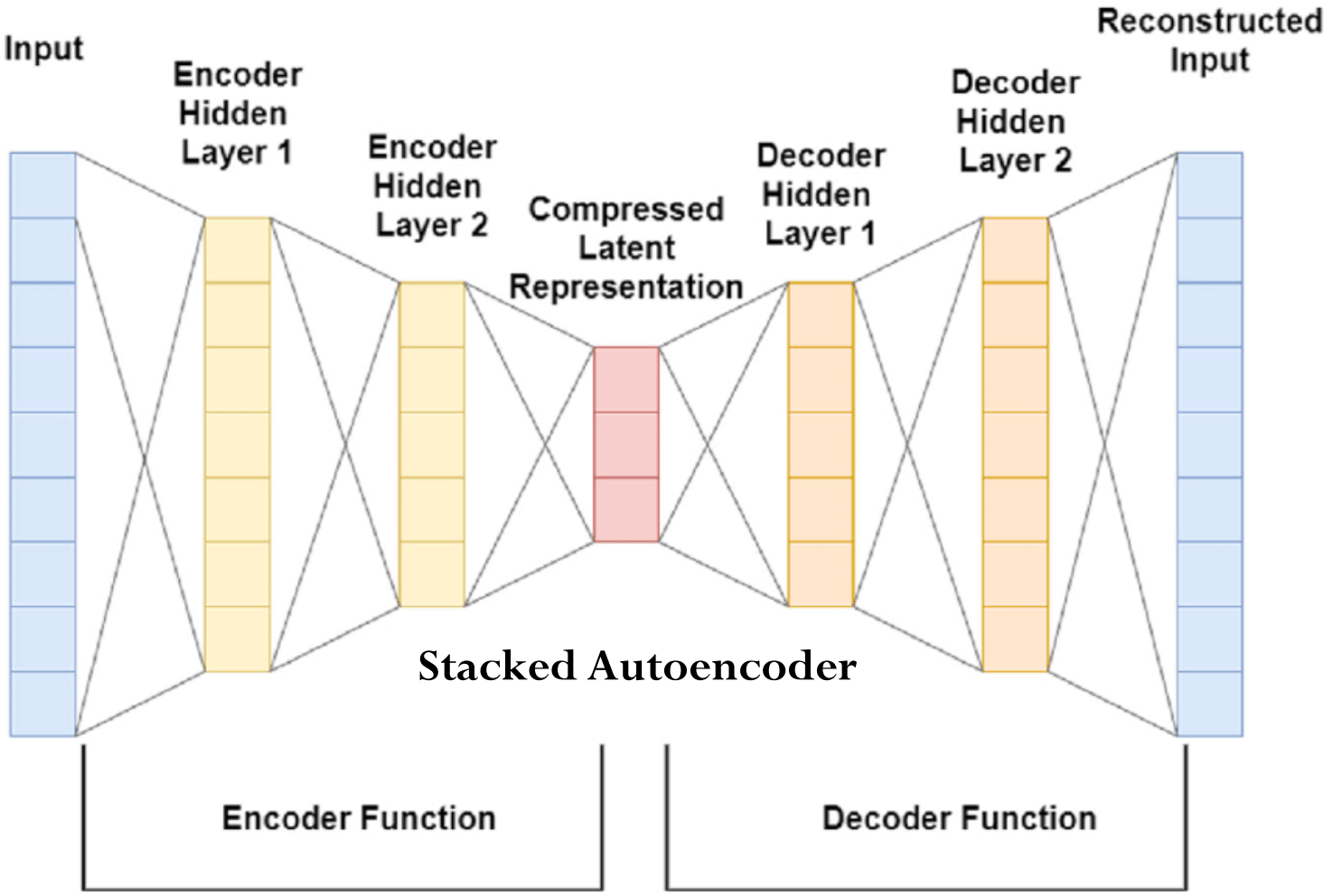
Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. "Restricted Boltzmann machines for collaborative filtering." Proceedings of the 24th International Conference on Machine learning. 2007.

Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. "A fast learning algorithm for deep belief nets." Neural computation 18.7 (2006): 1527-1554.

Autoencoder and Stacked Autoencoder



Autoencoder

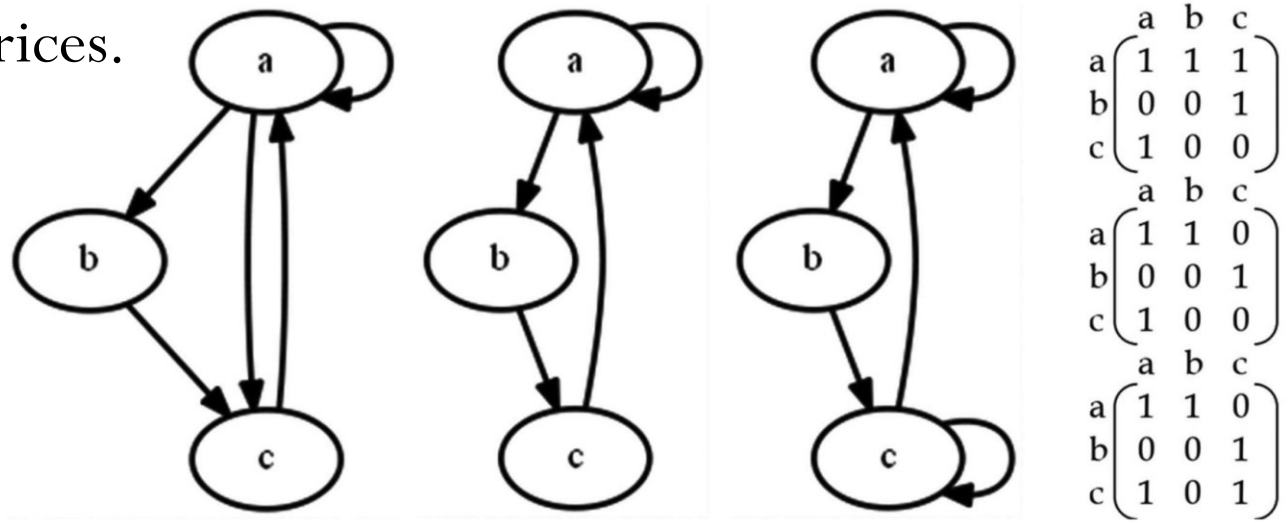


Stacked Autoencoder

Pascal Vincent, et al. "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion." Journal of Machine Learning Research 11.12 (2010).

System Graph Evolution Learning

- Graph evolution example, when it is evolved in three incremental states with three similar temporal patterns and matrices.



- To do System Graph Evolution Learning, we extended 3 deep learning (RBM, DBN, and dA) as three different deep evolution learning approaches.
- For each evolving system, we used the 3 *deep evolution learning* approaches separately, which resulted in three different ***System Neural Networks***.

Remodeling of Equations

$$g(\mathbf{ER}) = P(\boldsymbol{h}|\mathbf{EDSM}) = \prod_{l=1}^{L} P(\boldsymbol{h_l}|\mathbf{EDSM}).$$

Restricted Boltzmann Machines (RBM)

$$= \sum_i (a_i\, ec_{jk}) + \sum_l (b_l\, h_l) + \sum_i \sum_l (ec_{jk}\, w_{il}\, h_l)$$

$$\frac{d \log P(\boldsymbol{h}|\mathbf{EDSM})}{d\boldsymbol{W}_{jkl}} \approx < ec_{jk} h_l >_{\text{data}} - < ec_{jk} h_l >_{\text{reconstruct}}$$

ec_{jk} means entity connection between jth row and kth column of Evolving Design Structured Matrix (EDSM)

Deep Belief Networks (DBN)

$$g(\mathbf{ER}) = P\left(\mathbf{EDSM}, h^1,\ h^2 \ldots h^L\right) = \left(\prod_{l=0}^{L-2} P(h^l|h^{l+1})\right) P\left(h^{L-1}, h^L\right)$$

Autoencoder

$$\text{RE} = -\log P\left(\text{EDSM}|c\left(\boldsymbol{EDSM}\right)\right)$$

$$\text{RE} = -\sum_i \text{ec}_{jk} \log f_i\left(c\left(\boldsymbol{EDSM}\right)\right) + (1 - \text{ec}_{jk}) \log\left(1 - f_i\left(c\left(\boldsymbol{EDSM}\right)\right)\right)$$

Denoising Autoencoder dA

$$RE = -\log\left(g\left(\mathbf{ER}\right)\right) = -\log P(\mathbf{EDSM}|c(\widetilde{\mathbf{EDSM}}))$$

System Structure Learning (SSL) that uses **Deep Evolution Learner** (DEL)

Evolution Representor

Evolving Design Structure Matrix (EDSM), where $vector_i$ represents i^{th} state

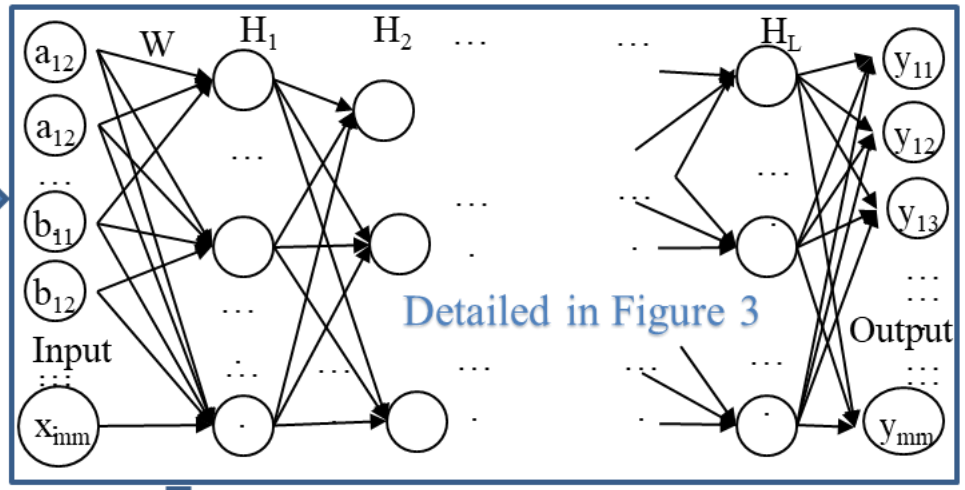
$vector_1$ $[\{a_{11}.. a_{jk} \dots a_{1m}\}, \quad \{a_{21}.. a_{jk} \dots a_{2m}\} \dots \quad \{a_{m1}.. a_{jk} \dots a_{mm}\}]$

$vector_2$ $[\{b_{11}.. b_{jk} \dots b_{1m}\}, \quad (b_{21}.. b_{jk} \dots b_{2m}) \dots \quad (b_{m1},.. b_{jk} \dots b_{mm})]$

$vector_N$ $[(x_{11}.. x_{jk} \dots x_{1m}), \quad (x_{21}.. x_{jk} \dots x_{2m}) \dots \quad (x_{m1}.. x_{jk} \dots x_{mm})]$

DEL

Deep System Neural Network (Deep SysNN)



DEL

0	0	….	0

zero_vector (m × m)

Conversion

y_{11}	y_{12}	….	y_{mm}

output_vector (m × m)

Conversion

0	0	….	0
….	….	….	….
….	….	….	….
….	….	….	….
0	….	….	0

Zero Matrix of size (m × m)

y_{11} , y_{12} … y_{1k} … y_{1m}
…. …. …. ….
y_{j1} , y_{j2} , … y_{jk} … y_{jm}
…. …. …. ….
y_{m1} , y_{m2} … y_{mk} … y_{mm}

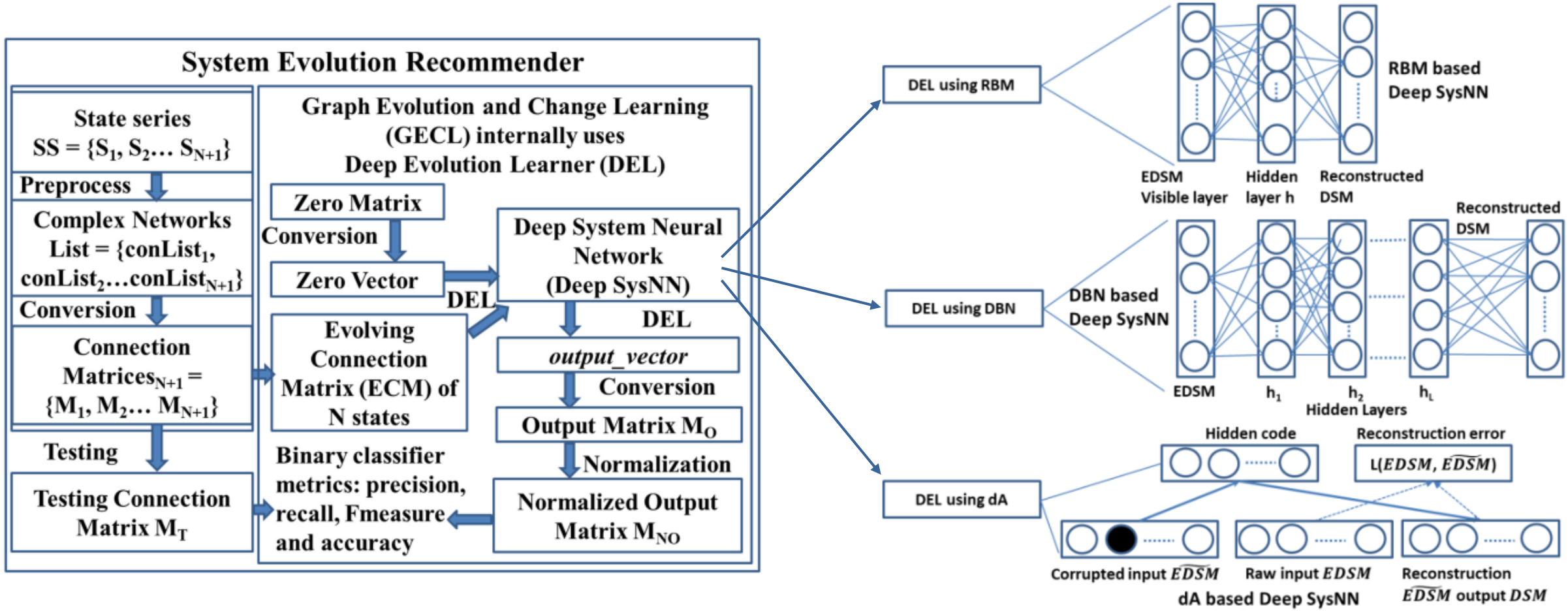
Output Matrix M_O (m × m)

