## 一、背景介绍

海量的网络配置请求需要通过控制平面下发到各南向计算节点。为了解决大规模云计算平台架构中控制平面为不同租户的 VPC 下发配置消息的问题，需要为控制平面 Alcor 设计并实现高并发、低时延、可拓展、低冗余的消息分发服务。

Alcor 的消息服务主要分为两个部分：（1）Avaiable Zone（AZ）/Region 级别的数据平面管理器（DataPlaneManager，DPM）负责计算正确网络配置并实现大规模消息下发通道；（2）Cluster 级别的 NetworkConfigrationManager（NCM）负责网络配置的按需下发。其中 DPM 已完成主体设计但还有较大的改进空间。我们开发的消息服务模块在 MQ 中为每个 VPC 创建一个 topic，在大规模云平台下 topic 数过多，且可能存在许多小的 vpc，导致资源浪费。因此需要在 NCM 中设计网络配置按需下发的算法，从而减少消息传输的冗余度。

在处理来自控制面的网络配置时，NetworkConfigManager(NCM)服务会立即下放必须的配置（Port、Gateway 等），同时保留非必须的配置（NeighborInfo、SecurityGroup 等）。当数据面需要对应配置时，向 NCM 请求下发所需配置。所以需要对留存配置设计按需下发逻辑和算法。

目前采用的方法是，在处理来自控制面的网络配置时，由 DataPlaneManager 通过 MQ\gRPC 将全量网络配置下发给数据面。然而当 VPC 中节点数量巨大的时候，一次性下放巨量配置可能会影响整个系统的性能 。当节点数量巨大的时候，过于频繁的请求会造成 NCM 服务负载过大及 IO 拥塞。同时，请求造成的时延也会降低系统性能。

为了解决上述问题， 需要设计 NetworkConfigManager(NCM)服务来分析节点当前真正需要的配置，设计配置处理逻辑，对已到达的消息进行语义合并，对不需要立即下发的网络配置进行批量延迟下发、分次下发，实现数据面网络配置的按需下发，。从而平滑系统负载曲线，降低通道峰值压力，优化网络配置下发延迟。

## 二、网络配置分析

在配置库里留存的的两种配置 Neighbor、Security Group。静态配置属性数据可以作为模型输入特征。

### 2.1、Neighbor

```
enum NeighborType {
    L2 = 0; // the default type
    L3 = 1;
}

message NeighborConfiguration {
    uint32 revision_number = 1;

    string request_id = 2;
    string id = 3;
    UpdateType update_type = 4; // DELTA (default) or FUL
    string vpc_id = 5;
    string name = 6;
    string mac_address = 7;
    string host_ip_address = 8;

    message FixedIp {
        NeighborType neighbor_type = 1;
        string subnet_id = 2;
        string ip_address = 3;
    }

    message AllowAddressPair {
        string ip_address = 1;
        string mac_address = 2;
    }

    repeated FixedIp fixed_ips = 9;
    repeated AllowAddressPair allow_address_pairs = 10;
}
```

## 2.2、Security Group

```
message SecurityGroupConfiguration {
    uint32 revision_number = 1;

    string request_id = 2;
    string id = 3;
    UpdateType update_type = 4; // DELTA (default) or FULI
    string vpc_id = 5;
    string name = 6;

    enum Direction {
        EGRESS = 0;
        INGRESS = 1;
    }

    message SecurityGroupRule {
        OperationType operation_type = 1;
        string security_group_id = 2;
        string id = 3;
        Direction direction = 4;
        EtherType ethertype = 5;
        Protocol protocol = 6;
        uint32 port_range_min = 7;
        uint32 port_range_max = 8;
        string remote_ip_prefix = 9;
        string remote_group_id = 10;
    }

    repeated SecurityGroupRule security_group_rules = 7;
}
```

