深度学习第二课：个性化推荐

大家好，我是来自 PaddlePaddle 开源社区的李钊（@livc），目前是一名大三学生。我曾经在手机百度实习，参与推荐算法和反作弊的研发工作，目前是 IDL 的一名实习生。很开心作为 Paddle Tutorials 系列的作者之一参加 GitChat 的分享。

在 Paddle 深度学习系列 Chat 的第一课中，官方开发组的张睿卿同学通过介绍一些深度学习的应用场景，带领大家了解深度学习的基本原理和工作方式，我们先来简单回顾下。

“人工智能”并不是一个很新的概念，它其实已经有 60 岁了，它的发展经历了三起三落，像极了数学史上的“三次危机”。作为燃料的大数据和硬件（GPU）腾兴带来的并行运算，促成了深度学习在 2012 年左右的大爆发。深度学习有很多有趣的应用，比如，搭载 GoPro 的小车的“自动驾驶”可以视为一个回归问题，普通的照片可以模仿出著名艺术家画作的风格，在机器翻译、序列生成等领域也有所突破。此外，深度学习并不“完美”，还有很多理论基础问题等待我们去解决，比如说存在可解释性的局限：很多东西不能称为“方法”，只能称为“窍门”（trick），南大周志华教授将其比作“老中医看病”。

从去年年底开始，Paddle 社区将理论与实践结合，开始撰写一份深度学习教程，其中包括：新手入门、识别数字、图像分类、词向量、情感分析、文本序列标注、机器翻译、个性化推荐。这份教程的每一章都对应一个真实问题，从背景介绍到代码实践，带领大家完整地解决问题。

本次 Chat 的主题是个性化推荐。在系列教程个性化推荐一文中，我们介绍了推荐系统的背景和经典模型，并以电影推荐为例，使用 MovieLens 数据集和 PaddlePaddle 训练了一个神经网络模型。

什么是推荐系统

> 随着信息技术和互联网的发展，人们逐渐从信息匮乏的时代走入了信息过载（information overload）的时代。在这个时代，无论是信息消费者还是信息生产者都遇到了很大的挑战：作为信息消费者，如何从大量信息中找到自己感兴趣的信息是一件非常困难的事情；作为信息生产者， 如何让自己生产的信息脱颖而出，受到广大用户的关注，也是一件非常困难的事情。推荐系统就是解决这一矛盾的重要工具。
>
> —— 项亮 《推荐系统实践》

乔布斯曾说，“消费者并不知道自己需要什么，直到我们拿出自己的产品，他们就发现，这是我要的东西”。同样，我们也可以说，信息爆炸的时代，面对琳琅满目的商品，用户

很可能不知道自己真正喜欢什么，如果没有推荐系统，用户也许永远不知道有更喜欢、更适合的商品没有浏览到。

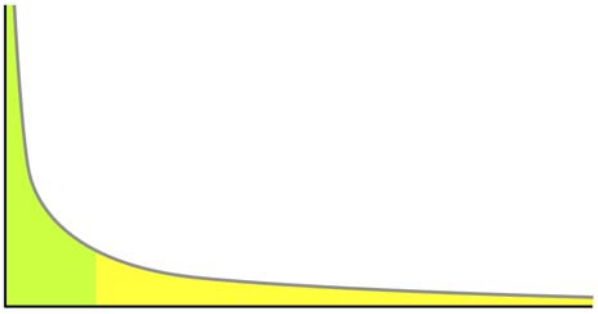
推荐系统和搜索引擎是人们获取信息的两种主要方法，与搜索引擎相比，推荐系统并不需要用户主动地寻找信息或商品，也不需要用户输入难以用简练文字描述的需求。但二者并不矛盾，在很多业务场景上推荐和搜索是相互结合的，比如说，搜索“周杰伦”时侧栏会推荐《听妈妈的话》。

从用户的角度讲，人们往往喜欢花 2 个小时看一部电影，却不愿意花 20 分钟去挑选一部电影；从企业的角度看，Data Science Central 编辑总监 Bill Vorhies 曾撰文[1]表示，“据估计，对亚马逊和 Netflix 这样的主要电商平台来说，个性化推荐的用户可能带来多达 10％到 25％ 的增量收入”，这就是推荐系统的意义。