Normalization

n_{11} , n_{12} … n_{1k} … n_{1m}
…. …. …. ….
n_{j1} , n_{j2} , … n_{jk} … n_{jm}
…. …. …. ….
n_{m1} , n_{m2} … n_{mk} … n_{mm}

Symbol	Description
SS	($vector_1$, $vector_2$ …. $vector_N$)
ER	EDSM = f(SS)
$\mathbf{a_{jk}}$ $\mathbf{b_{jk}}$ $\mathbf{x_{jk}}$	Elements of EDSM
input	*zero_vector*
g(**ER**)	H_1, H_2, …. H_L
Y	*output_vector* (m × m)
$\mathbf{M_{NO}}$	binary output matrix (m × m)

System Graph Evolution Learning



System Structure Learning

$$\mathbf{ER} = \mathrm{f}(\mathbf{SS})$$

$$\mathbf{ER} = \mathrm{f}(\{a_{11}..\, a_{jk}..\, a_{mm}\}, \{b_{11}..\, b_{mm}\}\ \ldots\ \{x_{11}..\, x_{jk}..\, x_{mm}\})$$

$$\mathrm{EDSM} = \mathrm{f}(\mathbf{SS}) = \boldsymbol{evolutionRepresentor}\,(N_{DSMs})$$

Algorithm ***SSL***(*N_DSMs*)

Initialize a zero matrix as M_Z
zero_vector = ***matrixToRowVector***(M_Z)
EDSM = ***evolutionRepresentor***(*N_DSMs*)
output_vector = ***deep_evolution_learner*** (*EDSM*, *zero_vector*)
M_O = ***rowVectorToMatrix***(*output_vector*)
M_{NO} = ***normalize***(M_O)
Return M_{NO}

Algorithm ***evolutionRepresentor***(*N_DSMs*)

For each *DSM_i* in *N_DSMs* where i ∈ state number
vector_i = ***matrixToRowVector***(*DSM_i*)

$$EDSM = \left\{ \begin{array}{c} vector\_i \\ + \\ EDSM \end{array} \right\}$$

End for
Return *EDSM*

Algorithm ***deep_evolution_learner*** (*EDSM*, *zero_vector*)

Initialize List *trainParameters*< LR, Ep, L >
Initialize a matrix *Deep_SysNN*
Deep_SysNN = ***delTrain***(*EDSM*, *trainParameters*)
// three variant of Deep SysNN based on Equation 4 to 10
output_vector = ***delReconstruct***(*Deep_SysNN*, *zero_vector*)
Return *output_vector*

$$\mathbf{Y} = \mathrm{g}(\mathbf{ER}) = \mathrm{g}(\mathrm{f}(\mathbf{SS}))$$

$$\boldsymbol{Y} = (\{y_{11}..\, y_{jk}..\, y_{1\,m}\}, \{y_{21}..\, y_{2\,m}\} \ldots \{y_{m1}..\, y_{jk}\ \ldots y_{mm}\})$$

Experiments on Evolving Systems

INFORMATION ABOUT IMBALANCED DATASET USED

Evolving Systems	Testing matrix (M_T)	Number of 1s	Number of 0s
Hadoop HDFS	Version 2.7.2	2938	9787703
List of Bible Translation	20th Century	46	579
List of Multi-sport Events	201 i.e. 2010-2017 decade	8	188
Frequent Market Basket	1211 i.e. Dec. 2011	617	13307
Positive sentiment of movie genres	201 i.e. 2010-2019 decade	47	578
Negative sentiment of movie genres	201 i.e. 2010-2019 decade	177	1848

Binary Metrics

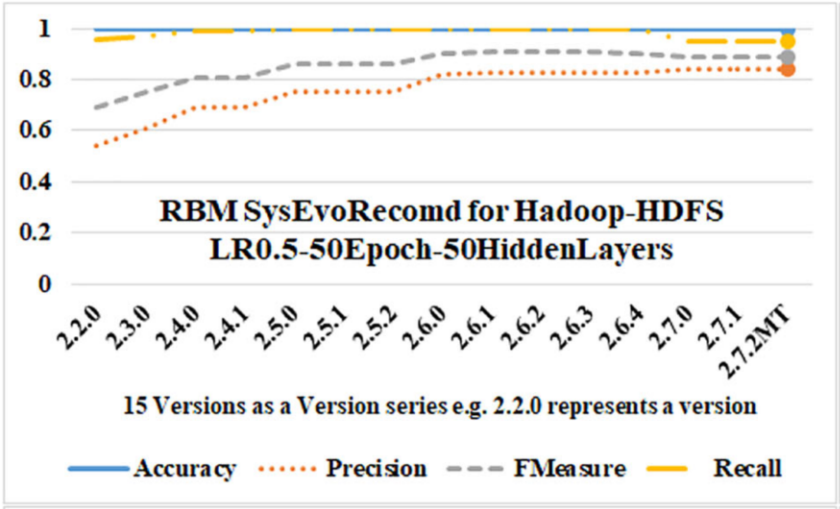
$$\#\text{ denotes “count of”}$$

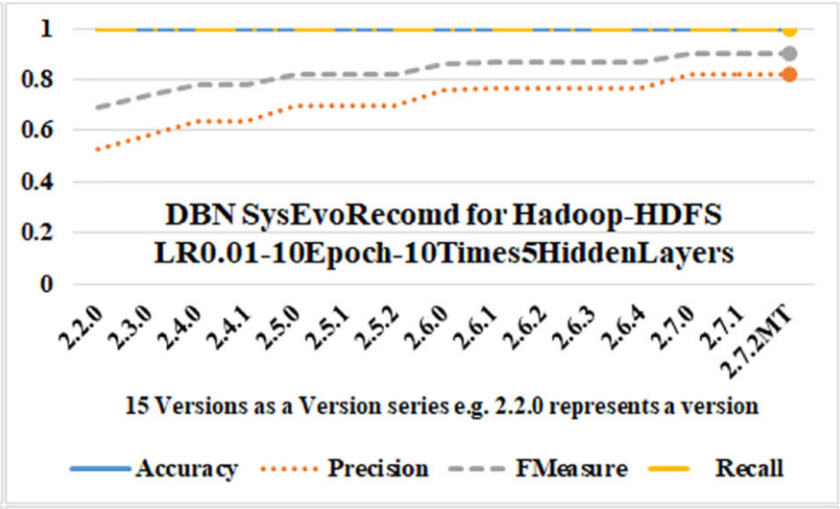
$$\text{accuracy} = \frac{\left(\#\begin{array}{c}\text{correctly recommended connections +}\\ \text{correctly recommended no connections}\end{array}\right)}{\#\text{all possible entity connections i.e. matrix size}}$$

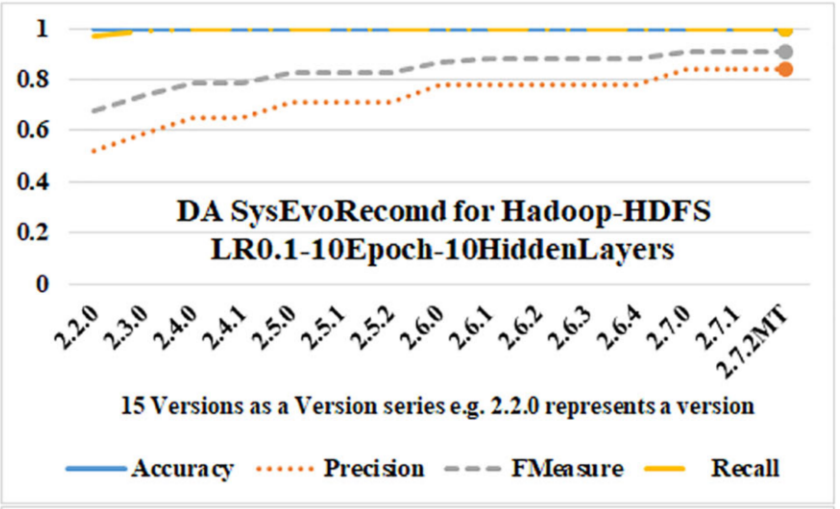
$$\text{precision} = \frac{\#\text{correctly recommended connections}}{\#\text{all connections recommended by the tool}}$$

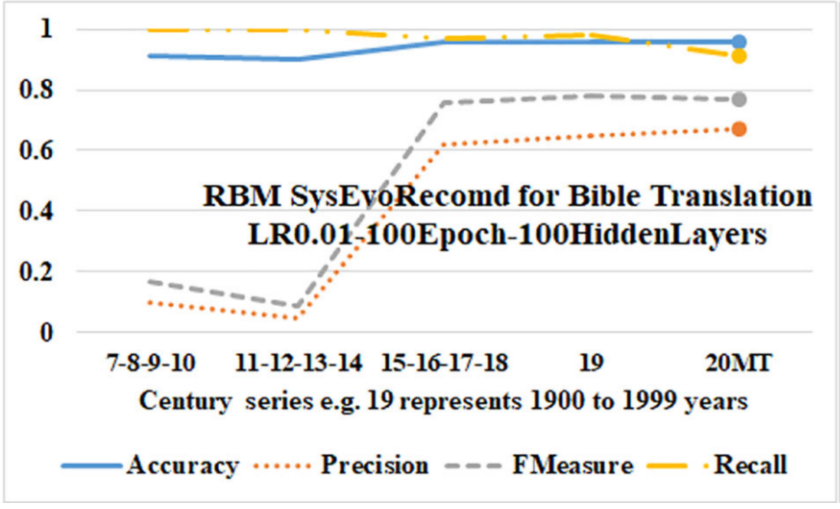
$$\text{recall} = \frac{\#\text{correctly recommended connections}}{\left(\#\begin{array}{c}\text{correctly recomended connections + incorrectly}\\ \text{recommended connections as no connections}\end{array}\right)}$$

