Continuous Incident Triage for Large-Scale Online Service Systems

Junjie Chen

Tenured Associate Professor

Tianjin University

天津大学

Tianjin University

Microsoft Research

Microsoft

Online Service System (OSS)

OSSs become increasingly popular in recent years

Skype has about 300 millions of active users as of October 2018

Office 365 has about 135 million monthly active users in 2018

There are on average 120K new Azure customer subscriptions per month in 2017

OSS incidents

Source code bugs

Misconfiguration

Network traffic

Incidents: unplanned interruptions and outages of the service

cause

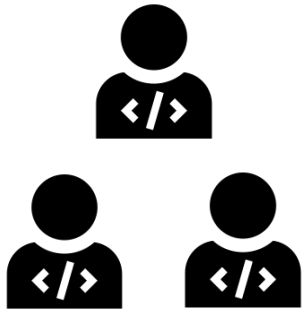
Huge economic loss and serious consequences

Example 1: The estimated cost of the one-hour downtime for Amazon.com on Prime Day in 2018 is up to $100 million

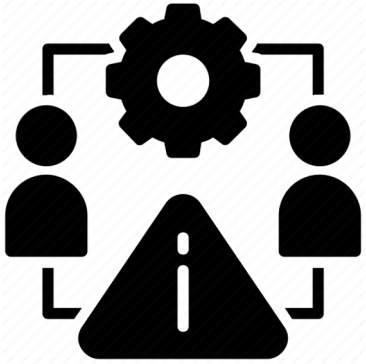
Example 2: The average cost of service downtime has steadily increased from $505,502 in 2010 to $740,357 in 2016 for 63 data center organizations in the U.S.

Incident Triage

Incidents



Incident triage: *assigning a new incident to the responsible team*



Incident mitigation ASAP

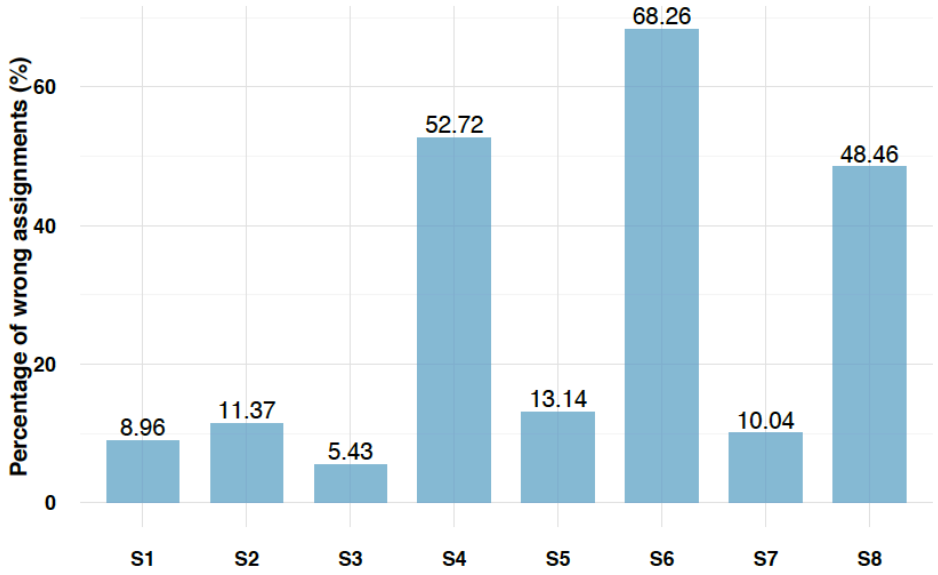
Accurate and efficient incident triage is very challenging for OOS

Reason	More Explanation
Incidents are automatically reported by monitors rather than people	created based on certain simple templates; no detailed textual descriptions

Accuracy of Incident Triage in the Beginning (Manual Triage)

