

**DGA域名检测报告**

目 录

[1背景与研究现状 4](#_Toc501959287)

[1.1 DGA简介 4](#_Toc501959288)

[1.2 DGA域名攻击方式 4](#_Toc501959289)

[1.3 研究现状 4](#_Toc501959290)

[1.3.1 Retrospective 4](#_Toc501959291)

[1.3.2 Real-time 5](#_Toc501959292)

[2 LSTM模型 5](#_Toc501959293)

[2.1 LSTM模型简介 5](#_Toc501959294)

[2.2 论文模型结构 6](#_Toc501959295)

[3. Attention机制 7](#_Toc501959296)

[3.1 Attention简介 7](#_Toc501959297)

[3.2 RNN Attention详解 7](#_Toc501959298)

[3.3 结果计算方式对比 9](#_Toc501959299)

[4 基于ATTENSION机制的DGA检测方法 10](#_Toc501959300)

[4.1 加入Attention后的模型结构 10](#_Toc501959301)

[4.2 线性Attention 11](#_Toc501959302)

[4.3 非线性Attention 12](#_Toc501959303)

[5 数据集构造 12](#_Toc501959304)

[5.1数据来源 12](#_Toc501959305)

[5.2 二分类数据集 13](#_Toc501959306)

[5.3 多分类数据集 14](#_Toc501959307)

[6 结果分析 15](#_Toc501959308)

[6.1 二分类结果分析 15](#_Toc501959309)

[6.1.1 整体分析 15](#_Toc501959310)

[6.1.2 分类分析 15](#_Toc501959311)

[6.1.3 leave-out分析 17](#_Toc501959312)

[6.2 多分类结果分析 17](#_Toc501959313)

[6.3 业务数据测试 18](#_Toc501959314)

[6.4 结论 18](#_Toc501959315)

[6.5 检测工具结构设计 18](#_Toc501959316)

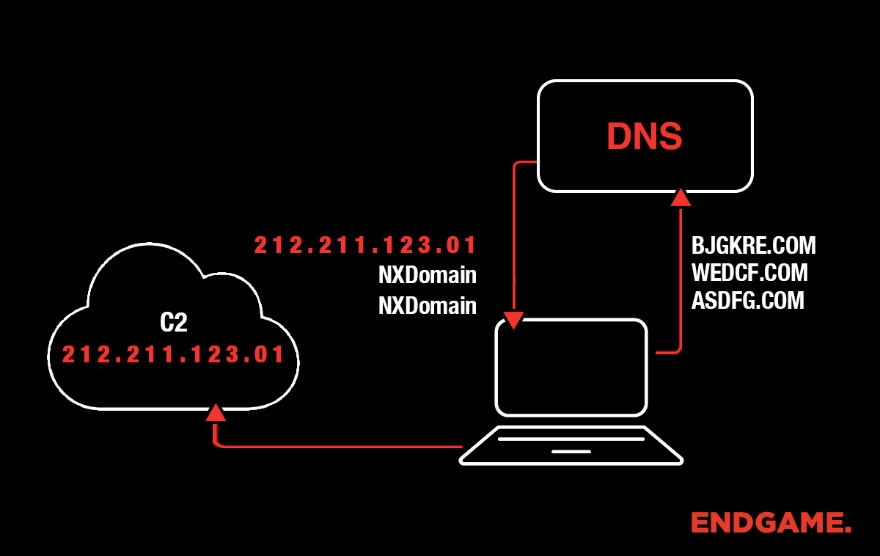
# 1背景与研究现状

## DGA简介

DGA是一种域名生产算法，它每天都可以根据种子生成大量符合域名要求的伪随机字符串。黑客们会从中选出一小部分进行恶意行为的绑定，假如想要检测出这些恶意域名，安全机构就需要将每天生成的大量伪随机域名全都加入到恶意库中，这显然是不现实的。黑客们常常会使用这种算法来规避黑名单列表的检测。

## DGA域名攻击方式

首先攻击者运行算法并随机选择少量的域（可能只有一个），然后攻击者将该域注册并指向其C2服务器。在受害者端恶意软件运行DGA并检查输出的域是否存在，如果检测为该域已注册，那么恶意软件将选择使用该域作为其命令和控制（C2）服务器。如果当前域检测为未注册，那么程序将继续检查其它域。



如图所示恶意软件会尝试连接三个域：asdfg.com，wedcf.com和bjgkre.com。前两个域未被注册，并从DNS服务器接收到NXDomain响应。第三个域已被注册，因此恶意软件会使用该域名来建立连接。

