1. 数据

1.1 数据集

大规模测试组测试场景时的所有配置，包括subnet、port、security group、node等配置，每个配置具有时间戳，可以根据时间戳确定每个VM的配置下发时序。

以subnet配置为例，配置由多个属性组成：

```
"{\"networks\":[{\"default\":true,\"id\":\"9192a4d4-ffff-4ece-b3f0-
8d36e3d88001\",\"project_id\":\"3dda2801-d675-4688-a63f-
dcda8d327f50\",\"tenant_id\":\"3dda2801-d675-4688-a63f-
dcda8d327f50\",\"name\":\"sample_vpc\",\"description\":\"vpc\",\"cidr\":\"172.16
.0.0/16\",\"routes\":
[{\"destination\":\"172.16.0.0/16\",\"target\":\"Local\",\"priority\":0,\"associ
atedType\":\"VPC\",\"associatedTableId\":\"453adbdc-3e37-458b-a126-
9fa648ef6c6a\",\"id\":\"9c326041-551e-4db6-a6e9-
2ea166e21b72\",\"project_id\":\"3dda2801-d675-4688-a63f-
dcda8d327f50\",\"tenant_id\":\"3dda2801-d675-4688-a63f-
dcda8d327f50\",\"name\":\"default_route_rule\",\"description\":\"\"}],\"router\":
{\"id\":null,\"project_id\":null,\"tenant_id\":null,\"name\":null,\"description\
":null,\"neutron_router_routetable\":null,\"neutron_subnet_routetables\":null,\"
subnet_Ids\":null,\"routetables\":null,\"vpc_default_route_table_id\":null,\"own
er\":null,\"router_extra_attribute_id\":null,\"gateway_ports\":null,\"admin_stat
e_up\":false,\"status\":null,\"created_at\":null,\"updated_at\":null},\"admin_st
ate_up\":true,\"dns_domain\":\"domain\",\"mtu\":1400,\"port_security_enabled\":t
rue,\"provider:network_type\":\"vxlan\",\"provider:physical_network\":\"\",\"pro
vider:segmentation_id\":15847424,\"router:external\":false,\"segments\":
[],\"shared\":false,\"vlan_transparent\":false,\"is_default\":true,\"availabilit
y_zone_hints\":[],\"availability_zones\":
[\"Nova\"],\"qos_policy_id\":\"\",\"revision_number\":1,\"status\":\"ACTIVE\",\"
tags\":[],\"created_at\":\"2022-06-15 11:28:41\",\"updated_at\":\"2022-06-15
11:28:41\",\"ipv4_address_scope\":\"\",\"ipv6_address_scope\":\"\",\"l2_adjacenc
y\":\"\",\"subnets\":[\"8182a4d4-ffff-4ece-b3f0-8d36e3d88001\"]}]}"
```


解析之后的配置数据格式如下：

```
id              8182a4d4-ffff-4ece-b3f0-8d36e3d88001
project_id              3dda2801-d675-4688-a63f-dcda8d327f50
tenant_id              3dda2801-d675-4688-a63f-dcda8d327f50
name              subnet1
description
network_id              9192a4d4-ffff-4ece-b3f0-8d36e3d88001
cidr              172.16.0.0/16
availability_zone
gateway_ip              172.16.0.1
gatewayPortId              0fa8e166-af9e-47cc-9824-fb5cf000ca48
gateway_port_detail              {'gateway_macAddress': 'aa:bb:cc:d2:8a:5c',
'gateway_port_id': '0fa8e166-af9e-47cc-9824-fb5cf000ca48'}
attached_router_id
port_detail              {}
enable_dhcp              True
primary_dns
secondary_dns
dns_list              []
ip_version              4
ipv4_rangeId              8234e0a7-7dfe-4e73-89ac-1d964eef2da1
ipv6_rangeId
ipv6_address_mode
ipv6_ra_mode
revision_number              1
segment_id
```

```
shared
sort_dir
sort_key
subnetpool_id
dns_publish_fixed_ip           False
tags           []
tags-any
not-tags
not-tags-any
fields
dns_nameservers           []
allocation_pools           [{'start': '172.16.0.1', 'end': '172.16.255.254'}]
host_routes           [{'destination': '172.16.1.0/24', 'nexthop':
'172.16.1.1'}]
prefixlen
use_default_subnet_pool           False
service_types           []
created_at           2022-06-15 11:28:46
updated_at           2022-06-15 11:28:46
```

每个配置所含的属性不同。

1.2 数据处理

需要将每个配置转化为DL模型可以接受的向量。

对于每个配置，输入向量的维度为1×N，N为配置文本中提取出词的数目，每个词由词典中的数表示。

配置文本中包含四种词：

1. 具有明确含义的词：属性名（project_id、cidr、allocation_pools等）、name（subnet1）、description、True/False
2. 表示ID的随机字符串： 8182a4d4-ffff-4ece-b3f0-8d36e3d88001
3. IP、MAC等具有网络特点含义的词：172.16.0.1、aa:bb:cc:d2:8a:5c
4. 表示数量的数字：4

在一般的NLP文本处理中，【1,3,4】词有相应的含义，可以作为一类一起处理，【2】没有具体含义，词之间没有关联。

1.2.1 step1: 分词

首先，需要提取配置文本中有意义的数据，剔除无效的符号等。由于配置的类型不同，各自属性也不同，不能统一处理，这里采用NLP中的分词算法，自动提取配置中的词。

目前英文词中常用的分词算法有NLTK、SpaCy、StanfordCoreNLP、BertWordPiece等：

example：采用四种方法对以下配置分词：

```
"[{'destination': '172.16.0.0/16', 'target': 'Local', 'priority': 0,
'associatedType': 'VPC', 'associatedTableId': '453adbdc-3e37-458b-a126-
9fa648ef6c6a', 'id': '9c326041-551e-4db6-a6e9-2ea166e21b72', 'project_id':
'3dda2801-d675-4688-a63f-dcda8d327f50', 'tenant_id': '3dda2801-d675-4688-a63f-
dcda8d327f50', 'name': 'default_route_rule', 'description': ''}]"
```