Benchmark for evaluation

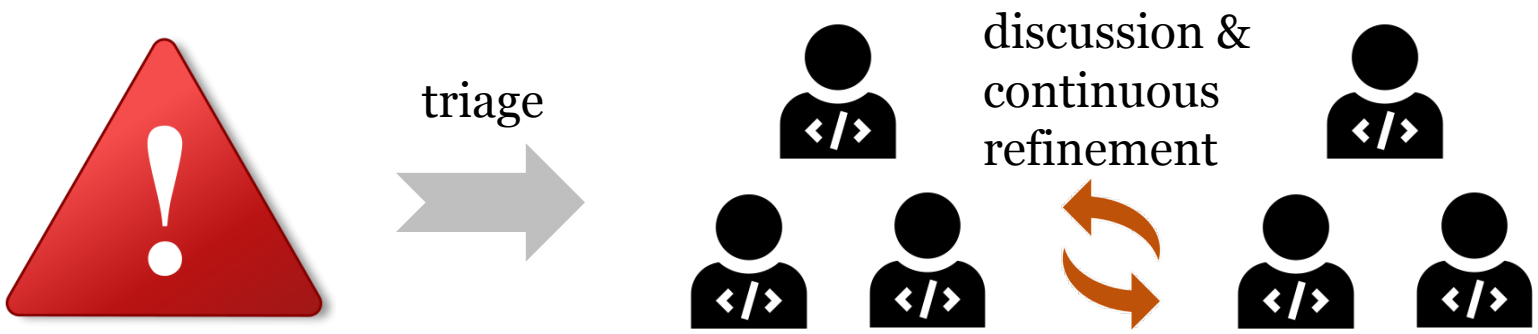
- 8 industrial large-scale OOS in Microsoft
- Six months of resolved incidents



The average percentage is 27.3%

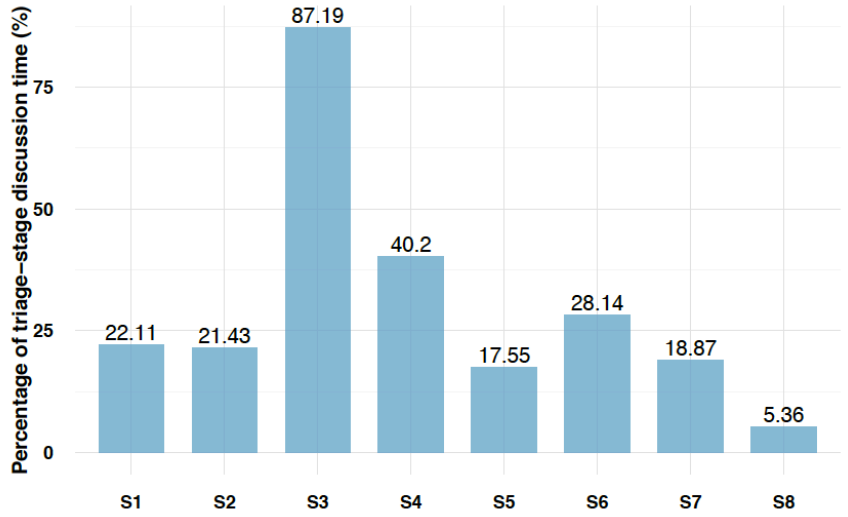
Even though the OCEs have rich experience and domain knowledge, they still make many mistakes during incident triage in the beginning due to the limited information provided by the incident reports

Continuous Incident Triage



Continuous Incident Triage
(practical scenario)

Sys.	S1	S2	S3	S4	S5	S6	S7	S8
Num.	8.45	20.55	10.75	15.81	6.42	8.59	13.56	6.42
Per.(%)	52.29	55.53	78.41	38.26	51.40	74.18	47.31	62.27



More than half of discussion items are conducted to refine incident triage

The discussion time spent on incident triage is non-trivial

Motivate to propose an effective approach to continuously refining incident triage based on incrementally provided discussions.

