人工智能学习笔记八——基于深度学习的钓鱼（垃圾）短信检测

本文将用两种深度学习模型，BiLstm 和 transformer 进行对照实验，分别对于垃圾短信进行检测。

经过搜索，网上目前主流的垃圾短信数据集都已经是十几年前的数据集了，随着 5G 时代的来临，短信的形式变化的太快了，现在收到的短信一般以验证码以及手机流量充值为主，人们已不在用短信进行聊天。通过搜索，找到登录 - 数据集市 (shujujishi.com)了这个数据集，包含了 100 万条短信数据，但是没有标注，因而笔者自行抽选了其中的部分短信进行标注，最终得到了 1 万条正例短信，和 1 万条负例短信。

小杨,北京如何凉皮餐饮店等5家企业查看/回复了你投递的简历，点击http://i.58.com/2F9_查看详情，主动联
【华为云】尊敬的daba123123123：华为云中国站计划于2019/08/1100:30-04:30对平台进行升级。升级过程事宜做好安排。给您带来的不便，敬请谅解。更多详细信息请查看官网公告。
【8868交易平台】您购买的世界OL（世界online）的游戏币交易成功。咨询热线：0757-63523999。
【话费查询】尊敬的客户，您好！您2018年06月的话费总额为:17.74元。话费查询请拨1008611。中国移动
曹先生先生你申请的岗位已通过，适合我司180-280/天，请及时联系Q：976554791正式录取
【有钱花】尊敬的用户，您的订单己冻结，与本人信息不符，请登入联系客服！
【58同城】杨伟伟,兰州恒巨商贸有限公司等31家企业查看/回复了你投递的简历，点击http://i.58.com/2F9_查看详情，
【UGGame】您好，您在UGgame投诉的【英雄联盟】订单，已经帮您处理，我们帮您重置了该游戏的试玩机会，快去体验吧！
【58到家】您的保洁师（李丽丽，中间号17071222322）将于8月9日08:30，前往滨江豪园6栋1单元202
【神马出行】感谢您乘坐神马专车，本次行程消费0.00元，充送优惠后相当于0.00元，稍后可对行程管家进行评价，祝您生活愉快。
【鼎诺光华】亲爱的戴涛，感谢您对鼎诺光华的支持与信赖。今天是您的生日，全体鼎诺人祝您生日快乐，阖家幸福！
【58到家】您预约的日常保洁服务（8月9日08:30，滨江豪园6栋1单元202）将由保洁师李丽丽为您服务。为保护您的
【丰巢】请凭取件码‘44204892’至英国宫四服六期南门西侧丰巢智能柜取韵达快递的包裹。
【德邦快递】尊敬的客户您好！您的订单WK1**5866，因客户重复下单，已被快递员刘福东，15298512319取消，如有异议您可联系快递员
【云道活动】尊贵的嘉宾您好！您已成功报名云智中国?百度智能云ABC赋能企业智能化转型（武汉站），请提前出示您的报名信息方便签到入场，谢
【小遛共享】等你好久，怎么还不回来，临时锁车也是持续计费的，家里有矿吗？记得及时还车呀。
【GBLive】您的验证码是：968752。请不要把验证码泄露给其他人。
【WPS云服务】验证码651295，您正在尝试修改WPS登录密码，请妥善保管账户信息。
【小红书】您的验证码是:117645，3分钟内有效。请勿向他人泄漏。如非本人操作，可忽略本消息。
【交易猫】您已成功支付订单并租到账号！游戏账号：951647326，密码：tgb6。您也可登录交易猫-租号专区-个人中心进行查看。
【昌呈广告】您的验证码是：769728[DGClub]
【西山居】验证码：881748（有效期5分钟），您正在进行注册金山通行证操作，官方绝不会索取此验证码，请勿告知他人。
【他趣】验证码:996727，请勿泄露。
【凯浩贸易】您的验证码为：2459（5分钟有效），为保证账户安全，请勿向任何人提供此验证码。
【街兔电单车】验证码：（507976），您正在使用短信验证码登录功能，5分钟内有效。转发可能导致帐号被盗，请勿泄露给他人
【就过科技】您的验证码是831877。如非本人操作，请忽略本短信
【58同城】验证码750524。请勿向任何人提供验证码，以防账号被盗。
【口袋科技】您正在找回密码，您的验证码是5993
您的验证码是：4784。请不要把验证码泄露给其他人。
【SoulAPP】您的验证码：4226。验证码有效时间为5分钟，请勿向他人泄露。您正在登录Soul，如非本人操作，可忽略本消息。
【马蜂窝】您正在使用手机修改密码！短信验证码是741549，请在30分钟内使用
【嘿嘿连载】感谢您的注册，本次注册验证码为6965，请于3分钟内正确输入，切勿泄露他人。
【他趣】验证码:246219，请勿泄露。
【他趣】验证码:253382，请勿泄露。
【哔哩哔哩】821526为你的找回登录密码的验证码，请在5分钟内完成身份认证。验证码请勿泄漏，如非本人操作，请忽略或回复T退订。
【趣头条】您的验证码是2843。如非本人操作，请忽略本短信
【金数据】您的验证码是926238。如非本人操作，请忽略本短信
【Apple】您的AppleID验证码为：076940