三种分词结果如下：

```
NLTK:
['destination', '172160016', 'target', 'Local', 'priority', '0',
'associatedType', 'VPC', 'associatedTableId',
'453adbdc3e37458ba1269fa648ef6c6a', 'id', '9c326041551e4db6a6e92ea166e21b72',
'projectid', '3dda2801d6754688a63fdcda8d327f50', 'tenantid',
'3dda2801d6754688a63fdcda8d327f50', 'name', 'defaultrouterule', 'description']


Spacy:
['destination', '172160016', 'target', 'Local', 'priority', '0',
'associatedType', 'VPC', 'associatedTableId',
'453adbdc3e37458ba1269fa648ef6c6a', 'i', 'd',
'9c326041551e4db6a6e92ea166e21b72', 'projectid',
'3dda2801d6754688a63fdcda8d327f50', 'tenantid',
'3dda2801d6754688a63fdcda8d327f50', 'name', 'defaultrouterule', 'description']


Bert wordpiece:
['destination', '1721', '##60', '##01', '##6', 'target', 'local', 'priority',
'0', 'associated', '##type', 'vp', '##c', 'associated', '##table', '##id', '45',
'##3', '##ad', '##b', '##dc', '##3', '##e', '##37', '##45', '##8', '##ba',
'##12', '##6', '##9', '##fa', '##64', '##8', '##ef', '##6', '##c', '##6', '##a',
'id', '9', '##c', '##32', '##60', '##41', '##55', '##1', '##e', '##4', '##db',
'##6', '##a', '##6', '##e', '##9', '##2', '##ea', '##16', '##6', '##e', '##21',
'##b', '##7', '##2', 'project', '##id', '3d', '##da', '##28', '##01', '##d',
'##6', '##75', '##46', '##8', '##8', '##a', '##6', '##3', '##f', '##dc', '##da',
'##8', '##d', '##32', '##7', '##f', '##50', 'tenant', '##id', '3d', '##da',
'##28', '##01', '##d', '##6', '##75', '##46', '##8', '##8', '##a', '##6', '##3',
'##f', '##dc', '##da', '##8', '##d', '##32', '##7', '##f', '##50', 'name',
'default', '##rou', '##ter', '##ule', 'description']
```

分词算法	效果
NLTK	可以正确提取配置中的所有类型词
SpaCy	相比NLTK，分的更细，拆开了部分词破坏了语义
StanfordCoreNLP	
Bert wordPiece	对网络背景的词处理不好

综上，我们采用NLTK分词算法对配置文本分词，subnet配置的分词效果如下：

```
['id', '8182a4d4ffff4eceb3f08d36e3d88001']
['project_id', '3dda2801d6754688a63fdcda8d327f50']
['tenant_id', '3dda2801d6754688a63fdcda8d327f50']
```

```
['name', 'subnet1']
['description']
['network_id', '9192a4d4ffff4eceb3f08d36e3d88001']
['cidr', '172160016']
['gateway_ip', '1721601']
['gatewayPortId', '0fa8e166af9e47cc9824fb5cf000ca48']
['gateway_port_detail', 'gatewaymacAddress', 'aabbccd28a5c', 'gatewayportid',
'0fa8e166af9e47cc9824fb5cf000ca48']
['attached_router_id']
['port_detail']
['enable_dhcp', 'True']
['ipv4_rangeId', '8234e0a77dfe4e7389ac1d964eef2da1']
['revision_number', '1']
['subnetpool_id']
['dns_publish_fixed_ip', 'False']
['allocation_pools', 'start', '1721601', 'end', '17216255254']
['host_routes', 'destination', '172161024', 'nexthop', '1721611']
['use_default_subnet_pool', 'False']
```

1.2.2 step2：word2vec

分词后，需要将所有词转化为向量。配置信息不存在上下文关联，当前采用bags of words 模型，统计所有词并建立词典。对于随机字符串的处理有以下几种方法：

方法一：不考虑随机字符串词

将和随机字符串有关的词删去，只保留有意义的词。

```
['id', 'project_id', 'tenant_id', 'name', 'subnet1', 'description',
'network_id', 'cidr', '172160016', 'gateway_ip', '1721601', 'gatewayPortId',
'gateway_port_detail', 'gatewaymacAddress', 'aabbccd28a5c', 'gatewayportid',
'attached_router_id','port_detail', 'enable_dhcp', 'True', 'ipv4_rangeId',
'revision_number', '1', 'subnetpool_id', 'dns_publish_fixed_ip', 'False',
'allocation_pools', 'start', '1721601', 'end', '17216255254', 'host_routes',
'destination', '172161024', 'nexthop', '1721611', 'use_default_subnet_pool',
'False']
```

建词典：

```
{'1': 1, '172160016': 2, '1721601': 3, '172161024': 4, '1721611': 5,
'17216255254': 6, 'False': 7, 'True': 8, 'aabbccd28a5c': 9, 'allocation_pools':
10, 'attached_router_id': 11, 'cidr': 12, 'description': 13, 'destination': 14,
'dns_publish_fixed_ip': 15, 'enable_dhcp': 16, 'end': 17, 'gatewayPortId': 18,
'gateway_ip': 19, 'gateway_port_detail': 20, 'gatewaymacAddress': 21,
'gatewayportid': 22, 'host_routes': 23, 'id': 24, 'ipv4_rangeId': 25, 'name':
26, 'network_id': 27, 'nexthop': 28, 'port_detail': 29, 'project_id': 30,
'revision_number': 31, 'start': 32, 'subnet1': 33, 'subnetpool_id': 34,
'tenant_id': 35, 'use_default_subnet_pool': 36}
```

转化向量：

```
[24, 30, 35, 26, 33, 13, 27, 12, 2, 19, 3, 18, 20, 21, 9, 22, 11, 29, 16, 8, 25,
31, 1, 34, 15, 7, 10, 32, 3, 17, 6, 23, 14, 4, 28, 5, 36, 7]
```