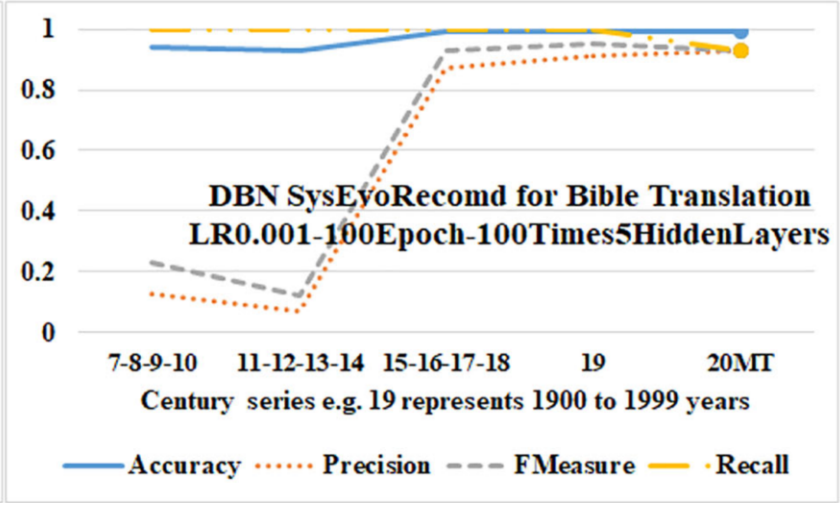
System Evolution Recommendations

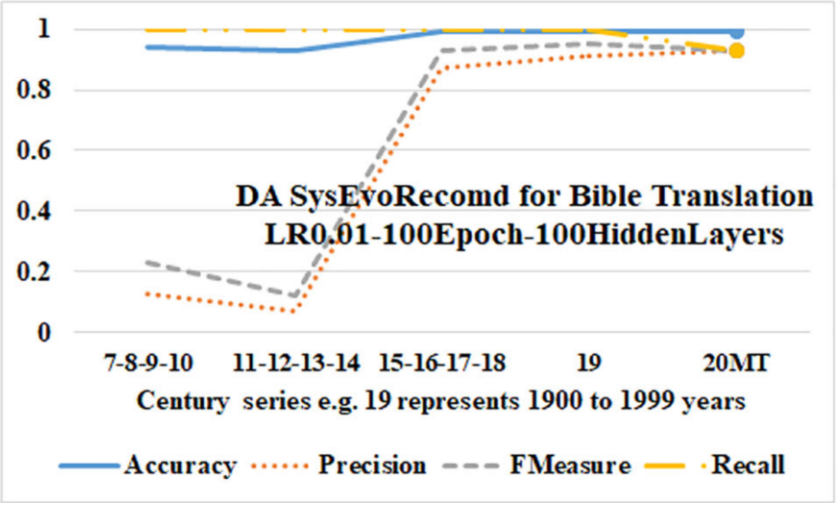




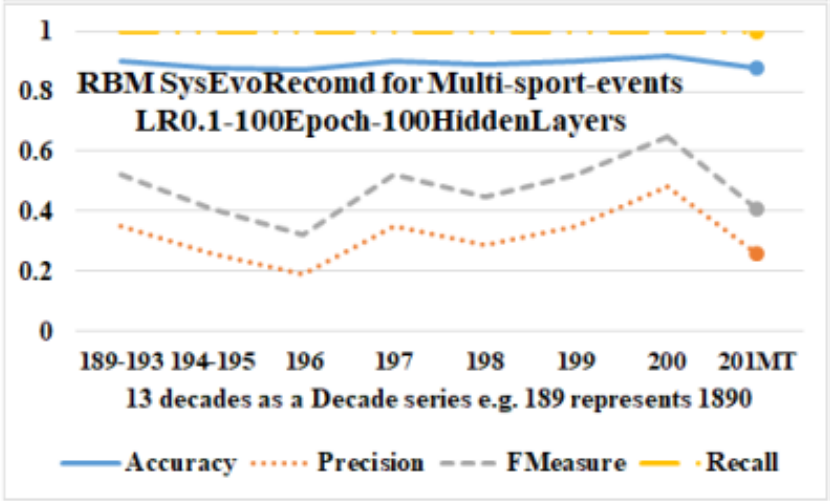


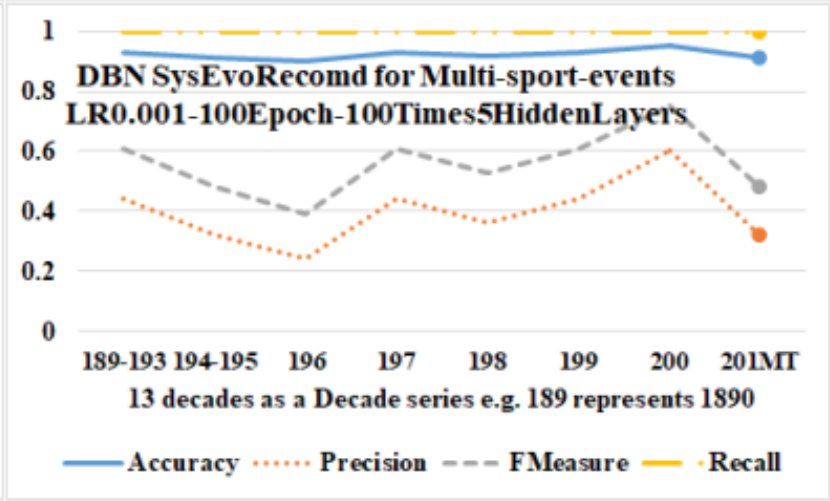


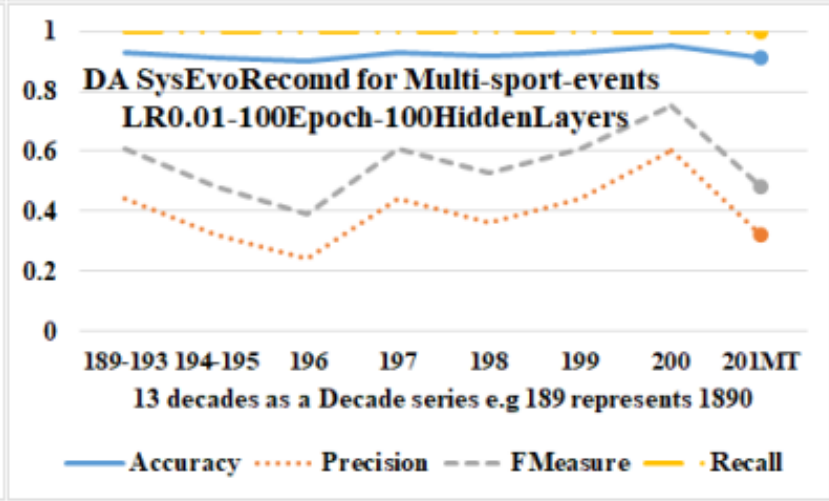


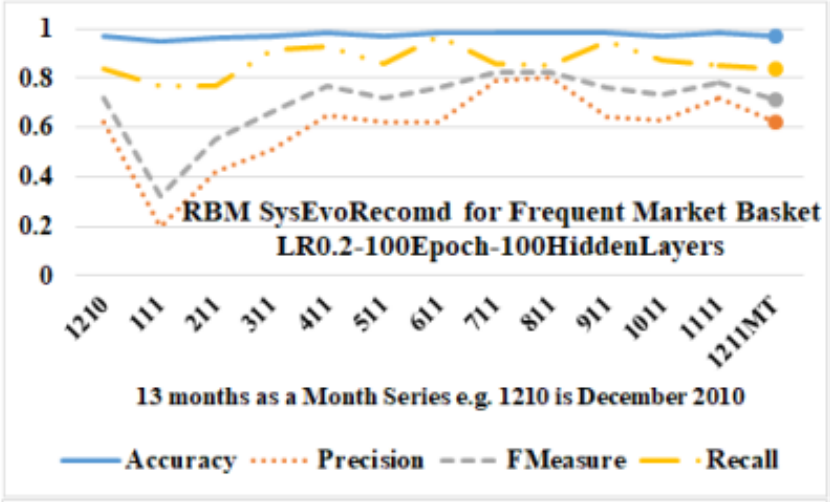


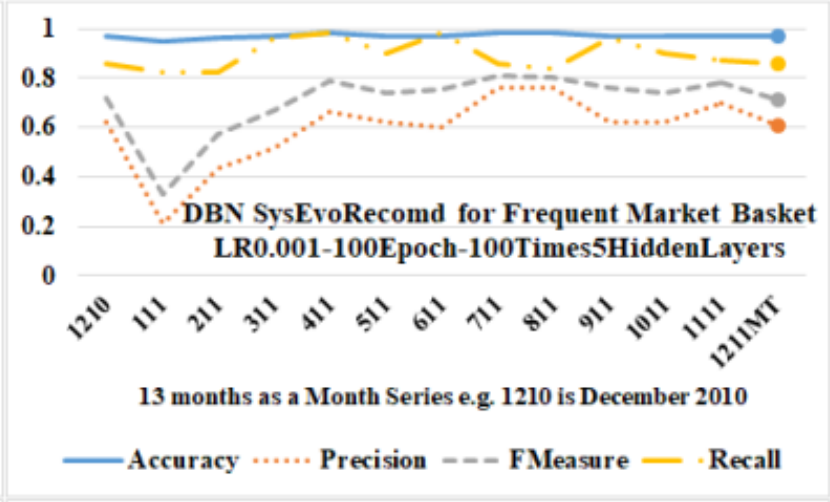
Recommendation

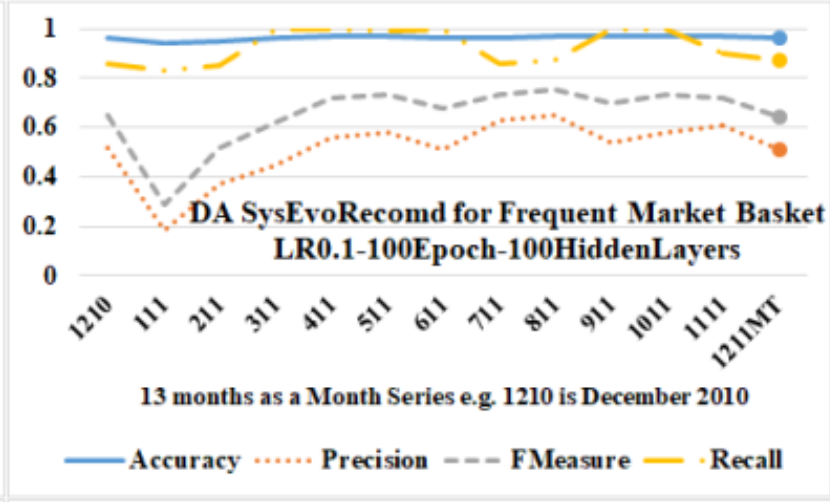




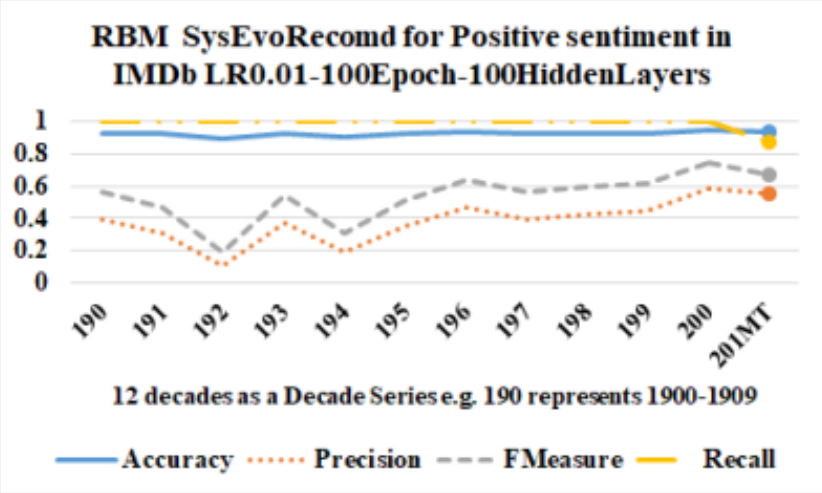


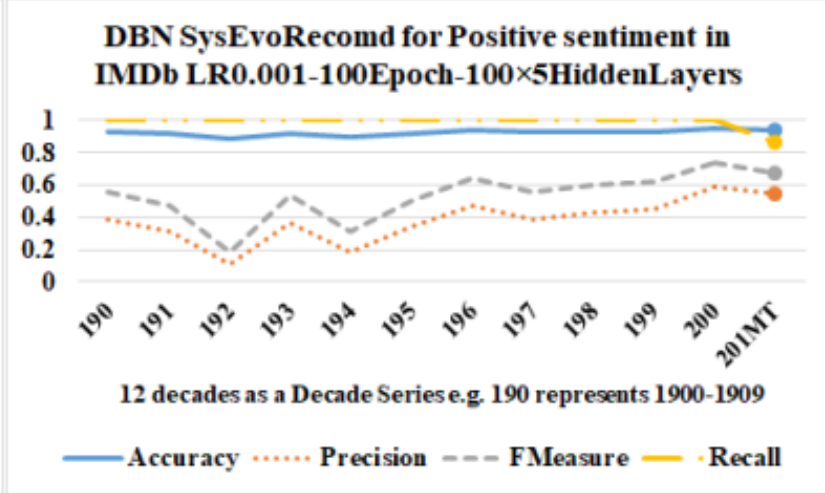


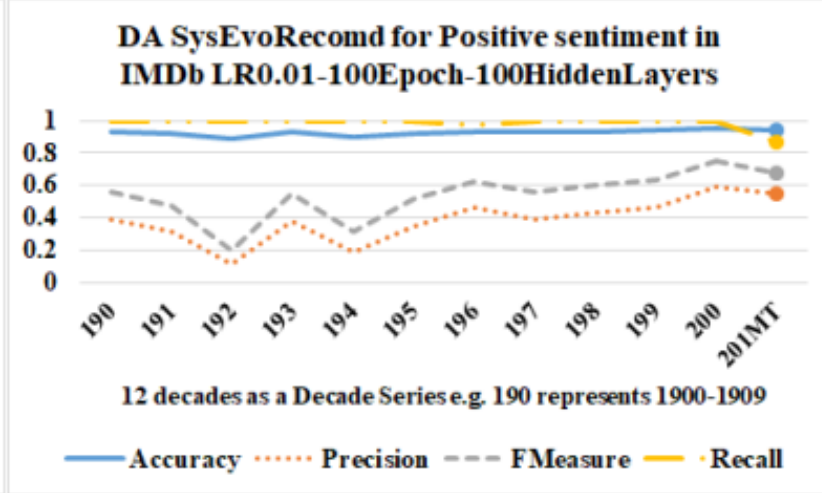


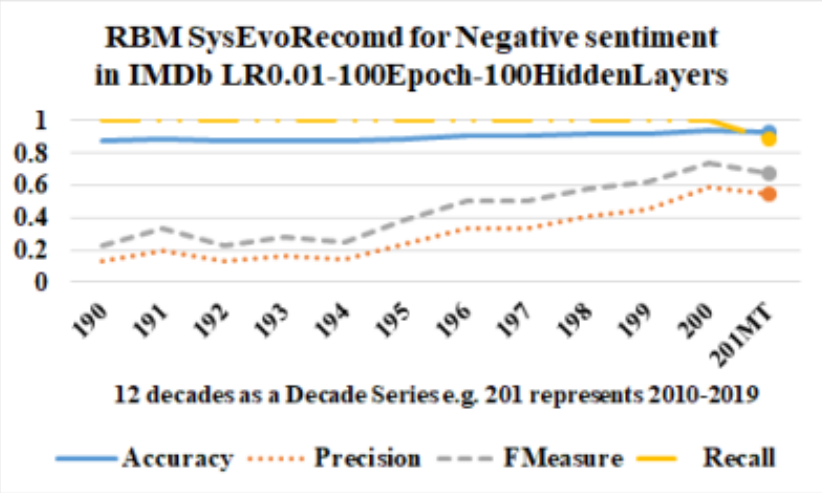


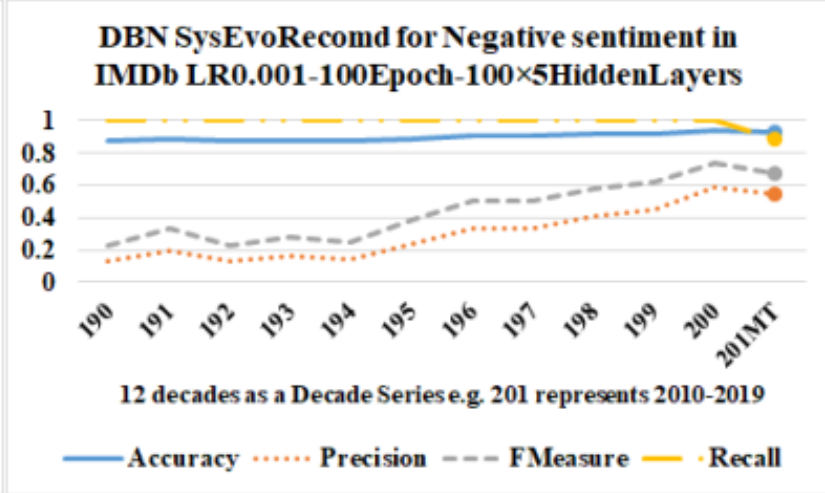
Recommendation

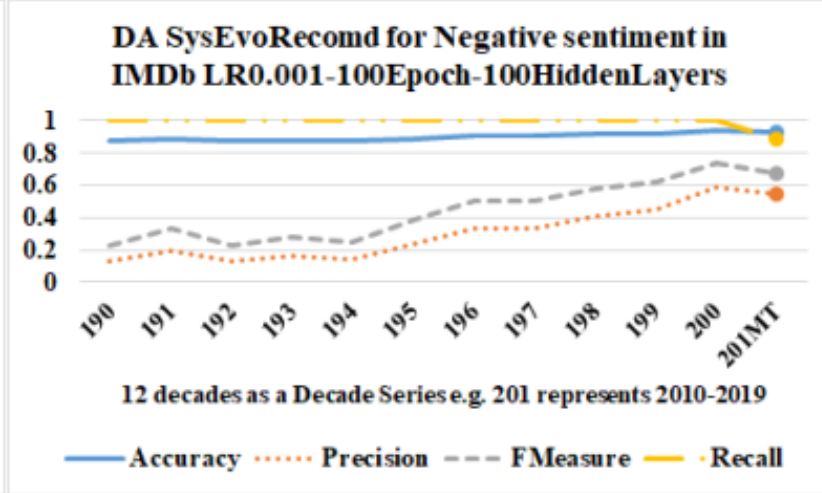








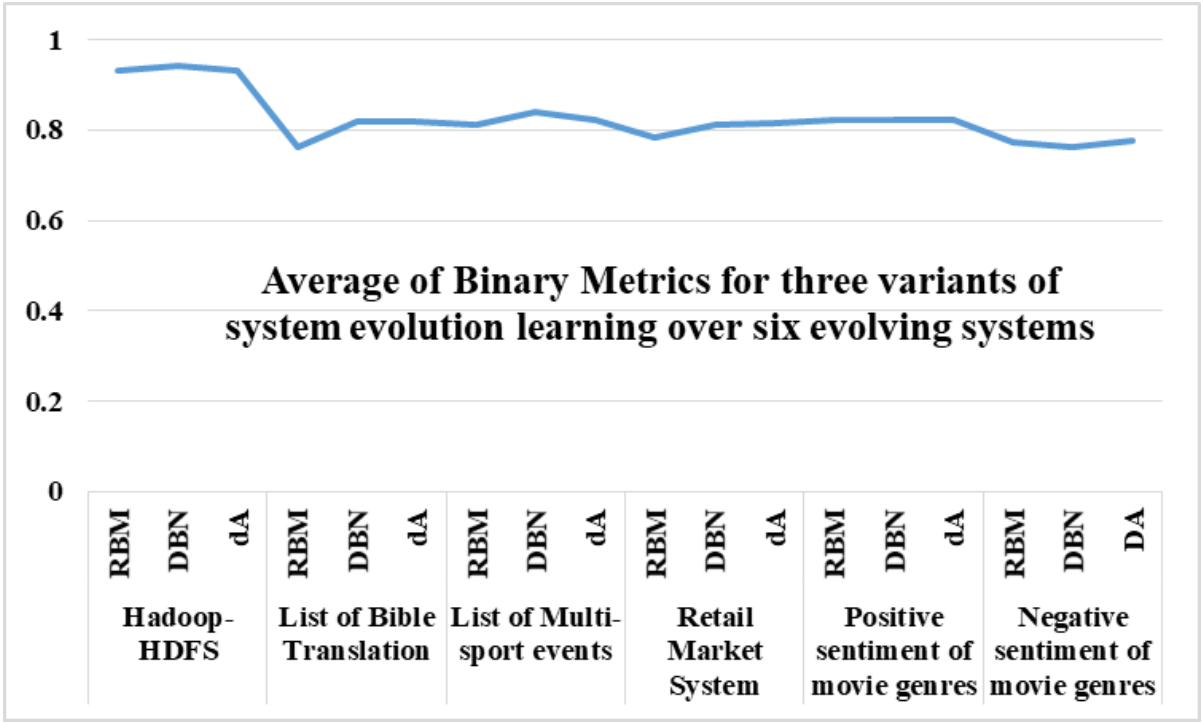




Experiments on Evolving Systems

Evolving systems	Training States (N)	Deep Evolution learning	ABM
Hadoop HDFS-Core	15	RBM	0.888
		DBN	0.886
		dA	0.883
Bible Translation	13	RBM	0.649
		DBN	0.749
		dA	0.749
Multi-sport Events	13	RBM	0.662
		DBN	0.714
		dA	0.696
Retail market system	13	RBM	0.755
		DBN	0.760
		dA	0.737