## 1.3 研究现状

### 1.3.1 Retrospective

Retrospective检测方式是在恶意行为发生之后，通过分析DNS查询信息、HTTP报文信息等建立分类器，对识别出的DGA域名再进一步分析，查询其来源。

这种方式成本太大，不适合装在客户机上，而且只能在恶意行为发生之后才能进行检测。

### 1.3.2 Real-time

Real-time 检测方式是对域名进行字符串分析，针对DGA域名的特点人工的设计一些特征，比如长度、元/辅音字母个数等。

这种方式虽然可以进行实时检测，但是当有种子更新时就需要重新设计特征，费时费力，效果也不太好。

# 2 LSTM模型

## 2.1 LSTM模型简介

LSTM模型是RNN的一种特殊类型，解决了一般RNN模型的长期依赖问题。LSTM 通过刻意的设计来避免长期依赖问题。记住长期的信息在实践中是 LSTM 的默认行为，而非需要付出很大代价才能获得的能力。

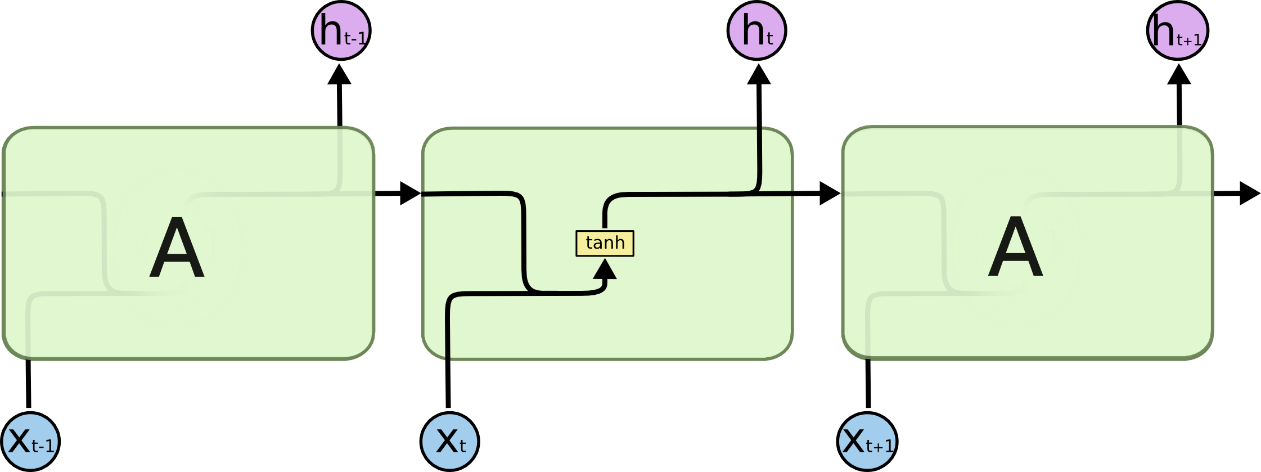