长尾效应

长尾（The Long Tail）最初由《连线》的总编辑克里斯·安德森（Chris Anderson）于 2004 年提出，用来描述诸如亚马逊和 Netflix 之类的网站的商业和经济模式，指那些不受到重视的销量小但种类多的产品或服务，由于总量巨大，累积起来的总收益超过主流产品的现象。在互联网领域，长尾效应尤为显著[2]。

如下图所示，图中横轴表示数据类型，纵轴表示频率，大部分数据的频率都很低，但都是大于零的（图中右侧黄色部分），这就是长尾。比如，人们生活中常用的汉字其实并不多，但因频率较高，所以这些为数不多的汉字占据了左侧绿色区域，而绝大部分的汉字罕有使用，它们就属于长尾。



一个优秀的推荐系统不仅能推荐全局热点，更应该能够准确地理解**“长尾”需求**：通过挖掘某种用户群体的小众需求，将符合条件但并不热门的商品或信息推荐给用户。由于并非每个人的偏好都与主流完全一致，长尾数据的成功挖掘将带来远远高于平均的效益。

百度研究院的王益老师曾在《分布式机器学习系统》系列讲座上分享过一个真实的 case：用户搜索“红酒木瓜汤”，如果推荐系统能够理解出“丰胸”、“美容”、“减肥”等方面的语义，那点击（或购买）的几率将远远高于平均，推荐系统的任务就是将长尾需求和用户偏好挖掘出来并匹配。亚马逊高级副总裁 Steve Kessel 曾说“如果我有 10 万种书，哪怕一次仅卖掉一本，10 年后加起来它们的销售就会超过最新出版的《哈利·波特》！”说的其实也是这个道理。

传统的推荐方法

传统的推荐方法可以分为协同过滤推荐、基于内容过滤推荐和组合推荐，其中协同过滤的应用最为广泛，我们的教程中有更详细的介绍。

推荐方法	优点	缺点
协同过滤推荐	个性化程度高	冷启动、稀疏问题
基于内容过滤推荐	简单	不能发现新商品

协同过滤推荐和基于内容过滤推荐各有优缺点，所以在工业界中往往采用模型的组合方式，克服各自的缺点，达到更好的效果。在刚刚结束不久的 AAAI-17 大会上，1999 年的一篇论文因发现了将协同过滤与基于内容过滤结合起来的有效方式，被评为经典论文提名奖（Honorable Mention）。

深度学习具有优秀的提取特征的能力，能够学习多层次的抽象特征表示，并对异质或跨域的内容信息进行学习，因此近年来在推荐系统上的应用和探索也渐渐增多。

基于深度学习的推荐系统

这一部分，我们会介绍 Google 提出的 YouTube 深度神经网络推荐模型和宽度&深度学习模型，以及我们使用 PaddlePaddle 实现的融合推荐模型。

YouTube 的深度神经网络推荐系统

经常上 YouTube 看视频的同学可能知道，它的首页视频**几乎全部**是个性化的，足以见得推荐系统对这个世界上最大的视频网站的重要性。

YouTube 的推荐算法系统经历过几次改动，其团队也发布了很多相关的论文。在 2016 年 9 月的 RecSys 会议（推荐系统领域顶级会议）上，Google 发布了 YouTube 的深度神经网络推荐模型[3]。

这个模型由两个神经网络组成：候选生成网络和排序网络。这样划分是一个常见的做法：为了节省计算资源，首先从大规模样本中召回候选集，降低数据规模，然后进行更精细的运算，得到 top k。

候选生成网络将推荐问题建模为一个类别数极大的多类分类问题（如下图所示），它首先将用户的历史信息（如观看历史、搜索历史）和其他特征拼接成向量，输入给非线形多层感知器（MLP）。

在训练阶段，将 MLP 的结果输入 Softmax 进行多分类，预测时计算用户的综合特征（MLP 的输出）与所有视频的相似度，取得分较高的 k 个视频输入给排序网络。

这里 YouTube 团队介绍了“视频发布时间”（也可以称作 Example Age，样本年龄）这一特征，因为经过观察，用户更喜欢新发布的视频，哪怕有点和自己不相关，对于这样一个视频数目庞大的网站，新视频的推荐也是极其重要的。由于机器学习系统都是使用历史的行为数据来训练，这样就对过去存在一个隐式的偏差（bias），因此把 Example Age 特征加入模型后，可以发现模型结果和经验上的分布更相符。