图 1 （部分正例短信图）

【中国人寿】尊敬的客户：生活中充满风险，家人的健康和安全是我们每个人最大的心愿。中国人寿愿与您相伴，撑起家庭保护伞，无惧风险，为爱担当！保险产品详情请点击
【滴滴外卖】来就送最高(6元无门槛)立减券，滴滴食堂(0元配送)更省钱！戳https://z.didi.cn/2pMMB退订TD
【悦美APP】来吧，“鼻”个美。精致眼鼻低至2999，领券最高立减3500元~详戳http://t.cn/AiYufay2退订回N
【V24】Hi，今日立秋！云南蒙自石榴券后价11.8元，青桔9.9元/斤，西梅29.9元/斤，铁棍山药8.8元/400g，茭白笋5.8元/300g。以上应季好物另有满35减5元优惠券！更多新鲜好物尽在V24生鲜时选。退订回T
【滴滴外卖】来就送最高(6元无门槛)立减券，滴滴食堂(0元配送)更省钱！戳https://z.didi.cn/2pMMB退订TD
享券等你来拿，点击链接直达会场报名https://m.tb.cn/Q.1k0VH-回复T退订
【adidas】三叶草推出重磅鞋款OZWEEGO，携易烊千玺、王嘉尔等再掀复古风潮！http://weq.me/JG5Sk。回QX退订
【赤狐CRM】快速获取2万企业名录，覆盖100行业，300城市，支持批量下载导出。登录m.foxsaas.com/dmtg获取，回复T退订
【腾讯科技】经典沙盒玩法，正版乐高授权，来BonBon游戏下载《乐高无限》手游，释放想象力！点我下载url.cn/5Q248BF回T退订
【亨氏官方旗舰店】番茄酱2元，铁锌钙米粉325g包邮价21.9元，更多优惠上苏宁抢t.suning.cn/9aHywCD回T退订
【联想惠商】重磅福利：惠商8日独家首发联想K66鼠标，首发价仅19.9元/件！https://hub2.cn/wosu,退订回TD
【财团App】转发收益翻倍活动即刻开启，单价升为0.4元！更有现金红包大奖等你拿！点击下载财团：https://dwz.cn/wKBZ7SdJ?客服微信：caituankefu2
【阿里云】阿里云IoT88物联网狂欢节，全场商品下单返利5%，总有一款是你想要的，立即进入会场http://a.aliyun.com/f1.5sVV6回td退订
【亿滋官方旗舰店】奥利奥促销来袭专区2件7折，闲趣优冠限时拼购，更多优惠上苏宁抢t.suning.cn/6aHTUgc回T退订
【V24】Hi，今日立秋！云南蒙自石榴券后价11.8元，青桔9.9元/斤，西梅29.9元/斤，铁棍山药8.8元/400g，茭白笋5.8元/300g。以上应季好物另有满35减5元优惠券！更多新鲜好物尽在V24生鲜时选。退订回T
【您的Gap奥莱】超值T恤，季末底价！全场T恤￥39起！另有超多精选夏季商品减价基础上再享2件8折！进店详询。回TD退订
【联想惠商】重磅福利：惠商8日独家首发联想K66鼠标，首发价仅19.9元/件！https://hub2.cn/wosu,退订回TD
【易果生鲜】亲爱的会员，全网99减30、188减50两档优惠券在您账号等候多时了！新人大礼包可别错过哟！回TD退订
【美赞臣旗舰店】上苏宁领160元券买铂睿2段只需197/罐，下单满额再选赠品；t.suning.cn/9aGDNeC回T退订
【每日优鲜】送你满79减20元红包！水果上新啦→超甜巨大的硒砂大西瓜低至2元/斤！每月17日，超级生鲜节t.cn/AiT58SNh退订N
【食行生鲜】惊喜！<2元立秋滋补券>已送至您账户，今明有效，1.99元吃哈密瓜，戳s.34580.cn/jcs退订回TD
【秦丝科技】老板，您已经注册秦丝进销存7天啦，学会库存查询补货功能，让您告别糊涂账！dh3t.cn/or05Yq回T退订
【秦丝科技】多个门店账号手机、电脑、平板数据实时同步！下载进销存app请点击：https://www.qinsilk.com，手机端也能操作！咨询培训请联系顾问老师:17154832150(微信同号),回T退订
【云汉芯城】您有一张500元京东卡即将过期，请及时查看！http://tinyurl.com/yyl9xdtg回复TD退订
【惠氏官方旗舰店】约惠狂夏为爱臻选！辣妈奶爸必囤好物—惠氏启赋、铂臻，更多福利立即t.suning.cn/4aHzxYL回T退订
【滴滴外卖】来就送最高(6元无门槛)立减券，滴滴食堂(0元配送)更省钱！戳https://z.didi.cn/2pMMB退订TD
【海信空调官方旗舰店】海信空调限时特惠→领券下单享优惠，来苏宁易购爆款空调0元抢t.suning.cn/7aAQiKm回T退订
【苏宁】2元神券已到账，拼购日狂欢24小时，8枚雪媚娘蛋黄酥券后仅7.9元，抢t.suning.cn/9ahExlf回T退订
【亨氏官方旗舰店】番茄酱2元，铁锌钙米粉325g包邮价21.9元，更多优惠上苏宁抢t.suning.cn/9aHywCD回T退订
【精灵云测】留不住用户怎么办？快来测测服务器的承载能力！只需一键，即可免费测试，测试PC登录：www.fairytest.com，感谢您的支持
【1药网】亲爱的小先生/女士，姿美堂胶原蛋白粉，限时128，买2赠1，规律服用，效果更佳速抢>t.cn/AiY7Xnxy退订回复TD
【腾讯科技】经典沙盒玩法，正版乐高授权，来BonBon游戏下载《乐高无限》手游，释放想象力！点我下载url.cn/5Q248BF回T退订
【adidas】neo潮流衣橱周，速来get明星同款搭配！即日起至8/15，购买指定neo正价产品2件及以上，还可享该等产品7折！戳http://weq.me/O05Ry。回QX退订
【货车帮】嗨老铁，每月要参与分20万吗？300元代金券需要吗？“攒油兑大奖”的活动正在如火如荼的进行，好多师傅都后悔参加的晚了。邀你链接直达，一般人我不告诉他：t.56qq.cn/qN8EPXbboc回复TD退订
【途虎养车网】您关注的保养特惠啦！100元通用神券拼团购！全场保养低至149元！戳>app.tuhu.cn/vCKRD回T退订
【两鲜】七夕送啥才够奇葩？30元红包奉上，邀您99元享4斤+大榴莲！有10%几率买到8斤的哦，快>dh3t.cn/FHtykGTD退订
【拉勾网】马斯克，这些大厂正在高薪招聘网络工程师，点击查看http://t.cn/E2rb1rI回T退订
【每日优鲜】@GG：今天立秋~你的80元食鲜红包已到账，红提限时7.9元1斤，拼团三黄鸡8.8元1只t.cn/AiT5M1EF退订N