Positive sentiment of movie genres	16	RBM	0.68
		DBN	0.68
		dA	0.680
Negative sentiment of movie genres	16	RBM	0.623
		DBN	0.639
		dA	0.631

- **Average of Binary Metrics** (ABM)
- $$\text{ABM} = \frac{(\text{accuracy} + \text{precision} + \text{fmeasure} + \text{recall})}{4}$$



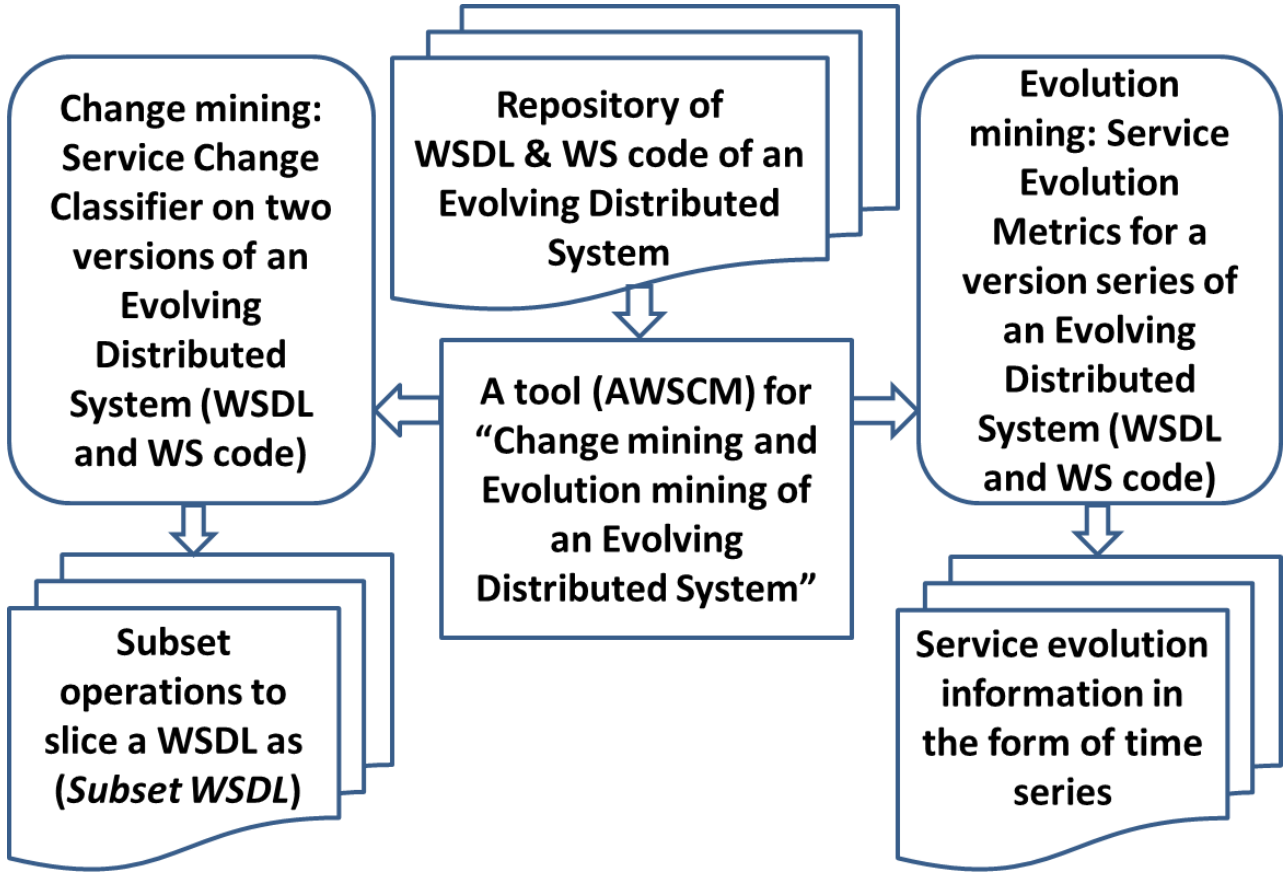
Applied Data Science Research

Service Evolution Analytics: Change and Evolution Mining of a Distributed System

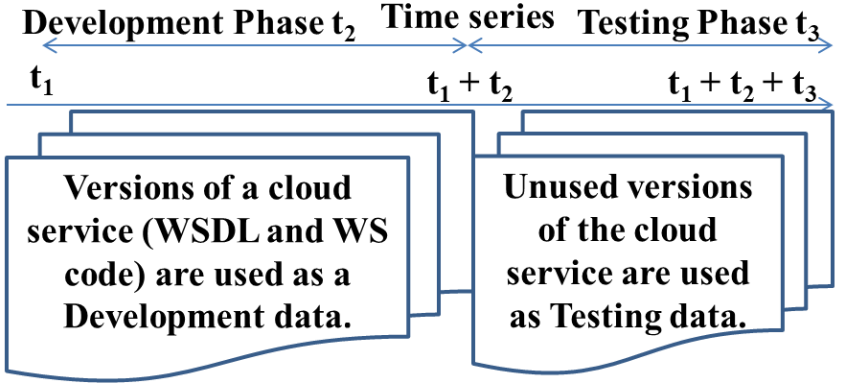
1. **Service Evolution Analytics: Change and Evolution Mining of a Distributed System**
2. minOffense: Inter-Agreement Hate Terms for Stable Rules, Concepts, Transitivities, and Lattices

Service Evolution Analytics Model and Tool

Built an Automated Web Service Change Management (AWSCM) tool that supports change and evolution mining of an evolving distributed system.



Web Services Description Language (**WSDL**) is an **Interface** of a **Web Service** (**WS**)

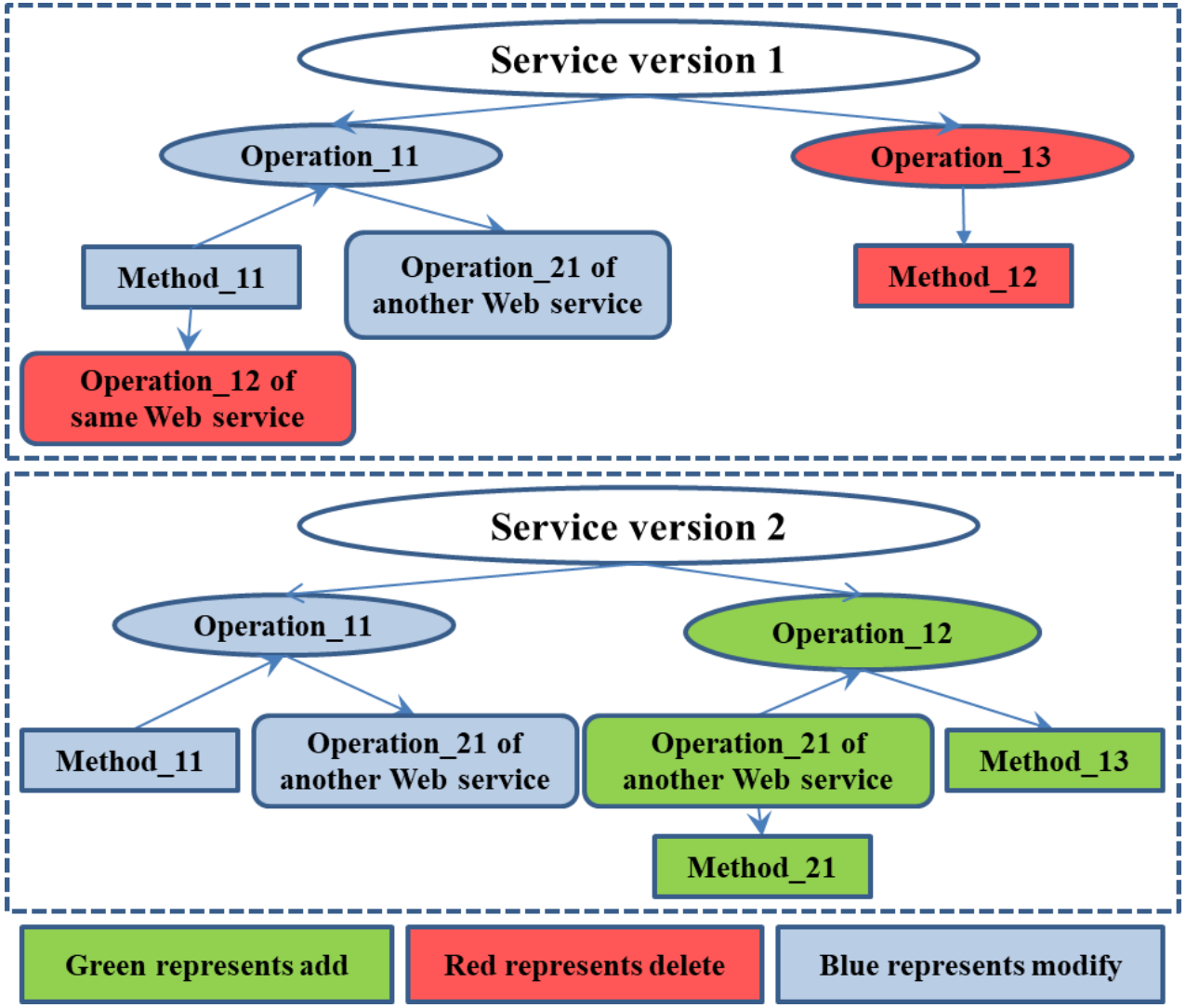


Development and testing phases of the tool.

