Automatically and Adaptively Identifying Severe Alerts for Online Service Systems

Nengwen Zhao, Panshi Jin, Lixin Wang, Xiaoqin Yang,
Rong Liu, Wenchi Zhang, Kaixin Sui, Dan Pei

INFOCOM 2020

Outline

Background

Design

Evaluation

Operational Experience

Background

Design

Evaluation

Operational Experience

Service quality

Online service systems have become indispensable parts of our daily life

Search Engine

Online Shopping

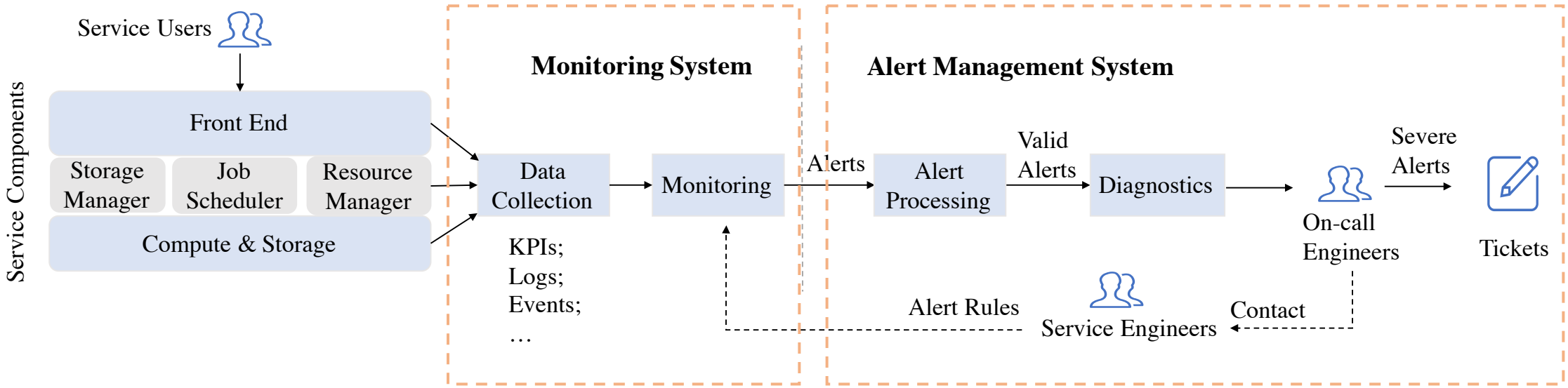
Social Network

Service quality and user experience is vital

one-hour downtime

Loss $100 million !

Monitoring systems and Alert management systems

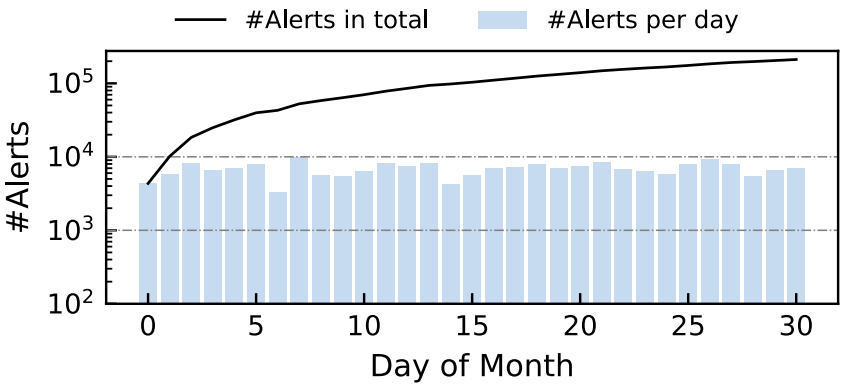


e.g., CPU utilization exceeds 90% log file has error keywords

Monitoring data Alert rules

Alert Fatigue

However, due to the large scale and complexity of online service system, the number of alerts is way more than what on-call engineers can investigate (Alert fatigue)



The number of alerts in a large commercial bank in a given month.