图 2 （部分负例短信图）

正例的数据放在了 good.txt 的文件内，负例的数据放在了 bad.txt 内。

然后是数据的筛选，观察图 1 和图 2 可以发现，短信里面含有大量的 url 链

接，以及电话号码，QQ 号码等许多信息，这边将把这些无用的信息全部删除，只保留中文字符。但有些中文字符也是没有用的，例如“它”，“的”，“但是”等连接词或者指示代词，无论是正例还是负例，都能见到这些词，我们称这些词为停用词。这边引用了 stopwords/scu_stopwords.txt at master · goto456/stopwords (github.com)这篇文章的停用词列表，放在了 stopword.txt 内，代码如下：

```python
import re
import jieba
fr2 = open("bad.txt","r",encoding="utf-8")
databad = fr2.readlines()
fr2.close()
fr3 = open("good.txt","r",encoding="utf-8")
datagood = fr3.readlines()
fr3.close()
fr4 = open("stopword.txt","r",encoding="utf-8")
stopword = fr4.readlines()
fr4.close()
stopword = [i.strip() for i in stopword]
datagood_select = []
databad_select = []
for i in datagood:
    out = re.findall(r'[\u4e00-\u9fa5]', i) #把非中文字符都替换掉
    outcon = "".join(out)
    outcut = list(jieba.cut(outcon))
    outfinal = []
    for word in outcut:
        if word not in stopword:
            outfinal.append(word)
    datagood_select.append(outfinal)
for i in databad:
    out = re.findall(r'[\u4e00-\u9fa5]', i)
    outcon = "".join(out)
```

```python
outcut = list(jieba.cut(outcon))
outfinal = []
for word in outcut:
    if word not in stopword:
        outfinal.append(word)
databad_select.append(outfinal)
```

```
啊
哎
唉
俺
按
吧
把
甭
别
嘿
很
乎
会
或
既
及
啦
了
们
你
您
哦
砰
啊
你
我
他
她
它
```

图 3 （部分停用词列表）

然后是分词器，这边笔者规定分词器只保留出现频次最高的 1200 个词，规定输入数据的最大长度为 30，不足 30 的补零，超过 30 的进行丢弃操作，正例

标记标签为 1，负例标记标签为 0，按照 7：3 的比例切分训练集和测试集，代码如下：

```
from keras.preprocessing.text import Tokenizer
from keras_preprocessing.sequence import pad_sequences
from sklearn.model_selection import train_test_split
import numpy as np
MAX_LEN = 30#句子的最大长度
num_char=1200#保留 1200 个词
tokenizer_str = Tokenizer(num_words=num_char)
tokenizer_str.fit_on_texts(datagood_select+databad_select)
#print("word_index: \n",tokenizer_str.word_index)
datagood_ls=[]
databad_ls=[]
for i in datagood_select:
    if len(i)<MAX_LEN:
        out = tokenizer_str.texts_to_sequences([i])
        datagood_ls.append(out[0])
for i in databad_select:
    if len(i)<MAX_LEN:
        out = tokenizer_str.texts_to_sequences([i])
        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 30 个词，转化成 numpy 数组
data_all_mat = pad_sequences(dataall, maxlen=MAX_LEN, padding='post')
dataout = np.array(dataout)
#print(len(data_all_mat))
data_train, data_test, dataout_train, dataout_test = train_test_split(data_all_
mat, dataout, test_size=0.3, random_state=42, shuffle=True)
```

接着是对于模型的搭建，这边笔者分别用了 BiLstm 模型和 transformer 两个模型，对于 BiLstm 的模型的介绍可以查看这篇文章：[基于 BiLstm 模型的恶意 url 检测 (caodong0225.github.io)]。这边就不在过多的赘述，用的模型和上文的模型一模一样。而对于 transformer 模型，它是一个利用注意力机制来提高

