Mobile App Fingerprinting through Automata Learning and Machine Learning

Fatemeh Marzani, Fatemeh Ghassemi, Zeynab Sabahi-Kaviani

University of Tehran

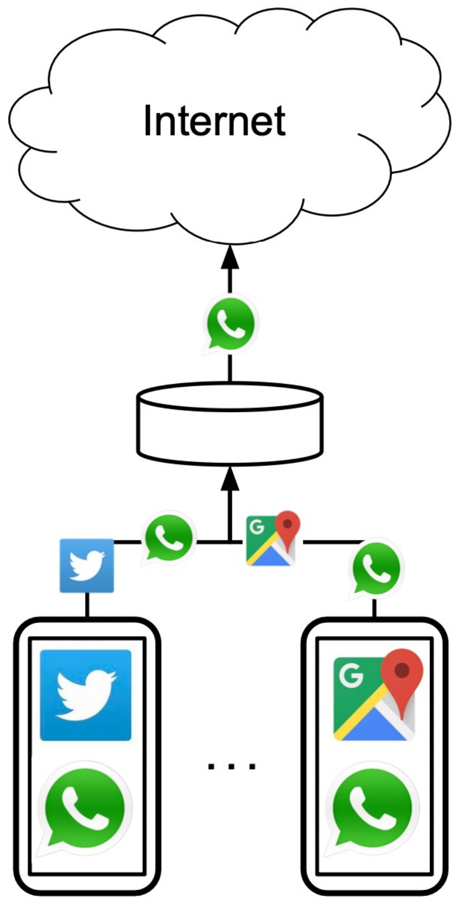
Thijs van Ede, Maarten van Steen

University of Twente

Contact: f.marzani.ut@gmail.com

Problem Definition

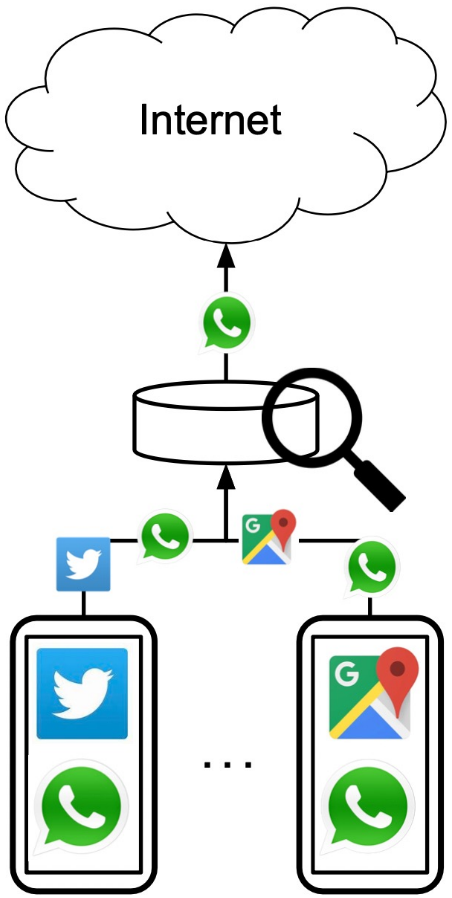
- Most of apps communicate with the internet



Picture is taken from FlowPrint

Problem Definition

- Most of apps communicate with the internet
- How can we identify running apps from network traffic?



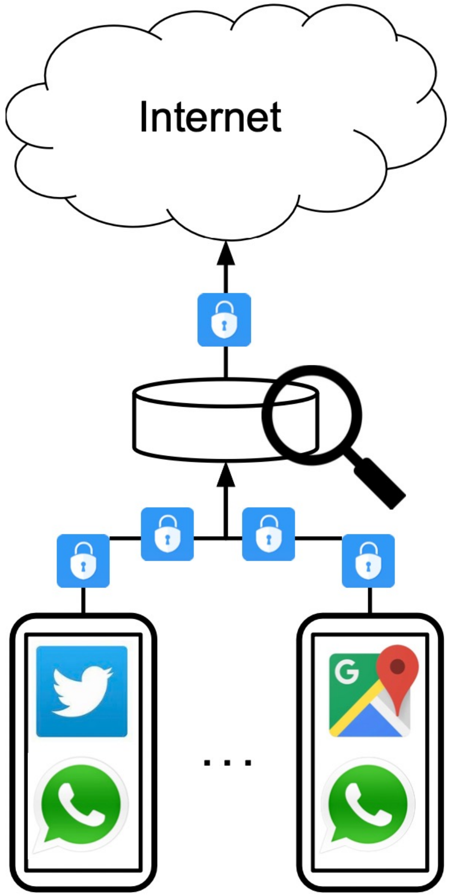
Picture is taken from FlowPrint

Problem Definition

- Most of apps communicate with the internet
- How can we identify running apps from network traffic?
- Sorry! It is not that easy!

Challenges

- Encrypted traffic
- Homogeneous traffic
- Dynamic traffic
- Evolving traffic
- Complexity of mathematical models
- Lack of sufficient and diverse data



Picture is taken from FlowPrint

Fingerprinting Usage

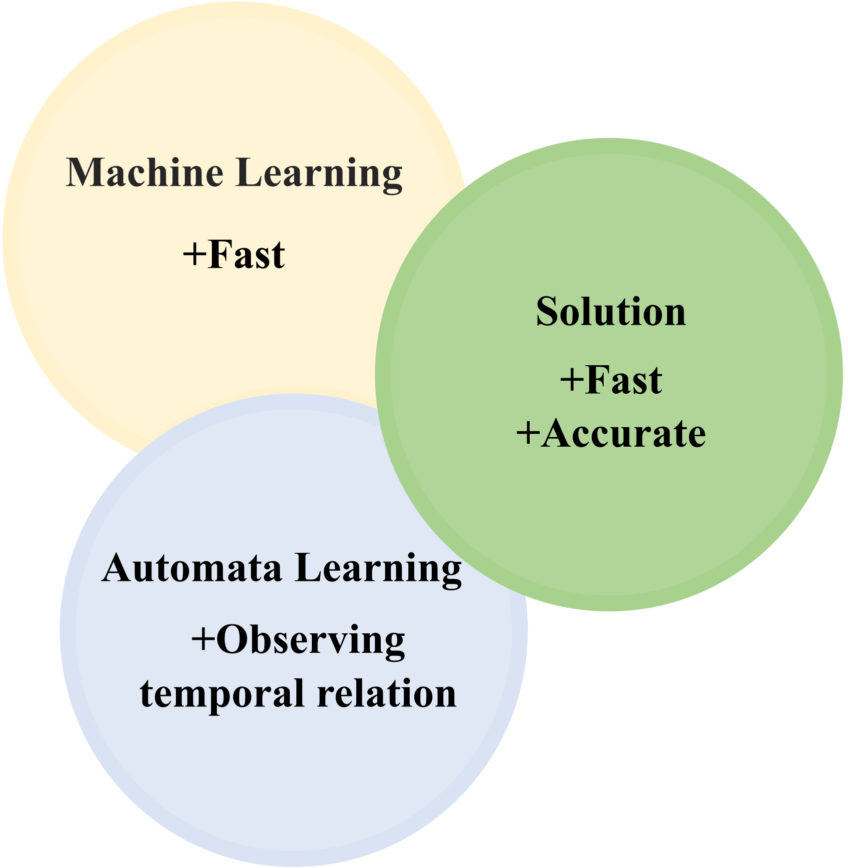
- Network management
- Security and intrusion detection
- Advertising and market research