方法二：一起处理

随机字符串和其他词同等看待，并处理。

所有词：

```
['id', '8182a4d4ffff4eceb3f08d36e3d88001', 'project_id',
'3dda2801d6754688a63fdcda8d327f50', 'tenant_id',
'3dda2801d6754688a63fdcda8d327f50', 'name', 'subnet1', 'description',
'network_id', '9192a4d4ffff4eceb3f08d36e3d88001', 'cidr', '172160016',
'gateway_ip', '1721601', 'gatewayPortId', '0fa8e166af9e47cc9824fb5cf000ca48',
'gateway_port_detail', 'gatewaymacAddress', 'aabbccd28a5c', 'gatewayportid',
'0fa8e166af9e47cc9824fb5cf000ca48', 'attached_router_id', 'port_detail',
'enable_dhcp', 'True', 'ipv4_rangeId', '8234e0a77dfe4e7389ac1d964eef2da1',
'revision_number', '1', 'subnetpool_id', 'dns_publish_fixed_ip', 'False',
'allocation_pools', 'start', '1721601', 'end', '17216255254', 'host_routes',
'destination', '172161024', 'nexthop', '1721611', 'use_default_subnet_pool',
'False']
```

建词典：

```
{'0fa8e166af9e47cc9824fb5cf000ca48': 1, '1': 2, '172160016': 3, '1721601': 4,
'172161024': 5, '1721611': 6, '17216255254': 7,
'3dda2801d6754688a63fdcda8d327f50': 8, '8182a4d4ffff4eceb3f08d36e3d88001': 9,
'8234e0a77dfe4e7389ac1d964eef2da1': 10, '9192a4d4ffff4eceb3f08d36e3d88001': 11,
'False': 12, 'True': 13, 'aabbccd28a5c': 14, 'allocation_pools': 15,
'attached_router_id': 16, 'cidr': 17, 'description': 18, 'destination': 19,
'dns_publish_fixed_ip': 20, 'enable_dhcp': 21, 'end': 22, 'gatewayPortId': 23,
'gateway_ip': 24, 'gateway_port_detail': 25, 'gatewaymacAddress': 26,
'gatewayportid': 27, 'host_routes': 28, 'id': 29, 'ipv4_rangeId': 30, 'name':
31, 'network_id': 32, 'nexthop': 33, 'port_detail': 34, 'project_id': 35,
'revision_number': 36, 'start': 37, 'subnet1': 38, 'subnetpool_id': 39,
'tenant_id': 40, 'use_default_subnet_pool': 41}
```

转化向量：

```
[29, 9, 35, 8, 40, 8, 31, 38, 18, 32, 11, 17, 3, 24, 4, 23, 1, 25, 26, 14, 27,
1, 16, 34, 21, 13, 30, 10, 36, 2, 39, 20, 12, 15, 37, 4, 22, 7, 28, 19, 5, 33,
6, 41, 12]
```

方法三：分开处理

随机字符串和其他词分开处理，分别建立词典，在模型中embedding之后再拼接。

```
有语义词：
['id', 'project_id', 'tenant_id', 'name', 'subnet1', 'description',
'network_id', 'cidr', '172160016', 'gateway_ip', '1721601', 'gatewayPortId',
'gateway_port_detail', 'gatewaymacAddress', 'aabbccd28a5c', 'gatewayportid',
'attached_router_id','port_detail', 'enable_dhcp', 'True', 'ipV4_rangeId',
'revision_number', '1', 'subnetpool_id', 'dns_publish_fixed_ip', 'False',
'allocation_pools', 'start', '1721601', 'end', '17216255254', 'host_routes',
'destination', '172161024', 'nexthop', '1721611', 'use_default_subnet_pool',
'False']


随机字符串词：
['8182a4d4ffff4eceb3f08d36e3d88001', '3dda2801d6754688a63fdcda8d327f50',
'3dda2801d6754688a63fdcda8d327f50', '9192a4d4ffff4eceb3f08d36e3d88001',
'0fa8e166af9e47cc9824fb5cf000ca48', '0fa8e166af9e47cc9824fb5cf000ca48',
'8234e0a77dfe4e7389ac1d964eef2da1']
```

分别建词典：

```
有语义词：
{'1': 1, '172160016': 2, '1721601': 3, '172161024': 4, '1721611': 5,
'17216255254': 6, 'False': 7, 'True': 8, 'aabbccd28a5c': 9, 'allocation_pools':
10, 'attached_router_id': 11, 'cidr': 12, 'description': 13, 'destination': 14,
'dns_publish_fixed_ip': 15, 'enable_dhcp': 16, 'end': 17, 'gatewayPortId': 18,
'gateway_ip': 19, 'gateway_port_detail': 20, 'gatewaymacAddress': 21,
'gatewayportid': 22, 'host_routes': 23, 'id': 24, 'ipV4_rangeId': 25, 'name':
26, 'network_id': 27, 'nexthop': 28, 'port_detail': 29, 'project_id': 30,
'revision_number': 31, 'start': 32, 'subnet1': 33, 'subnetpool_id': 34,
'tenant_id': 35, 'use_default_subnet_pool': 36}


随机字符串词：
{'0fa8e166af9e47cc9824fb5cf000ca48': 1, '3dda2801d6754688a63fdcda8d327f50': 2,
'8182a4d4ffff4eceb3f08d36e3d88001': 3, '8234e0a77dfe4e7389ac1d964eef2da1': 4,
'9192a4d4ffff4eceb3f08d36e3d88001': 5}
```