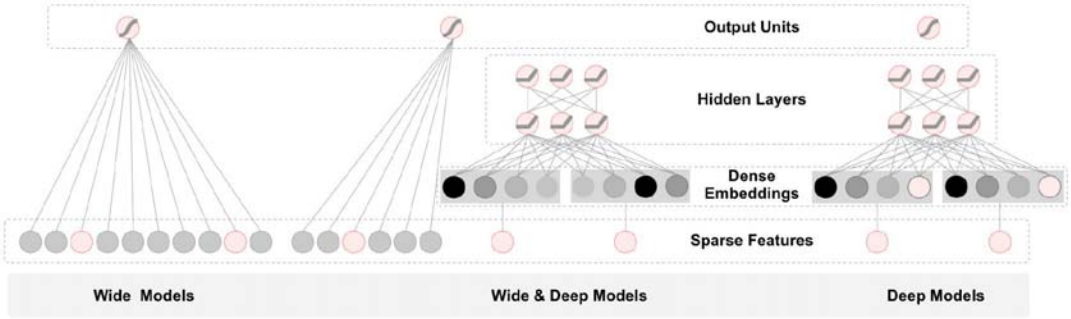
候选生成网络结构[4]

排序网络模型结构与候选生成网络类似，它添加了许多用于描述用户和视频相关性的更精细的特征，从而进行更细致的打分，比如用户很可能根据首页视频的缩略图去选择。此外，排序网络顶部使用加权逻辑回归（weighted logistic regression）进行训练，使用 e^x 作为测试阶段的激活函数。

Google 的宽度&深度学习（Wide & Deep learning）

Google 在 2016 年 6 月发布了一篇关于“宽度&深度学习”的论文[5]，业内一些公司也在纷纷学习。这里的推荐场景是 Google Play 应用商店，但其实 Wide & Deep 的方法可以泛化应用在更广义的推荐场景上。

简单来说，人脑就是一个不断**记忆（memorization）**并且**归纳（generalization）**的过程。比如说人们通过记忆“麻雀会飞”和“鸽子会飞”，归纳出“有翅膀的动物就会飞”的结论。由此获得启发，将宽线性模型（用于记忆，下图左侧）和深度神经网络模型（用于归纳，下图右侧）结合，汲取各自优势形成了 Wide & Deep 模型用于推荐排序（下图中间），这是一个非常有启发的探索。



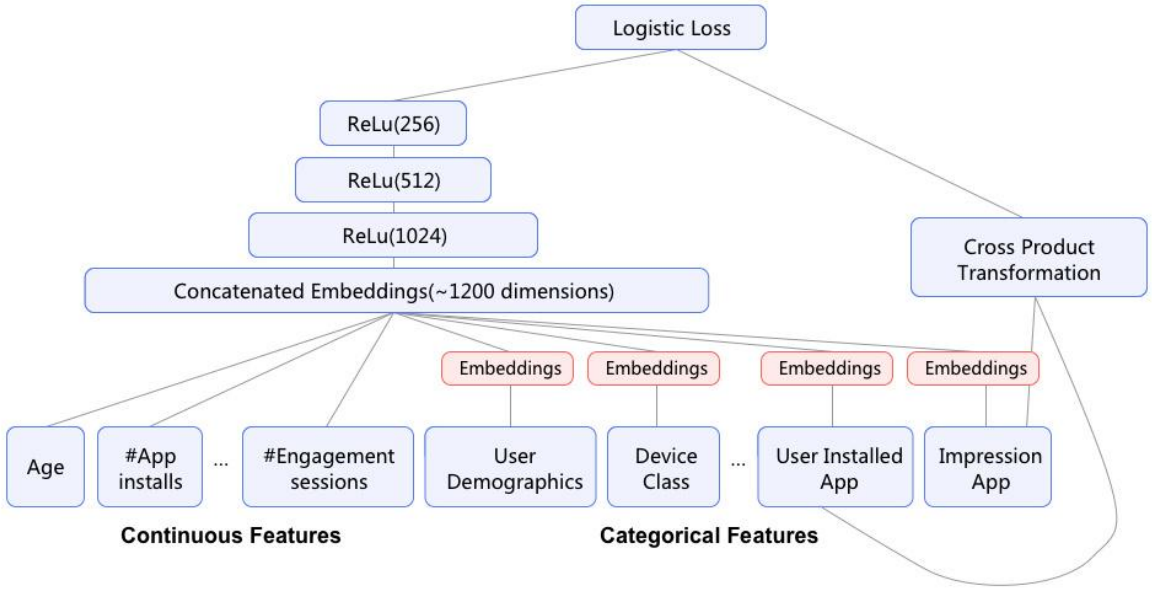
宽度&深度模型[6]