Fingerprinting Methods

Method	Advantages	Disadvantages
Port-based	Fast and simple	Infeasible for dynamic ports
Payload-based	Accurate classification	Infeasible for encrypted payload
Statistics-based	Fast	Ignore temporal relation among flows/packets High false positive rate
Correlation-based	High accuracy	High computational overhead
Behavior-based	Deal with encrypted traffic	Manual domain-specific

Motivation

- Can we take advantage of combining methods to create faster and more accurate solution?



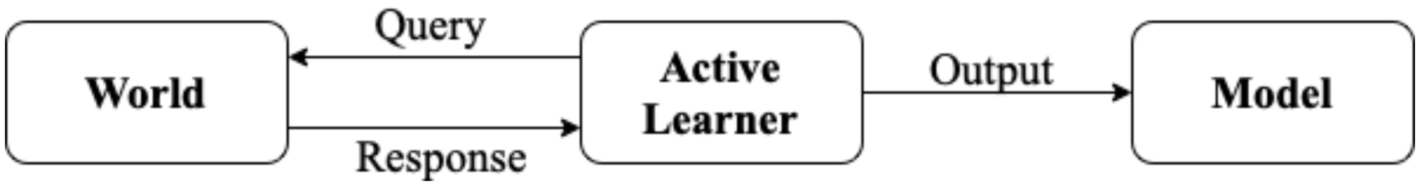
Automata Learning

Try to find one of the smallest automata with the set of given samples.

Automata Learning

Try to find one of the smallest automata with the set of given samples.

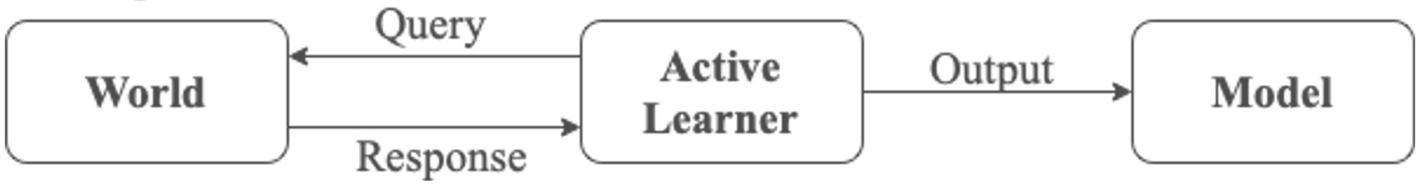
- **Active Learning**



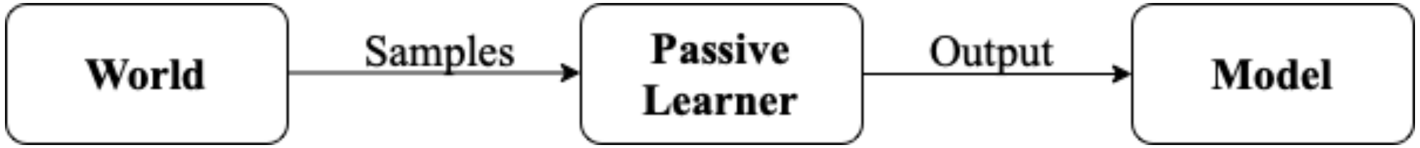
Automata Learning

Try to find one of the smallest automata with the set of given samples.

- Active Learning



- **Passive Learning**



K-TSS: K-Testable Language in the Strict Sense

- Learn a **regular language** from samples:

- Create **k-test Vector** of samples :

  $Z = \langle \Sigma, I, F, T \rangle$

  Σ: Finite alphabet

  I: Prefixes of length $(k-1)$

  F: Suffixes of length $(k-1)$

  T: Segments of length k

- Learn a regular language with k-test vector:

$$L(Z) = [(I\Sigma^* \cap \Sigma^* F) - \Sigma^*(\Sigma^k - T)\Sigma^*]$$

k-test Vector Example

If word = **abbac** and k-window size = **3**, define k-test vector $\langle \Sigma, I, F, T \rangle$

- **alphabet** = Σ = {a, b, c}

k-test Vector Example

If word = **abbac** and k-window size = **3**, define k-test vector $\langle \Sigma, I, F, T \rangle$

- **alphabet** = Σ = {a, b, c}
- **segments** = T = {abb}

abbac

k-test Vector Example

If word = **abbac** and k-window size = **3**, define k-test vector $\langle \Sigma, I, F, T \rangle$

- **alphabet** = Σ = {a, b, c}
- **segments** = T = {abb, bba}

abbac

k-test Vector Example

If word = **abbac** and k-window size = **3**, define k-test vector $\langle \Sigma, I, F, T \rangle$

- **alphabet** = Σ = {a, b, c}
- **segments** = T = {abb, bba, bac}

abbac

k-test Vector Example

If word = **abbac** and k-window size = **3**, define k-test vector $\langle \Sigma, I, F, T \rangle$

- **alphabet** = Σ = {a, b, c}
- **segments** = T = {abb, bba, bac}
- **prefixes** = I = {ab}