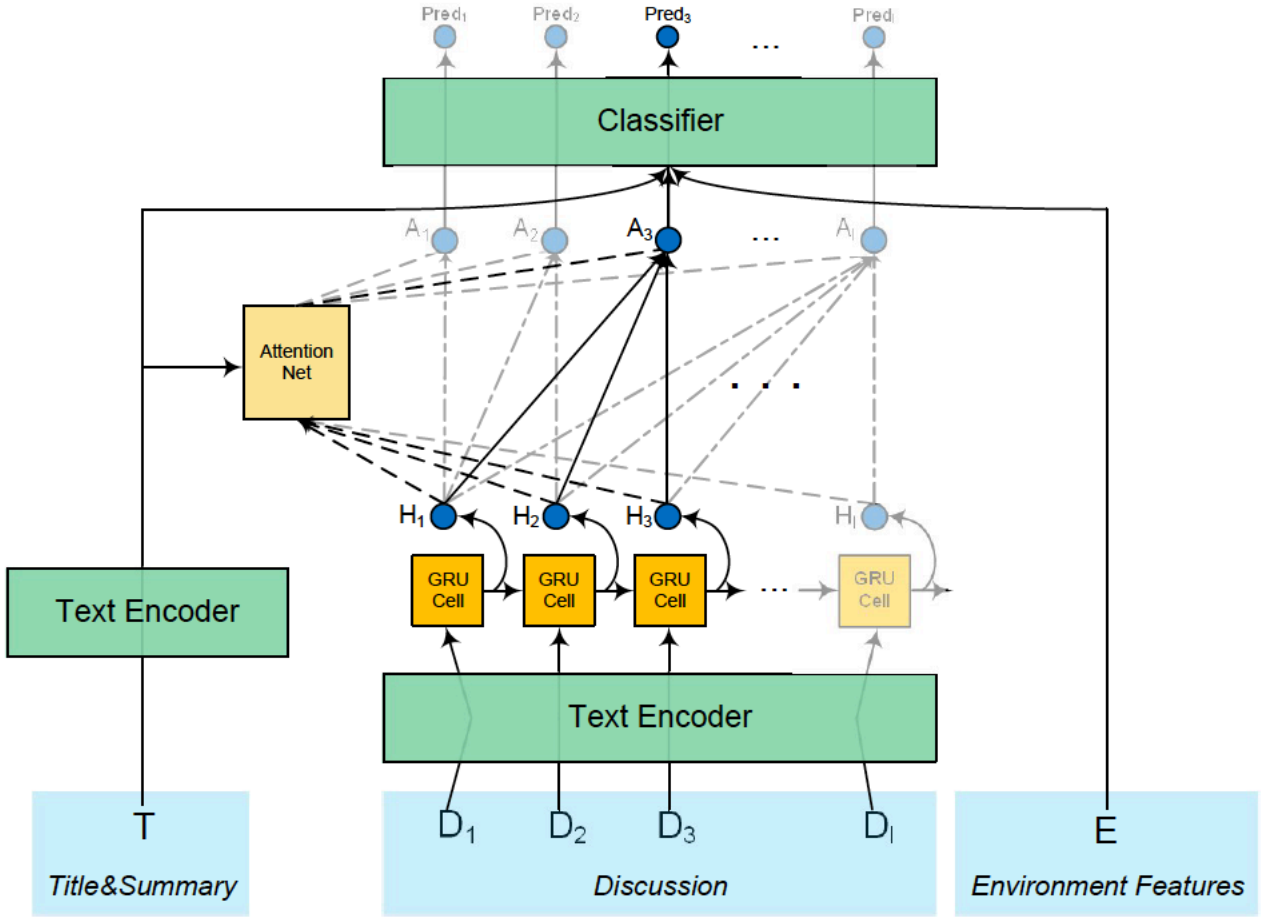
Approach — DeepCT

Existing triage approaches cannot work well in this real-world scenario

- either ignore discussions or simply treat all discussions as a whole
- without considering their characteristics, i.e., incremental creation

Challenges in continuous incident triage

- how to learn knowledge from incremental discussions to fit the scenario of continuous incident triage
- how to reduce the impact of noise introduced by manual discussions (like conversations) on incident triage



Input Data

title and summary of an incident report

- the textual description about the symptom when an incident is reported

incremental discussions about an incident

- manually written by engineers incrementally like conversations
- core information for continuous incident triage