转化向量：

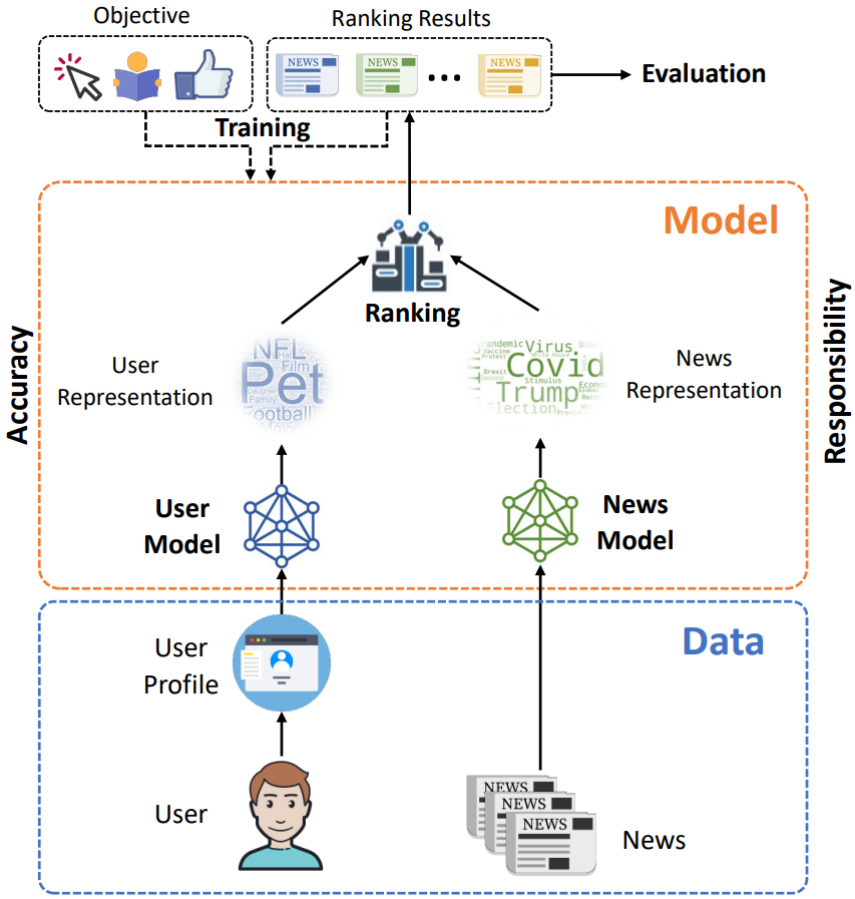
```
有语义词：
[24, 30, 35, 26, 33, 13, 27, 12, 2, 19, 3, 18, 20, 21, 9, 22, 11, 29, 16, 8, 25,
31, 1, 34, 15, 7, 10, 32, 3, 17, 6, 23, 14, 4, 28, 5, 36, 7]

随机字符串：
[3, 2, 2, 5, 1, 1, 4]
```

embedding之后拼接

1.3 参考场景

1.3.1 新闻推荐

新闻推荐场景中，输入为每个用户的历史点击新闻，用户的个人信息，以及用户可能要预览的候选新闻集合。模型通过用户的个人信息和历史点击新闻信息对用户建模，通过新闻的类型、标题、内容等具体信息对新闻建模。最终获得某用户对所有候选新闻的点击得分，来预测用户接下来最有可能点击的新闻。



1.3.2 日志检测

系统日志文件一般都是半结构化数据，通常包括时间戳、错误信息、IP地址、目标组件等。可以用于异常检测和故障定位。我们的配置数据和日志数据类似，在数据处理方面可以参考。

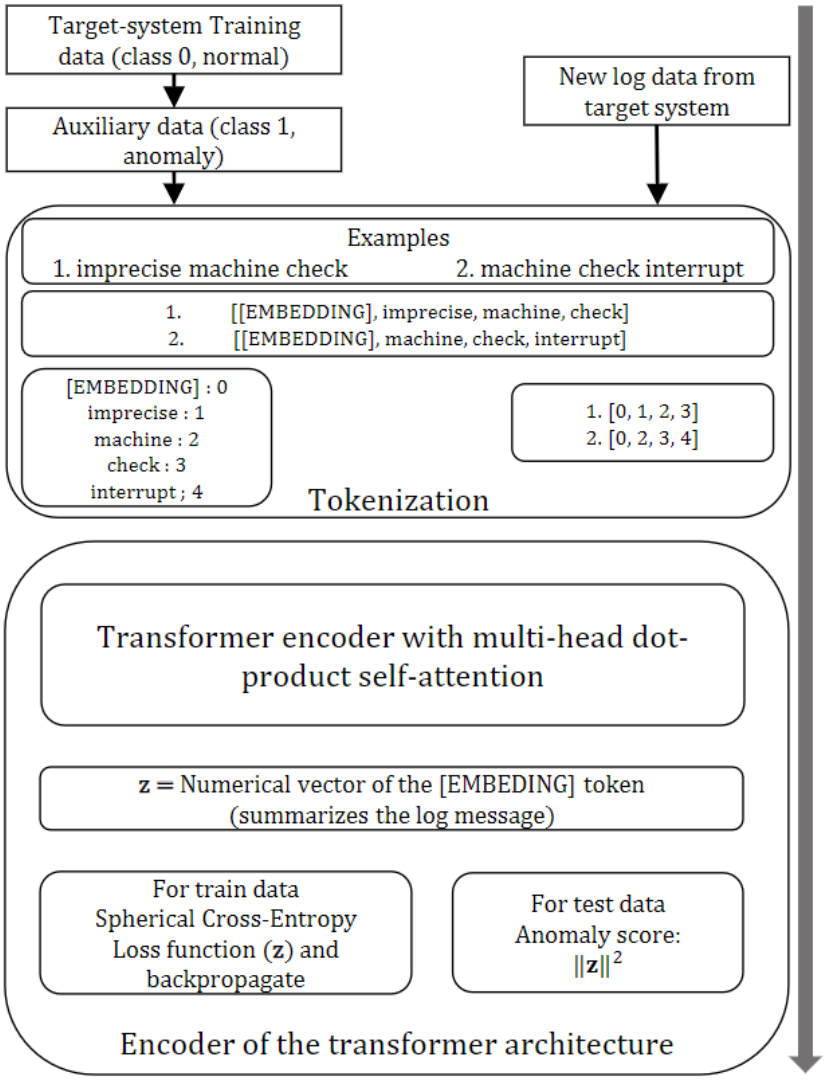
Log Parsing：解析原始日志，一般包括固定值，变量（IP、名称、id等）。