abbac

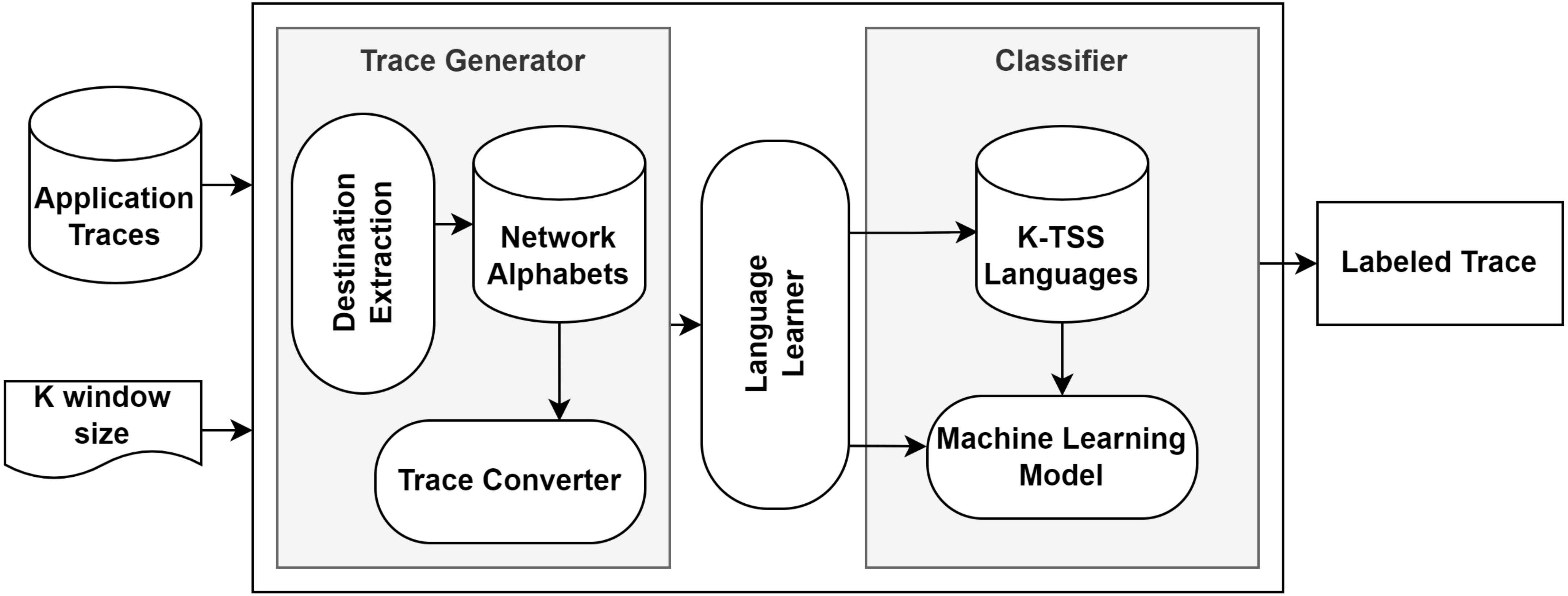
k-test Vector Example

If word = **abbac** and k-window size = **3**, define k-test vector $\langle \Sigma, I, F, T \rangle$

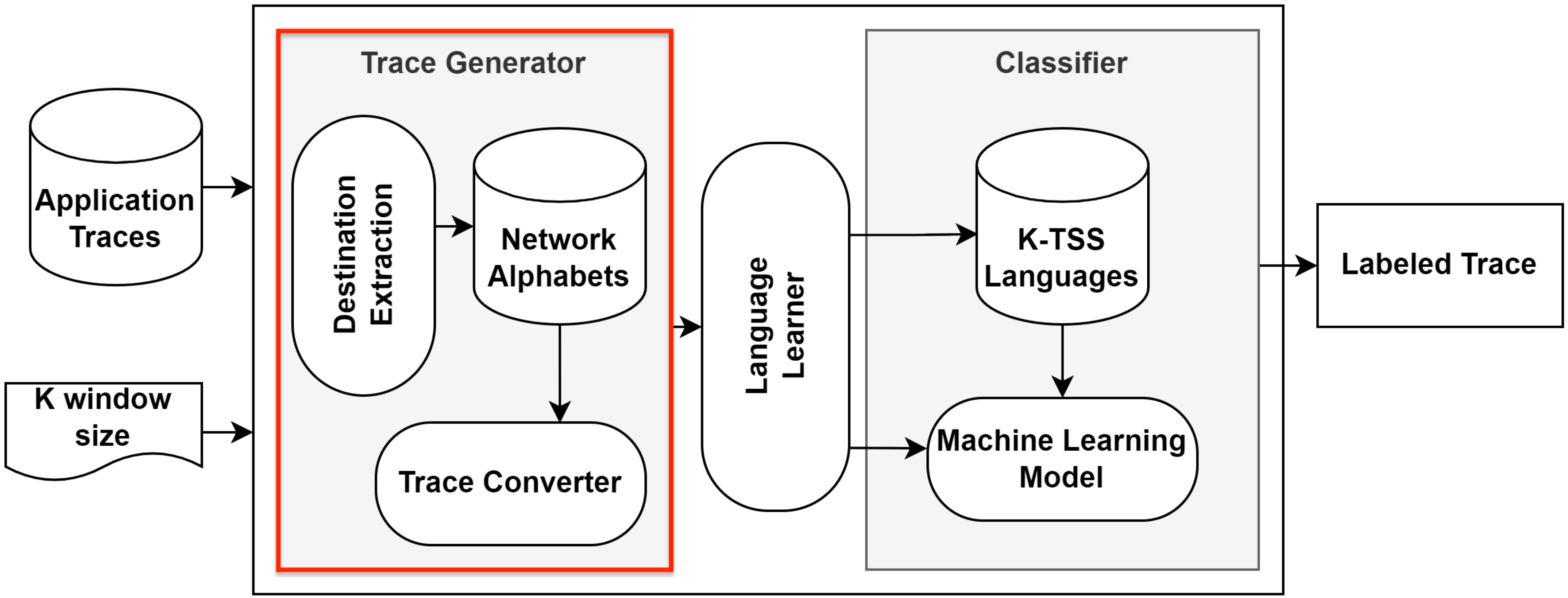
- **alphabet** = Σ = {a, b, c}
- **segments** = T = {abb, bba, bac}
- **prefixes** = I = {ab}
- **suffixes** = F = {ac}

abbac

ML-NetLang


Application Traces
K window size
Trace Generator
Destination Extraction
Network Alphabets
Trace Converter
Language Learner
Classifier
K-TSS Languages
Machine Learning Model
Labeled Trace

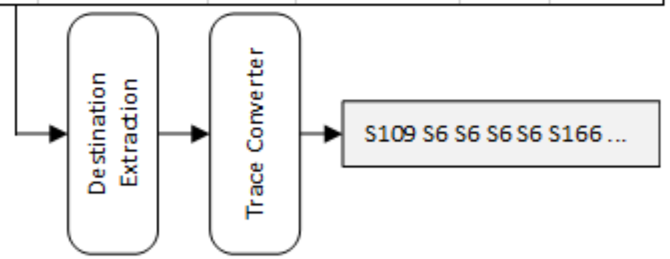
ML-NetLang


Trace Generator
Classifier
Application Traces
K window size
Destination Extraction
Network Alphabets
Trace Converter
Language Learner
K-TSS Languages
Machine Learning Model
Labeled Trace

Trace Generator

Converts the network traffic of each app into a list of words that are in the language of that app.