宽度模型的输入是用户安装应用和为用户展示（impression）的应用间的向量积（叉乘），模型通常训练 one-hot 编码后的二值特征（比如安装 netflix app 并展示了 pandora app 是 1，没有展示是 0），这种操作不会归纳出训练集中未出现的特征对。

基于 embedding 的深度模型可以探索出过去从未或很少出现的新的特征组合，提升了推荐商品的多样性。它可以添加小颗粒特征（比如安装了视频类应用，展示的是音乐类应

用），同时也需要手动完成特征工程。高维稀疏的类别特征（如人口学特征和设备类别）映射为低纬稠密的向量后，与其他连续特征（用户年龄、应用安装数等）拼接在一起，输入 MLP 中，最后输入逻辑输出单元。

预测（服务）时，宽度&深度学习模型会将所有候选应用的分数从高到低排序后返回给用户。

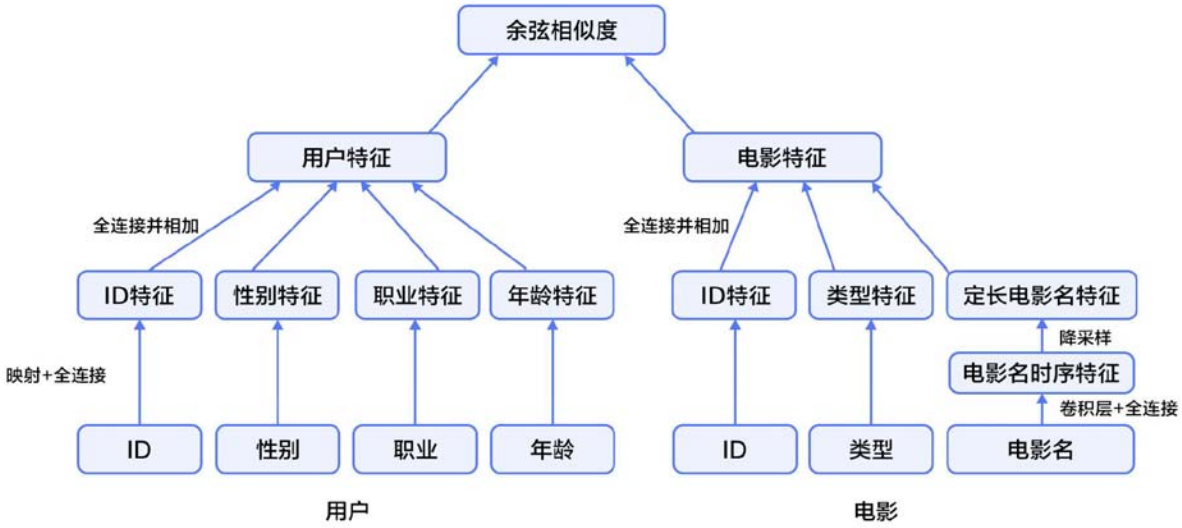


应用推荐中的宽度&深度模型[7]

尽管“ 宽度&深度学习”这一想法很简单，但经过试验，它显著提高了 Google Play 商店中应用的下载率，同时满足了训练和测试阶段的速度要求。值得一提的是，宽度&深度学习模型和集成（ensemble）学习并不是一回事，因为集成学习中的模型是分别独立训练的，互不干扰，只有在预测时才会联系在一起。

融合推荐模型

我们将使用 Paddle 实现电影推荐模型，数据集包含了 6,000 位用户对 4,000 部电影的 1,000,000 条评价（评分范围 1~5 分，均为整数），训练完成后，通过输入电影和用户的 ID，模型能够预测出该用户对该电影的评分，以代表喜好程度。这里只介绍主要的网络配置，完整版请见教程。



1. 设置 batch size、网络初始学习率，使用 RMSProp 优化方法。

```
settings(
   batch_size=1600, learning_rate=1e-3,
learning_method=RMSPropOptimizer())
```