Figure 传统RNN

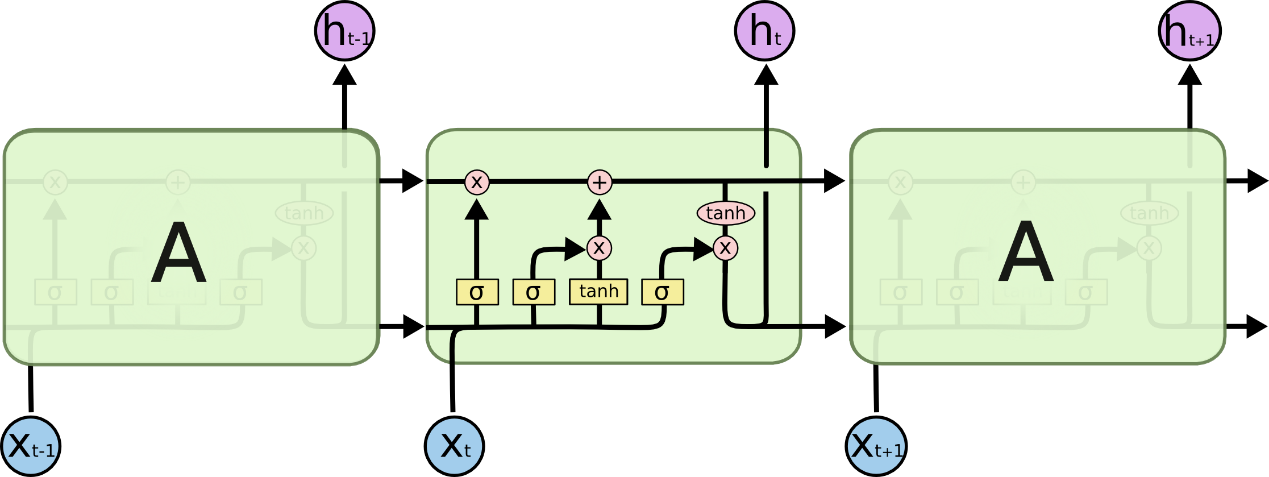


Figure LSTM

## 2.2 论文模型结构

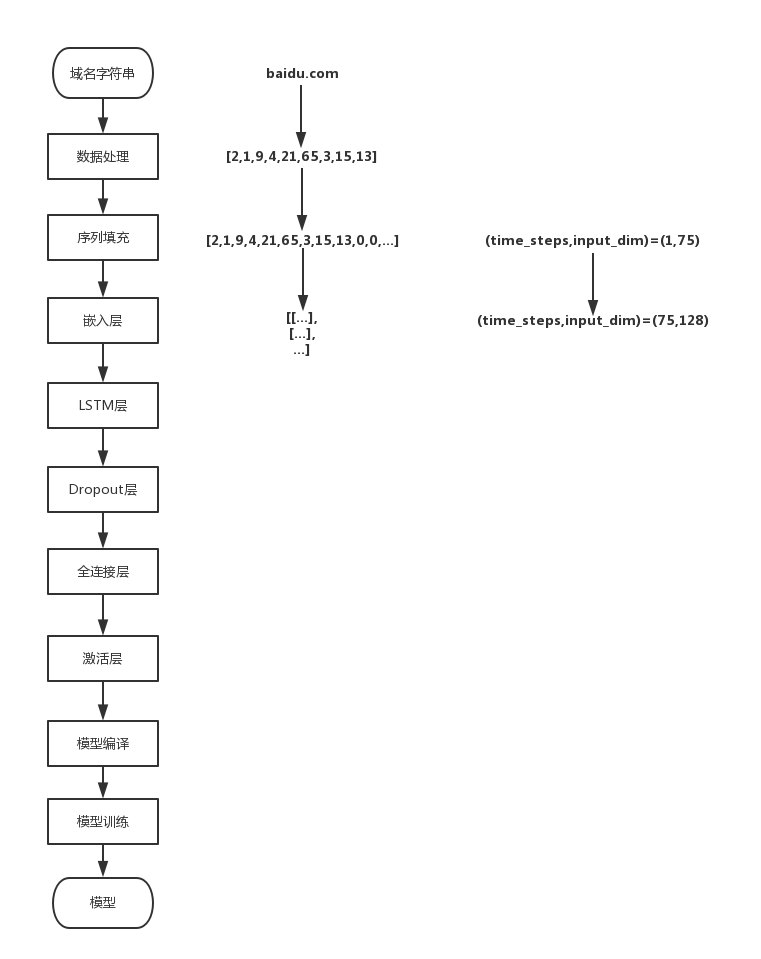


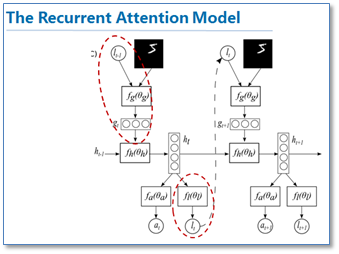
Figure 论文模型结构

# 3. Attention机制

## 3.1 Attention简介

Attention机制一开始被提出时是被应用于图像处理，代表性论文是《Recurrent Models of Visual Attention》。在这篇论文中，Attention机制的目的主要是用来解决图像处理中计算开销太大的问题。

该Attention机制会在训练的时候学习一个location()，他会告诉模型，对于输入的图像，我们应该关注哪些位置，去处理注意力部分像素，而不是图像的全部像素。