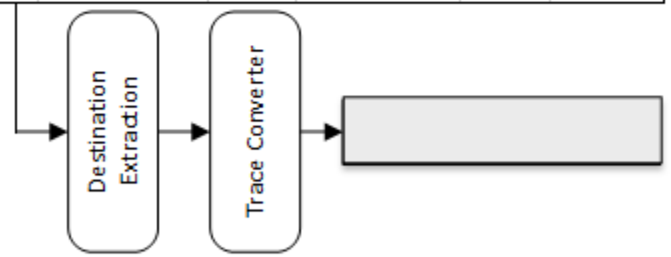
#	Time	Source	S-Port	Destination	D-Port	Protocol
1	20:25:56.71	10.11.2.4	49268	172.217.5.238	443	TCP
2	20:25:56.71	172.217.5.238	443	10.11.2.4	49268	TCP
3	20:25:57.03	10.11.2.4	49268	172.217.5.238	443	TCP
4	20:25:57.03	172.217.5.238	443	10.11.2.4	49268	TCP
5	20:26:07.81	10.11.2.4	31877	8.8.8.8	53	DNS
6	20:26:07.83	8.8.8.8	53	10.11.2.4	31877	DNS
7	20:26:07.87	10.11.2.4	8774	8.8.8.8	53	DNS
8	20:26:07.89	8.8.8.8	53	10.11.2.4	8774	DNS
9	20:26:08.45	10.11.2.4	15428	8.8.8.8	53	DNS
10	20:26:08.46	8.8.8.8	53	10.11.2.4	15428	DNS
11	20:26:08.52	10.11.2.4	30988	8.8.8.8	53	DNS
12	20:26:08.53	8.8.8.8	53	10.11.2.4	30988	DNS
13	20:26:08.56	10.11.2.4	55705	34.196.47.203	443	TCP
14	20:26:08.56	34.196.47.203	443	10.11.2.4	55705	TCP
...	...	...	...	...	...	...
...	...	...	...	...	...	...



Trace Generator

- Extract its network traffic flows.

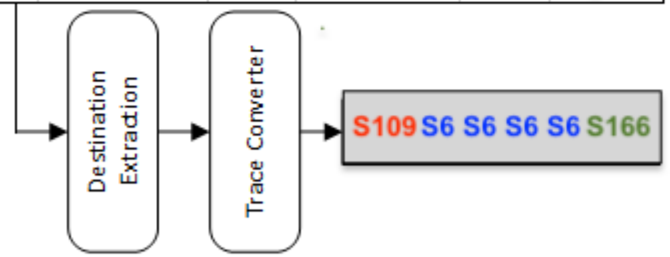
#	Time	Source	S-Port	Destination	D-Port	Protocol
1	20:25:56.71	10.11.2.4	49268	172.217.5.238	443	TCP
2	20:25:56.71	172.217.5.238	443	10.11.2.4	49268	TCP
3	20:25:57.03	10.11.2.4	49268	172.217.5.238	443	TCP
4	20:25:57.03	172.217.5.238	443	10.11.2.4	49268	TCP
5	20:26:07.81	10.11.2.4	31877	8.8.8.8	53	DNS
6	20:26:07.83	8.8.8.8	53	10.11.2.4	31877	DNS
7	20:26:07.87	10.11.2.4	8774	8.8.8.8	53	DNS
8	20:26:07.89	8.8.8.8	53	10.11.2.4	8774	DNS
9	20:26:08.45	10.11.2.4	15428	8.8.8.8	53	DNS
10	20:26:08.46	8.8.8.8	53	10.11.2.4	15428	DNS
11	20:26:08.52	10.11.2.4	30988	8.8.8.8	53	DNS
12	20:26:08.53	8.8.8.8	53	10.11.2.4	30988	DNS
13	20:26:08.56	10.11.2.4	55705	34.196.47.203	443	TCP
14	20:26:08.56	34.196.47.203	443	10.11.2.4	55705	TCP
...	...	...	...	...	...	...
...	...	...	...	...	...	...



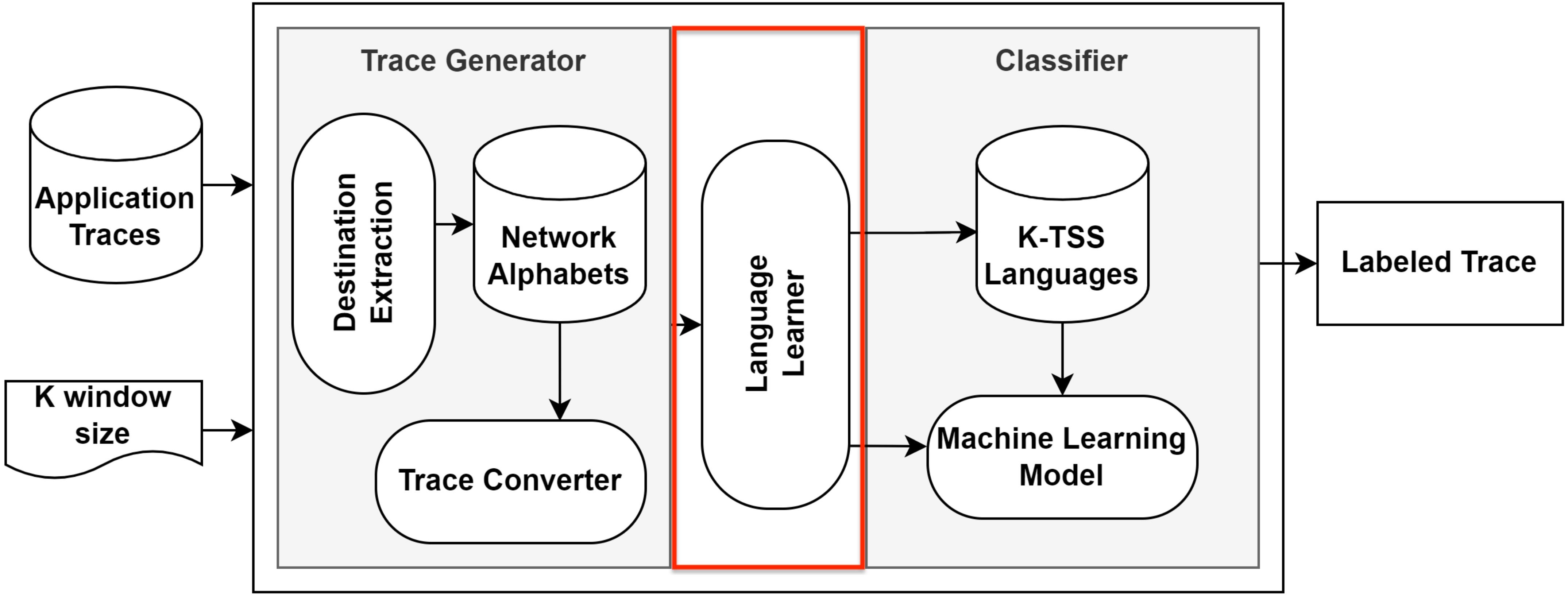
Trace Generator

- Extract its network traffic flows.
- Flows are categorized based on either **tuple of destinations (IP, Port) or TLS certificates** and given a symbol based on their category.
- S6 means, it's 6^{th} unique destination address seen among all apps