配置的一些基本属性，如 Id、type、name 等可以作为模型的输入特征，而其他作用不太的描述性属性可以舍弃。

配置的属性大都是 string 以及 int 类型。

```
{
  "host_info": {
    "node_id" : "9192a4d4-ffff-4ece-b3f0-8d36e3d85002",
    "local_ip": "10.213.43.161",
    "mac_address": "90:17:ac:c1:34:64",
    "node_name": "node2",
    "server_port": 8080,
    "veth": "eth0",
    "host_dvr_mac": "74-70-FD-73-09-07"
  }
}
```

```
],
"id": "9192a4d4-ffff-4ece-b3f0-8d36e3d00108",
"mac_learning_enabled": true,
"name": "port108",
"network_id": "9192a4d4-ffff-4ece-b3f0-8d36e3d88038",
"network_ns": "string",
"port_security_enabled": true,
"project_id": "3dda2801-d675-4688-a63f-dcda8d327f50",
"qos_network_policy_id": "string",
"qos_policy_id": "string",
"revision_number": 0,
"security_groups": [
  "3dda2801-d675-4688-a63f-dcda8d111111"
],
```


## 三、相关调研

调研了机器学习在网络方面应用的论文，以及推荐系统用到的一些深度模型，在设计模型时主要参考以下两篇论文。

### 3.1、NPA：Neural News Recommendation with Personalized Attention

#### 3.1.1、基本介绍

这篇论文解决的问题是新闻推荐，根据用户的兴趣点推荐其最可能点击到的新闻。主要使用用户的历史浏览新闻序列来提取用户的兴趣点，从候选新闻中找出用户接下来可能点击到的新闻。主要包含以下两个模型。

News representation model：使用 CNN 来学习新闻标题的上下文表示。Attention 机制用来表示不同词对不同新闻的重要程度。

User representation model：根据用户的历史新闻点击来学习用户的表示。Attention 机制用来表示不同新闻对不同用户的重要程度。

Personalized Attention：相同的词和新闻对于不同用户的信息量可能不同，通过使用用户 ID 的嵌入作为词和新闻级注意力网络的查询向量来区分根据用户的喜好关注重要的词语和新闻。