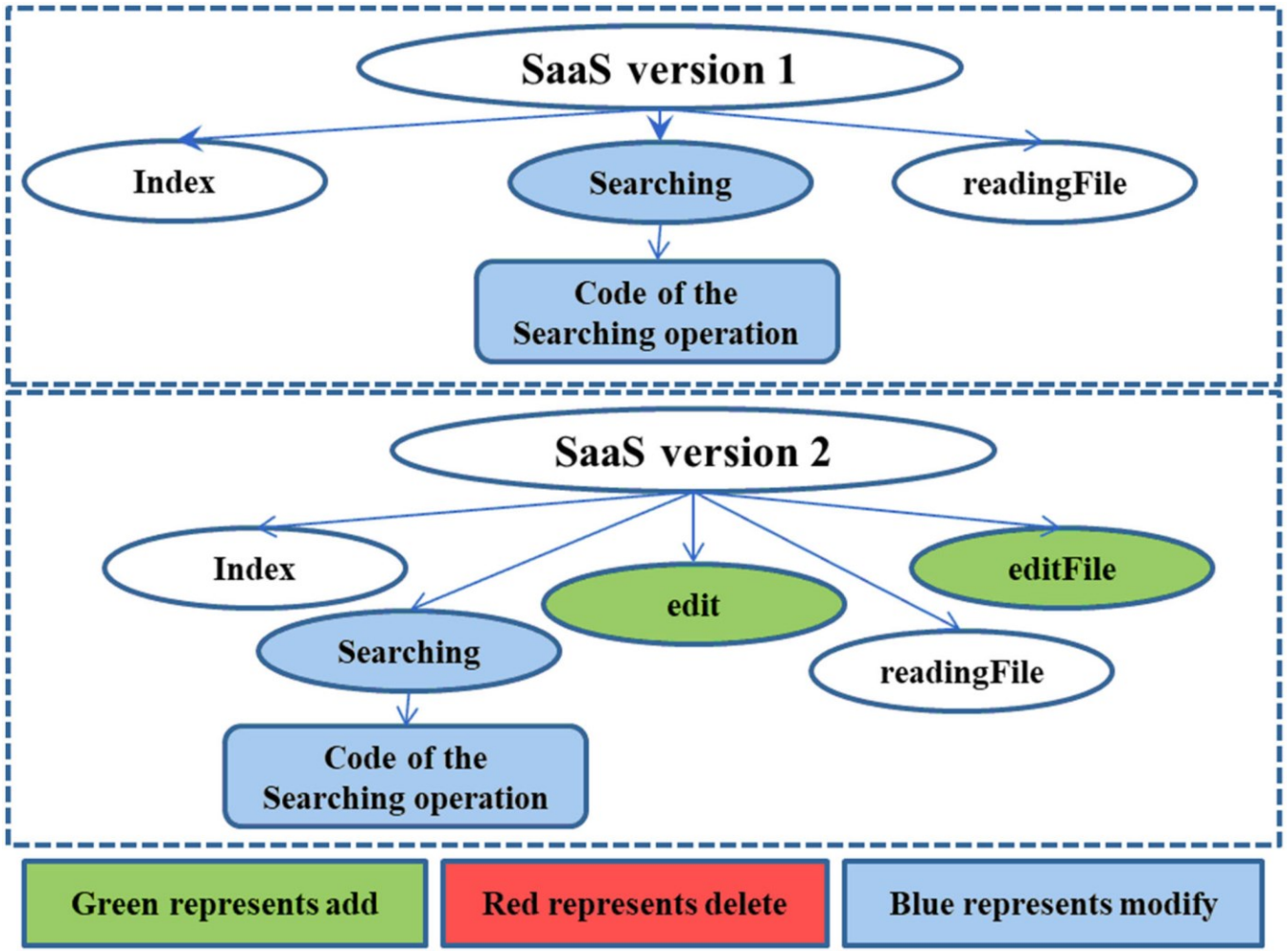
Starting at time t_1 the *development phase* runs for time t_2.

The subsequent testing phase starts at time $t_1 + t_2$ a runs for time t_3. Assuming the both phases end at time $t_1 + t_2 + t_3$.

Change mining of two versions: labeling

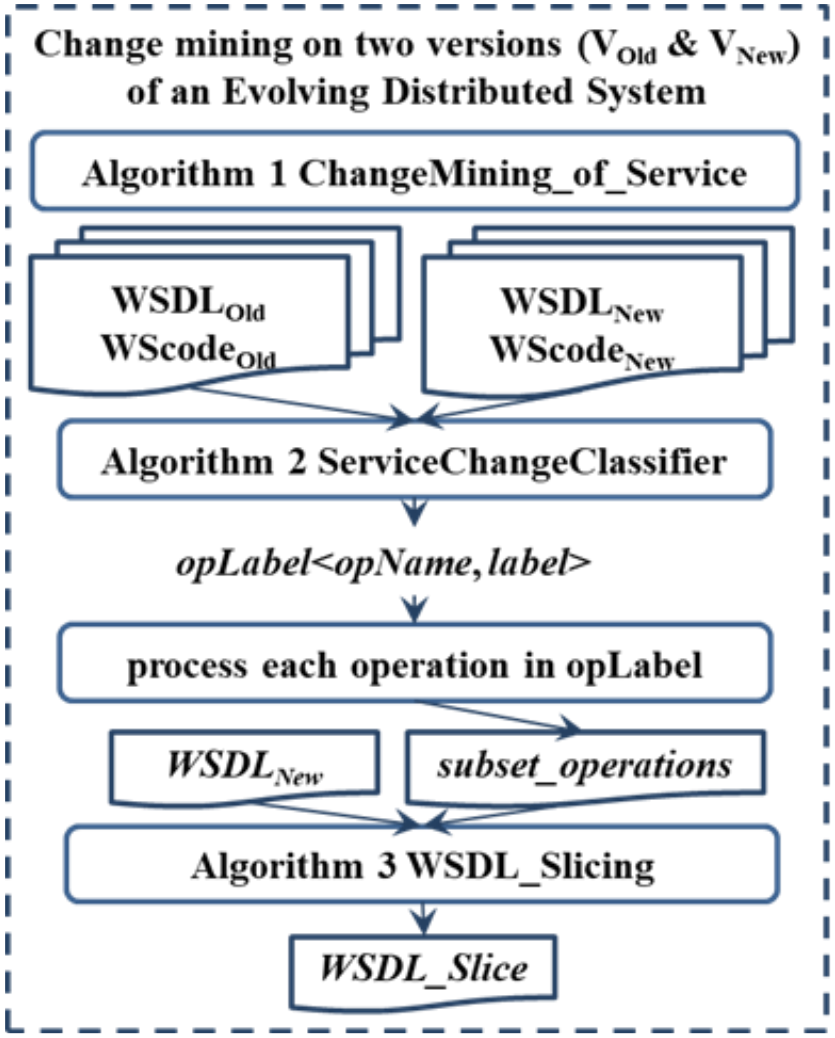


Change mining of two versions: labeling



Service Change Classification

- **Subset WSDL** or **WSDL Slice** for Change analysis
  - a *service change classifier* algorithm to assign change labels to a service's operations and then extracts a *WSDL slice*.



ChangeMiningOfService ($WSDL_{Old}$, $WSDL_{New}$, $WScode_{Old}$, $WScode_{New}$)

1. *opLabels* = ***ServiceChangeClassifier*** (*$WSDL_{Old}$*, *$WSDL_{New}$*, *$WScode_{Old}$*, *$WScode_{New}$*)
2. Initialize an empty set *subset_operations*
3. **For each** pair < *opName*, *label* > in *opLabels*
   a. **If** *label* = = "*inserted*" or *label* = = "*modified*"
   subset_operations ← *subset_operations* ∪ *opName*

End for

4. *WSDL_Slice* = ***WSDL_Slicing*** (*$WSDL_{New}$*, *subset_operations*)

Return *WSDL_Slice*

Algorithm 1: ***ServiceChangeClassifier*** (*$WSDL_{Old}$*, *$WSDL_{New}$*, *$WScode_{Old}$*, *$WScode_{New}$*)

1. *$WScode_{Changes}$* = ***semanticCodeDiff***(*$WScode_{Old}$*, *$WScode_{New}$*)
2. *$WSDL_{Changes}$* = ***semanticWSDLDiff***(*$WSDL_{Old}$*, *$WSDL_{New}$*)
3. Initialize array list *opLabels*<*opName, label*> with *unchanged* label, and retrieve *opName_change* from *$WScode_{Changes}$* and *$WSDL_{Changes}$* along with the change labels.
4. **For** each *opName_change* in *$WSDL_{Changes}$*
   a. **If** (*opName_change* has label *insert*)
   Assign label *inserted* to *opName* in *opLabels*
   b. **If** (*opName_change* has label *delete*)
   Assign label *deleted* to *opName* in *opLabels*
   c. **If** (*opName_change* has label *modify*)
   Assign label *modified* to *opName* in *opLabels*

End For

5. **For** each *opName_change* in *$WScode_{Changes}$*
   a. **If** (*opName_change* has label *insert*)
   Assign label *inserted* to *opName* in *opLabels*
   b. **If** (*opName_change* has label *delete*)
   Assign label *deleted* to *opName* in *opLabels*
   c. If (opName_change has label *modify*)
   Assign label *modified* to *opName* in *opLabels*

End For

Return *opLabels*<*opName*, *label*>

Algorithm 2: ***WSDL_Slicing*** (*$WSDL_{New}$*, *subset_operations*)

String *cStartDef*, *cXSD*, *cMsg*, *cPort*, *cBinding*, *cService*, *cEndDef*

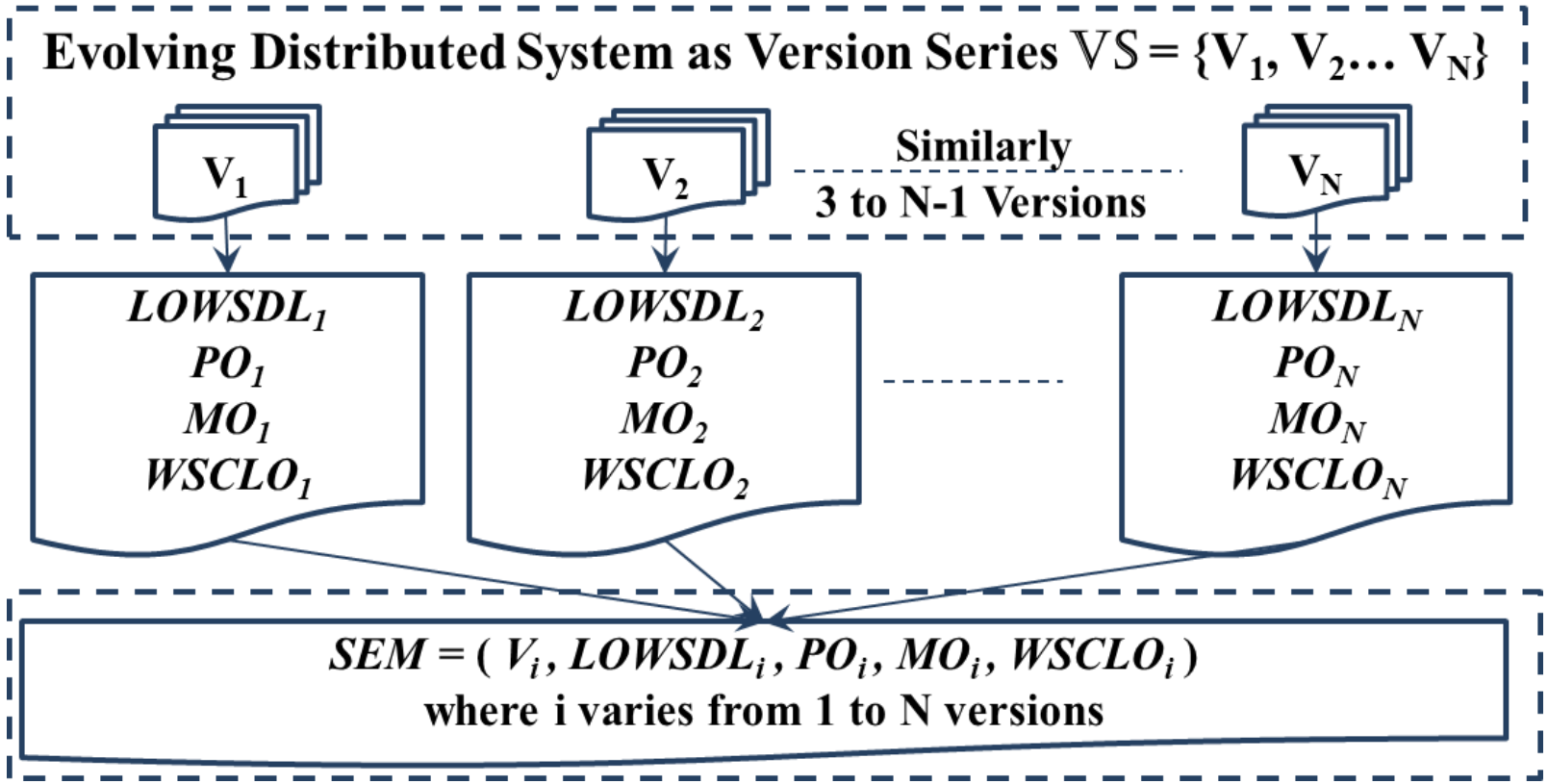
Array of String[] *$opWSDL_{New}$*

1. [*subset_operations* ∈ *$opOfWSDL_{New}$* |
   subset_operations = operation required to construct *WSDL_Slice*]
2. *cStartDef* = **sliceStartDefinition**(*$WSDL_{New}$*)
   cXSD = **sliceXSD**(*$WSDL_{New}$*, *subset_operations*)
   cMsg = **sliceMessage**(*$WSDL_{New}$*, *subset_operations*)
   cPort = **slicePort**(*$WSDL_{New}$*, *subset_operations*)
   cBinding = **sliceBinding**(*$WSDL_{New}$*, *subset_operations*)
   cService = **sliceService**(*$WSDL_{New}$*)
   cEndDef = **sliceEndDef**(*$WSDL_{New}$*)
3. *WSDL_Slice* = *cStartDef* + *cXSD* + *cMsg* + *cPort* + *cBinding* + *cService* + *cEndDef*

Return *WSDL_Slice*}

Evolution mining of a Version series: Service Evolution Metrics

- four *Service Evolution Metrics* to study cloud service evolution and to deduce facts from a version series



Evolution mining of a Version series: Service Evolution Metrics

Four *Service Evolution Metrics* to study cloud service evolution

1. ***Lines per operation in the WSDL***

LOWSDL $=\{(V_1, LOWSDL_1)\ldots (V_i, LOWSDL_i)\ldots (V_N, LOWSDL_N)\}$

$$LOWSDL_i = \frac{Number\ of\ source\ lines\ in\ the\ WSDL\ for\ version\ i}{Number\ of\ operations\ in\ version\ i}$$

2. ***Parameters per operation in the WSDL***

PO $= \{(V_1, PO_1)\ldots (V_i, PO_i)\ldots (V_N, PO_N)\}$

$$PO_i = \frac{Number\ of\ parameters\ in\ the\ WSDL\ for\ version\ i}{Number\ of\ operations\ for\ version\ i}$$