异常检测模型：根据时序日志预测下一个日志事件，并与实际情况对比，判断是否异常。

Related Works

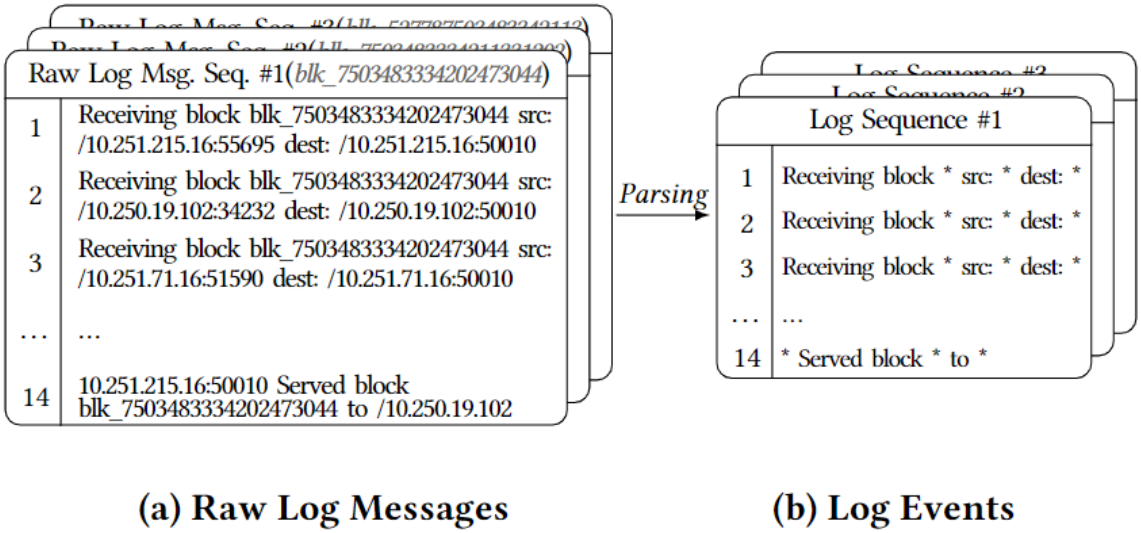
paper 1：[Self-Attentive Classification-Based Anomaly Detection in Unstructured Logs]

处理方法类似于我们的方法一，删除变量词，只保留语句，并使用NLTK分词，建立词典并转化为向量。

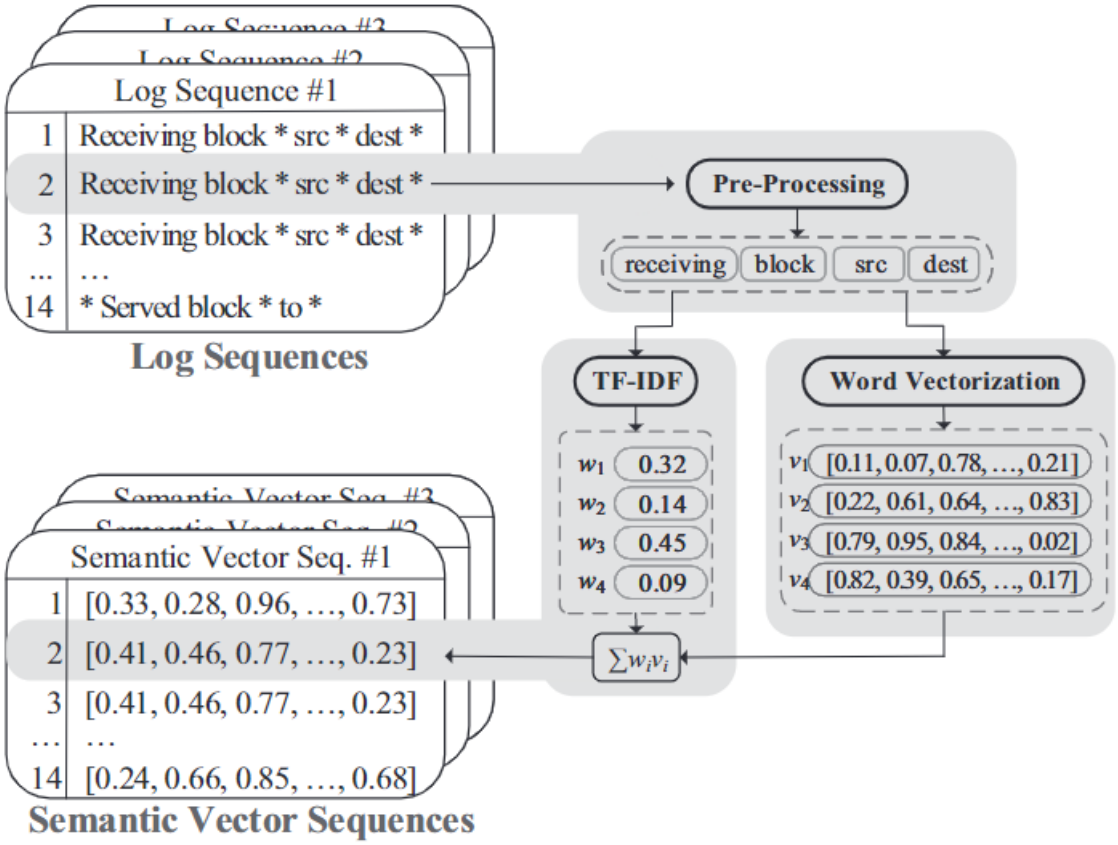


paper 2：**Robust Log-Based Anomaly Detection on Unstable Log Data**

处理日志文本时，同样不考虑参数变量，并将每条剩下的语义词作为整体，考虑上下文语义。使用预训练词向量（FastText在Common Crawl Corpus数据集上训练）



(a) Raw Log Messages **(b) Log Events**



paper 3：**SwissLog: Robust and Unified Deep Learning Based Log Anomaly Detection for Diverse Faults**