#### 3.1.2、模型介绍

模型的具体结构如下图，主要包含三个部分：

News Encoder：根据新闻标题学习新闻表示，包括三个模块

① Word Embedding：新闻标题中每个词表示为一个低维 vector

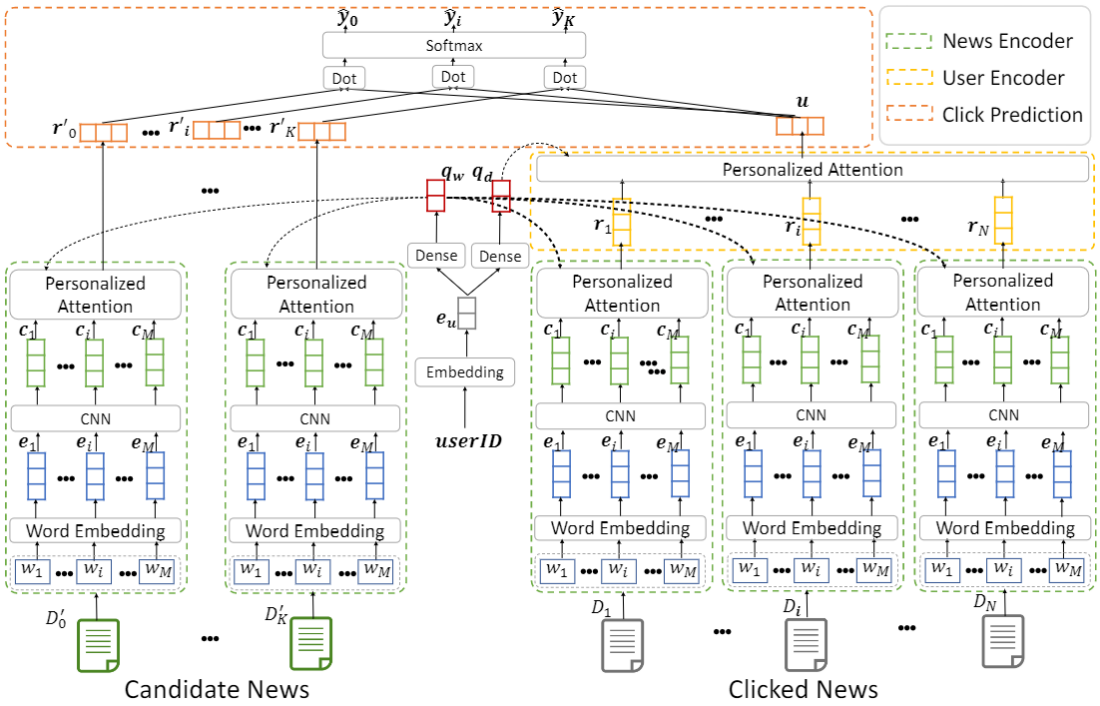
② CNN：捕捉新闻的上下文信息

③ Personalized Attention：词级 Attention，不同词对同一新闻的表示权重不同。而对于不同用户，同一词也有不同表示信息量，使用 user_id 作为 Attention 中的 query

User Encoder：根据用户历史点击新闻学习用户表示

① Personalized Attention：新闻级 Attention

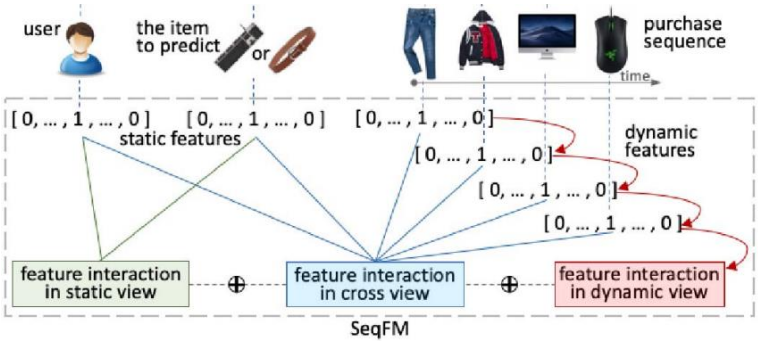
Click Prediction：预测一系列候选新闻的点击分数




## 3.2、 SeqFM：Sequence-Aware Factorization Machines for Temporal Predictive Analytics

### 3.2.1、基本介绍

这篇论文解决的问题是推荐系统，根据用户过去的购买顺序预测下一个可能购买的商品。用户历史购买商品的顺序对预测结果有影响，下一个可能购买商品可能和近期内购买商品相关度更高，而不是远期。为了实现时序感知的预测模型，采用 FFM 以及 Attention 机制。模型的逻辑如下图，特征会分为静态特征和动态特征，动态特征具有时序特征。



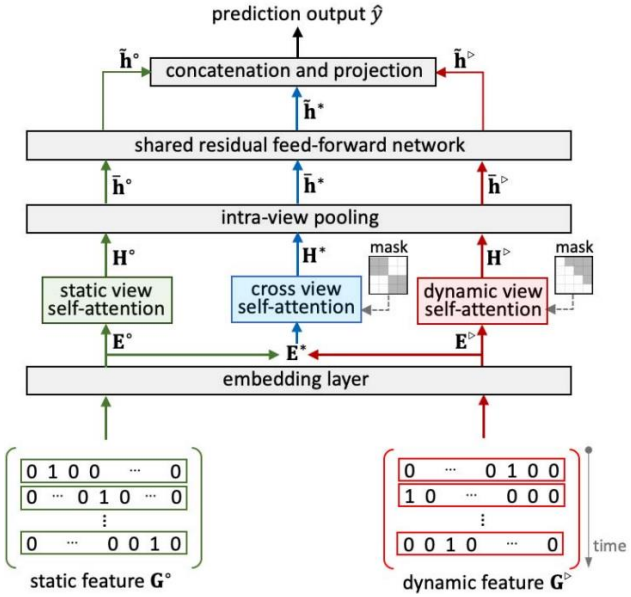

### 3.2.2、模型介绍

模型的具体细节如下图，主要模块有：

①Embedding layer：每个特征使用 one-hot 表示，通过全连接层降维为密集 vector。
FFM：特征分为静态特征（用户信息、候选商品信息）和动态特征（历史购买顺序等具有时序特征的数据，按照时序构建），cross view 增加特征之间的交互。