模型训练速度的模型，关于它的工作原理，这边就不做介绍了。

模型的流程图如下：

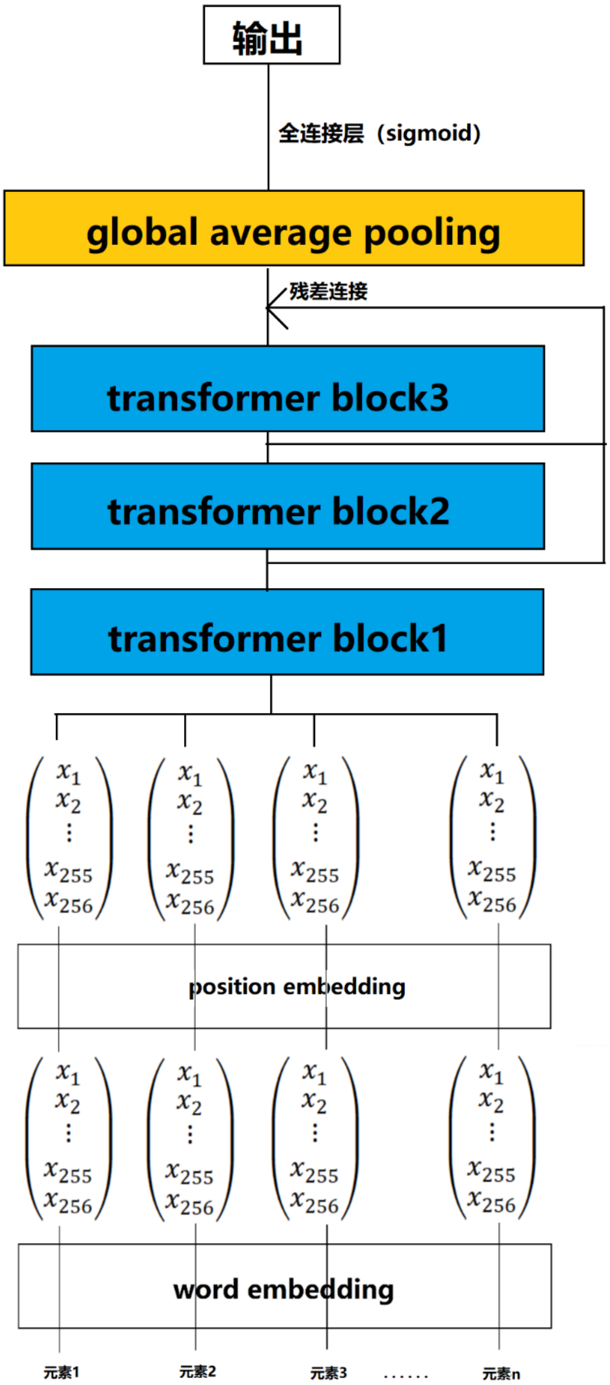


图 4 （transformer 模型流程图）

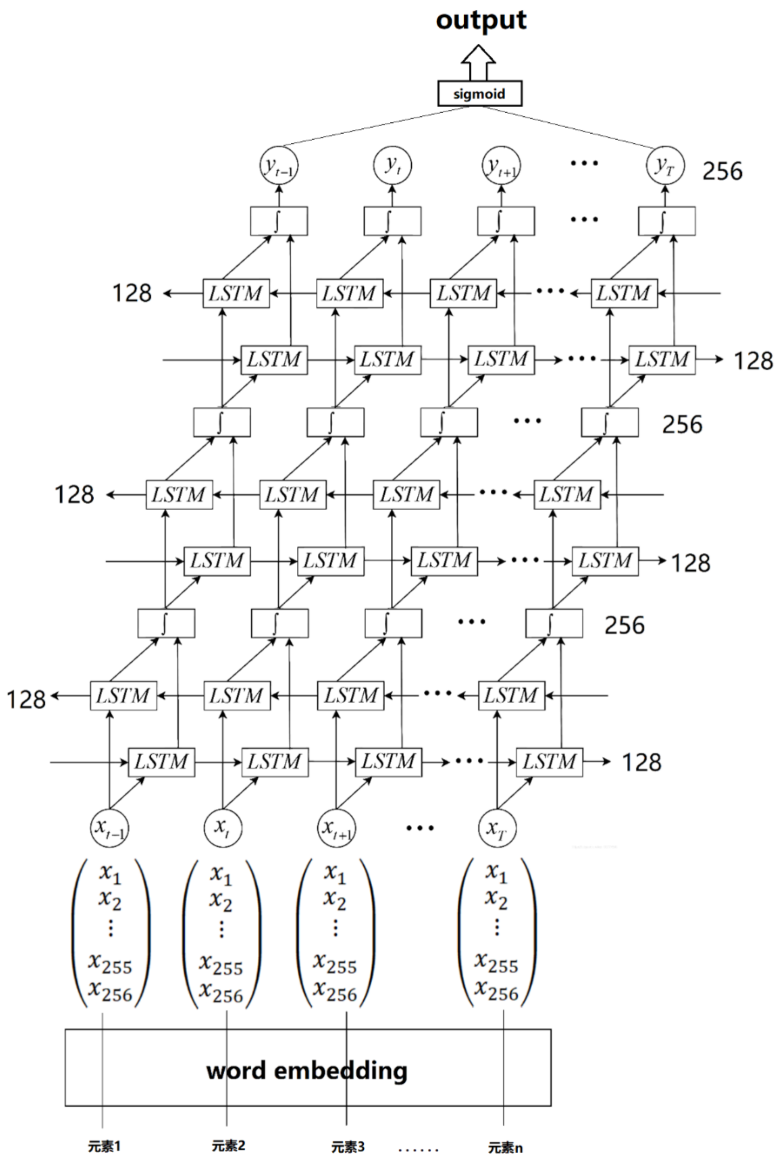
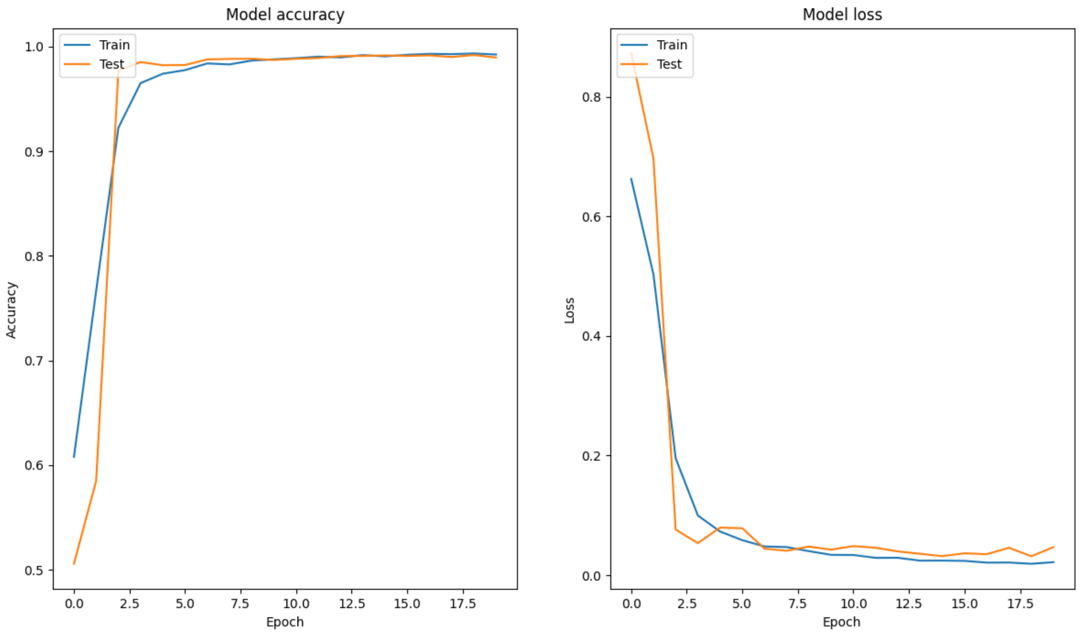