Alert severity

Manual rules are used to classify alerts into different priority levels

Fixed threshold for KPIs:

- CPU utilization over 95% is error
- CPU utilization over 80% is warning

Keywords matching for logs:

- Fail, warning, error…

Alert severity

However

1. Simple manual rules cannot sufficiently capture the patterns of complex and interacting factors that influence the priorities of the alerts.

2. It is labor intensive for engineers to manually define and maintain rules

1) there are many types of alerts
2) new types of alerts might be added due to system changes
3) engineers might have different priority preferences

Alert severity

Manual rules are used to classify alerts into different priority levels

Rule-based approach performs not well,often results in missed severe alerts and lengthened time-to-repair, or results in wasted troubleshooting time on non-severe alerts.

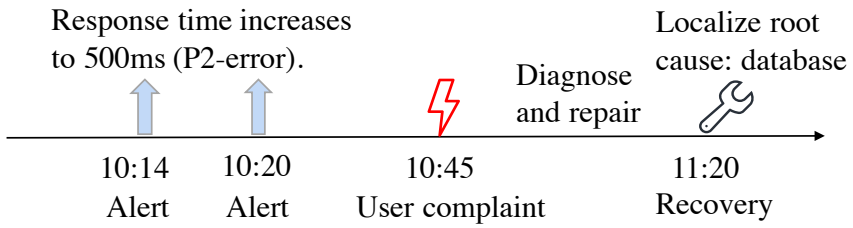
Fixed threshold for KPIs:

- CPU utilization over 95% is error
- CPU utilization over 80% is warning

Keywords matching for logs:

- Fail, warning, error…

Motivation case study



A missing failure case due to the unsuitable rule-based strategy

- Failure was discovered at 10:45 by user complaint
- Before that, there are some related alerts indicating the failure, but they are P2-error
- **wasting 31 minutes of repair time**

Therefore, it is in an urgent need to design an effective algorithm that fully utilizes comprehensive factors to identify severe alerts accurately from numerous alerts.

Challenges

1. Labeling overhead

A large number of alerts every day

2. Large varieties of alerts

Manually define rules for each kind of alerts is unrealistic

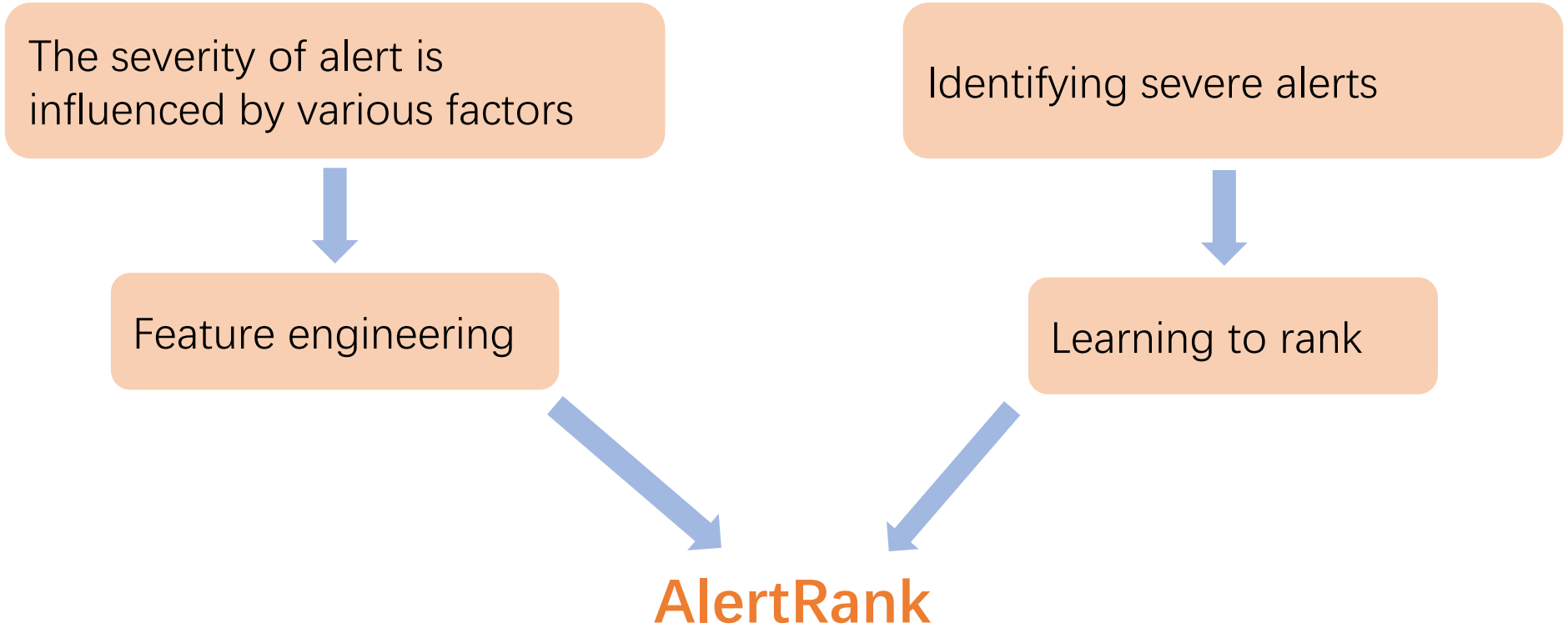
3. Complex and dynamic online service systems