2. 构造用户、电影特征（以用户特征为例）


```
# 将用户ID，性别，职业，年龄四个属性分别映射到其特征隐层。

user_id_emb = embedding_layer(input=user_id, size=embsize)
user_id_hidden = fc_layer(input=user_id_emb, size=embsize)

gender_emb = embedding_layer(input=gender, size=embsize)
gender_hidden = fc_layer(input=gender_emb, size=embsize)

age_emb = embedding_layer(input=age, size=embsize)
age_hidden = fc_layer(input=age_emb, size=embsize)

occup_emb = embedding_layer(input=occupation, size=embsize)
occup_hidden = fc_layer(input=occup_emb, size=embsize)


# 将这四个属性分别全连接并相加形成用户特征的最终表示。

user_feature = fc_layer(
   input=[user_id_hidden, gender_hidden, age_hidden,
occup_hidden],
   size=embsize)
```

3. 计算余弦相似度，定义损失函数和网络输出。

```
similarity = cos_sim(a=movie_feature, b=user_feature,
scale=2)


# 训练时，采用regression_cost作为损失函数计算回归误差代价，并作为网络
的输出。


# 预测时，网络的输出即为余弦相似度。

if not is_predict:
   lbl=data_layer('rating', size=1)
   cost=regression_cost(input=similarity, label=lbl)
   outputs(cost)
else:
   outputs(similarity)
```

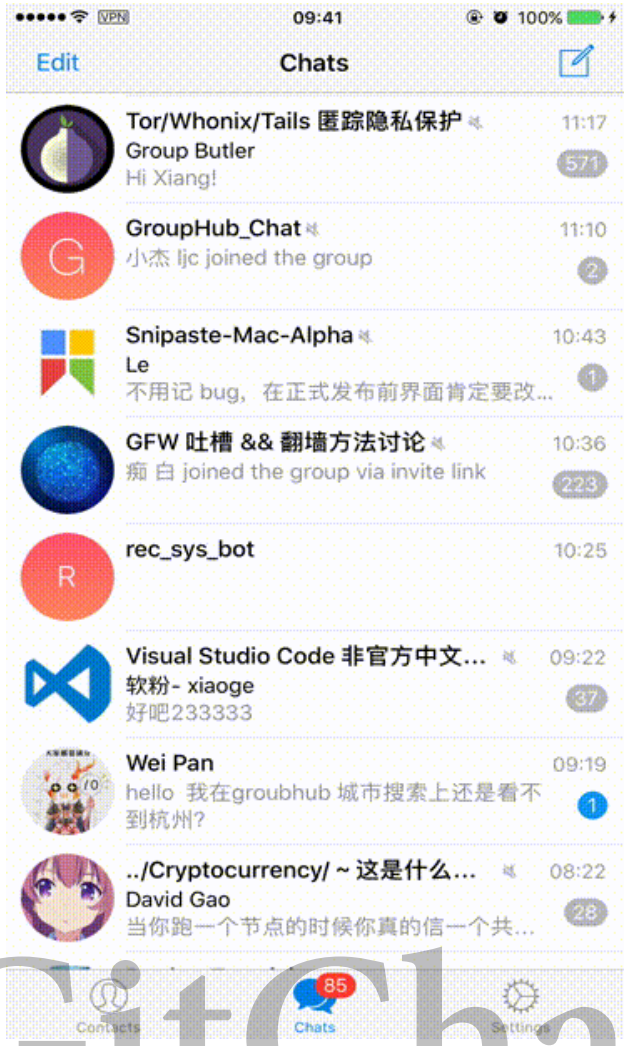
训练完成后，我们可以通过 `./evaluate.py log.txt` 评估模型，找出效果最好的模型轮数。接下来，我们搭建一个简单的 ChatBot 完成电影推荐的预测，作为融合推荐模型的应用。

融合推荐模型的 ChatBot 应用

近些年涌现出一大批聊天机器人和智能家庭设备，它们几乎全部支持个性化，比如“识别不同的人”，“根据不同人的喜欢推荐不同的内容”。Facebook 创始人扎克伯格使用多种 AI 技术为自己家里构建了一个自动控制系统，命名为 Jarvis，它能够根据家庭成员的喜好播放不同风格的音乐。所以，基于聊天机器人的个性化服务是未来的趋势。