#	Time	Source	S-Port	Destination	D-Port	Protocol
1	20:25:56.71	10.11.2.4	49268	172.217.5.238	443	TCP
2	20:25:56.71	172.217.5.238	443	10.11.2.4	49268	TCP
3	20:25:57.03	10.11.2.4	49268	172.217.5.238	443	TCP
4	20:25:57.03	172.217.5.238	443	10.11.2.4	49268	TCP
5	20:26:07.81	10.11.2.4	31877	8.8.8.8	53	DNS
6	20:26:07.83	8.8.8.8	53	10.11.2.4	31877	DNS
7	20:26:07.87	10.11.2.4	8774	8.8.8.8	53	DNS
8	20:26:07.89	8.8.8.8	53	10.11.2.4	8774	DNS
9	20:26:08.45	10.11.2.4	15428	8.8.8.8	53	DNS
10	20:26:08.46	8.8.8.8	53	10.11.2.4	15428	DNS
11	20:26:08.52	10.11.2.4	30988	8.8.8.8	53	DNS
12	20:26:08.53	8.8.8.8	53	10.11.2.4	30988	DNS
13	20:26:08.56	10.11.2.4	55705	34.196.47.203	443	TCP
14	20:26:08.56	34.196.47.203	443	10.11.2.4	55705	TCP
...	...	...	...	...	...	...
...	...	...	...	...	...	...



ML-NetLang


Application Traces
K window size
Trace Generator
Destination Extraction
Network Alphabets
Trace Converter
Language Learner
Classifier
K-TSS Languages
Machine Learning Model
Labeled Trace

Language Learner

The k-test vector $\langle \Sigma, I, F, T \rangle$ is derived by the language learner, taking the list of words as input

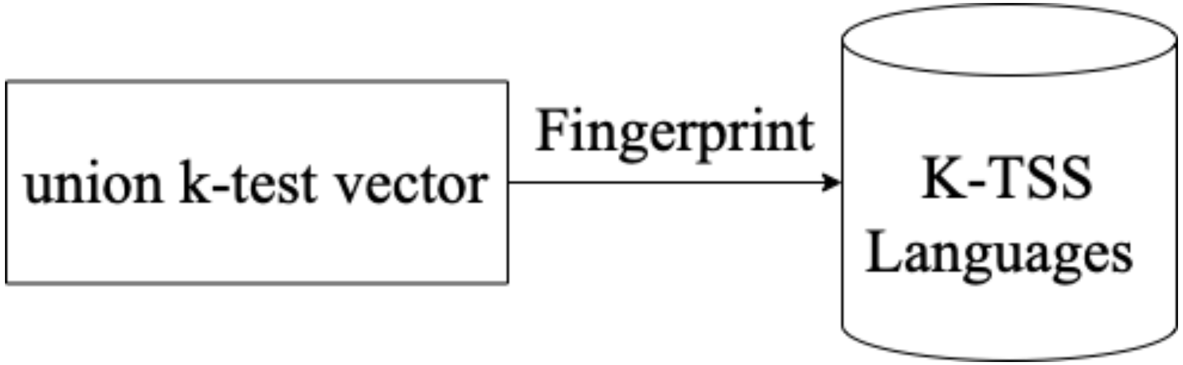
$$\text{app word} = \text{S109 S6 S6 S6 S6 S166}$$

k-window size = 3

$$\Sigma = \{\text{S109}, \text{S6}, \text{S166}\}$$
$$\text{I} = \{\text{S109 S6}\}$$
$$\text{F} = \{\text{S6 S166}\}$$
$$\text{T} = \{\text{S109 S6 S6}, \text{S6 S6 S6}, \text{S6 S6 S166}\}$$

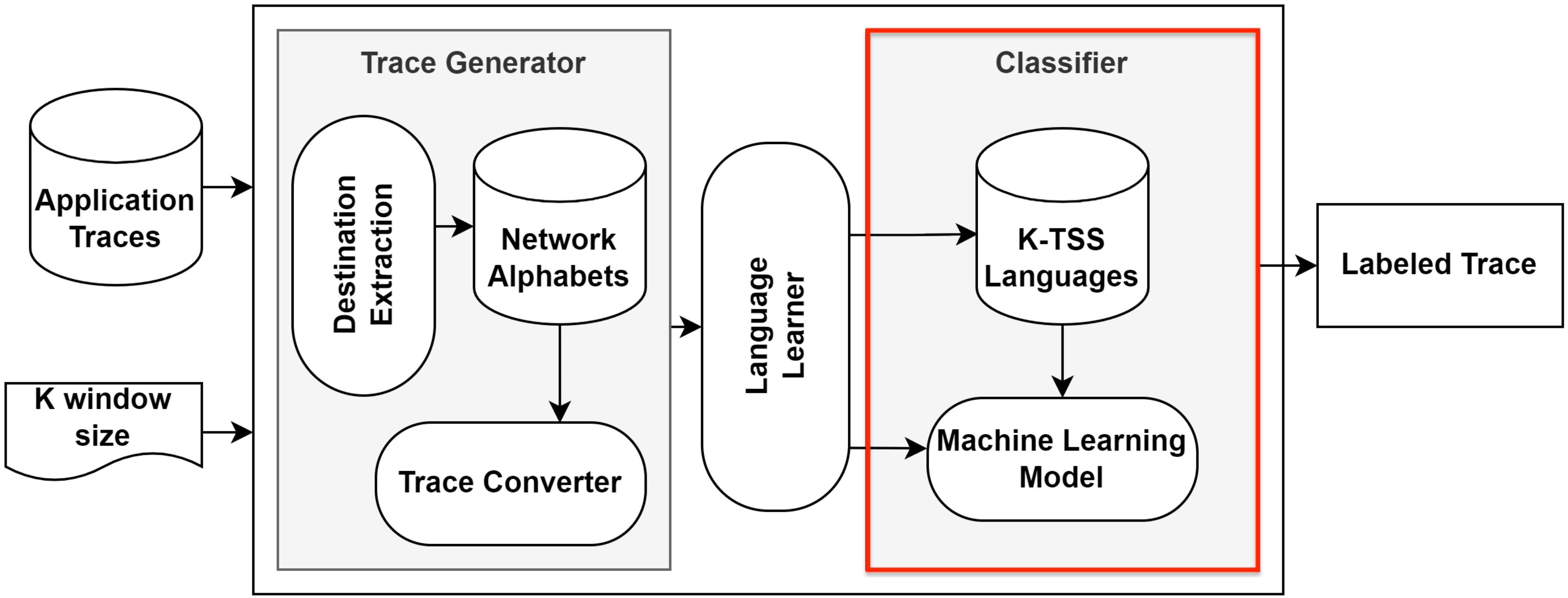
Language Learner

The **union k-test vector** of an app as its **fingerprint**:



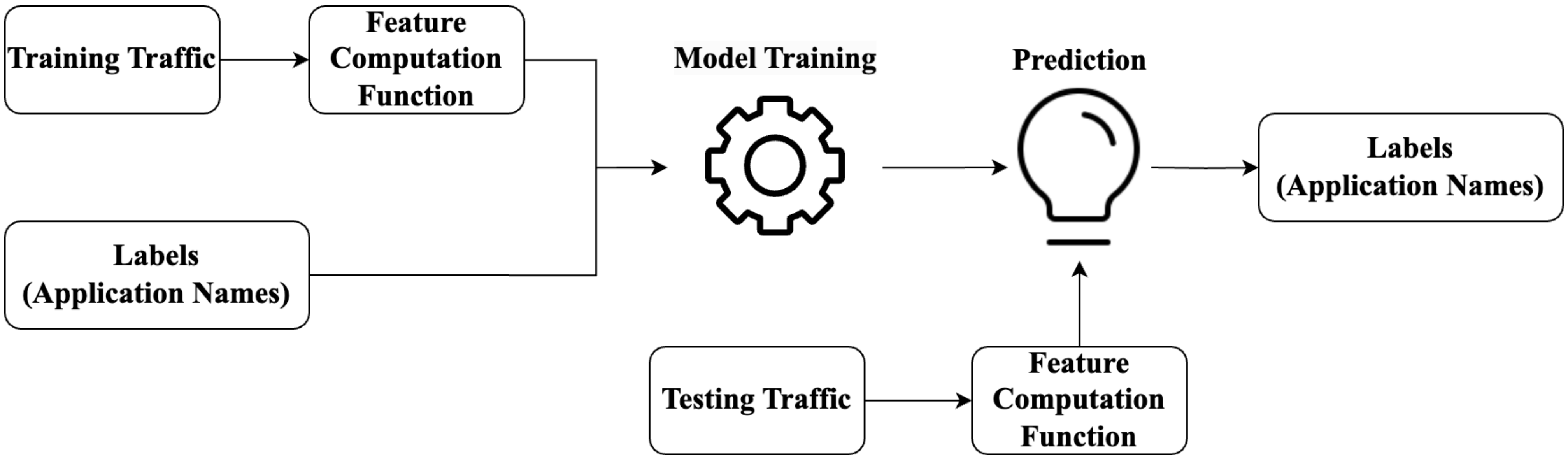
$$\mathrm{UZ}(\boldsymbol{App_i}) = [< \cup_{\mathrm{k=1}}^{m} \Sigma_k, \cup_{\mathrm{k=1}}^{m} \boldsymbol{I_k}, \cup_{\mathrm{k=1}}^{m} \boldsymbol{F_k}, \cup_{\mathrm{k=1}}^{m} \boldsymbol{T_k} >]$$

ML-NetLang

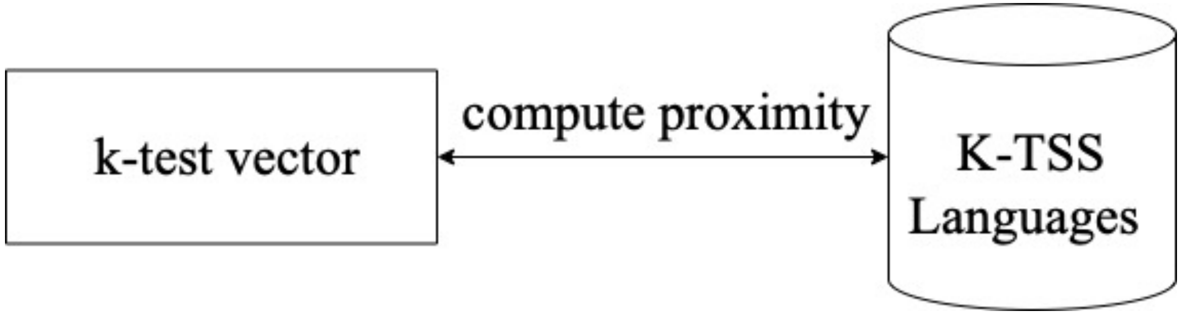

Application Traces
K window size
Trace Generator
Destination Extraction
Network Alphabets
Trace Converter
Language Learner
Classifier
K-TSS Languages
Machine Learning Model
Labeled Trace

Classifier

Aim is to label given network traffic according to learned fingerprints.



FC: Feature Computation Function



$$\Delta T_i = \frac{|T - T_i|}{|T|}, \Delta T_i' = \frac{|T_i - T|}{|T_i|}, \Delta\Sigma_i = \frac{|\Sigma - \Sigma_i|}{|\Sigma|}, \mathrm{SIM}\Sigma_i = \frac{|\Sigma \cap \Sigma_i|}{|\Sigma \cup \Sigma_i|}$$

$$FC(L(w), UZ(\mathrm{App}_i)) = (\Delta T_i, \Delta T_i', \Delta\Sigma_i, \mathrm{SIM}\Sigma_i)$$

Dataset

Dataset name	Encrypted	Homogeneous	Dynamic	Evolving
ReCon	✓	✓		✓
Cross Platform	✓	✓	✓	

ML-NetLang vs. NetLang

Dataset	ML-NetLang (Logistic Regression)			NetLang (Distance Function)		
	Precision	Recall	F1 Score	Precision	Recall	F1 Score
Cross Platform (Android)	0.94	0.95	0.94	0.26	0.36	0.29
Cross Platform(iOS)	0.96	0.97	0.96	0.20	0.27	0.22
Cross Platform(Average)	0.95	0.96	0.95	0.23	0.32	0.26
ReCon	0.97	0.97	0.96	0.29	0.28	0.26
Training Time	<sec			<1 hr		
Test Time	<sec			<milisec		

ML-NetLang vs. FlowPrint and AppScanner

Dataset	ML-NetLang (Logistic Regression)			FLOWPRINT (Jaccard Similarity)			AppScanner (Random Forest)		
	Precision	Recall	F1 Score	Precision	Recall	F1 Score	Precision	Recall	F1 Score
Cross Platform(Android)	**0.94**	**0.95**	**0.94**	0.90	0.87	0.87	0.91	0.89	0.87
Cross Platform(iOS)	**0.96**	**0.97**	**0.96**	0.94	0.93	0.93	0.85	0.15	0.24
Cross Platform(Average)	**0.95**	**0.96**	**0.95**	0.92	0.89	0.89	0.88	0.50	0.58
ReCon	**0.97**	**0.97**	**0.96**	0.95	0.94	0.95	0.90	0.43	0.58