Adapt to the dynamic environment

4. Imbalanced data

Only a small portion of alerts are severe

Key idea


The severity of alert is influenced by various factors
Identifying severe alerts
Feature engineering
Learning to rank
AlertRank

Contributions

1. This work is the first one that proposes an automatic and adaptive approach for identifying severe alerts for online service systems.
2. We design a comprehensive set of features from multiple data sources to characterize the severities of alerts.
3. We formulate the problem of identifying severe alerts as a ranking model, which can handle class imbalance and instruct engineers to repair which alert first
4. Experiments on real-world datasets show AlertRank is effective with a F1-score of 0.89 on average.

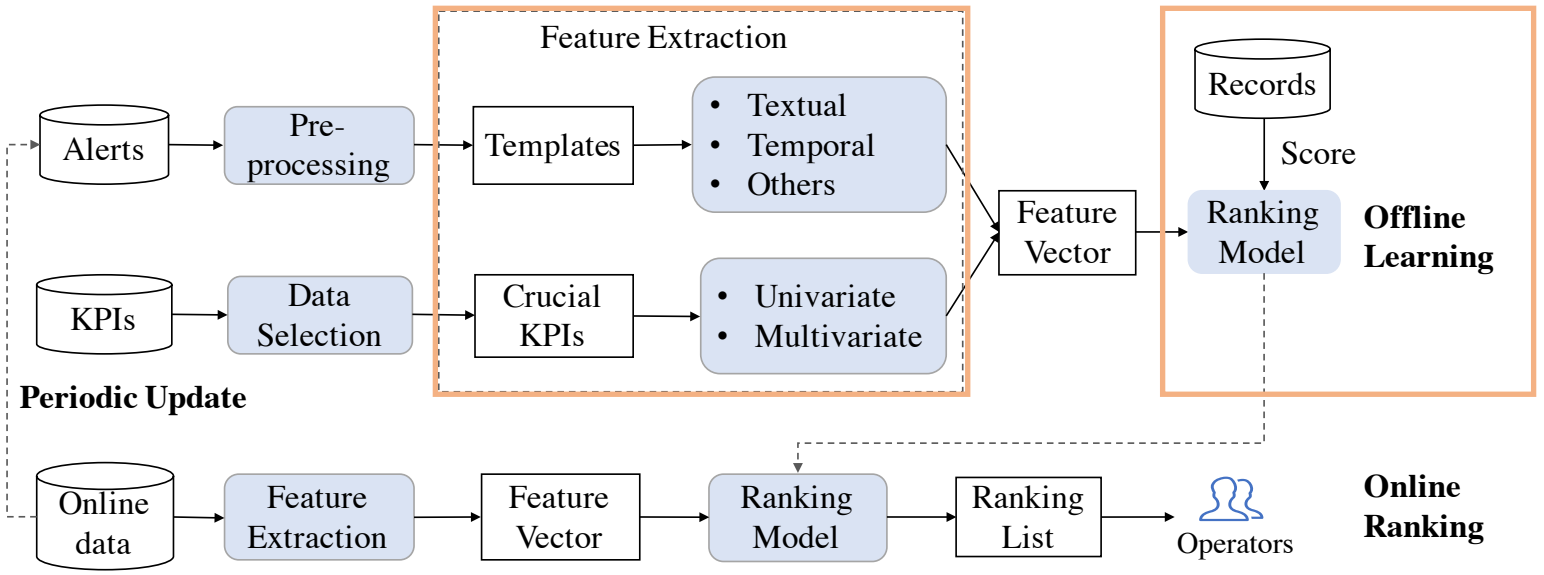
Background

Design

Evaluation

Operational Experience

Overview



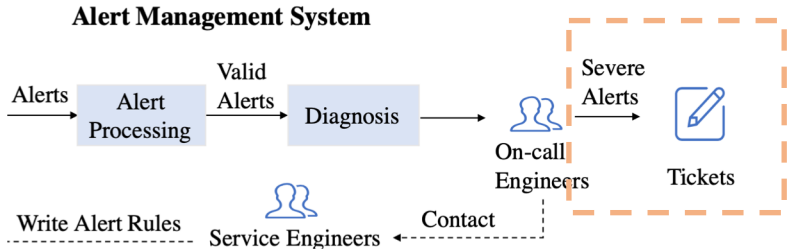
Core idea:

- Multi-feature fusion: alert features and KPI features
- Learning to rank problem

Data prepare

Obtain severity labels automatically from historical resolution records and tickets.

Binary label



Severe or non-severe

Continuous label

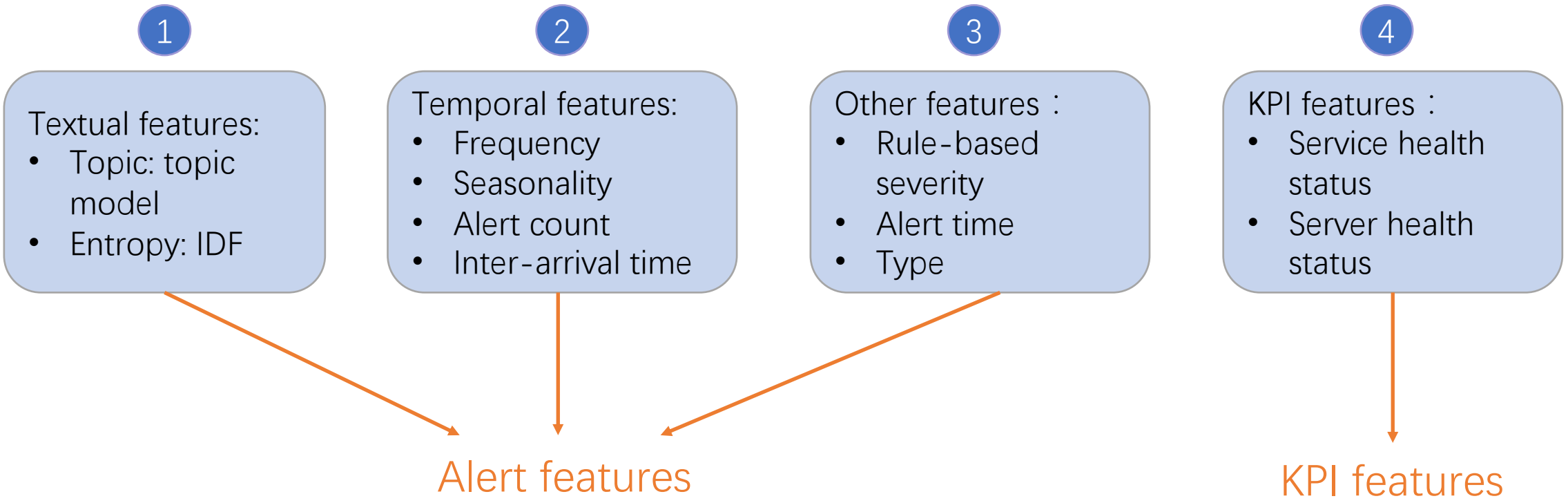
TABLE I: An example alert.