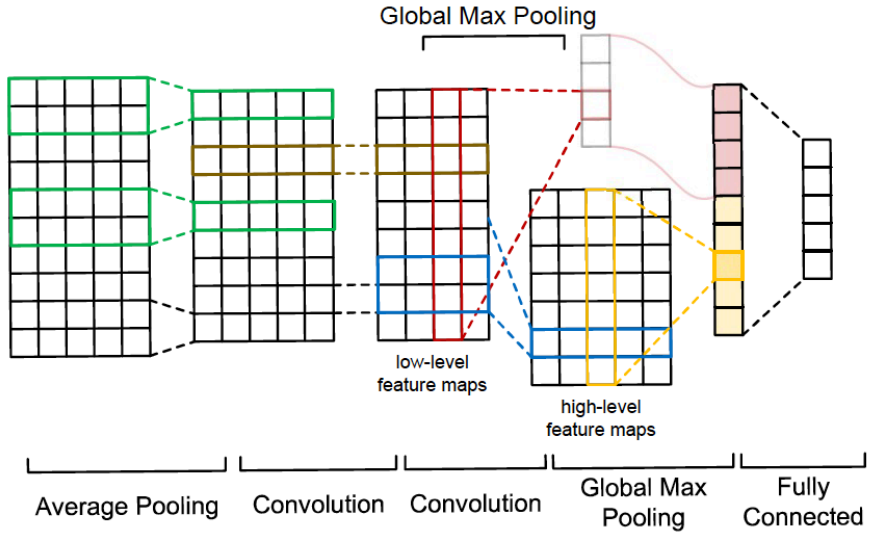
incident-occurring environment info

- including the monitor ID reporting the incident, the incident-occurring device, and the incident type (monitor reporting or human reporting)

Domain-Specific Text Encoding

There are many special terms in textual descriptions, such as API names and component names, which are helpful but cannot be well handled by traditional text encoding methods due to the small occurrence frequency of each special term.

build pre-trained subword vectors based on external corpus, and conducts fine tuning based on historical incident data to incorporate the domain knowledge



CNN-based encoder for the first and second type of input data

conduct representation learning to embed the third type of input data

Designed GRU-Based Model

Enhance the learning for the knowledge from earlier discussions so that correct assignments can be achieved with fewer discussions

GRU network

- Considering temporal relations among discussions
- Reset gate to forget some past information
- Update gate decide what to collect from previous discussion items