Conclusion

Combined automata learning and machine learning techniques:

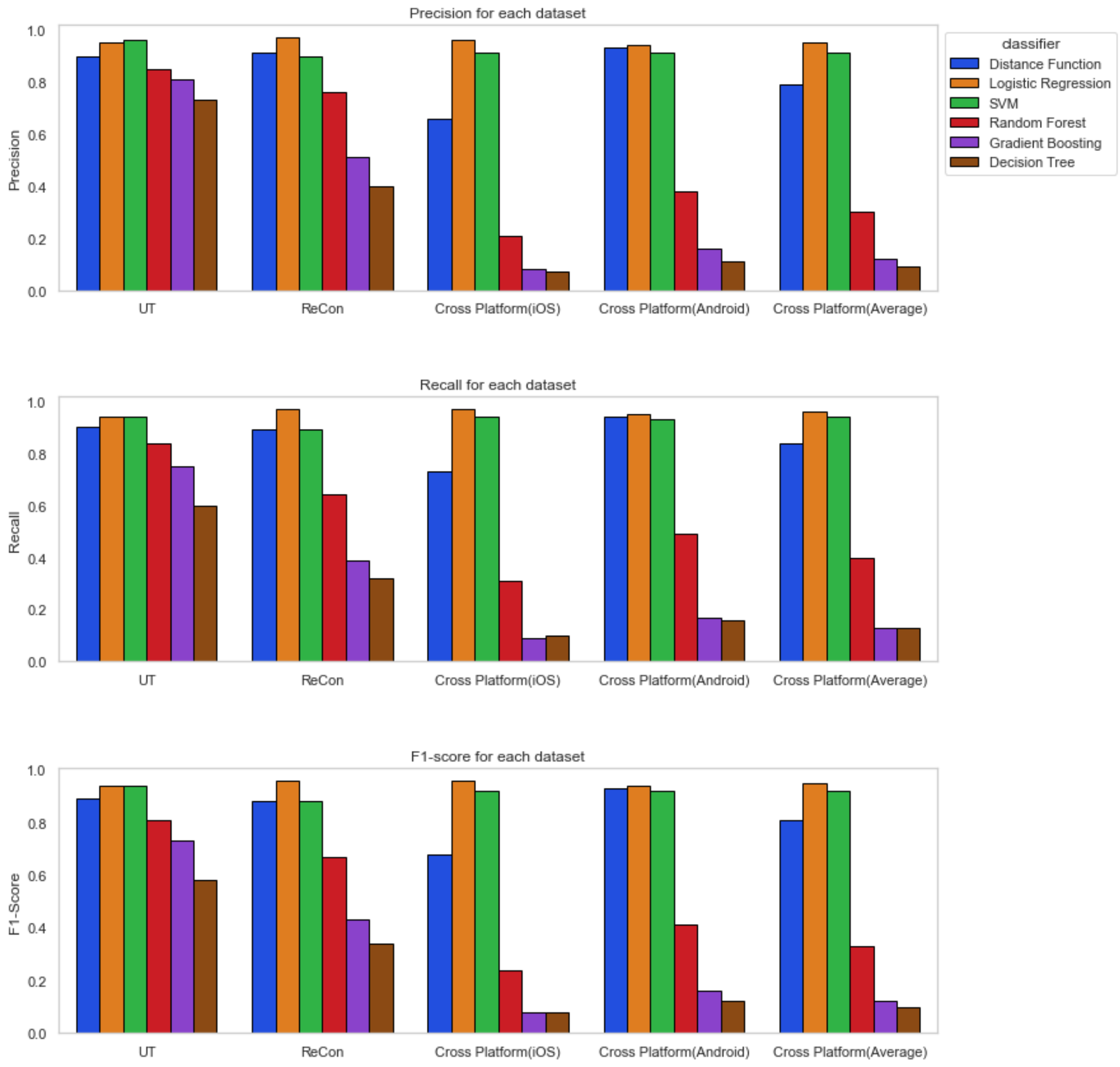
https://github.com/mlnetlang

- Advantages of using Automata Learning:
  - Automatically generating the alphabet of automata
  - Observe temporal relation among flows
- Advantages of using Machine Learning:
  - Upgrading the classification result by using ML algorithm
  - Fast classification