Time	Severity	Type
2019-02-20 10:04:32	P2-error	Memory
AppName	**Server**	**Close Time**
E-BANK	IP(*.*.*.*)	2019-02-20 10:19:45
Content		
Current memory utilization is 79% (Threshold is 60%).		
Resolution Record		
Contact the service engineers responsible for E-BANK and get a reply that there is no effect on business, then close the alert.		

1. None. (0; 65.1%)
2. This alert is in white list. (0.1; 4.2%)
3. This alert has been recovered automatically. (0.2; 7.8%)
4. Contact the service engineers and there is no effect on business. (0.4; 10.6%)
5. Known reasons. This alert has been resolved. (0.6; 6.4%)
6. Contact the service engineers and there is an effect on business. Already resolved. (0.8; 3.8%)
7. Create a ticket. (1; 2.1%)

Severity score

Feature engineering



Alert features

- Alert preprocessing
  - Tokenization
  - Alert parsing: remove parameters
- Textual features
  - Topic : Biterm Topic Model
  - Entropy: based on Inverse Document Frequency

Alert Contents:
A1: Memory utilization current value is 67%. It exceeds the threshold.
A2: TCP CRITICAL - 0.7 second response time on port 3306.
A3: The number of processes is abnormal (instance: TimeoutCtrl), current value is 0.
A4: TCP CRITICAL - 0.8 second response time on port 3302.
A5: Memory utilization current value is 73%. It exceeds the threshold.

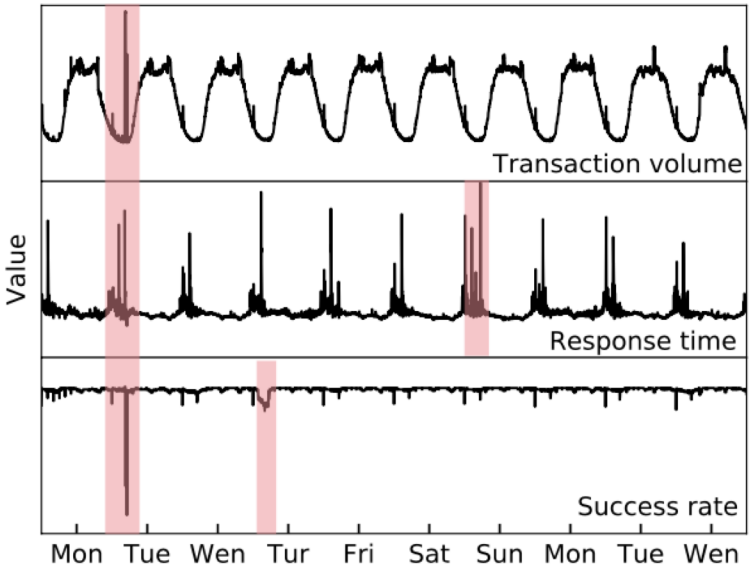
Alert Templates:
T1: Memory utilization current value is *. It exceeds the threshold.
T2: TCP CRITICAL - * second response time on port *
T3: The number of processes is abnormal (instance: *), current value is *.

T#1: oracle, connection, database, space, pool, process, lock
T#2: syslog, alert, error, stack, records, hardware, warning
T#3: monitor, environment, server, temporal, battery, power, voltage
…
T#13: unaccessible, export, response, packet, password, order, accounting
T#14: switch, virtual, communication, connection, health, network, report