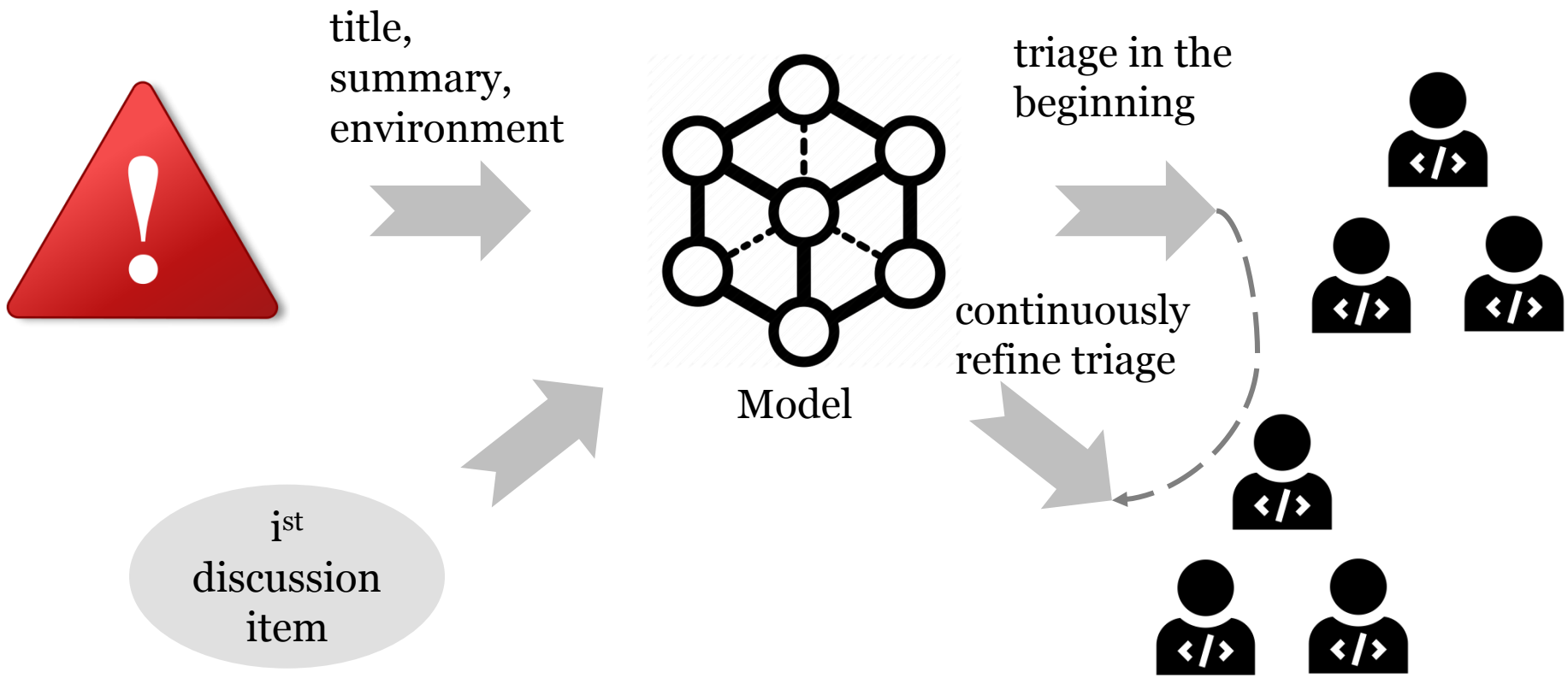
Attention-based mask strategy

- Noise can be masked by assigning them quite small weights
- Weights are calculated by the *softmax* function

Continuous loss function

- Our goal is to achieve the correct incident-triage result as much as possible at each time step
- Instead of calculating the loss at the last time step, it calculates the sum of the loss at each time step