3. ***Messages per operation in the WSDL***

MO $= \{(V_1, MO_1)\ldots (V_i, MO_i)\ldots (V_N, MO_N)\}$

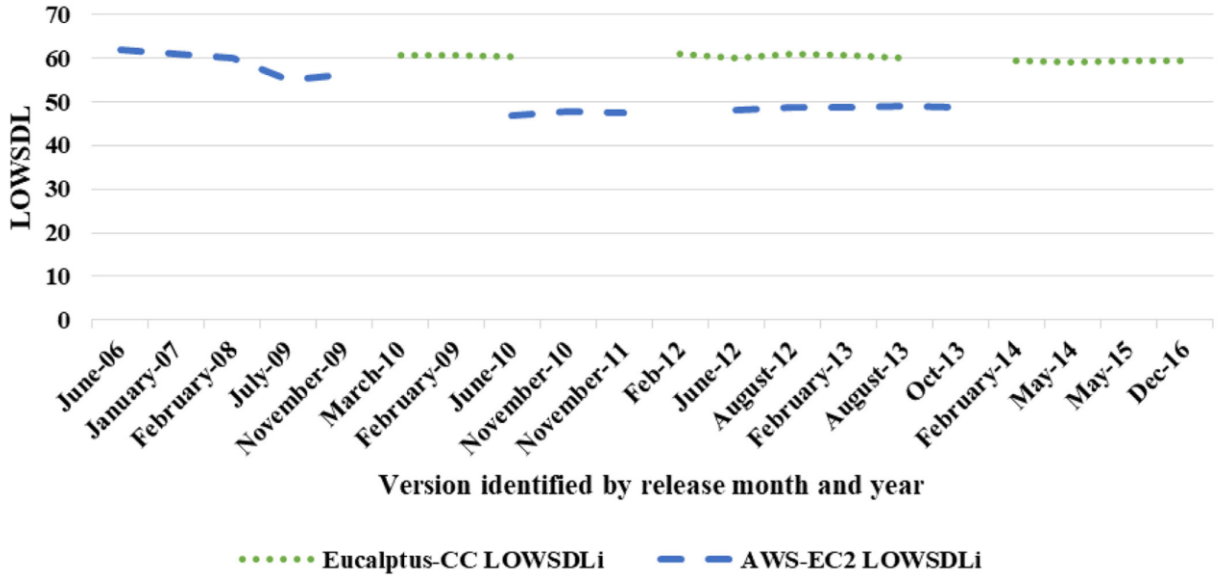
$$MO_i = \frac{Number\ of\ messages\ in\ the\ WSDL\ for\ version\ i}{Number\ of\ operations\ for\ version\ i}$$

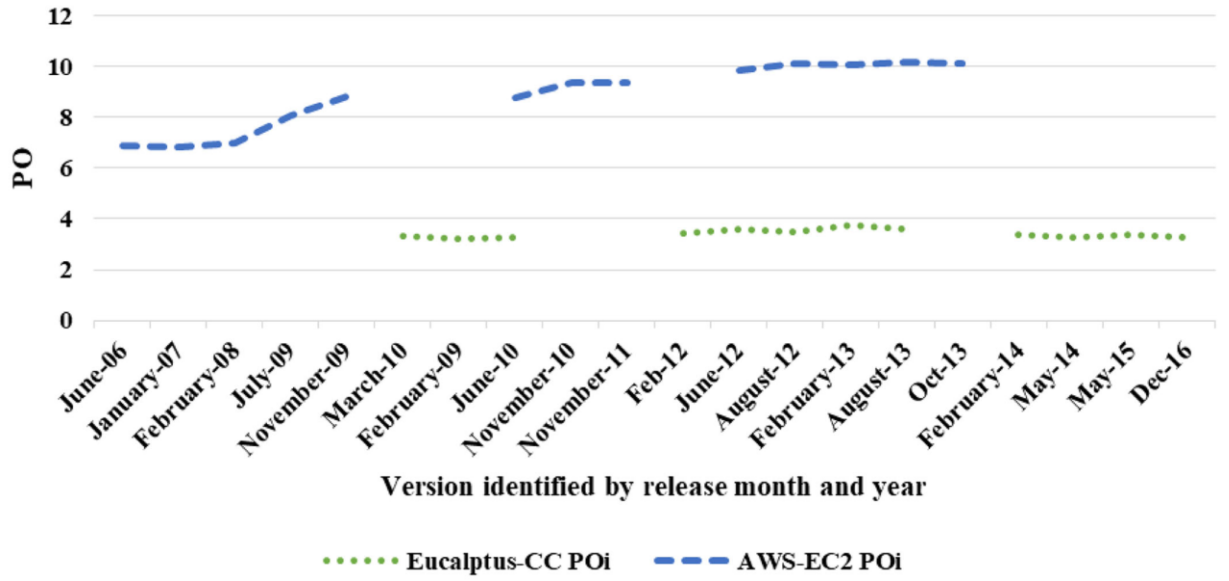
4. ***Code Lines per operation in the WS code***

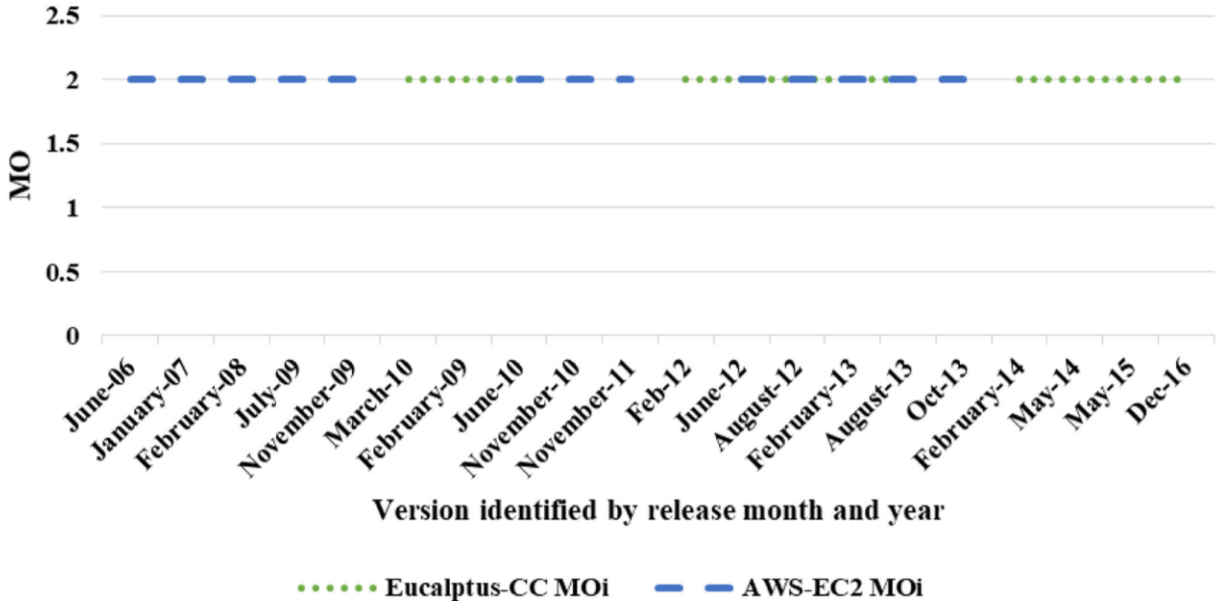
WSCLO $= \{(V_1, WSCLO_1)\ldots (V_i, WSCLO_i)\ldots (V_N, WSCLO_N)\}$

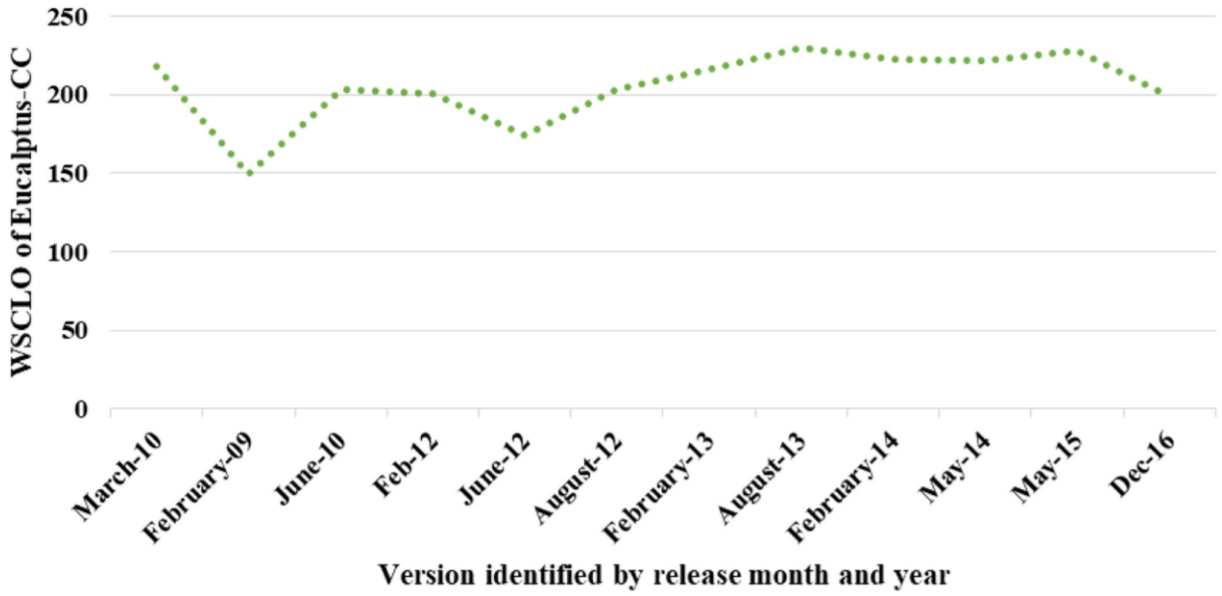
$$WSCLO_i = \frac{Number\ of\ code\ lines\ of\ WS\ code\ in\ version\ i}{Number\ of\ operations\ in\ version\ i}$$

Evolution of Eucalyptus - Cluster Controller (CC) and Amazon Web Service - Elastic Compute Cloud (EC2)









Applied Data Science Research

minOffense: Inter-Agreement Hate Terms for Stable Rules, Concepts, Transitivities, and Lattices

1. Service Evolution Analytics: Change and Evolution Mining of a Distributed System
2. **minOffense: Inter-Agreement Hate Terms for Stable Rules, Concepts, Transitivities, and Lattices**

Motivation and Problem

- For a given set of Hate Terms lists (HTs-lists) and Hate Speech data (HS-data), it is challenging to understand which hate term contributes the most for hate speech.
- Two approaches to the relationship between co-occurring Hate Terms (HTs).

1. Quantitative Analysis

- To create an ***Inter-agreement HTs-list***, which explains the contribution of an individual hate term toward hate speech.
- To produce a **Severe Hate Terms list** (***Severe HTs-list***)

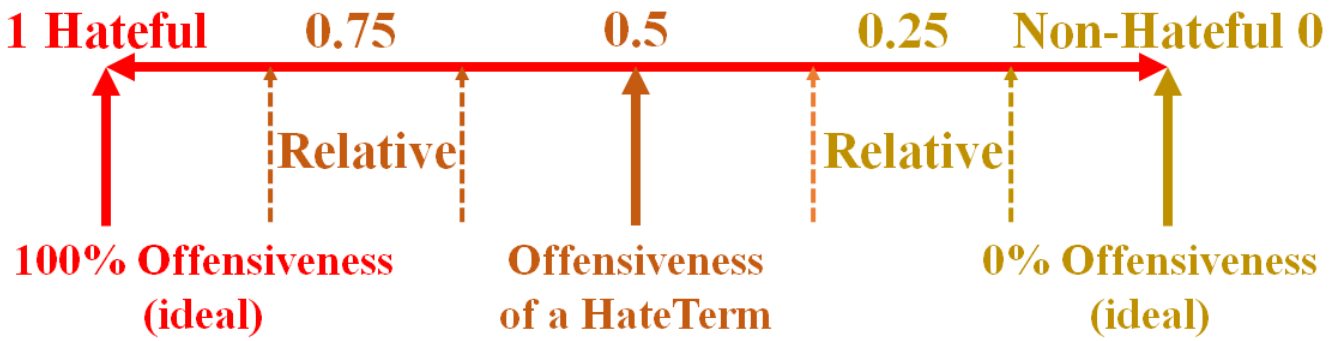
2. Qualitatively Analysis

- ***Stable Hate Rule*** (***SHR***) mining detects ordered frequently co-occurring HTs with ***minimum Stability*** (***minStab***). This form ***Stable Hate Rules*** and ***Concepts***.
- These rules and concepts are used to visualise the graphs of ***Transitivities*** and ***Lattices*** formed by HTs.