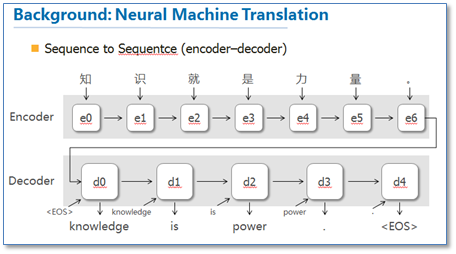


之后，Attention机制也被广泛的应用于编码器-解码器 (Encode-Decode) 结构的模型，因为在这种模型中，**输入序列不论长短都会被编码成一个固定长度的向量表示，而解码则受限于该固定长度的向量表示。**当输入序列比较长时，很多信息都会丢失，模型难以学到合理的向量表示，性能会变得很差。

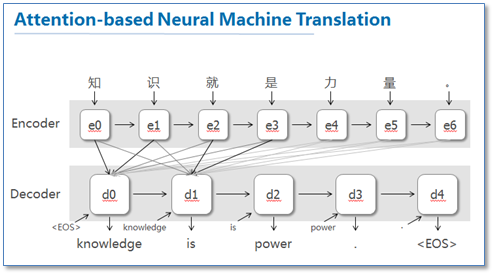
在这种结构中使用Attention机制，虽然不能从输入中获取更多的信息，但是对已经获取到的信息能够进行更有效的学习，在很多问题中取得了很好的效果。

## 3.2 RNN Attention详解

在传统的RNN中，输出是由之前的输出与状态决定的，模型结构如下图：



而在加入了attention机制之后，模型结构就变成了下图：



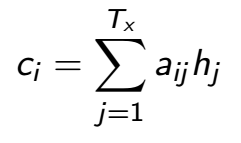
这里，每一个输出就不仅仅与之前输出和状态有关，还与所有的输入有关，计算函数如下：





是当前RNN的一个隐藏输出，从公式可以看到，无论还是都和有关，就是我们添加的attention层求出来的一个权重矩阵。

的计算和一般层里的参数是一样的，都是通过训练得到的，最终是一个constant vector，计算公式如下：



这里是根据输入求出的，可以理解为RNN模型对每个输入的解释，是校准函数。

## 3.3 结果计算方式对比

在论文中所使用的模型中，最终的输出只是由最后一个输入经过LSTM模型之后产生的中间结果计算得到，如图Figure 。而加入Attention机制之后，最终的输出则是由所有输入经过LSTM模型之后产生的中间结果共同决定的，如图Figure 5。

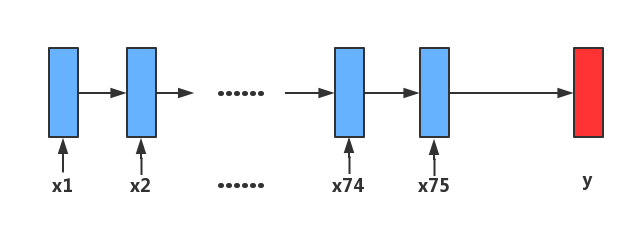


Figure 论文中结果的计算方式

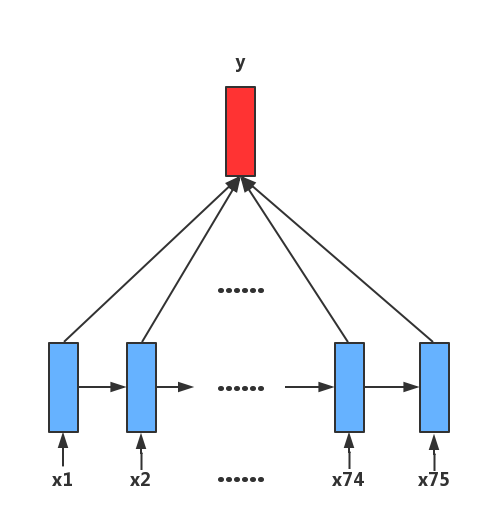
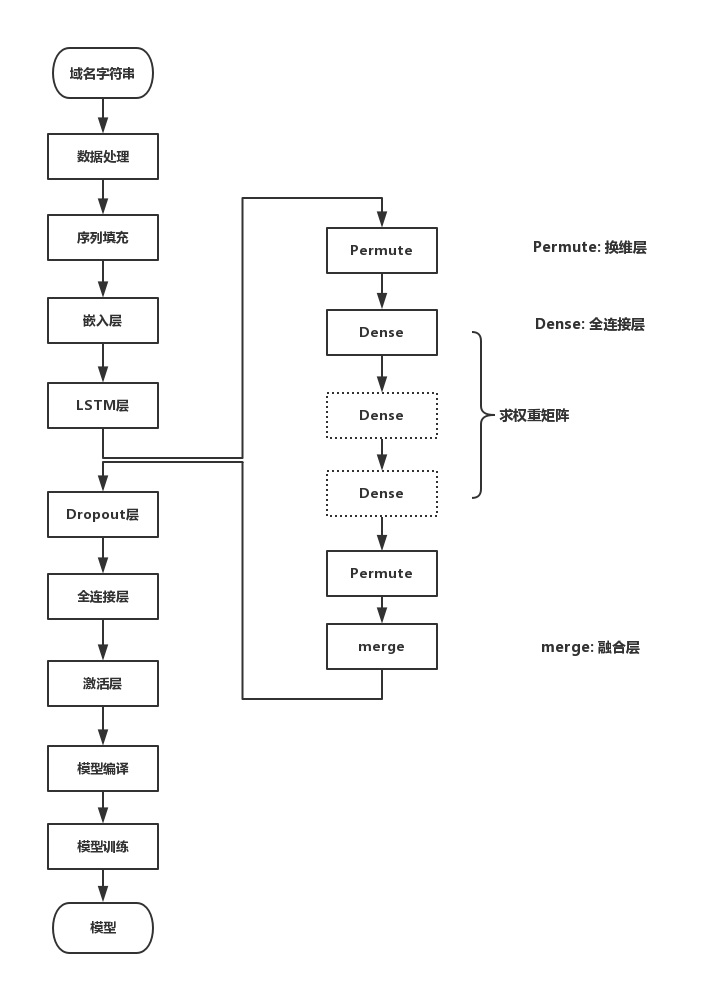