同样不考虑参数值。根据已知有效词典识别词

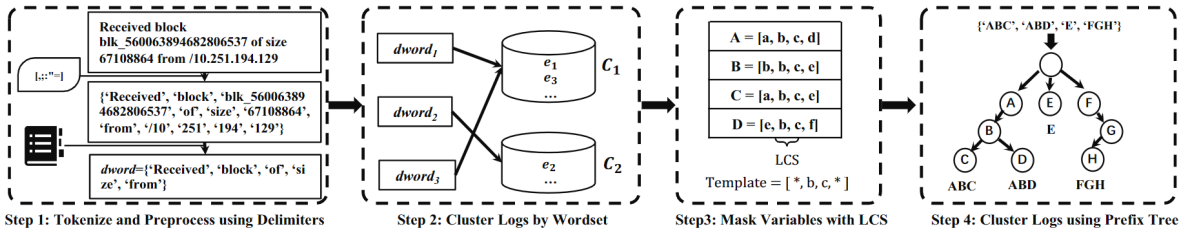


Fig. 3. The workflow of log parsing

提到的一种数据格式，类似于配置数据，使用于该模型中：

Case 1: “Expected quotacontroller.Sync to still be running but it is **blocked**. %v”,err

Case 2: “` {"metadata":{"ownerReferences":[{"apiVersion":"%s","kind":"%s","name":"%s","uid":"%s","controller":true,"**block**OwnerDeletion":true}],"uid":"%s"}}` ”,m.controllerKind.GroupVersion(), m.controllerKind.Kind, m.Controller.GetName(), m.Controller.GetUID(), rs.UID)

paper 4：[DeepLog | Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security]

每个日志事件解析为两部分：log key、参数值，考虑到参数，但是没有代入模型。

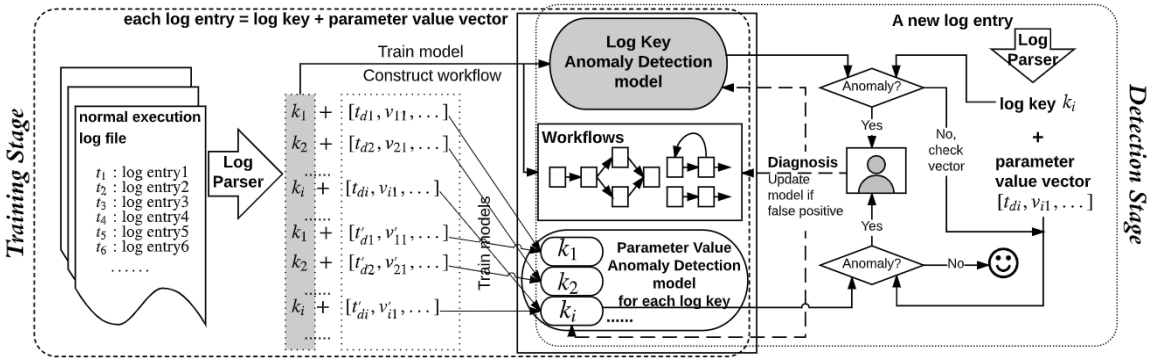
将日志条目视为遵循某些模式和语法规则的序列元素，日志建模为自然语言序列。

eg1：“Took 10 seconds to build instance”，其中，Took * seconds to build instance 为log key，参数值10单独抽取出来。

参数不加入模型的训练，而是作为预测之后的校验。

log message (log key underlined)	log key	parameter value vector
t_1 Deletion of file1 complete	k_1	$[t_1 - t_0, \text{file1Id}]$
t_2 Took 0.61 seconds to deallocate network ...	k_2	$[t_2 - t_1, 0.61]$
t_3 VM Stopped (Lifecycle Event)	k_3	$[t_3 - t_2]$
...	...	...

Table 1: Log entries from OpenStack VM deletion task.