Thank You ☺

- Questions?
- f.marzani.ut@gmail.com

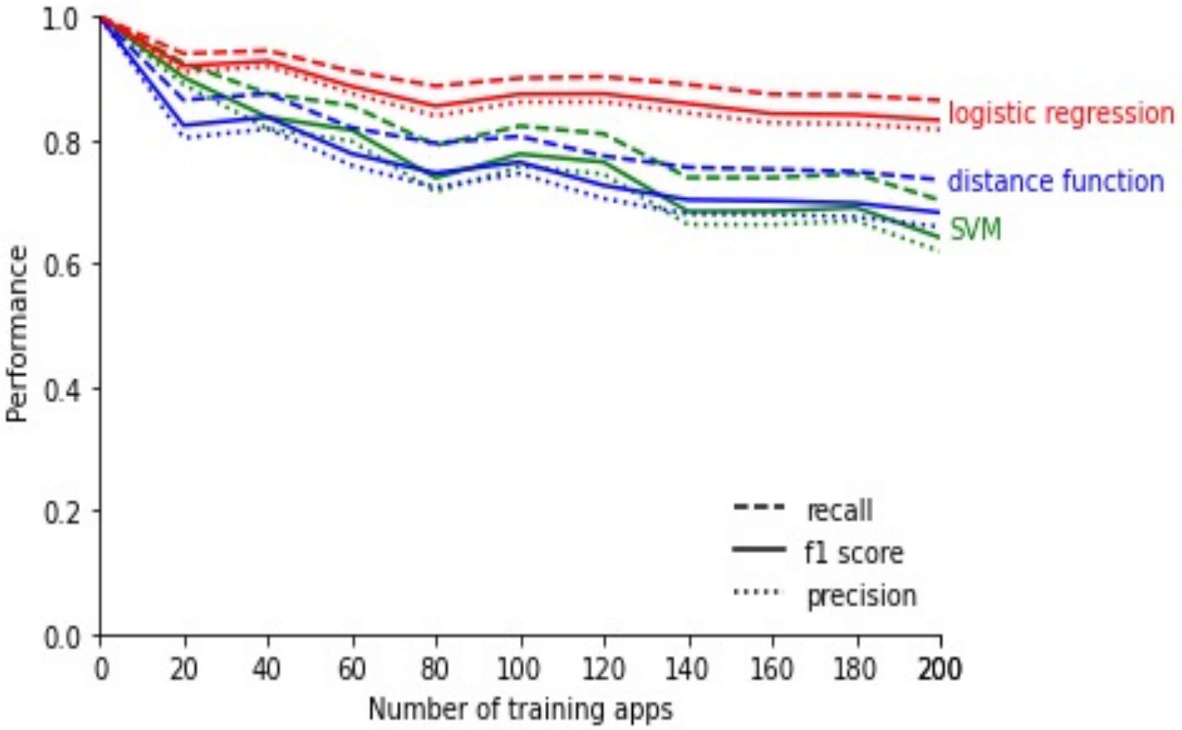
Classifiers



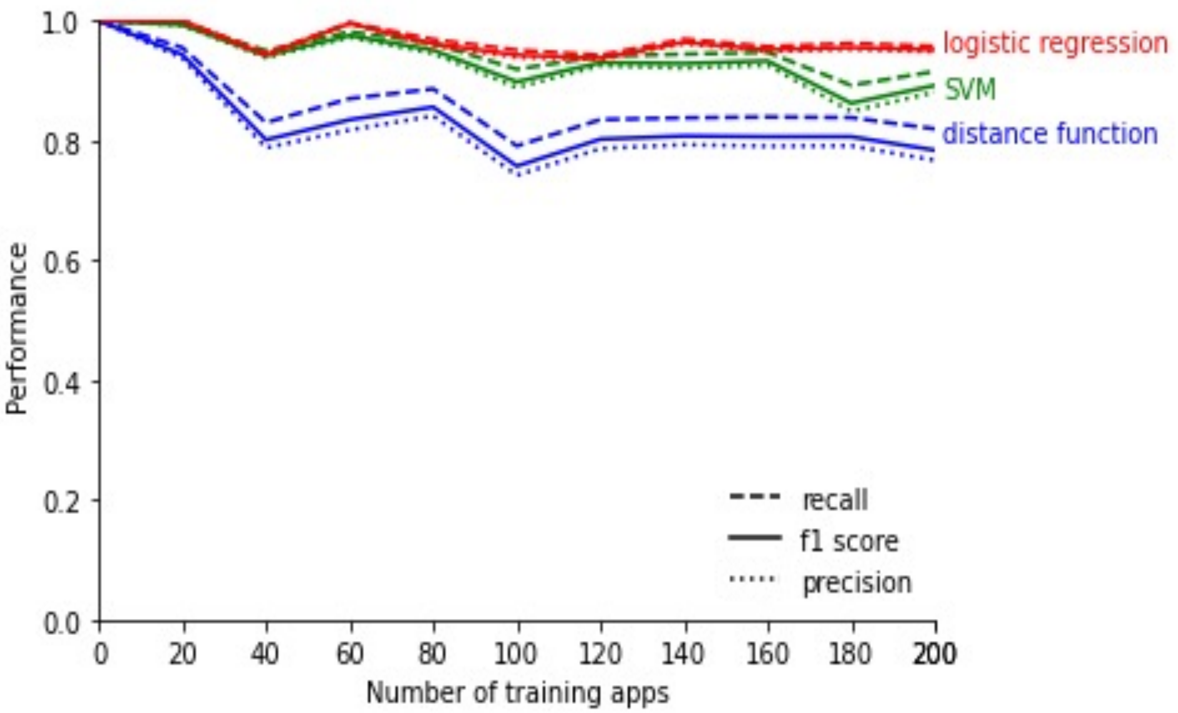
ML-NetLang Performance

- Stable performance if number of apps increase

ReCon

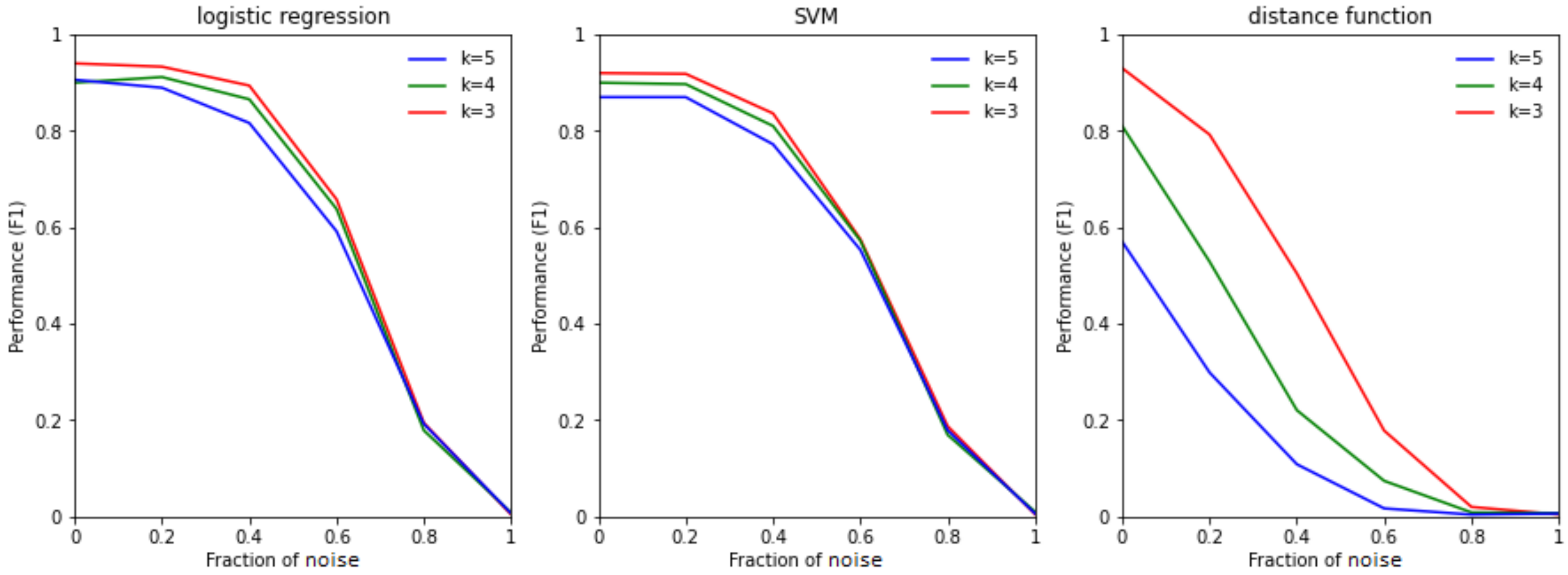


Cross Platform



Window Size (k) vs. Performance

Cross Platform (Android)



Window Size (k) vs. Performance

Cross Platform (iOS)