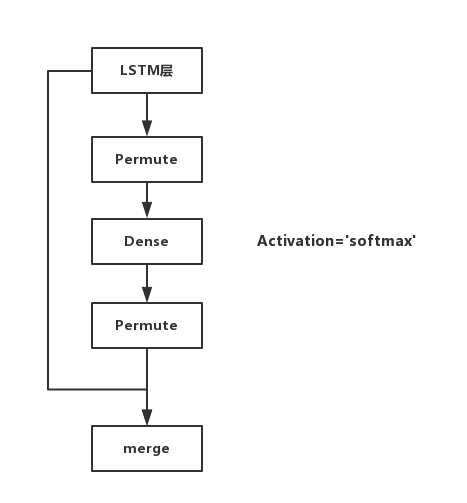
Figure 加入Attention后结果的计算方式

# 4 基于ATTENSION机制的DGA检测方法

## 4.1 加入Attention后的模型结构

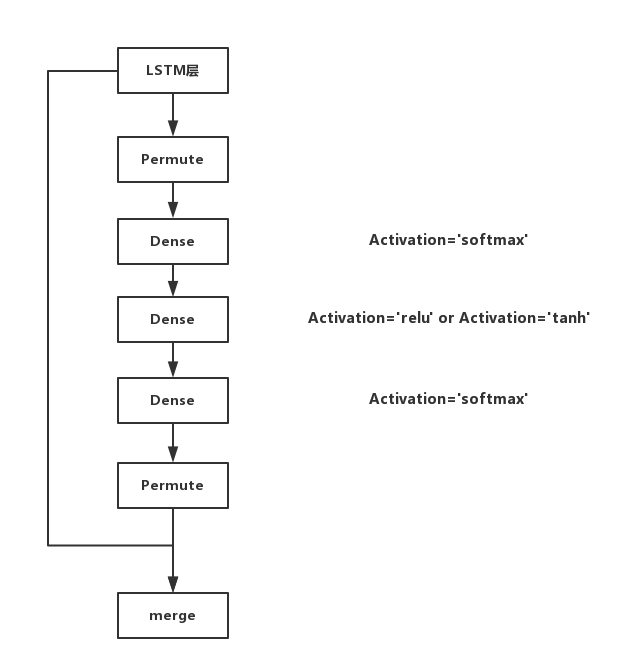


## 4.2 线性Attention



线性Attention在求权重矩阵时，只使用了一层全连接层，激活函数是softmax。

## 4.3 非线性Attention



非线性Attention在线性Attention的基础上，添加了一层中间层进行降维，激活函数可以是relu或者tanh，然后又加了一层全连接层还原其维度。

# 5 数据集构造

## 5.1数据来源

白样例：

Alexa全球域名点击排名

https://amazonaws-china.com/cn/alexa-top-sites/

黑样例：

<http://data.netlab.360.com/feeds/dga>

http://osint.bambenekconsulting.com/feeds

## 5.2 二分类数据集

白样例：1000078个

黑样例：60类，共1687806个

训练集：白样例+50类黑样例，每一类抽80%组成训练集

测试集：白样例+50类黑样例，除了训练集以外的部分

leave-out 测试集：未参与训练的10类黑样本，测试模型对新出现的DGA家族域名的检测能力。

Table 二分类域名类型

|  |  |  |
| --- | --- | --- |
| class | num | leave-out |
| alexa | 1000078 |  |
| bamital | 864 |  |
| banjori | 439344 |  |
| bedep | 368 | ✔ |
| beebone | 210 | ✔ |
| blackhole | 12 |  |
| chinad | 7536 |  |
| conficker | 3000 |  |
| corebot | 520 | ✔ |
| cryptolocker | 12000 |  |
| cryptowall | 94 | ✔ |
| dircrypt | 570 | ✔ |
| dyre | 15996 |  |
| emotet | 215422 |  |
| fobber | 600 | ✔ |
| fobber\_v1 | 298 |  |
| fobber\_v2 | 300 |  |
| gameover | 68000 |  |
| geodo | 1152 |  |
| gozi | 36 |  |
| gspy | 99 |  |
| hesperbot | 192 | ✔ |
| kraken | 8988 |  |
| locky | 8228 |  |
| matsnu | 1022 | ✔ |