②Self-Attention：表示特征之间的顺序和语义关系。

SeqFM 在处理具有顺序属性的特征时，使用 Masked self-attention，每个动态特征仅与当前时间之前的特征交互。




## 四、基于机器学习的网络配置按需下发算法

### 4.1、算法设计

为了预测 VM 需要的配置并提前下发，可以根据该 VM 的历史下发配置来学习下发逻辑，由于 VM 历史配置信息量不足，所以采用相似 VM 的配置下发序列来表示，相似 VM 的下发配置也具有一定的相关性。通过 VM 的表示 vector 与候选配置的交互来预测每个候选配置的下发可能性。参考新闻推荐模型 NPA，模型中在学习配置表示和 VM 表示的时候使用到 Personalized Attention 结构。

### 4.2、语义表示（Embedding）

语义表示指的是将数值、文本等类型的数据转化为模型可以处理的向量，输入到模型中。静态配置数据的属性大都是 string 以及 int 类型，而一些 string 类型属性通常为无意义的字符序列，与实际句子不同。语义表示：

本模型使用到的 embedding 方式有以下几种：

#### 4.2.1、one-hot 编码

机器学习应用中，特征可能不是连续值，而是一些分类值。

例如，地区：[“Europe”,” US”，“Asia” ]，共三种值，普通编码使用 00，01，10 表示，而 one-hot 编码使用三位表示，分别为[1, 0, 0]，[0, 1, 0]，[0, 0, 1]，

One-hot 编码比较稀疏，信息量少，适用于特征取值较少的情况。One-hot 编码之后的向量需要降维为低维密集向量。

4.2.2、Word2id 词典

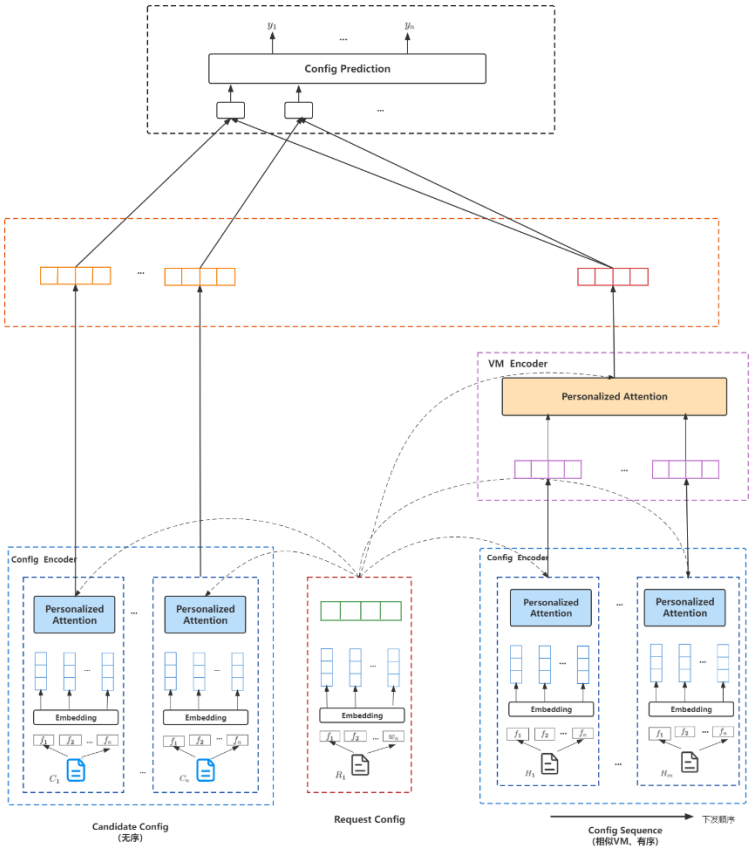
处理文本数据，转化为向量。

① 对于有具体含义的文本，先分词，所有词构成词典。
例如：'I am a boy.', 'How are you?', 'I am very lucky.'三个句子，取所有词构成 word2id 词典，可以按照出现频率排序，dict={i':0, 'am':1, 'a':2, 'boy':3, 'very':4, 'lucky':5, '.':6, 'how':7, 'are':8, 'you':9, '?':10}。三个句子分别表示为[0, 1, 2, 3, 6]，[7, 8, 9, 10, 0]，[0, 1, 4, 5, 6]。所有句子长度固定，不足补 0。

② 对于全是字符的特征值，一般为数字和字母组成，和一般文本不同，不具有具体含义。在将文本转化为向量时，无法分词，所以如“KLSDJsd323KENlmc”的特征值，每个字符作为一个词，同样建立 word2id 词典，处理方法同上。