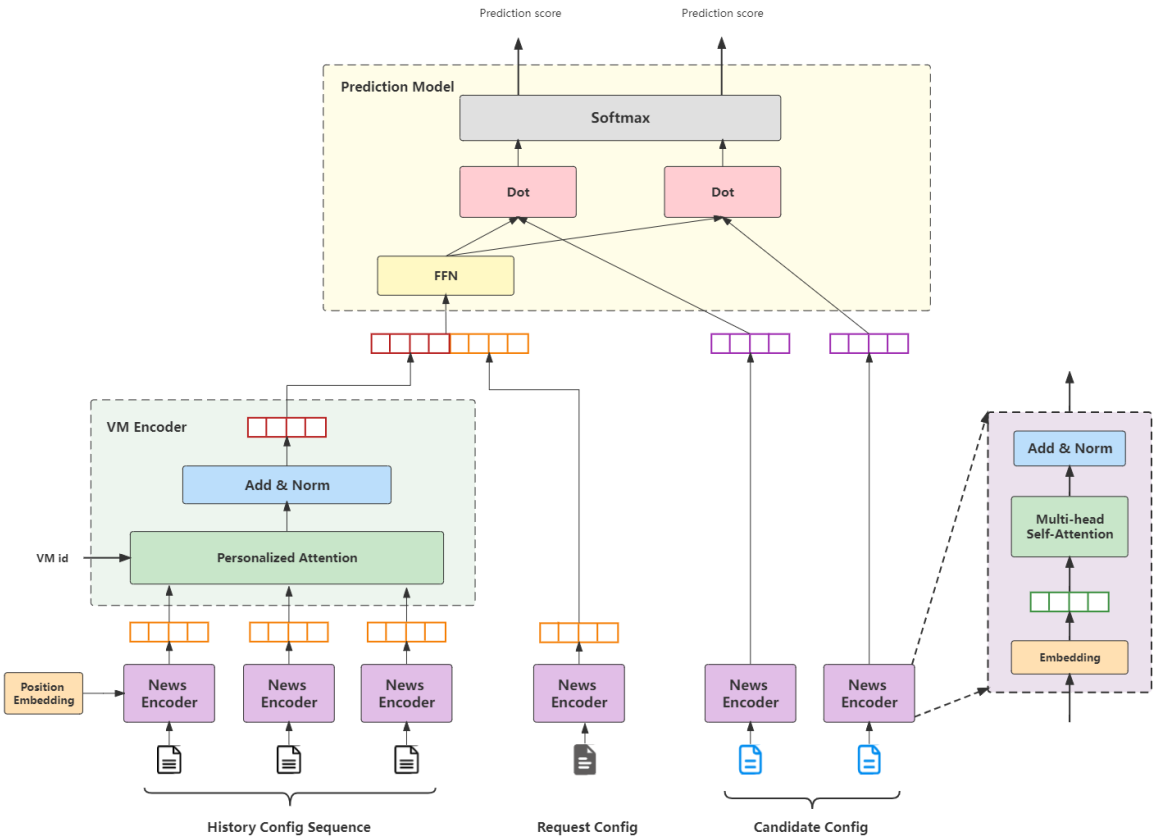


2. 模型

2.1 模型结构

2.1.1 模型v2

新版本的模型参考transformer框架中的部分思想设计，模型同样包含三个模块：*Config Encoder*，*VM Encoder*，*Prediction Model*，具体模型框架如下图：



「Input」

对每一组数据（每个VM），有：

1. 当前请求的配置Request Config
2. 该VM在数据库中即将要下发的候选配置集合Candidate Config
3. 该VM的id
4. 该VM的相似VM的历史下发配置序列History Config Sequence

「Config Encoder」

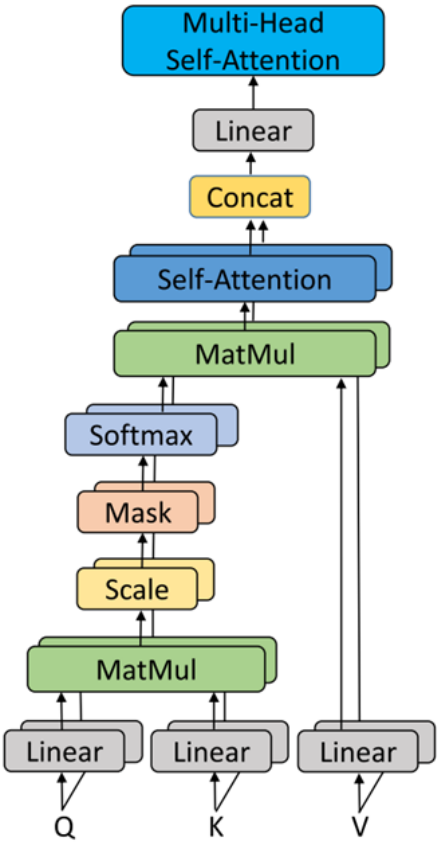
1. **Embedding**：词嵌入，将每个输入的配置的每个词映射为一个固定维度稠密向量。

   本模型中，如果所有配置相同，将(feature_size，word_size)维度的配置转化为(feature_size, embedding_size)。

   如果配置类型不同，将输入配置作为文本处理，所有类型的配置设定统一长度的词，即将维度(word_num, word_size)通过Embedding层处理为(word_num, embedding_size)。

2. **Multi-head Self-Attention**：

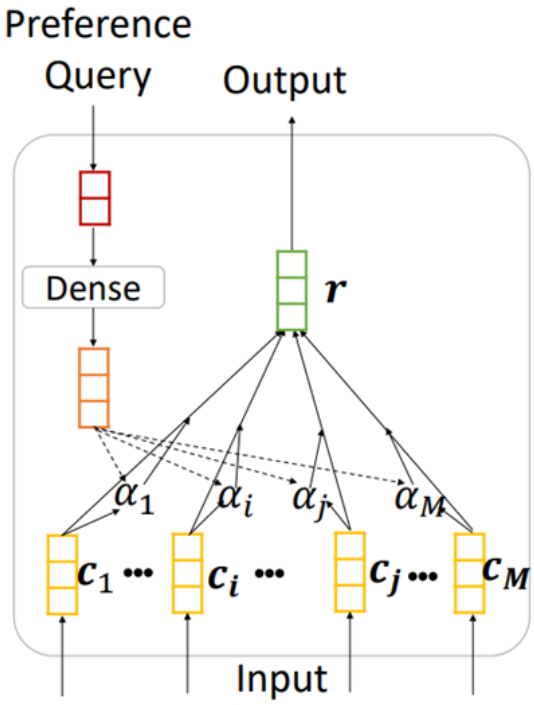
   不同的随机初始化映射矩阵W_q, W_k, W_v可以将输入向量映射到不同的子空间，这可以让模型从不同角度理解输入的序列。因此同时几个Attention的组合效果可能会优于单个Attention，这种同时计算多个Attention的方法就是Multi-Head Attention，或者多头注意力。Multi-head Attention的结构如下图：



Add&Norm：作用同上

「VM Encoder」

1. **Personalized Attention**：结构作用同模型v1。



2. **Add&Norm**：残差模块和归一化，可以有效的改善深层模型中梯度消失的问题，加速收敛，以帮助深层网络更好的训练。参考Transformer里的Add&Norm结构。