| murofet | 54434 |  |
| necurs | 49152 |  |
| nymaim | 12779 |  |
| p2p | 2000 |  |
| padcrypt | 1584 |  |
| pandabanker | 63 |  |
| pizd | 1020 |  |
| post | 132000 |  |
| proslikefan | 1560 |  |
| pushdo | 2040 |  |
| pykspa | 19064 |  |
| pykspa\_v1 | 44703 |  |
| pykspa\_v2\_fake | 4800 |  |
| pykspa\_v2\_real | 200 |  |
| qadars | 3534 |  |
| qakbot | 40000 |  |
| ramdo | 2000 |  |
| ramnit | 106906 |  |
| ranbyus | 20022 |  |
| rovnix | 179998 |  |
| shifu | 2554 |  |
| shiotob | 12521 |  |
| simda | 31267 |  |
| sisron | 84 |  |
| sphinx | 1536 |  |
| suppobox | 6426 |  |
| symmi | 4320 | ✔ |
| tempedreve | 255 | ✔ |
| tinba | 100927 |  |
| tofsee | 20 |  |
| unknowndropper | 60 |  |
| unknownjs | 360 |  |
| vawtrak | 3150 |  |
| vidro | 200 |  |
| virut | 60420 |  |
| volatile | 996 |  |

## 5.3 多分类数据集

数据集：对于所有样例数大于10000的类别中随机抽取10000个样例，总共20个类。

训练集：从数据集中的每一类样例随机抽取80%

测试集：数据集中除训练集以外的样例

# 6 结果分析

## 6.1 二分类结果分析

### 6.1.1 整体分析

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 实验还原 | Attention(linear) | Attention(relu) | Attention(tanh) |
| Precison | 98.72 | 99.15 | 99.04 | **99.23** |
| Recall | 98.82 | 99.16 | **99.24** | 99.16 |
| F1 | 98.77 | 99.15 | 99.14 | **99.20** |
| 误报率 | 2.15 | 1.42 | 1.61 | **1.29** |
| 漏报率 | 1.18 | 0.84 | **0.76** | 0.84 |