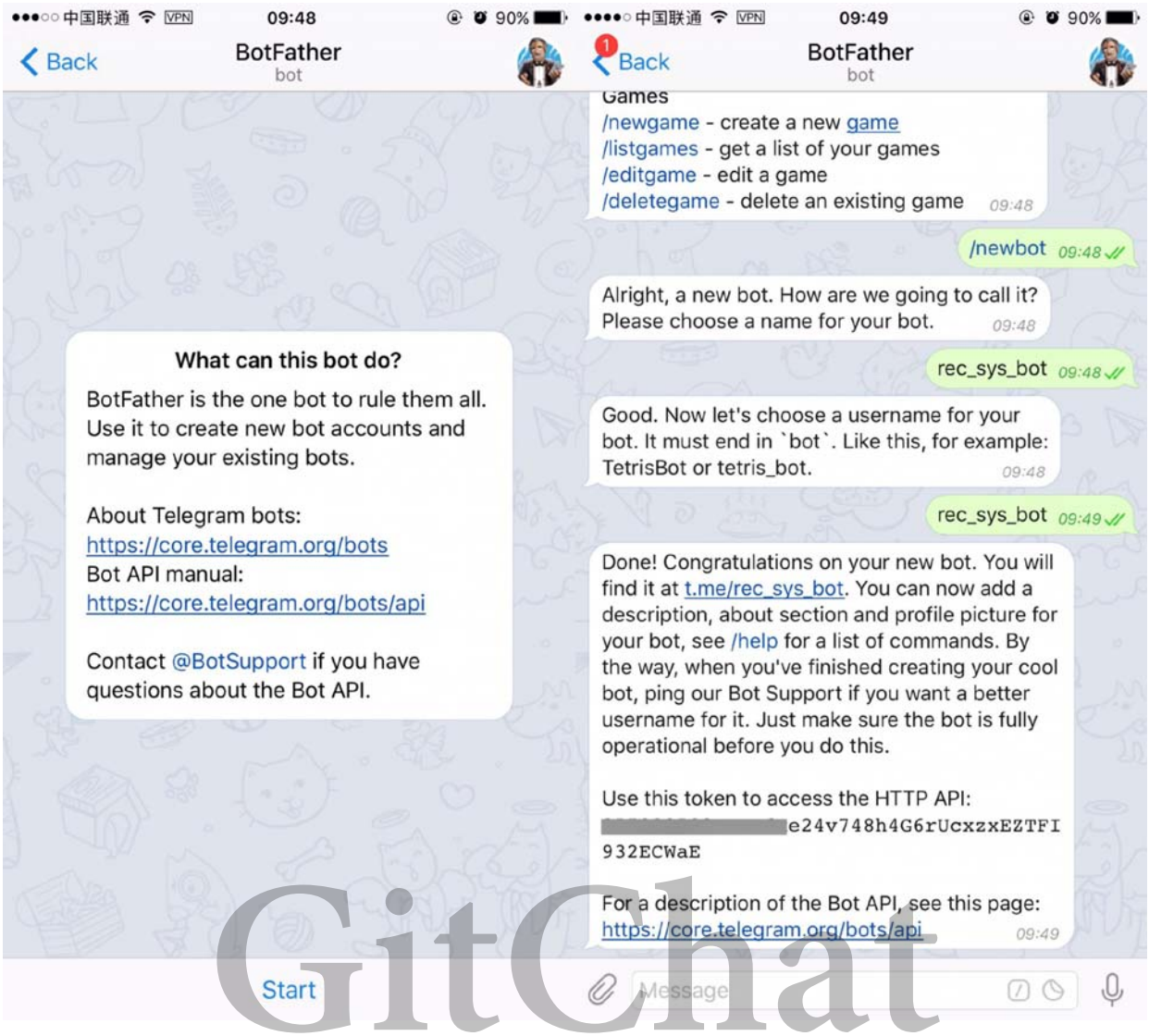
Bot 的开发**非常简单**，我们借助 Telegram 来完成这个任务。Telegram 是一款开源的即时通讯软件（类似微信、WhatsApp 等），它的机器人平台（Telegram Bot Platform）极大地丰富了生态，比如可以使用 Bot SSH 登录 VPS 、接收 RSS 订阅新闻或博客、下载 YouTube 视频、接收微信消息甚至是玩游戏等等。由于 Bot 是面向 API 的，我们可以开发某个 Workflow （比如 IFTTT）完成一系列的任务，有人为其创造了一个新名词，叫“r2r - robots 2 robots”。

由于其服务在中国的网络环境下并不容易访问，这里我推荐使用 proxychains 运行 Python 文件或 IPython 交互环境，当然也可以直接搭建在国外的 VPS 上。基于融合推荐模型的 ChatBot 最终效果如图：