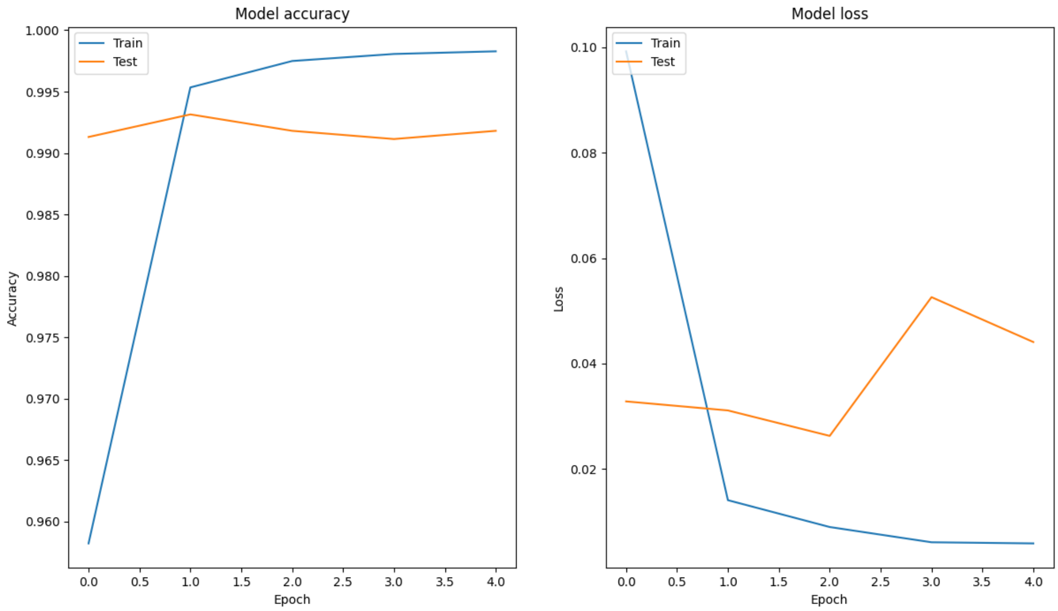
图 5 （Bilstm 模型流程图）

两个模型的训练效果图分别如下：




```
Epoch 1/20
110/110 - 53s - loss: 0.6628 - accuracy: 0.6080 - val_loss: 0.8726 - val_accuracy: 0.5059 - 53s/epoch - 477ms/step
Epoch 2/20
110/110 - 52s - loss: 0.5032 - accuracy: 0.7666 - val_loss: 0.6970 - val_accuracy: 0.5851 - 52s/epoch - 472ms/step
Epoch 3/20
110/110 - 52s - loss: 0.1956 - accuracy: 0.9224 - val_loss: 0.0761 - val_accuracy: 0.9769 - 52s/epoch - 471ms/step
Epoch 4/20
110/110 - 52s - loss: 0.0998 - accuracy: 0.9650 - val_loss: 0.0536 - val_accuracy: 0.9851 - 52s/epoch - 472ms/step
Epoch 5/20
110/110 - 54s - loss: 0.0730 - accuracy: 0.9740 - val_loss: 0.0795 - val_accuracy: 0.9821 - 54s/epoch - 493ms/step
Epoch 6/20
110/110 - 58s - loss: 0.0585 - accuracy: 0.9774 - val_loss: 0.0785 - val_accuracy: 0.9823 - 58s/epoch - 523ms/step
Epoch 7/20
110/110 - 63s - loss: 0.0480 - accuracy: 0.9838 - val_loss: 0.0443 - val_accuracy: 0.9876 - 63s/epoch - 571ms/step
Epoch 8/20
110/110 - 63s - loss: 0.0469 - accuracy: 0.9829 - val_loss: 0.0408 - val_accuracy: 0.9881 - 63s/epoch - 570ms/step
Epoch 9/20
110/110 - 63s - loss: 0.0401 - accuracy: 0.9865 - val_loss: 0.0478 - val_accuracy: 0.9883 - 63s/epoch - 571ms/step
Epoch 10/20
110/110 - 62s - loss: 0.0340 - accuracy: 0.9877 - val_loss: 0.0427 - val_accuracy: 0.9871 - 62s/epoch - 568ms/step
Epoch 11/20
110/110 - 59s - loss: 0.0337 - accuracy: 0.9889 - val_loss: 0.0486 - val_accuracy: 0.9883 - 59s/epoch - 538ms/step
Epoch 12/20
110/110 - 54s - loss: 0.0289 - accuracy: 0.9903 - val_loss: 0.0458 - val_accuracy: 0.9890 - 54s/epoch - 492ms/step
Epoch 13/20
110/110 - 52s - loss: 0.0291 - accuracy: 0.9895 - val_loss: 0.0397 - val_accuracy: 0.9908 - 52s/epoch - 475ms/step
Epoch 14/20
110/110 - 51s - loss: 0.0243 - accuracy: 0.9918 - val_loss: 0.0357 - val_accuracy: 0.9910 - 51s/epoch - 460ms/step
Epoch 15/20
110/110 - 51s - loss: 0.0244 - accuracy: 0.9905 - val_loss: 0.0318 - val_accuracy: 0.9915 - 51s/epoch - 461ms/step
Epoch 16/20
110/110 - 52s - loss: 0.0239 - accuracy: 0.9920 - val_loss: 0.0366 - val_accuracy: 0.9910 - 52s/epoch - 471ms/step
Epoch 17/20
110/110 - 52s - loss: 0.0210 - accuracy: 0.9930 - val_loss: 0.0351 - val_accuracy: 0.9915 - 52s/epoch - 468ms/step
Epoch 18/20
110/110 - 51s - loss: 0.0211 - accuracy: 0.9927 - val_loss: 0.0458 - val_accuracy: 0.9900 - 51s/epoch - 465ms/step
Epoch 19/20
110/110 - 51s - loss: 0.0191 - accuracy: 0.9933 - val_loss: 0.0317 - val_accuracy: 0.9918 - 51s/epoch - 465ms/step
Epoch 20/20
110/110 - 51s - loss: 0.0217 - accuracy: 0.9923 - val_loss: 0.0468 - val_accuracy: 0.9895 - 51s/epoch - 463ms/step
```

图 6 （transformer 模型训练图）




```
Epoch 1/5
110/110 - 56s - loss: 0.0993 - accuracy: 0.9582 - val_loss: 0.0328 - val_accuracy: 0.9913 - 56s/epoch - 505ms/step
Epoch 2/5
110/110 - 77s - loss: 0.0141 - accuracy: 0.9953 - val_loss: 0.0311 - val_accuracy: 0.9931 - 77s/epoch - 697ms/step
Epoch 3/5
110/110 - 86s - loss: 0.0090 - accuracy: 0.9975 - val_loss: 0.0263 - val_accuracy: 0.9918 - 86s/epoch - 780ms/step
Epoch 4/5