### 6.1.2 分类分析

分类分析里列出的参数是每一类的 ，support是每一类参与测试的样例个数。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Domain Type | 实验还原 | Attention(linear) | Attention(relu) | Attention(tanh) | support |
| Alexa | 0.9785 | 0.9858 | 0.9839 | 0.9871 | 200016 |
| bamital | 1 | 1 | 1 | 1 | 173 |
| banjori | 1 | 0.9999 | 1 | 0.9999 | 87869 |
| blackhole | 1 | 1 | 1 | 1 | 3 |
| chinad | 1 | 0.998 | 1 | 0.9993 | 1508 |
| conficker | 0.8317 | 0.8267 | 0.8167 | 0.835 | 600 |
| cryptolocker | 0.9933 | 0.9908 | 0.995 | 0.9933 | 2400 |
| dyre | 1 | 1 | 1 | 1 | 3200 |
| emotet | 0.9998 | 0.9997 | 0.9998 | 0.9998 | 43085 |
| fobber\_v1 | 1 | 1 | 1 | 1 | 60 |
| fobber\_v2 | 0.9833 | 0.9833 | 1 | 0.9833 | 60 |
| gameover | 1 | 1 | 1 | 1 | 13600 |
| geodo | 1 | 1 | 1 | 1 | 231 |
| gozi | 0.125 | 0.125 | 0.125 | 0.125 | 8 |
| gspy | 1 | 1 | 1 | 1 | 20 |
| kraken | 0.9872 | 0.9861 | 0.9894 | 0.9894 | 1798 |
| locky | 0.9727 | 0.9684 | 0.9714 | 0.9702 | 1646 |
| murofet | 0.9983 | 0.9983 | 0.9981 | 0.9983 | 10887 |
| necurs | 0.9827 | 0.9846 | 0.9834 | 0.9834 | 9831 |
| nymaim | 0.9386 | 0.9327 | 0.939 | 0.9382 | 2556 |
| p2p | 1 | 1 | 1 | 1 | 400 |
| padcrypt | 0.9685 | 0.9748 | 0.9432 | 0.959 | 317 |
| pandabanker | 1 | 1 | 1 | 1 | 13 |
| pizd | 0.3186 | 0.848 | 0.8971 | 0.8676 | 204 |
| post | 1 | 1 | 1 | 1 | 26400 |
| proslikefan | 0.9038 | 0.8782 | 0.875 | 0.9006 | 312 |
| pushdo | 0.8578 | 0.8775 | 0.8971 | 0.8824 | 408 |
| pykspa | 0.9725 | 0.9696 | 0.9753 | 0.9746 | 3813 |
| pykspa\_v1 | 0.9897 | 0.9924 | 0.9945 | 0.9945 | 8941 |
| pykspa\_v2\_fake | 0.9417 | 0.9469 | 0.9594 | 0.9635 | 960 |
| pykspa\_v2\_real | 0.85 | 0.95 | 1 | 0.925 | 40 |
| qadars | 0.9915 | 0.9844 | 0.9915 | 0.9887 | 707 |
| qakbot | 0.9925 | 0.9915 | 0.9941 | 0.994 | 8000 |
| ramdo | 0.9975 | 0.9925 | 0.9975 | 1 | 400 |
| ramnit | 0.9783 | 0.9794 | 0.9773 | 0.9797 | 21382 |
| ranbyus | 0.9965 | 0.9953 | 0.9968 | 0.9973 | 4005 |
| rovnix | 0.9999 | 0.9999 | 0.9998 | 0.9999 | 36000 |
| shifu | 0.9511 | 0.9667 | 0.9667 | 0.955 | 511 |
| shiotob | 0.9908 | 0.9856 | 0.9896 | 0.9908 | 2505 |
| simda | 0.9824 | 0.9882 | 0.9936 | 0.9901 | 6254 |
| sisron | 1 | 1 | 1 | 1 | 17 |
| sphinx | 0.9968 | 0.9935 | 1 | 0.9968 | 308 |
| suppobox | 0.3639 | 0.8569 | 0.902 | 0.9051 | 1286 |
| tinba | 0.9941 | 0.9959 | 0.9956 | 0.9957 | 20186 |
| tofsee | 0.75 | 0.25 | 0.5 | 1 | 4 |
| unknowndropper | 1 | 1 | 1 | 1 | 12 |
| unknownjs | 0.9167 | 0.9167 | 0.9306 | 0.9444 | 72 |
| vawtrak | 0.6286 | 0.7968 | 0.8127 | 0.827 | 630 |
| vidro | 0.975 | 0.975 | 1 | 0.975 | 40 |
| virut | 0.9149 | 0.9323 | 0.9401 | 0.9158 | 12084 |
| volatile | 0.935 | 0.985 | 0.935 | 0.975 | 200 |
| **average** | 0.9206 | 0.9373 | 0.9464 | **0.9549** |  |