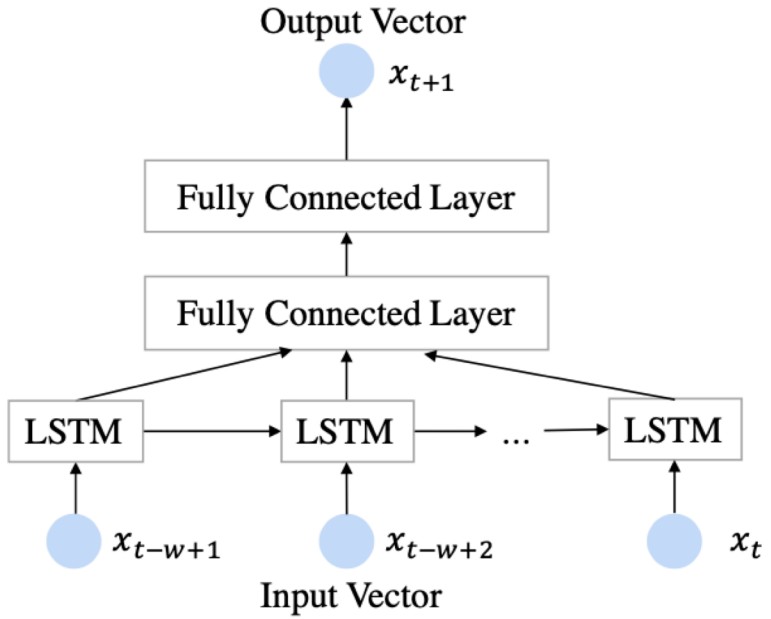
Extracted topics by BTM and some corresponding representative keywords

KPI features

Anomaly score > characterize the health status of servers and applications

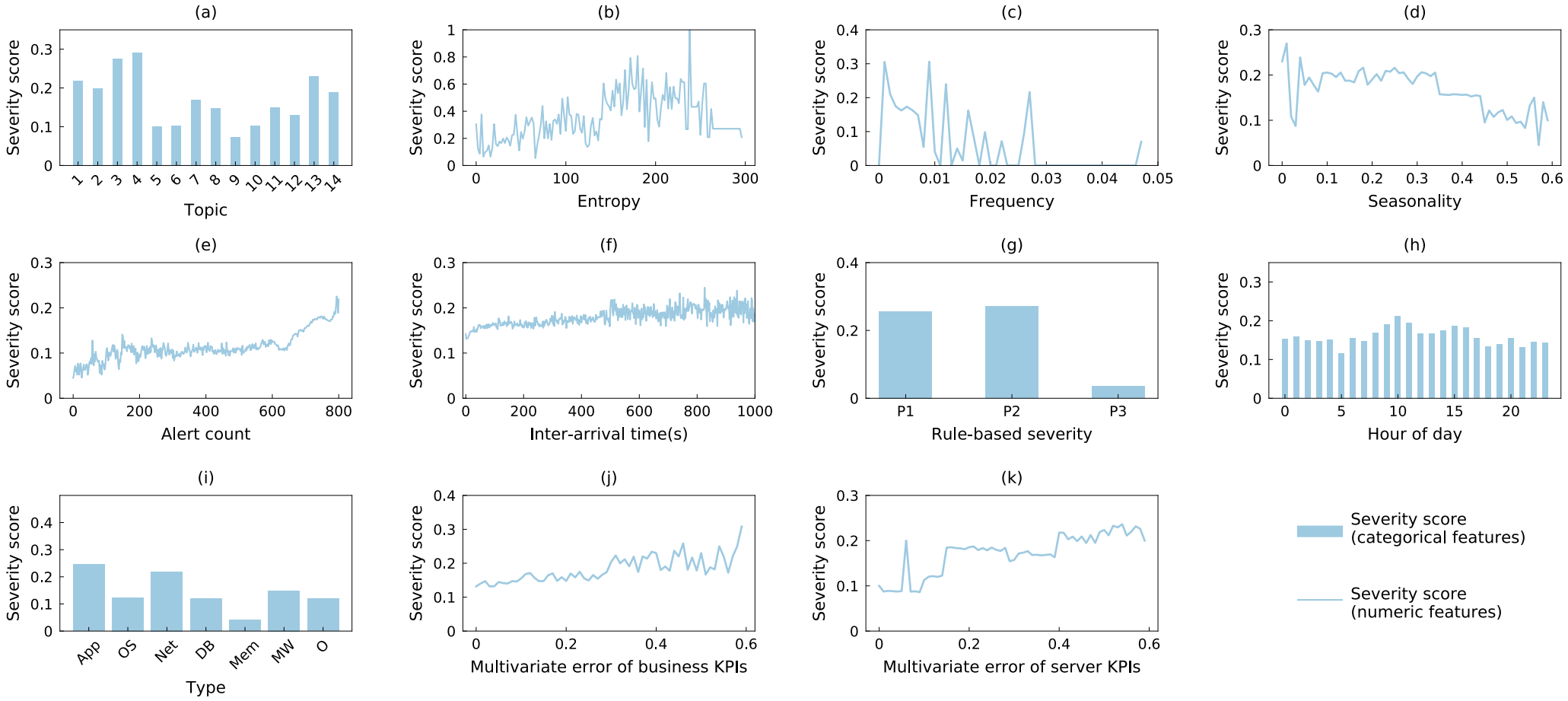


An example of business KPIs



LSTM model

Feature analysis



The qualitative relationship between severity scores and some representative features.