1. 创建 Bot 并安装依赖

首先需要找官方的机器人老爹 @BotFather 发送 `/newbot` 命令申请创建，设置 bot 的基本信息后会得到一串 Token，帮助访问 HTTP API，这是 Bot 的唯一标识，不能泄露出去。



在 GitHub 上，很多热心的开发者使用不同的语言封装了原生 API，让开发变得更加容易。我们使用 Python 完成开发，因此首先安装依赖 `pip install python-telegram-bot --upgrade`。

2. 接入 Paddle 预测文件

变量 `MODEL_PATH` 是模型评估 `./evaluate.py log.txt` 的结果，函数 `cal_with_paddle` 实际就是教程中 `prediction.py` 的功能，输入是电影和用户的 ID，输出预测的评分。

Paddle 社区即将发布新版 API，数据读入、训练、预测的过程将变得更加简洁，因此你完全可以不关心这里究竟做了什么。

```
from py_paddle import swig_paddle, DataProviderConverter
from common_utils import *
from paddle.trainer.config_parser import parse_config
try:
    import cPickle as pickle
except ImportError:
    import pickle
# 模型路径
```

```
MODEL_PATH = 'output/pass-00004/'
def cal_with_paddle(movie_id, user_id):
    # 加载参数
    swig_paddle.initPaddle('--use_gpu=0')
    conf = parse_config("trainer_config.py", "is_predict=1")
    network = swig_paddle.GradientMachine.createFromConfigProto(
        conf.model_config)
    assert isinstance(network, swig_paddle.GradientMachine)
    network.loadParameters(MODEL_PATH)
    # 读入数据并预测
    with open('./data/meta.bin', 'rb') as f:
        meta = pickle.load(f)
        headers = [h[1] for h in meta_to_header(meta, 'movie')]
        headers.extend([h[1] for h in meta_to_header(meta,
'user')])
        cvt = DataProviderConverter(headers)
        movie_meta = meta['movie'][movie_id]
        user_meta = meta['user'][user_id]
        data = [movie_id - 1]
        data.extend(movie_meta)
        data.append(user_id - 1)
        data.extend(user_meta)
        return '%.2f' % (network.forwardTest(cvt.convert([data]))
[0]['value'][0][0] + 3)
```


3. 变量声明与函数定义 替换`TOKEN`为实际申请的字符串，定义按键界面，和交互函数。

```
from telegram import ReplyKeyboardMarkup
from telegram.ext import (Updater, CommandHandler,
MessageHandler, Filters, RegexHandler,
                          ConversationHandler)
# 两种交互方式，分别是按键选择回复和输入文本回复
CHOOSING, TYPING_REPLY = range(2)
TOKEN ='123456789:AAG6xe24v748h4G6rUcxzxEZTFI932ECWaE'

# 选择回复界面的三个按键，分别是输入电影ID、用户ID和预测
reply_keyboard = [['Movie_ID', 'User_ID'],
                  ['Predict']]
markup = ReplyKeyboardMarkup(reply_keyboard,
one_time_keyboard=True)

# 定义输出格式
def facts_to_str(user_data):
    facts = list()
    for key, value in user_data.items():
        facts.append('%s : %s' % (key, value))
    return "\n".join(facts).join(['\n', '\n'])

# `/start` 命令
def start(bot, update):
```

```
        update.message.reply_text(
            "Welcome to the movie recommender bot!",
            reply_markup=markup)
        return CHOOSING

    # 记录按键（key）并要求用户输入对应文本(value)
    def regular_choice(bot, update, user_data):
        text = update.message.text
        user_data['choice'] = text
        update.message.reply_text('Input %s please!' % text)
        return TYPING_REPLY

    # 记录文本(value)
    def received_information(bot, update, user_data):
        user_data[user_data['choice']] = update.message.text
        del user_data['choice']
        update.message.reply_text("Neat! This is what you already
    told me:"
                                  "%s"
                                  % facts_to_str(user_data),
                                  reply_markup=markup)
        return CHOOSING

    # 调用Paddle预测函数并输出结果
    def predict(bot, update, user_data):
        if 'choice' in user_data:
            del user_data['choice']

        score = cal_with_paddle(int(user_data['Movie_ID']),
    int(user_data['User_ID']))
        update.message.reply_text("Predicting with Paddle!\n"
                                  "Prediction Score is %s" % score)
        user_data.clear()
        return ConversationHandler.END
```