110/110 - 118s - loss: 0.0061 - accuracy: 0.9981 - val_loss: 0.0526 - val_accuracy: 0.9911 - 118s/epoch - 1s/step
Epoch 5/5
110/110 - 112s - loss: 0.0059 - accuracy: 0.9983 - val_loss: 0.0441 - val_accuracy: 0.9918 - 112s/epoch - 1s/step
```

图 7 （BiLstm 模型图）

最终 BiLstm 模型在训练集上的准确度达到了 99.8%,在测试集达到了 99.1%。

transformer 模型在训练集上的准确度达到了 99.2%，在测试集达到了 98.9%。

完整代码如下：

Transformer:

```
#coding:gbk
from __future__ import print_function
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
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,model_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
```

```python
        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_shape[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_shape[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_num)) #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 + multihead(X))
    O_seq_fc1 = Dense(MODEL_DIM * 4,activation='relu',name=f'{prefix}_fc1')(O_seq_LN1) #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()
```

```python
    return model


fr2 = open("bad.txt","r",encoding="utf-8")
databad = fr2.readlines()
fr2.close()
fr3 = open("good.txt","r",encoding="utf-8")
datagood = fr3.readlines()
fr3.close()
fr4 = open("stopword.txt","r",encoding="utf-8")
stopword = fr4.readlines()
fr4.close()
stopword = [i.strip() for i in stopword]
datagood_select = []
databad_select = []
for i in datagood:
    out = re.findall(r'[\u4e00-\u9fa5]', i) #把非中文字符都替换掉
    outcon = "".join(out)
    outcut = list(jieba.cut(outcon))
    outfinal = []
    for word in outcut:
        if word not in stopword:
            outfinal.append(word)
    datagood_select.append(outfinal)
for i in databad:
    out = re.findall(r'[\u4e00-\u9fa5]', i)
    outcon = "".join(out)
    outcut = list(jieba.cut(outcon))
    outfinal = []
    for word in outcut:
        if word not in stopword:
            outfinal.append(word)
    databad_select.append(outfinal)


MAX_LEN = 30#句子的最大长度
num_char=1200#保留 1200 个词
tokenizer_str = Tokenizer(num_words=num_char)
tokenizer_str.fit_on_texts(datagood_select+databad_select)
#print("word_index: \n",tokenizer_str.word_index)
datagood_ls=[]
databad_ls=[]
for i in datagood_select:
```

```
    if len(i)<MAX_LEN:
        out = tokenizer_str.texts_to_sequences([i])
        datagood_ls.append(out[0])
for i in databad_select:
    if len(i)<MAX_LEN:
        out = tokenizer_str.texts_to_sequences([i])
        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 30 个词，转化成 numpy 数组
data_all_mat = pad_sequences(dataall, maxlen=MAX_LEN, padding='post')
dataout = np.array(dataout)
#print(len(data_all_mat))
data_train, data_test, dataout_train, dataout_test = train_test_split(data_all_mat,
 dataout, test_size=0.3, random_state=42, shuffle=True)
MODEL_DIM = 256
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=20, batch_size=128, validation
_data=(data_test, dataout_test),verbose=2)
# 保存模型
model.save('transformer.h5')

# 保存分词器
with open('transformer.pickle', 'wb') as handle:
    pickle.dump(tokenizer_str, handle, protocol=pickle.HIGHEST_PROTOCOL)


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()
```