Learning to rank

Feature engineering → Ranking model

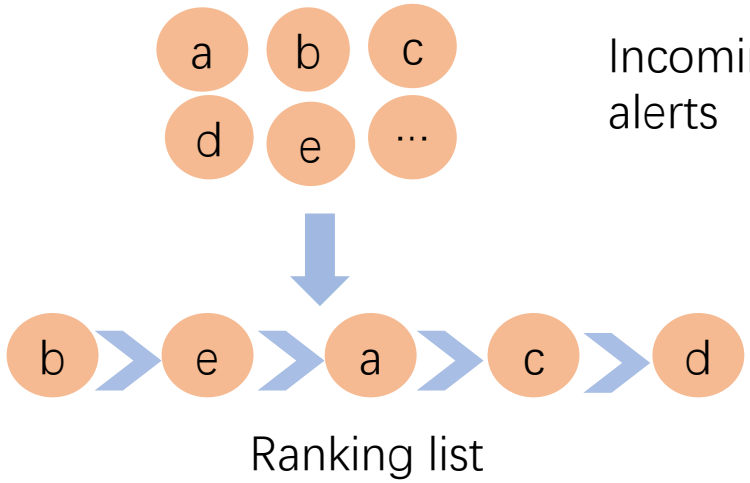
Features

Feature Type	Feature Name	#Feature
Alert Textural	BTM Topics (14), Entropy (1)	15
Alert Temporal	Frequency (1), Seasonality (1), Alert count (4), Inter-arrival time (1)	7
Alert Attributes	Original severity (1), Alert time (3), Type (1)	5
KPI Anomaly	Univariate anomaly (11), Multivariate anomaly (2)	13

Labels

1. None. (0; 65.1%)
2. This alert is in white list. (0.1; 4.2%)
3. This alert has been recovered automatically. (0.2; 7.8%)
4. Contact the service engineers and there is no effect on business. (0.4; 10.6%)
5. Known reasons. This alert has been resolved. (0.6; 6.4%)
6. Contact the service engineers and there is an effect on business. Already resolved. (0.8; 3.8%)
7. Create a ticket. (1; 2.1%)

Model: XGBoost ranking



Background

Design

Evaluation

Operational Experience

Research questions

1. How effective is AlertRank in identifying severe alerts?
2. How much can the alert features and KPI features contribute to the overall performance?
3. Is the ranking model adopted in AlertRank effective?
4. Is the incremental training pipeline useful?

Datasets and metrics

Datasets:

- Three datasets with time span of 6 months from a large commercial bank
- Label: based on tickets
- First 5 months as training set, and last one month as testing set

Metrics:

- Online evaluation
- Precision, recall and F-score
- Top-k precision

TABLE III: Details about the experimental datasets.

Datasets	Time span	#Alerts	#Severe alerts
A	2018/01/01∼2018/06/30	374940	7012
B	2018/07/01∼2018/12/30	429768	8482
C	2019/01/01∼2019/06/30	390437	7445

Overall performance

Baseline methods

- Rule-based: P1-critical, P2- error and P3-warning. Usually, engineers mainly focus on P1 alerts
- Bug-KNN: utilizes KNN to search for historical bug reports that are most similar to a new bug through topic and text similarity.