「Prediction Model」

1. **FFN：**包括两个线性层和激活函数

「Output」

对于每个VM的所有候选配置，都有一个范围(0,1)的预测分。根据预测分来确定预先下发哪些配置。

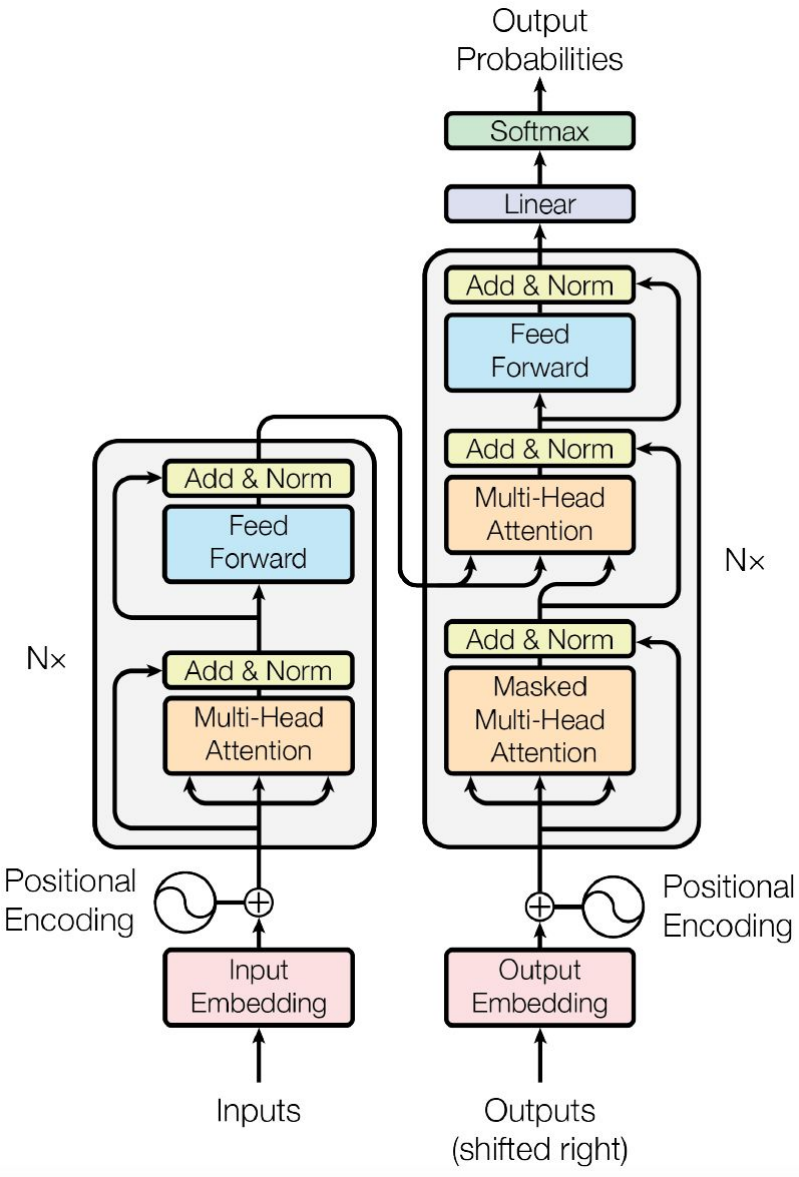
2.1.2 模型更新

模型的三个模块保持不变，具体的组成细节可以根据训练结果调整。

2.2 参考模型

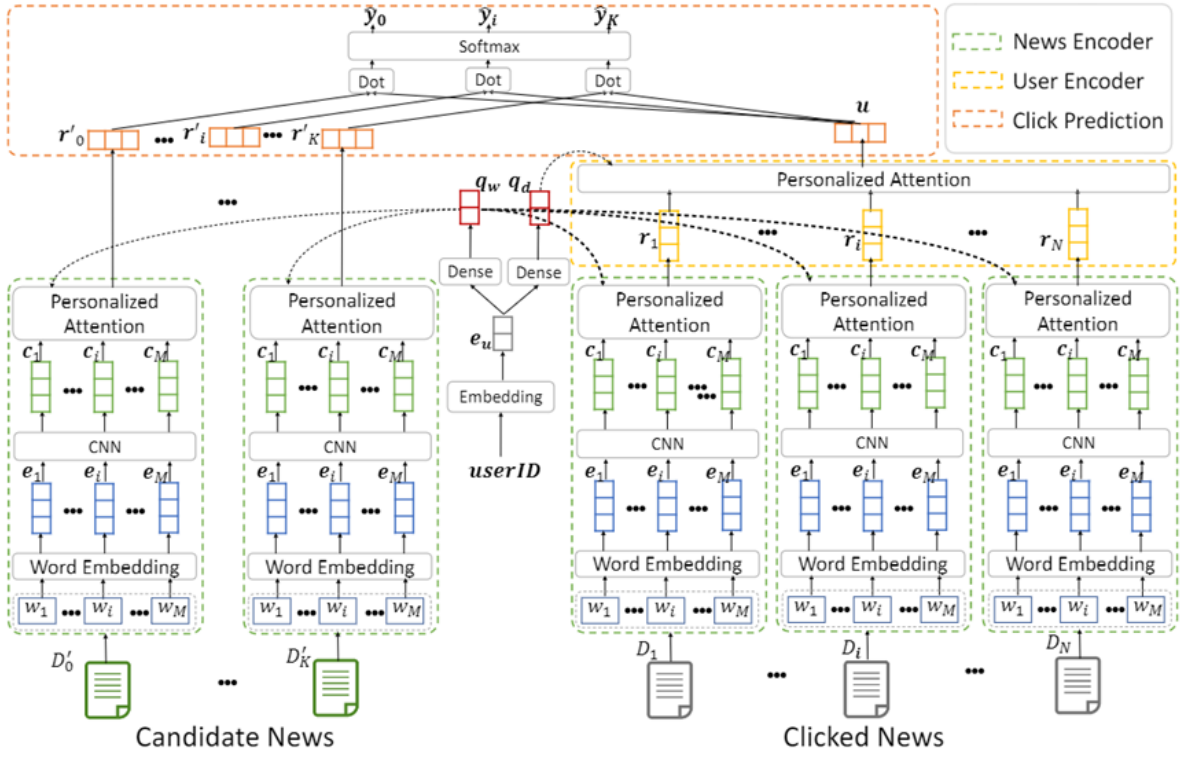
2.2.1 Transformer[1]

参考Transformer模型中的Multihead Attention、Add & Norm和Feed Forward部分。



2.2.2 NPA [2]

参考NPA中Personalized Attention 、Click Prediction部分。



Ref

1. Attention is All you Need (neurips.cc) ↩

2. NPA | Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining ↩