4. 开始运行

```
    updater = Updater(TOKEN)
    dp = updater.dispatcher

    # 定义对话handler
    conv_handler = ConversationHandler(
        # 以`/start`命令作为入口
        entry_points=[CommandHandler('start', start)],
        # 定义交互方式，支持正则匹配
        states={
            CHOOSING: [RegexHandler('^User_ID|Movie_ID$',
                                    regular_choice,
                                    pass_user_data=True),
                       ],
            TYPING_REPLY: [MessageHandler(Filters.text,
                                          received_information,
```

```
                                        pass_user_data=True),
                    ],
        },
        # 预测
        fallbacks=[RegexHandler('^Predict$', predict,
    pass_user_data=True)]
    )
    # 添加对话handler
    dp.add_handler(conv_handler)
    # 开始运行
    updater.start_polling()
```

至此，我们已经完成了 ChatBot 推荐模型的基本功能。Bot 中还有更丰富的功能值得探索，此外我们还可以接入云服务的 API，例如使用 Google Cloud Speech API 完成语音转文字的功能。

总结

近些年来，深度学习已经极大地推进了图像处理、语音识别、NLP 等领域的发展与进步，而在推荐系统上面的应用还处于早期阶段，同时也意味着有很大的发展空间。此外，深度学习正在为医学、生物信息学、逻辑推理、量化投资甚至围棋等领域带来新的启发与思考。我曾与学校的神经科学研究所合作，使用深度学习技术来分析食蟹猴基因特征，预测 microRNA 的碱基序列，获得了不错的效果，而最基本的神经网络结构也是从大脑的生物机理获得的启发，这形成了推动学科进步的良性循环。

2016 年的最后一天，罗振宇在他的“跨年演讲”中提到，“人工智能不是人的延伸，它是人的替代”；英伟达 CEO 黄仁勋在《智能工业革命》中认为：“继蒸汽机（发明）、大规模

生产以及自动化之后，AI 技术将引发第四次工业革命”；周志华教授在采访中说，“2017 年，机器学习技术将在更多行业带来更大价值”。各个行业的人们都在关注和见证着 AI 的发展，与此同时，很多工程师和社区（如 Paddle）正在努力着降低学习和应用的门槛。

我们有幸亲身经历了这次发展的浪潮，但仍需清醒地意识到其实还有很漫长的路等待人们的探索，我们期待更多如 GAN（生成对抗网络） 一样的新思想的爆发，这需要我们见素抱朴，不忘初心。

感谢

感谢订阅本次 Chat，个性化推荐这一章节的网络结构其实很简单，更多的知识和内容，还请关注该系列的后续分享。

大家熟知的许多任务，如：机器翻译，看图说话，为你写诗，对话机器人，标题党改写等等，背后都有着共同的模型。下一课我们将会介绍这些任务背后的深度神经网络模型，一起进入自然语言处理任务中一个非常有意思的问题：自动文本生成。

我们将在下一课介绍自然语言处理任务中的重要积木：循环神经网络。围绕循环神经网络，我们会一起讨论，如何对抗梯度消失和梯度爆炸，为什么需要深度循环神经网络，如何有效地训练深度循环神经网络。在此基础上，我们会继续开发本课中的对话机器人，引入神经图灵机的概念，介绍去年最火的技术之一：“注意力机制”，利用已有的积木，让循环神经网络从数据中学习，自动生成回复与用户进行有趣地交互。最后，我们会一起讨论现有技术面临的挑战，探讨一些加速文本生成任务的技术，期待大家的参与。

Paddle 不仅属于百度，更属于开源社区，我们希望对深度学习感兴趣的研究人员、工程师和开源爱好者能够加入 PaddlePaddle Tech Writer，撰写您所擅长的深度学习教程或设计有趣的示例，让更多的人感受到深度学习的魅力。如果您在使用 Paddle 过程中遇到任何问题，都可以去 GitHub 发起 Issue，社区的小伙伴们将在第一时间为您解答，希望 Paddle 与您共同成长。

参考资料

1. http://www.datasciencecentral.com/profiles/blogs/understanding-and-selecting-recommenders-1
2. https://zh.wikipedia.org/wiki/%E9%95%BF%E5%B0%BE
3. https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/45530.pdf
4. 引自论文[3]中图3
5. https://arxiv.org/abs/1606.07792
6. 引自论文[5]中图1
7. 引自论文[5]中图4

GitChat