**4.3、模型分析**

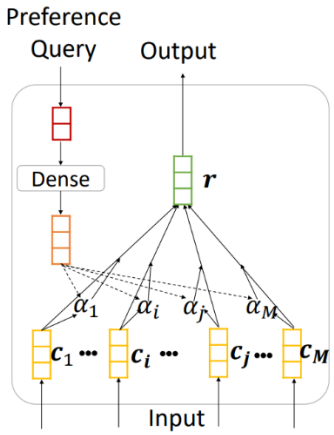
模型的具体结构如下图，主要包含三个模块：Config Encoder、VM Encoder 和 Config Prediction。




① Input：
VM 的候选配置库中的配置 Candidate Config
当前请求配置 Request Config

相似 VM 的历史下发配置 History Config Sequence。

②Personalized Attention：
Attention 结构如下，输入为多个特征，key 和 value 通过特征本身获得，query 使用其他数据，经过压缩后输入模型，输出为一个向量，用来表示 VM 或配置。分为特征级 Attention 和配置级 Attention。




③Config Encoder：根据配置的属性学习配置的表示。分为两个模块：Embedding 和 Personalized Attention。输入为每个配置的多个特征，输出为每个配置的向量表示。
Embedding：每个特征使用 one-hot 表示，维度大且稀疏，使用全连接层降维为密集 vector。每个配置特征表示为$C_i = [f_1, f_2, f_3, \dots, f_M]$，一共有 M 个特征。 经过 Embedding 之后，配置表示为$E_i = [e_1, e_2, e_3, \dots, e_M]$，$E_i \in R^{M \times N}$

特征级 Personalized Attention：
不同特征对同一配置表示的影响也不同，同一配置对不同 VM 表示的影响不同。
Attention 中的 key 以及 value 源于自身，而 query 使用 VM 的基本信息表示，可以使用 VM_id，或者该 VM 当前请求的配置。Config Encoder 中的 Personalized Attention 层的输入为 embedding 之后的向量，输出为一个向量表示该配置。

④VM Encoder：根据相似 VM 的历史下发配置学习 VM 的表示。使用 Personalized Attention 模型。输入为相似 VM 的历史下发配置的向量表示，输出为 VM 的向量表示。
配置级 Personalized Attention：
不同配置对同一用户的影响不同，同样，Attention 中的 key 和 value 源于自身，query 使用 VM 的基本属性特征来表示。VM Encoder 中的 Personalized Attention 层的输入为相似 VM 的历史下发配置序列的向量表示，输出为该 VM 的向量表示。

⑤Config Prediction：计算每个候选配置表示向量和 VM Encoder 向量内积，通过 softmax 激活函数归一化后，输出该候选配置的预测分数。每个候选配置有一个 0 到 1 之间的预测值，预测值越高，说明该候选配置是 VM 即将下发的配置的可能性更高。训练模型时，候选配置中有一部分为需要下发的，另一部分为不需要下发的。通过设定阈值来判断哪些配置需要下发。