Inter-Agreement HTs

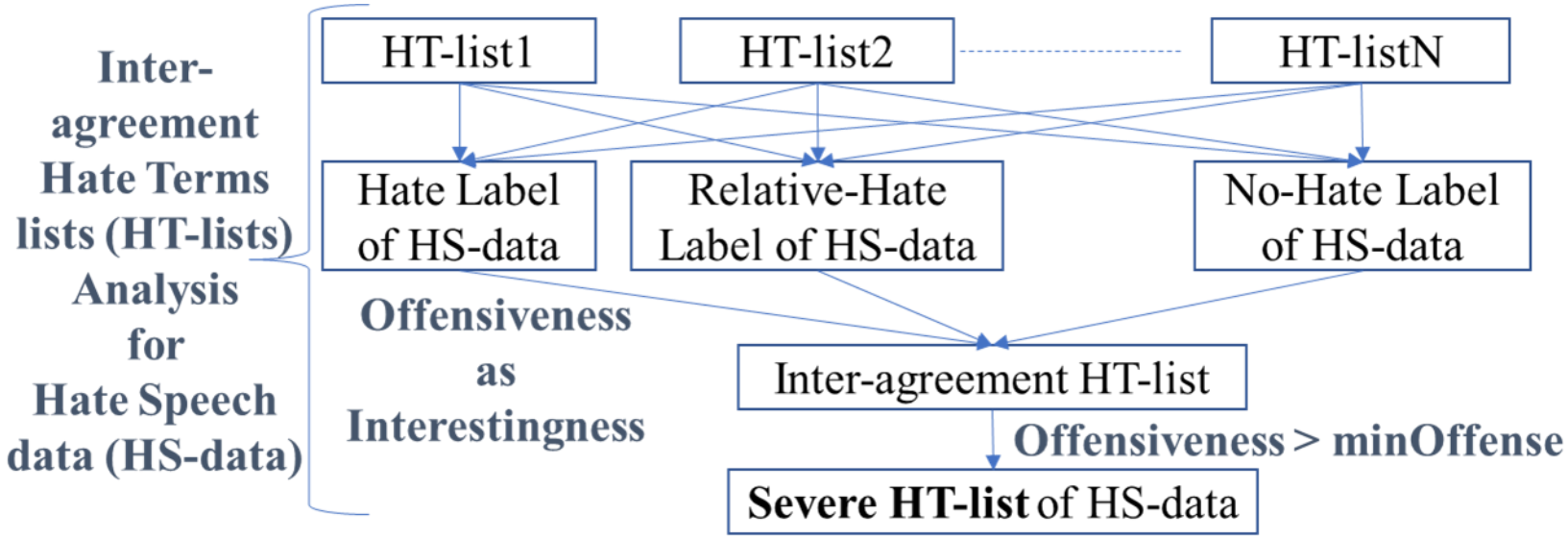
- Percentage contribution (i.e., overall input towards cost) of a hate term.
- Varying value of the Offensiveness of a HT for a given HS-data divided into classes (Hate, Relative-hate, and No-hate).
- HT will be most Hateful when its Offensiveness equals to **1**
- HT will be least Hateful when its Offensiveness value is equal to **0**

$$\text{Offensiveness} = \frac{2 \times \text{Hatefulness} \times \text{Relativeness}}{(\text{Hatefulness} + \text{Relativeness})}$$



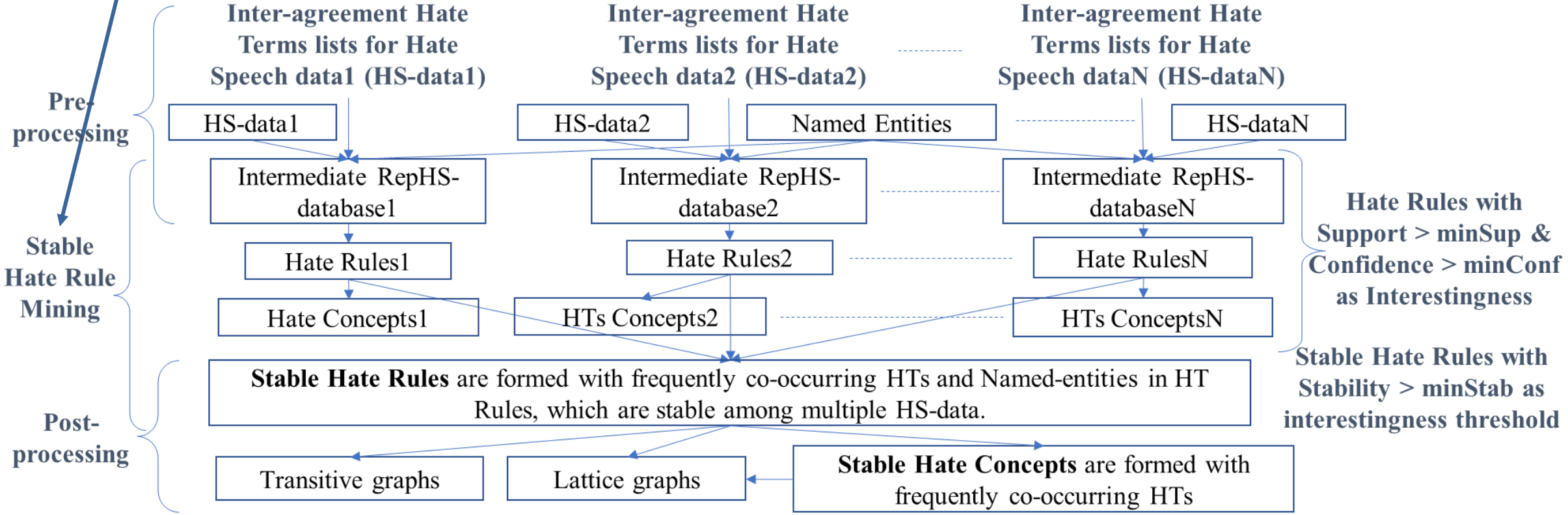
Severe Hate Terms-list

- Generate the Severe HTs-list from the Inter-Agreement HTs-list having HTs with ***Offensiveness* metric** values greater than a user-defined interestingness threshold ***minimum Offense*** (**minOffense**).
- Offensiveness provides help to separate out the highly severe HTs and the less severe HTs.
- High values of Offensiveness generate the Severe HTs-lists.
- Severe HTs-list helps in better hate speech classification as compared to the given set of HTs-lists.



Stable Hate Rule (SHR)

- **Stable Hate Rule (SHR) mining**
  - SHR mining over the database to discover co-occurring HTs.
  - This helps to discover and analyse the co-occurring concepts of HTs.



Inter-agreement Confusion-matrix

RANKING OF HTS-LIST NAME IN DECREASING ORDER OF INTER-AGREEMENT WITH THE HS-DATA.

HS-data Name	HTs-lists Names (number of HTs)
Davidson et al	**Offensiveness(Hate) (0.7, 298)**, Gorrell et al abuse-terms (403), HateBaseList (1015), Wiegand et al lexicon-of-abusive-words (7156), Hurtlex EN (5925), Union (13538), and Chandrasekharan et al Reddit hate lexicon (199).
de Gibert et al	**Offensiveness(Hate)(0.46, 578)**, Union (13538), Hurtlex EN (5925), Wiegand et al lexicon-of-abusive-words (7156), Chandrasekharan et al Reddit hate lexicon (199), HateBaseList (1015), Gorrell et al abuse-terms (403).
Gao et al	**Offensiveness(Hate)(0.75, 622)**, Union(13538), Wiegand et al lexicon-of-abusive-words (7156), Hurtlex EN (5925), Chandrasekharan et al Reddit hate lexicon (199), Gorrell et al abuse-terms (403), HateBaseList(1015).

Inter-agreement Confusion-matrix

FOR THE THREE HS-DATA, THE TABLE PROVIDES A COMPARISON OF THE SEVERE HTS-LIST WITH THE GIVEN HTS-LISTS.

HTs-list Name (minOffense, number of HTs)	HS-data Name and Class	Accuracy	Recall	Precision	F-Measure	Compute Time
Gorrell et al abuse-terms (-, 403)	Davidson_et_al_ 0Hate Vs. No-Hate	0.857	0.784	0.917	0.845	12 sec
	Davidson_et_al_ 0Hate+1Offensive Vs. No-Hate	0.845	0.761	0.915	0.831	
	Davidson_et_al_1 Offensive Vs. No-Hate	0.844	0.759	0.915	0.83	
Offensiveness(Hate) (0.7, 298)	Davidson_et_al_ 0Hate Vs. No-Hate	**0.921**	**0.946**	0.901	**0.923**	17 sec
	Davidson_et_al_ 0Hate+ 1Offensive Vs. No-Hate	**0.929**	**0.962**	0.903	**0.931**	
	Davidson_et_al_1 Offensive Vs. No-Hate	**0.93**	**0.963**	**0.903**	**0.932**	
Union (-, 13538)	de_Gibert_et_al_ 0Hate Vs. No-Hate	0.633	0.959	0.58	0.723	1 min 31 sec
	de_Gibert_et_al_0Hate +1RelationalHate Vs. No-Hate	0.629	0.951	0.578	0.719	
	de_Gibert_et_al_ 1RelationalHate Vs. No-Hate	0.6	0.893	0.563	0.69	
Offensiveness(Hate) (0.46, 578)	de_Gibert_et_al_ 0Hate Vs. No-Hate	**0.821**	0.832	**0.814**	**0.823**	**14 sec**
	de_Gibert_et_al_ 0Hate+ 1RelationalHate Vs. No-Hate	**0.8**	0.789	**0.806**	**0.797**	
	de_Gibert_et_al_ 1RelationalHate Vs. No-Hate	**0.646**	0.482	**0.718**	0.577	
Union (-, 13538)	Gao_et_al_ 0Hate Vs. No-Hate	0.46	0.772	0.475	0.588	15 sec
Offensiveness(Hate) (0.75, 622)	Gao_et_al_ 0Hate Vs. No-Hate	**0.541**	0.718	**0.53**	**0.61**	**5 sec**

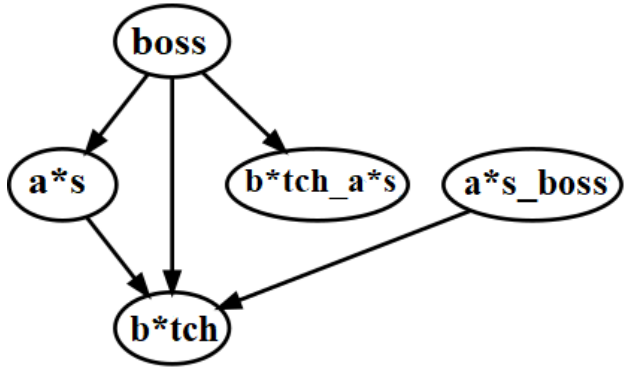
Our approach shown an improvement from 0.845 to 0.923 (best) as compared to the baseline.

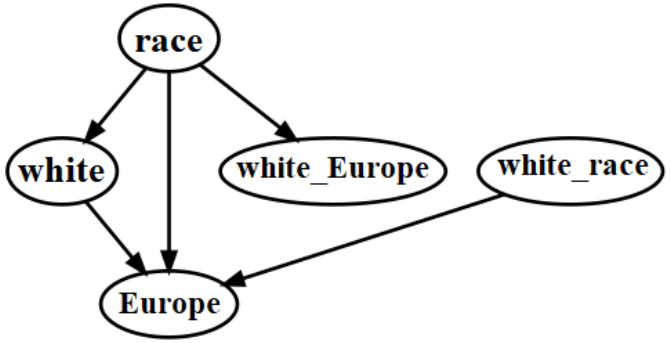
Severe HTs-list provides better results for confusion-matrix (precision, recall, f-measure, and accuracy).

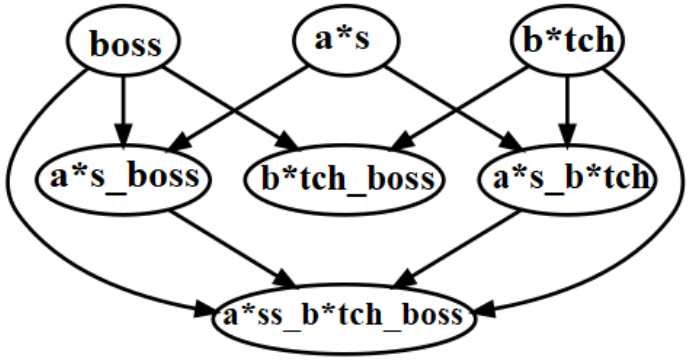
SHR mining to generate: Stable Hate Rules, Concepts, Transitivities, and Lattices

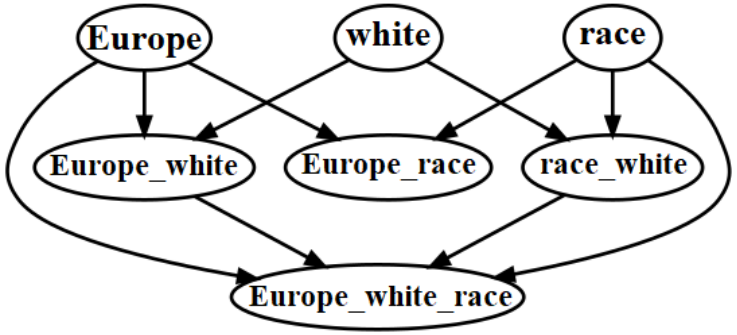
TWO HATE CONCEPTS (FIRST ROW) AND THEIR SHRS WITH SIMILAR HTS.

a*s b*tch boss 5	**Europe race white 5**
a*s → b*tch	white → Europe
boss → b*tch a*s	race → white Europe
a*s boss → b*tch	white race → Europe
boss → b*tch	race → white
boss → a*s	race → Europe









Publications, Research, and Teaching Interests

- Abstract of PhD Thesis on **7th Heidelberg Laureate Forum** 2019 Proceedings (Informal Publication). Selected as a **200 Young researcher** in Mathematics and Computer Science.
- The proceeding also contains work details of the **23 Laurates** i.e. ACM Turing award winners, IMU Field Medalist, and DNVA Abel Prize winners.
- Abstract of PhD Thesis on Dagstuhl Seminar 19401 Report (Informal Publication).

Major Contributions

System Network Analytics

- Stable Network Evolution Rule Mining
- Network Evolution Subgraph (Graphlet or Motif) Mining

System Evolution Metrics

- System Changeability and Stability
- System State Complexity and Evolving System Complexity

System Evolution Learning

- System Evolution Recommender (SysEvoRecomd)
- Graph Evolution and Change Learning to do Network Reconstruction

Analytics on Cloud Services and Big Data

- Service Change Classifier and Service Evolution Metrics
- Association Rules for Big Scholarly Data

System Network Evolution Rules and Subgraphs

1. Animesh Chaturvedi and Aruna Tiwari. "System Evolution Analytics: Evolution and Change Pattern Mining of Inter-Connected Entities". *48th 2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)* pp. 3877-3882. IEEE SMC Society DOI: 10.1109/SMC.2018.00750.
2. Animesh Chaturvedi, Aruna Tiwari, and Nicolas Spyratos. "minStab: Stable Network Evolution Rule Mining for System Changeability Analysis". *IEEE Transactions on Emerging Topics in Computation Intelligence*, Vol 5.2 (April 2019). DOI: 10.1109/TETCI.2019.2892734.
3. Animesh Chaturvedi and Aruna Tiwari. "System Network Complexity: Network Evolution Subgraphs of System State Series." *IEEE Transactions on Emerging Topics in Computational Intelligence,* Vol 4.2 (2018): 130-139. DOI: 10.1109/TETCI.2018.2848293. (IEEE Computer Society and IEEE Computational Intelligence Society).