### 6.1.3 leave-out分析

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Domain Type | 实验还原 | Attention(linear) | Attention(relu) | Attention(tanh) | support |
| bedep | 0.9946 | 0.9973 | 0.9946 | 0.9973 | 368 |
| beebone | 0 | 0 | 0 | 0 | 210 |
| corebot | 0.9981 | 0.9885 | 0.9942 | 0.9365 | 520 |
| cryptowall | 0.234 | 0.2128 | 0.2234 | 0.2447 | 94 |
| dircrypt | 0.9807 | 0.9842 | 0.9789 | 0.9789 | 570 |
| fobber | 0.9867 | 0.99 | 0.9933 | 0.9917 | 600 |
| hesperbot | 0.9271 | 0.9271 | 0.9531 | 0.9219 | 192 |
| matsnu | 0.001 | 0 | 0 | 0 | 1022 |
| **symmi** | 0.5331 | 0.4067 | 0.4919 | 0.3907 | 4320 |
| tempedreve | 0.8745 | 0.8824 | 0.8627 | 0.8824 | 255 |
| **average** | **0.6530** | 0.6389 | 0.64492 | 0.6344 |  |

beebone是一类特殊的DGA域名，它不是通过随机种子与时间生成的，而是有固定的格式，每部分都是从一个词库中随机抽样产生的，比如ns1.backdates13.biz 和ns1.backdates0.biz。matsnu和beebone类似。

这种类型的域名在训练集中是从未出现过的，所以无法识别出来。在论文中也证实了，如果这两种类型的域名不参与训练的话，用其他模型也是无法识别出来的。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Domain Type | 实验还原 | Attention(linear) | Attention(relu) | Attention(tanh) | support |
| bedep | 0.9946 | 0.9973 | 0.9946 | 0.9973 | 368 |
| corebot | 0.9981 | 0.9885 | 0.9942 | 0.9365 | 520 |
| cryptowall | 0.234 | 0.2128 | 0.2234 | 0.2447 | 94 |
| dircrypt | 0.9807 | 0.9842 | 0.9789 | 0.9789 | 570 |
| fobber | 0.9867 | 0.99 | 0.9933 | 0.9917 | 600 |
| hesperbot | 0.9271 | 0.9271 | 0.9531 | 0.9219 | 192 |
| **symmi** | 0.5331 | 0.4067 | 0.4919 | 0.3907 | 4320 |
| tempedreve | 0.8745 | 0.8824 | 0.8627 | 0.8824 | 255 |
| **average** | **0.8161** | 0.7986 | 0.8115 | 0.7930 |  |

## 6.2 多分类结果分析

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 实验还原 | Attentio(linear) | Attention(relu) | Attention(tanh) |
| Micro Precison | 0.7697 | **0.8317** | 0.8291 | 0.8295 |
| Micro Recall | 0.7697 | **0.8317** | 0.8291 | 0.8295 |
| **Micro F1** | 0.7697 | **0.8317** | 0.8291 | 0.8295 |
| Macro Precison | 07659 | **0.8350** | 0.8312 | 0.8320 |
| Macro Recall | 0.7697 | **0.8317** | 0.8291 | 0.8295 |
| **Macro F1** | 0.7678 | **0.8333** | 0.8301 | 0.8307 |

## 6.3 业务数据测试

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Feed name | url | support | DGA num | P |
| 360 DGA Feed | http://data.netlab.360.com/feeds/dga/dga.txt | 1094605 | 1087315 | 0.9933 |
| Bambenek DGA Domains | http://osint.bambenekconsulting.com/feeds/dga-feed.txt | 874269 | 868217 | 0.9931 |
| C2 Domains | http://osint.bambenekconsulting.com/feeds/c2-dommasterlist.txt | 498 | 421 | 0.8454 |
| ZeuS Tracker Domains | https://zeustracker.abuse.ch/blocklist.php?download=baddomains | 47 | 1 | 0.0213 |

这里是从业务系统中提取出的相关域名的测试结果，support是测试样例个数，DGA num是最终被模型识别为DGA域名的样例个数，P是这两者的比值。

## 6.4 结论

分析实验结果，在二分类问题中，使用非线性，中间层激活函数为tanh的Attention模型能够得到更好的效果；在多分类问题中，使用线性Attention模型能够取得更好的效果。

对于部署系统而言，使LSTM模型进行域名检测的最大优点在于不需要人工的去设计特征，即使有新的DGA家族产生，也只需要收集一些样本加入到训练集中重新训练一个模型即可，节省了大量的人力成本。

同时，LSTM模型是基于Python的Keras库实现的，部署起来也比较简单。而且在训练集比较完善的情况下，LSTM模型能够得到非常好的识别效果。

用论文中的话来说，该系统是目前DGA域名检测问题上效果最优秀，最便于部署的系统。

## 6.5 检测工具结构设计

检测工具主要分为两部分——模型训练和模型预测。根据我们的实验结果，在训练二分类时，使用LSTM+非线性Attention(tanh)模型，在训练多分类时，使用LSTM+线性Attention模型。