### 4.4、Example

输入：

预测 VM1，相似 VM2

相似 VM2 的历史配置下发序列：$H_1$，$H_2$，$H_3$,$H_4$，$H_5$

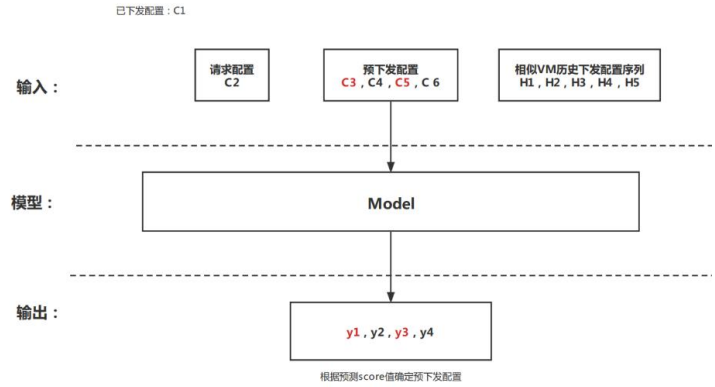
VM1 的已下发序列：$C_1$

请求配置：$C_2$

候选配置：$C_3$，$C_4$，$C_5$，$C_6$

输出：每个候选配置的预测值$y_{C_3}$，$y_{C_4}$，$y_{C_5}$，$y_{C_6}$

设定阈值 k=2，则选择 score 最高的两个配置作为即将下发的配置。




**4.5、数据介绍**

4.5.1、验证模型的数据（试用）

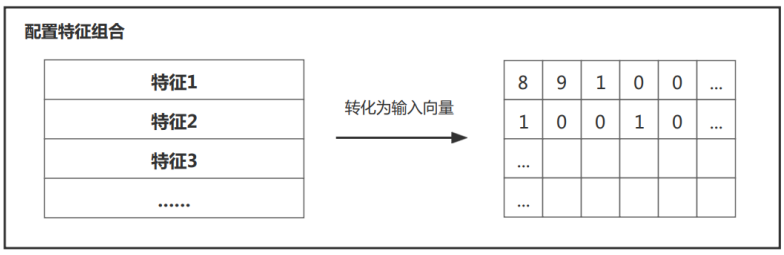
使用 google 的 Borg 集群数据，取 InstanceEvents table 中的数据。

根据时间戳来确定每组数据的时序，作为下发顺序。将同一组数据作为候选数据。

| | |
|---|---|
| Timestamp | 根据时间戳确定 event 的顺序 |
| Machine_id | 作为输入配置的特征 |
| Type | |
| Platform_id | |
| …… | |

4.5.2、配置数据

所有的网络配置都可以用特征组合的形式来表示，Interger 数据直接用数字表示 (IP、Port、type、Id 等)，string 类型数据可以分词、建立字典并用向量表示（MAC、info 等)。如下图，一行数字表示一个配置特征。多个特征表示一个配置。