BiLstm 模型的代码如下:

```
import re
from keras import Sequential
from keras.layers import Embedding
from keras.models import Model,load_model
from keras.layers import Conv2D, MaxPooling2D, Dropout, Activation, Flatten, Dense,
BatchNormalization,LSTM,Bidirectional
import pickle
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
import jieba
from keras.preprocessing.text import Tokenizer
import numpy as np
from keras_preprocessing.sequence import pad_sequences
fr2 = open("bad.txt","r",encoding="utf-8")
databad = fr2.readlines()
fr2.close()
fr3 = open("good.txt","r",encoding="utf-8")
datagood = fr3.readlines()
fr3.close()
fr4 = open("stopword.txt","r",encoding="utf-8")
stopword = fr4.readlines()
fr4.close()
stopword = [i.strip() for i in stopword]
num_char=1200
MAX_LEN = 30
datagood_select = []
databad_select = []
for i in datagood:
    out = re.findall(r'[\u4e00-\u9fa5]', i)
    outcon = "".join(out)
    outcut = list(jieba.cut(outcon))
    outfinal = []
    for word in outcut:
        if word not in stopword:
```

```
            outfinal.append(word)
    datagood_select.append(outfinal)
for i in databad:
    out = re.findall(r'[\u4e00-\u9fa5]', i)
    outcon = "".join(out)
    outcut = list(jieba.cut(outcon))
    outfinal = []
    for word in outcut:
        if word not in stopword:
            outfinal.append(word)
    databad_select.append(outfinal)
tokenizer_str = Tokenizer(num_words=num_char)
tokenizer_str.fit_on_texts(datagood_select+databad_select)
#print("word_index: \n",tokenizer_str.word_index)
datagood_ls=[]
databad_ls=[]
for i in datagood_select:
    if len(i)<MAX_LEN:
        out = tokenizer_str.texts_to_sequences([i])
        datagood_ls.append(out[0])
for i in databad_select:
    if len(i)<MAX_LEN:
        out = tokenizer_str.texts_to_sequences([i])
        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 30 个词，转化成 numpy 数组
data_all_mat = pad_sequences(dataall, maxlen=MAX_LEN, padding='post')
dataout = np.array(dataout)
#print(len(data_all_mat))
data_train, data_test, dataout_train, dataout_test = train_test_split(data_all_mat,
 dataout, test_size=0.3, random_state=42, shuffle=True)
model = Sequential()
model.add(Embedding(num_char, 256, input_length=MAX_LEN))
model.add(Bidirectional(LSTM(128, return_sequences=True)))
model.add(Dropout(0.5))
model.add(Bidirectional(LSTM(128, return_sequences=True)))
model.add(Dropout(0.5))
model.add(Bidirectional(LSTM(128, return_sequences=False)))
model.add(Dropout(0.5))
model.add(Dense(1))
model.add(Activation('sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
model.summary()
```

```python
history = model.fit(data_train,dataout_train, epochs=5, batch_size=128, validation_data=(data_test, dataout_test),verbose=2)
# 保存模型
model.save('bilstm.h5')

# 保存分词器

with open('bilstm.pickle', 'wb') as handle:
    pickle.dump(tokenizer_str, handle, protocol=pickle.HIGHEST_PROTOCOL)


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()
```

为了测试训练的成果，笔者用自己的手机短信进行测试，笔者选取了自己手机上的 200 条短信，并将其分成三类：正常短信（标签为 1），垃圾短信（标签为 0）和可以分成垃圾也可以不算垃圾的短信（标签为 2）。