Datasets	A			B			C		
Methods	P	R	F1	P	R	F1	P	R	F1
AlertRank	**0.85**	**0.93**	**0.89**	**0.82**	**0.90**	**0.86**	**0.93**	**0.92**	**0.93**
Rule-based	0.43	0.68	0.53	0.47	0.70	0.56	0.41	0.74	0.53
Bug-KNN	0.72	0.76	0.74	0.79	0.62	0.70	0.80	0.53	0.64

AlertRank outperforms other baseline method, and achieves the F1-score of 0.89

Feature contribution

Alert features alone

0.89

0.76

KPI features alone

0.89

0.36

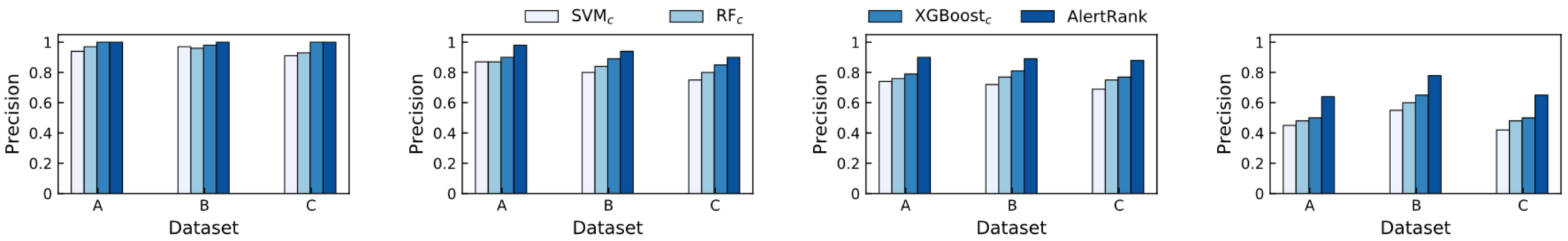
Datasets	A			B			C		
Methods	P	R	F1	P	R	F1	P	R	F1
AlertRank	**0.85**	**0.93**	**0.89**	**0.82**	**0.90**	**0.86**	**0.93**	**0.92**	**0.93**
Alert Only	0.82	0.79	0.80	0.75	0.80	0.77	0.67	0.77	0.72
KPI Only	0.42	0.40	0.41	0.32	0.39	0.35	0.36	0.31	0.33

- Our model benefits from the ensemble features extracted from multiple data sources
- Alert features are more powerful than KPI features.

Effectiveness of ranking model

Datasets	SVM_c	RF_c	$XGBoost_c$	*AlertRank*
A	0.75	0.79	0.80	**0.89**
B	0.70	0.77	0.78	**0.86**
C	0.80	0.81	0.85	**0.93**

The ranking model adopted in AlertRank is indeed effective compared with classification models.



(a) Precision@1. (b) Precision@2. (c) Precision@3. (d) Precision (when recall=1).

Incremental training pipeline

Online service system is under constant change

Adding new alert types and rules

e.g., new applications deployment, software upgrade, or configuration change

Methods	W/o update			Weekly update			**Daily update**		
Time	P	R	F1	P	R	F1	P	R	F1
~ Apr.19	0.82	0.87	0.84	0.83	0.87	0.85	**0.85**	**0.89**	**0.87**
Apr.19 ~	0.76	0.62	0.68	0.80	0.82	0.81	**0.87**	**0.89**	**0.88**

Incremental training is indeed essential to keep our models in tune with highly dynamic online systems

Background

Design

Evaluation

Operational Experience

Success story

1. Using traditional rule-based strategy, engineers need to investigate more alerts, and waste much time in investigating non-severe alerts (low precision) but still miss many severe alerts (not high recall).

2. AlertRank can significantly reduce the number of alerts which engineers need to investigate, while ensuring high precision and recall

Datasets		A	B	C
Rule-based	#Alerts	1996	2536	2094
	P/R	0.43/0.68	0.47/0.70	0.41/0.74
AlertRank	#Alerts	1380	1869	1148
	P/R	0.85/0.93	0.82/0.90	0.93/0.92

Conclusion

1. It is time consuming and error prone for on-call engineers to investigate each alert manually without any guidelines.

2. We novelly propose a ranking-based framework with multi-feature fusion named AlertRank that identifies severe alerts automatically and adaptively

3. AlertRank can significantly reduce the number of alerts which engineers need to investigate while ensuring high precision and recall.

Thank you!

Q&A

znw17@mails.tsinghua.edu.cn

INFOCOM 2020