Usage of DeepCT



Evaluation

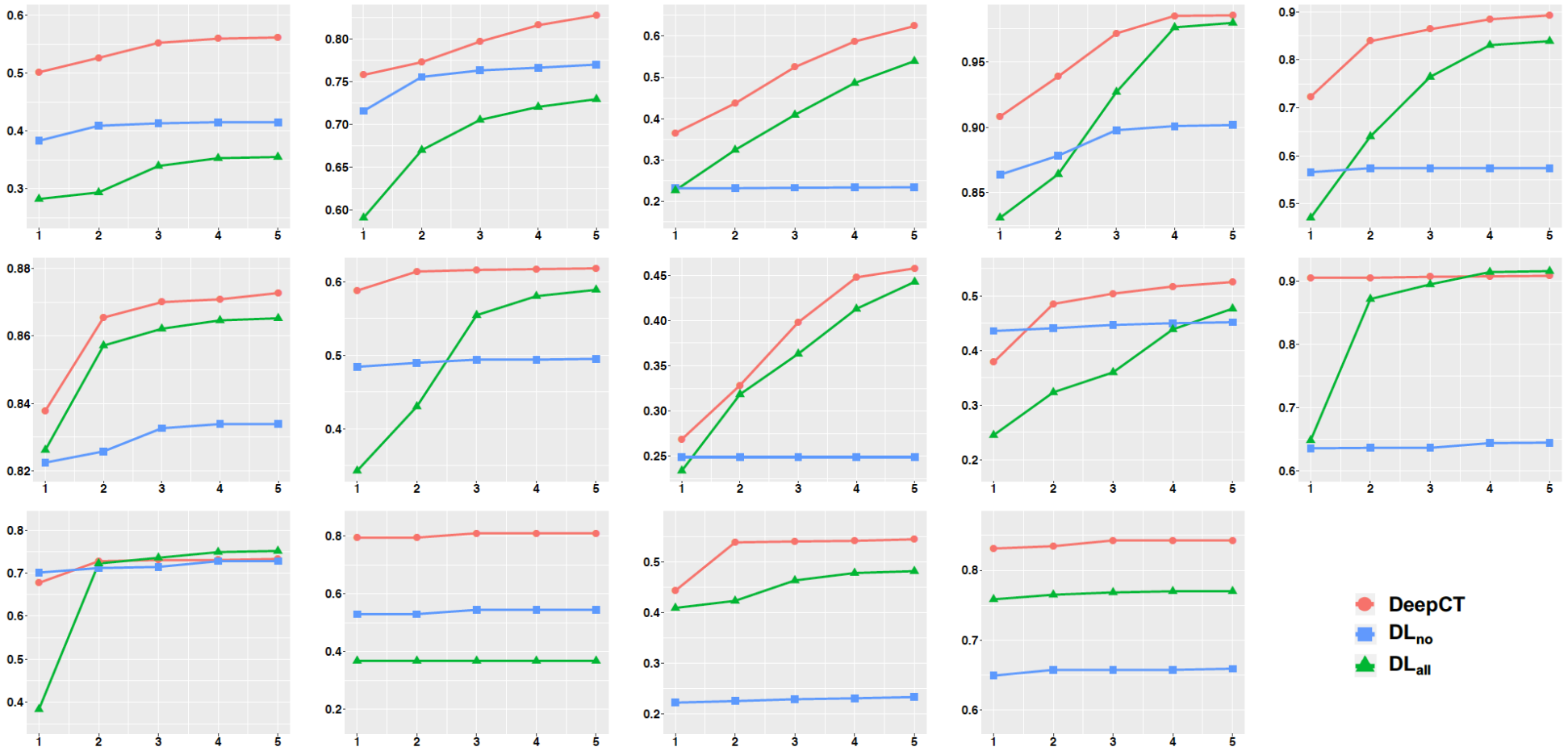
Benchmark

- 14 industrial large-scale OOS in Microsoft
- different application areas and developed by different product groups
- six months of incident data (resolved)
- over 90GB incident reports
- about 2000 teams
- former four months for training, the latter two months for predicting

Compared approaches

- State-of-the-art deep learning based bug triage (DL): use CNN to train the classifier based on textual descriptions (title & summary)
- DL_{no}: ignore discussions during training
- DL_{all}: treat all discussions as a whole during training

Effectiveness of DeepCT



Effectiveness comparison among DeepCT, DL_{no}, and DL_{all} for each studied online service system (the x-axis represents the number of discussion items and the y-axis presents the accuracy of incident triage)

Effectiveness of DeepCT

Statistical analysis

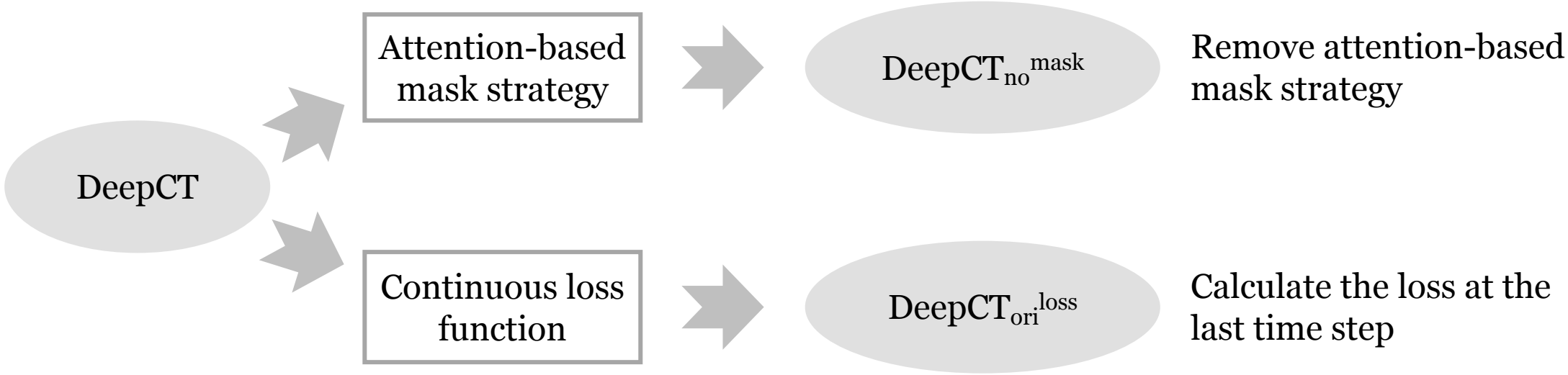
Approach		# Discussion Items 1	2	3	4	5
Avg.	DeepCT	0.641	0.686	0.709	0.722	0.729
	DL_{no}	0.535	0.544	0.549	0.551	0.552
	DL_{all}	0.473	0.562	0.608	0.639	0.650
⇑(%)	vs DL_{no}	19.81	26.10	29.14	31.03	32.07
	vs DL_{all}	35.52	22.06	16.61	12.99	12.15
p-val	vs DL_{no}	**0.004**	**0.000**	**0.000**	**0.000**	**0.000**
	vs DL_{all}	**0.000**	**0.000**	**0.000**	**0.002**	**0.002**