System Neural Network

1. Animesh Chaturvedi, Aruna Tiwari, Shubhangi Chaturvedi, and Pietro Lio'. "System Neural Network: Evolution and Change based Structure Learning", *IEEE Transactions on Artificial Intelligence*. Vol. 3.3, pp. 426 - 435, June 2022. DOI: 10.1109/TAI.2022.3143778. (IEEE Computer Society and IEEE Computational Intelligence Society)
2. Animesh Chaturvedi, Aruna Tiwari, and Shubhangi Chaturvedi "SysEvoRecomd: Network Reconstruction by Graph Evolution and Change Learning", *IEEE Systems Journal*, Vol. 14.3, pp. 4007 - 4014, Sept. 2020 IF: 4.463 DOI:10.1109/JSYST.2020.2988037.
3. Animesh Chaturvedi, and Aruna Tiwari. "System Evolution Analytics: Deep Evolution and Change Learning of Inter-Connected Entities". 48th *IEEE International Conference on Systems, Man, and Cybernetics* (SMC), Miyazaki Japan, October 2018, pp. 3075-3080. IEEE SMC Society DOI: 10.1109/SMC.2018.00657
4. Animesh Chaturvedi, and Aruna Tiwari. "SysEvoRecomd: Graph Evolution and Change Learning based System Evolution Recommender". 18th *IEEE International Conference on Data Mining Workshops* (ICDMW). (Core A*) Singapore, 2018, pp. 1499-1500. IEEE Computer Society DOI: 10.1109/ICDMW.2018.00217

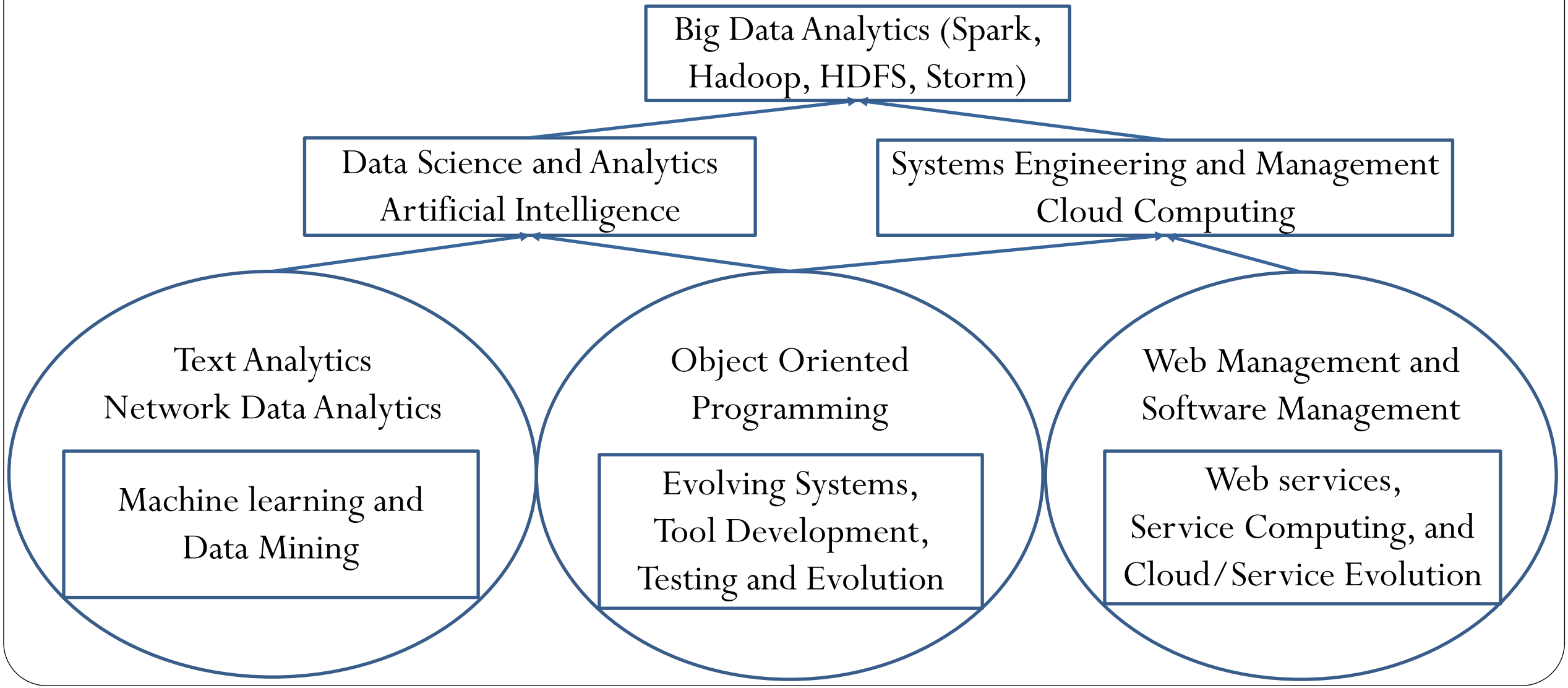
Service Evolution Analytics

1. Animesh Chaturvedi, Aruna Tiwari, Shubhangi Chaturvedi, and Dave Binkley "Service Evolution Analytics: Change and Evolution Mining of a Distributed System", *IEEE Transactions on Engineering Management*, Vol. 68.1, pp 137 - 148, Feb-2021, DOI: 10.1109/TEM.2020.2987641 IF: 2.784. (ABDC A).
2. Animesh Chaturvedi, and Dave Binkley. "Web Service Slicing: Intra and Inter-Operational Analysis to Test Changes". *IEEE Transactions on Services Computing* Vol. 14.3 (May-June 2021): 930-943. IF: 4.91 DOI: 10.1109/TSC.2018.2821157 (CORE A*).
3. Animesh Chaturvedi, "Subset WSDL to access Subset Service for Analysis", *6th IEEE International Conference on Cloud Computing Technology and Science* (IEEE CloudCom), Singapore, Dec 2014, pp 688-691. *IEEE Computer Society* and *IEEE Cloud Computing* DOI: 10.1109/CloudCom.2014.149.
4. Animesh Chaturvedi, "Automated Web Service Change Management AWSCM - A Tool" *6th IEEE International Conference on Cloud Computing Technology and Science* (IEEE CloudCom), Singapore 2014, pp 715-718. *IEEE Computer Society* and *IEEE Cloud Computing* DOI: 10.1109/CloudCom.2014.144.

Recent accepted papers

1. Animesh Chaturvedi, "Call Graph Evolution Analytics over a Version Series of an Evolving Software System" *37th IEEE/ACM International Conference on Automated Software Engineering (ASE'22)*. (Core A*) Accepted
2. Animesh Chaturvedi, Aruna Tiwari, and Nicolas Spyratos. "System Network Analytics: Evolution and Stable Rules of a State Series". *IEEE 9th IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, 2022 (Core A) Accepted.
3. Animesh Chaturvedi and Rajesh Sharma. "minOffense: Inter-Agreement Hate Terms for Stable Rules, Concepts, Transitivities, and Lattices". *IEEE 9th International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE, 2022. (Core A) Accepted.

Research Expertise and Area



Research Background

- Six approaches and theories are proposed during my Ph.D Thesis work
  - Network Evolution Rule Mining
  - Network Evolution Subgraph Mining
  - Deep System Graph Evolution Learning
  - Web Service Slicing and Service Evolution Analytics
  - Big Data Evolution mining
  - Hate Speech Analytics
- I have been published in reputed journals such as
  - *IEEE Transactions on Artificial Intelligence (TAI)*
  - *IEEE Transactions on Emerging Topics in Computation Intelligence (TETCI)* (2 papers)
  - *IEEE Transactions on Engineering Management (TEM)* (ABDC A)
  - *IEEE Transactions on Services Computing (TSC)* (CORE A*)
  - *IEEE Systems Journal (ISJ)*
  - *IEEE International Conference on Systems, Man, and Cybernetics (SMC)* (2 papers) (Core B)
  - *IEEE International Conference on Data Mining Workshops (ICDMW)* (Core A*)
  - *IEEE International Conference on Cloud Computing Technology and Science (CloudCom)* (2 papers) (Core A)
  - *IEEE International Conference on Data Science and Advanced Analytics (DSAA)* (2 papers) (Core A)
  - *IEEE/ACM International Conference on Automated Software Engineering (ASE'22)*. (Core A*)

Research Talks at

- Motorola Mobility Bangalore (worked as Student Intern for 6 months 2013)
- IIT - Kanpur (worked as Research and Project Associate for 7 months 2014)
- TU/Eindhoven Netherlands, (IEEE MESOCA and IEEE ICSME 2013)
- NTU-Singapore, (IEEE CloudCom 2014)
- Sheraton-grande hosted by University of Miyazaki Japan, (IEEE SMC 2018)
- Heidelberg Laureate Forum (HLF) 2019*
- Schloss Dagstuhl (known for DBLP) Seminars 2018
- IIT-Bombay (CSE department) 2021
- Higher Education Academy of Karnataka University, 2022

* Selected as 200Young Researchers by *Heidelberg Laureate Forum* (HLF) an event of ACM and IMU in HLF 2019 and 2020.

Prof. Leslie Lamport

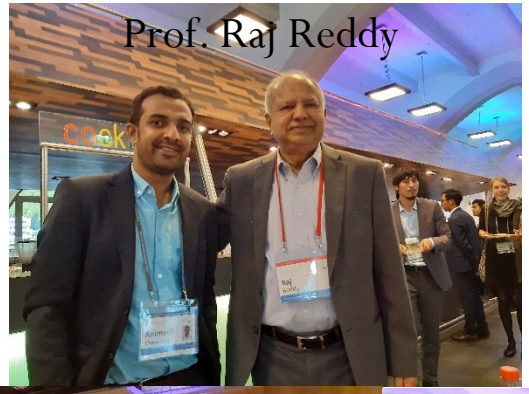
Prof. John Hopcroft

The Laureates

Prof. Robert Tarjan

Prof. Efim Zelmanov

Prof. Deffie & Prof. Helman (Public and Private Key)



Prof. Raj Reddy

Prof. Raj Reddy
Prof. P. J. Narayanan

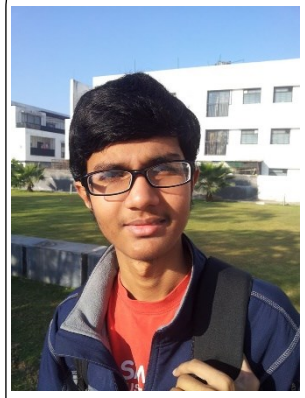
Prof. Leslie Lamport

Prof. Richard E. Stearns
Computational complexity theory.

The Heidelberg Laurate Forum collage

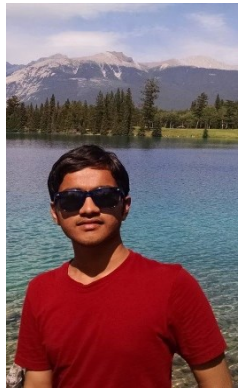
Prof. Shwetak Patel

Prof. Carlos Kenig
(President, IMU)



Krishna Chaitanya

Harsh Mohan



Vraj Shah

Dr. Srinivas Padmanabhuni

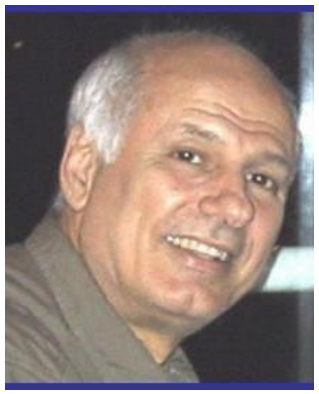
Dr. Aruna Tiwari

Shubhangi Chaturvedi

Aditya Jain

Kunal Gupta

Thanks to the collaborators



Emeritus Prof. Nicolas Spyratos
University of Paris-South, France

Prof. Dave Binkley
Loyola University of Maryland, USA

Prof. Pietro Lio
University of Cambridge (U.K.)

Prof. Nishant Sastry
University of Surrey, Alan Turing Institute (U.K.)

Dr. Rajesh Sharma
University of Tartu, Estonia

References

1. Rakesh Agrawal , Tomasz Imieliński, and Arun Swami. "Mining association rules between sets of items in large databases." SIG-MOD. 1993.
2. Ramakrishnan Srikant, and Rakesh Agrawal. "Mining Generalized Association Rules."VLDB 1995.
3. Usama M. Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. "Knowledge Discovery and Data Mining: Towards a Unifying Framework." *KDD*. 1996.
4. Jiawei Han, Jian Pei, and Yiwen Yin. "Mining frequent patterns without candidate generation." *ACM SIGMOD Record* 29.2 (2000): 1-12.
5. Haoyuan Li, et al. "PFP: Parallel FP-Growth for query recommendation." *Proceedings of the 2008 ACM Conference on Recommender systems*. 2008.
6. T. H. Cormen, C. E. Leiserson, R. L. Rivest, & C. Stein. (2009). *Introduction to Algorithms* (Vol. 3, pp. 624-642). Cambridge: MIT press.
7. Matei Zaharia, et al. "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing." 9th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 12). 2012.
8. Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." *Nature* 521.7553 (2015): 436-444.
9. Matei Zaharia, et al. "Apache spark: a unified engine for big data processing." Communications of the ACM 59.11 (2016): 56-65.
10. https://spark.apache.org/
11. https://www.kdnuggets.com/2016/03/data-science-process-rediscovered.html/2

ขอบคุณ
Thai

Grazie
Italian

תודה רבה
Hebrew

धन्यवादः
Sanskrit

ಧನ್ಯವಾದಗಳು
Kannada

Ευχαριστώ
Greek

Thank You
English

Gracias
Spanish

Спасибо
Russian

Obrigado
Portuguese

شكراً
Arabic

https://sites.google.com/site/animeshchaturvedi07

Merci
French

多謝
Traditional
Chinese

धन्यवाद
Hindi

Danke
German

多谢
Simplified
Chinese

நன்றி
Tamil
Tamil

ありがとうございました
Japanese

감사합니다
Korean