1【肯德基】363614（动态验证码），请在30分钟内填写
0 尊敬的客户您好，为持续提升中国移动客户服务质量，诚邀您对手机业务办理规范性打分，请回复10-1间任意数字，10表示非常规范，1表示非常不规范。(72小时有效)【中国移动】
0 【客户调研】尊敬的客户，诚邀您为上海移动业务及服务打分（1-10分），请回复1-10中任意数字，10分表示非常满意，1分表示非常不满意（48小时有效）。我们百倍努力，愿您10分满意！【中国移动】
0 【满格移动，服务上海】上海移动持续开展4/5G网络升级优化！为答谢您的信任与支持，特赠送您2GB国内通用流量（24小时内回复“10”领取，即时生效，次月底失效）。我们百倍努力，愿您10分满意！【中国移动】
2 【您的流量即将到期】尊敬的用户，您的双V账户流量额度即将过期清零，请升级后尽快兑换，每月仅需1.99元，享最高10G通用流量。立即查看http://a.10086.cn/c/u/722121924 回TD关闭短信通知【中国移动X支付宝双V会员】
1【阿里云】您正在进行短信登录，验证码933907，切勿将验证码泄露于他人，本条验证码有效期15分钟。
1【未来设计师】亲爱的用户，您的注册验证码是：043197，5分钟内有效。如非本人操作，请忽略。
2【上海移动】尊敬的客户182****7715，4月话费账单已送达您的139邮箱，点击查看账单详情 http://y.10086.cn/t/jSSvKzJCskkv1c 回Q关闭通知。客户端查账单更方便 http://y.10086.cn/p/f10 【中国移动 139邮箱】
2【上海移动】8月话费账单已送达您的139邮箱！点击查看账单详情 http://y.10086.cn/t/jSSvKzLhA2IJ1c 回Q关闭通知。体验APP领视频卡骑行券，y.10086.cn/x/qyAPPyl2，全心全意为您服务，期待您的十分满意【中国移动 139邮箱】
0主题：您的信用卡推荐办理通道已开通！[AD] 正文：最高10万额度，最快当天出结果，申请：http://y.10086.cn/x/gd007，以审批为准[未完] 发件人：139邮箱 查阅邮件：
http://y.10086.cn/n/jSSvKzvCijTd1E 回Q关闭通知【中国移动 139邮箱
2【中国移动X支付宝】尊敬的18217437715用户，您的双V账户有1.1GB流量即将过期，请您留意兑换使用，现在升级额外再得1GB。双V会员流量/生活/充值特权，仅需1.99元/月。查看详情
http://y.10086.cn/t/jSSvKzMdeTzw19
0【限时专享活动，优惠即将到期】恭喜您获得上海移动VIP超值套餐优惠礼包，即刻订购VIP特惠套餐68元档，可额外获得每月20-50G通用流量和500-1000分钟语音等限量礼包，总计12个月。马上回复“TH1850”一键升级套餐后，立享礼包！
1【充值提醒】尊敬的客户，您好！充值金额50.00元，余额43.53元。了解本月已使用话费情况，请发送102到10086查询【中国移动】【权益精选】PRO会员可领取5元话费充值礼，回复“开通月会员”查看详情！
0您的【套餐升级限时特惠权益】即将到期，回复短信“1850限时“至10086，或点击链接 https://dx.10086.cn/X8q-Cg 直接办理，也可通过下载登录上海移动和你APP-“我的礼包”领取（活动截止本月底）。【中国移动】
0上海移动提醒服务：您的30元授信额度即将失效，不可以申请或者修改授信额度，请及时交费保持良好的信用记录。【中国移动】
2【话费账单】尊敬的182****7715客户，您2021年02月01日-2021年02月28日共消费48.20元，实际应付48.20元。 主要消费项目包括： - 套餐及固定费48.00元； - 套餐外短彩信费0.20元。 查询费用详情，请点击
https://10086.cn/d/VVnQfi ，账单页面访问免流量，更多信息可详询10086。 【中国移动】
1上海移动提醒服务：至2021年3月7日0时，您余额已少于15元，为5.53元，请及时充值。推荐您办理“易充值业务”（免费），手机关联银行卡，自动充值交帐单，回复“易充值”了解详情。点击链接
http://dx.10086.cn/6yYVJVf 下载/打开“上海移动和你”App，话费余额随时查询。【中国移动】Your account balance is below 15 yuan. Please visit a local China Mobile store or our website at sh.10086.cn for recharge. For details or inquiries, please call 10086.【China Mobile】【充值优惠】领取5元话费充值礼，回复“开通月会员”查看详情！
0【和你尊享权益礼】通话省钱小妙招：全国亲情网，群内亲友之间享无限畅打，加在群里，也放到心上。点击 https://dx.10086.cn/jXOjAQ 查看或发送“全国畅打”到10086办理，全国移动手机号码均可加入。现在办理，还可领取i百联1小时停车券，先到先得！您也可在“上海移动和你APP”-“我的礼包”领取办理。【中国移动】
0【引领全千兆】5G有多快，和彩云网盘就有多快！仅需1元/月，享1T和彩云网盘空间，加享200G定向流量！一键备份照片视频，从此告别手机空间不足！点击链接即可查看 https://dx.10086.cn/GkFoBw ，详询10086。
0您的20元专属优惠券已到账！原30元5G特惠包现立减20元，再加赠5GB流量！仅需10元即可享10GB流量，更享视频彩铃、视频会员等权益。月底前回复“活动30”至10086即可办理，或点击
https://dx.10086.cn/08QZj2ui 直接办理。【中国移动】
0恭喜您获得上海移动超值套餐升级限时特惠权益！每月仅需68元，含流量10GB、语音80分钟，还可加赠至少20GB国内流量、500分钟语音通话。回复短信“1850限时“至10086，或点击 https://dx.10086.cn/X8q-Cg 直接办理，也可通过下载登录上海移动和你APP-“我的礼包”领取（活动截止本月底）。【中国移动】
0【好网络好体验】好消息！上海移动4/5G网络质量升级啦，为答谢您一直以来的信任与支持，特此赠送2GB全国通用流量，欢迎体验。请在本月内登录“上海移动和你App-我的礼包” 领取吧！我们全心全意，愿您10分满意。【中国移动】
0【限时流量通话礼包提醒】恭喜您获得上海移动VIP超值套餐优惠礼包，即刻订购VIP特惠套餐68元档（50元流量+18元语音），可额外获得每月20-50G通用流量和500-1000分钟语音等限量礼包，总计12个月。马上回复

图 8 （笔者自己手机的短信）

笔者用刚刚训练得到的 BiLstm 和 transformer 两个模型分别在此验证机上进行预测，最终 BiLstm 模型达到了 95%的准确度，transformer 模型达到了 93%的准确度。可见效果还是不错的。由于该数据集含有笔者个人的隐私信息，故不放出来公开。