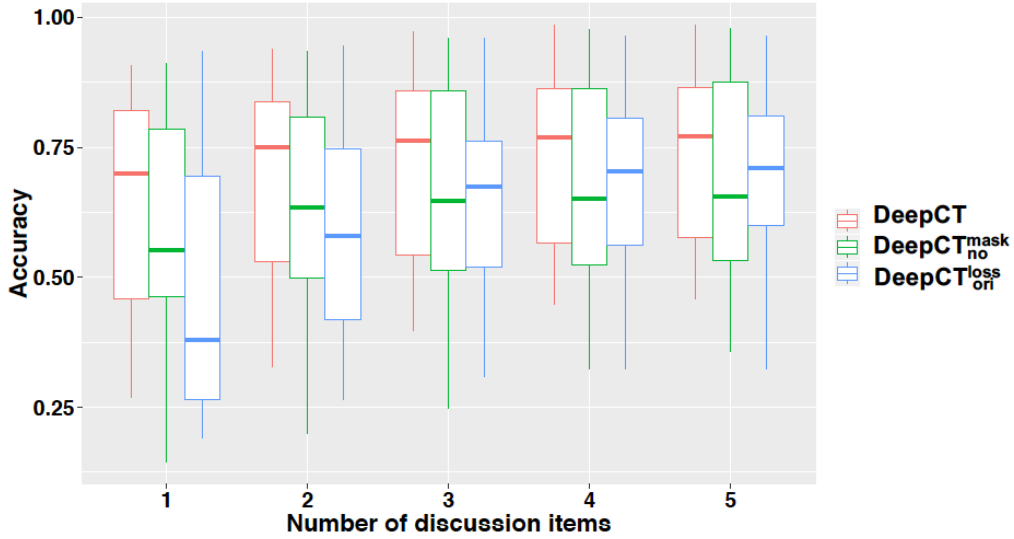
DeepCT significantly improves DL_{no} by 18.92%~30.88% and improves DL_{all} by 12.15%~35.52%

when the number of discussion items is small, the title&summary has a larger impact on prediction, and thus DL_{no} is more suitable than DL_{all}

when the number of discussion items becomes larger, the impact of discussions also increases, and thus DL_{all} is more suitable than DL_{no}