**4.6、模型实现和测试**

(1) 模型部分已使用 Pytorch 实现

(2) 已用 google 部分数据集训练和测试模型，测试集的准确率在 60%左右

(3) 测试集结果分析：准确率不高的原因可能有：

① 模型不适用该数据集

② 该数据集可用性不高，数据特征与结果的关联不大，无法预测

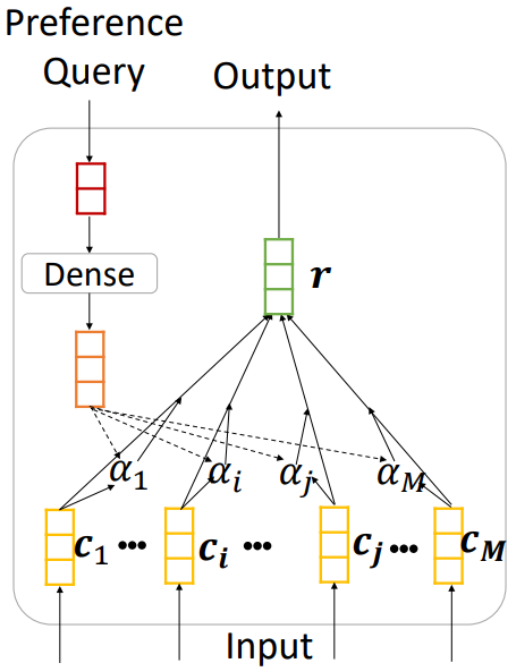
③ 数据量小

New Model：

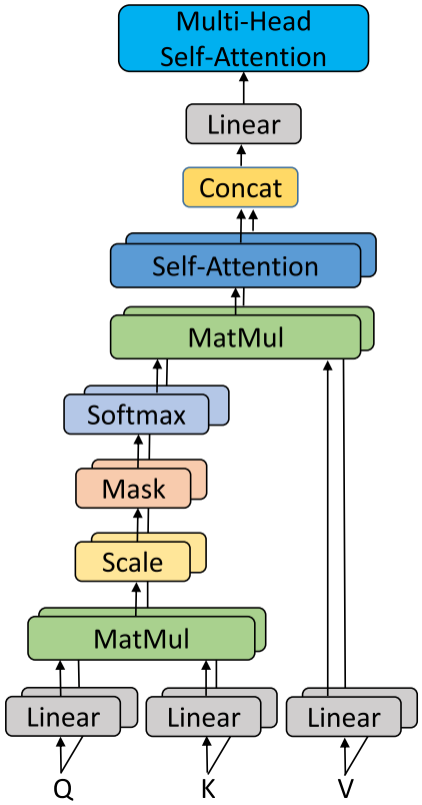
Position Embedding：补充位置信息

Add Norm：残差连接+归一化 LayerNorm

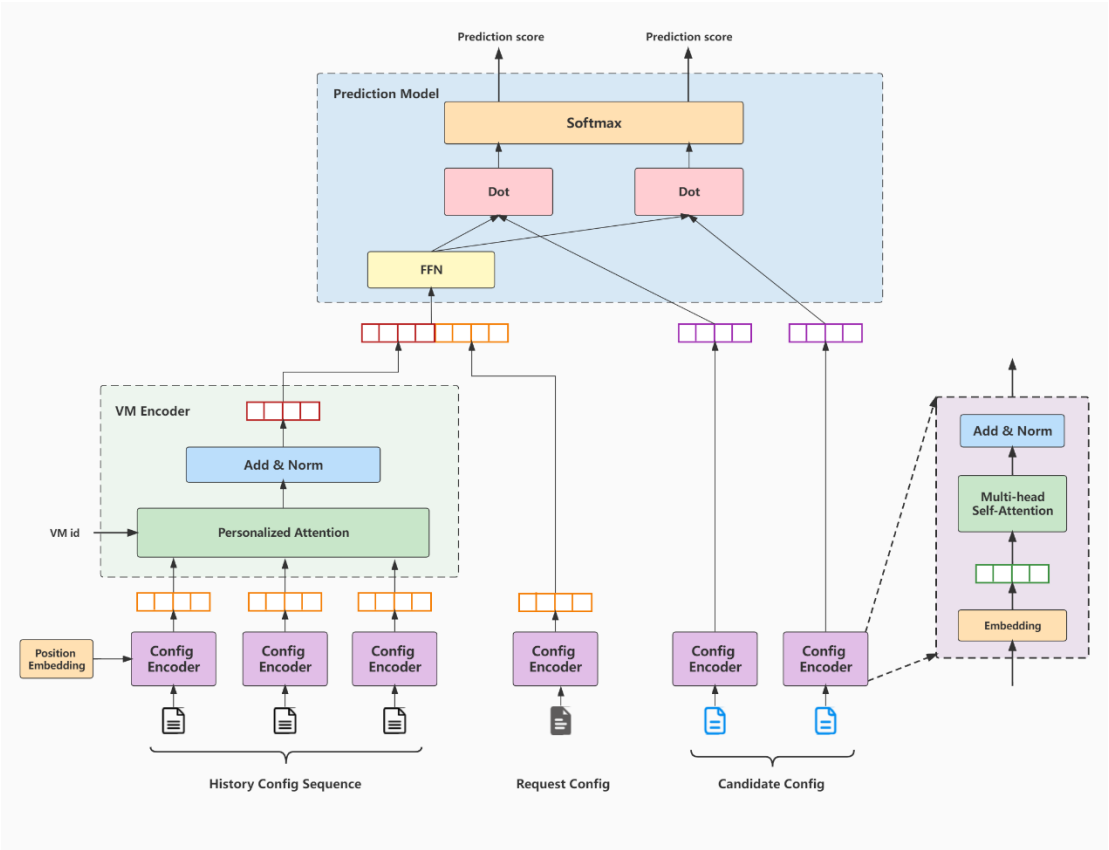
① Personalized Attention




② Multi-head Self-attention


Multi-Head
Self-Attention
Linear
Concat
Self-Attention
MatMul
Softmax
Mask
Scale
MatMul
Linear
Linear
Linear
Q
K
V

③ Main Model


Prediction score
Prediction score
Prediction Model
Softmax
Dot
Dot
FFN
VM Encoder
Add & Norm
VM id
Personalized Attention
Position Embedding
Config Encoder
Config Encoder
Config Encoder
Config Encoder
Config Encoder
Config Encoder
Add & Norm
Multi-head Self-Attention
Embedding
History Config Sequence
Request Config
Candidate Config