Contribution of Each Main Component

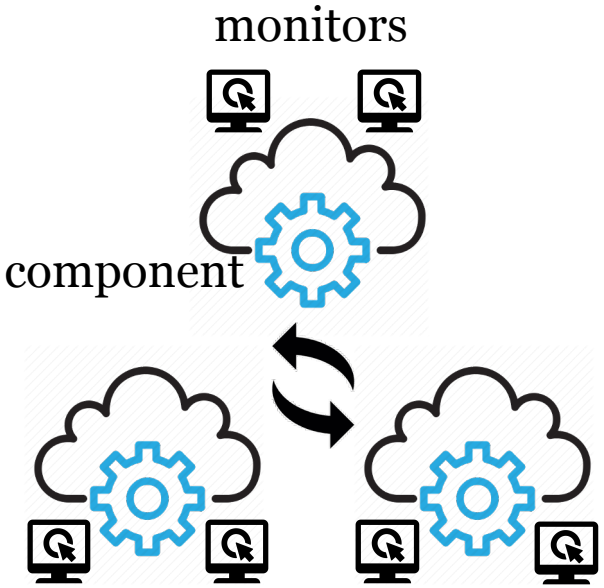




Both of attention-based mask strategy and continuous loss function contribute to DeepCT

Continuous loss function makes more contributions when the number of discussion items is small, while attention-based mask strategy makes more contributions when the number of discussion items is large.

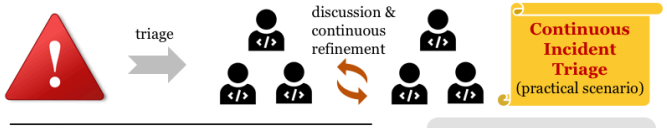
Lessons Learned



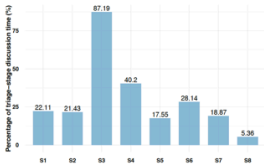
1. Many fault-tolerant techniques are designed, and thus an incident to an individual component may not affect the overall system and an incident to the overall system may be reflected by many components
2. (OCEs) Hard to fully understand the entire system and are often confused by the actual causes of an incident
3. Product teams that are responsible for maintaining individual components may not understand the details about other components and the entire system
4. Many incidents, reported by different monitors, have the same root cause and are duplicated or linked

Summary

Continuous Incident Triage



Sys.	S1	S2	S3	S4	S5	S6	S7	S8
Num.	8.45	20.55	10.75	15.81	6.42	8.59	13.56	6.42
Per.(%)	52.29	55.53	78.41	38.26	51.40	74.18	47.31	62.27



More than half of discussion items are conducted to refine incident triage

The discussion time spent on incident triage is non-trivial

Motivate to propose an effective approach to continuously refining incident triage based on incrementally provided discussions.

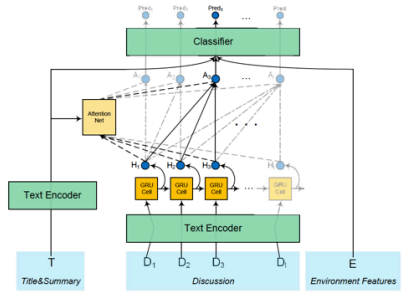
Approach — DeepCT

Existing triage approaches cannot work well in this real-world scenario

- either ignore discussions or simply treat all discussions as a whole
- without considering their characteristics, i.e., incremental creation

Challenges in continuous incident triage

- how to learn knowledge from incremental discussions to fit the scenario of continuous incident triage
- how to reduce the impact of noise introduced by manual discussions (like conversations) on incident triage



Evaluation

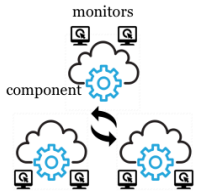
Benchmark

- 14 industrial large-scale OOS in Microsoft
- different application areas and developed by different product groups
- six months of incident data (resolved)
- over 90GB incident reports
- about 2000 teams
- former four months for training, the latter two months for predicting

Compared approaches

- State-of-the-art deep learning based bug triage (DL): use CNN to train the classifier based on textual descriptions (title & summary)
- DL_{no}: ignore discussions during training
- DL_{all}: treat all discussions as a whole during training

Lessons Learned



1. Many fault-tolerant techniques are designed, and thus an incident to an individual component may not affect the overall system and an incident to the overall system may be reflected by many components
2. (OCEs) Hard to fully understand the entire system and are often confused by the actual causes of an incident
3. Product teams that are responsible for maintaining individual components may not understand the details about other components and the entire system
4. Many incidents, reported by different monitors, have the same root cause and are duplicated or linked