Towards Conversational Recommender Systems分析重点

1.面向对话的推荐系统

2.首先提出一个场景来引出本文要解决的问题：

如果你来到一个陌生的城市，饿了，问当地人，想吃东西，让他们给点建议。他们会问你一些问题的，然后再给你推荐。比如：Do you like seafood? Do you have a car?

这两个问题和推荐餐馆有什么关系？

第一个问题跟吃的有关，第二个问题跟位置有关。所以，如果提出第一个问题的时候，你的回答是yes，而附近恰好有一家海鲜店，那么第二个问题就不需要问了。这里体现了两重含义：1.问题的上下文关系（举个例子：一个卖衣服的推荐系统在冬天喝夏天应该给用户推荐不同种类的服装。推荐系统不能因为用户在夏天喜欢过某件T恤就在冬天也给用户推荐类似的T恤）；2.问题的实时性（一问一答）。

3. 我们的推荐系统就要像这样的：

他给你推荐的时候，先要问你一些问题，

你有多饿？——不是很饿

你喜欢中国菜还是印度菜？——中国菜

OK，去Orchid怎样？——谢谢

这是一种对话，一问一答。

通过提问来捕捉用户的潜在特征，然后给出合理的推荐策略。

4. 总体的路子就是：（大纲）

（1）提出模型，搞出来item和用户之间能有效的传播反馈的隐藏结构；

（2）explore-exploit策略，即探索-利用策略，探测item空间，进行持续学习；

（3）feedback elicitation(反馈诱导)机制来选择绝对或相对问题。

5. 这里要建立的模型就是：概率矩阵分解模型

本页描述的就是绝对模型。

概率矩阵分解模型总体思路：在矩阵分解模型的基础上，加入了两种假设（一种是用户、物品的隐式特征向量服从高斯分布，另一种是已观测的评分数据条件概率也服从高斯分布），使得矩阵分解成为了一种生成模型，通过最大化对数后验来获得潜在特征矩阵。稍后解释概率矩阵分解。

6. 解释在线和离线的概念：在线体现了实时性，一旦用户做出某些反应，系统都会察觉，并且根据得到的反馈会进行计算更新，前面提到要实现这种对话形式的推荐，就是要实现在线实时分析。离线是对已有的数据进行分析，得出结果，就像给我们一堆数据集，让我们分析。而在线呢是数据集会发生变化，你要实时的给出分析结果。

这种实时包含了一个概念

因为这里要体现出来在线的意思，所以，作者把数据的状态（包含模型的各种参数）定义成了先验和后验。prior代表先验状态，就是没有经过计算处理的上一步的数据状态。posterior代表后验状态，就是先验状态经过一次完整的计算处理后的状态。

论文提到：

在线阶段的初始化是利用离线阶段的数据来完成的。从观测到的记录中学习离线属性。

假定所有在线阶段的item，在离线数据中都有。

就是用离线阶段的item潜在特征向量和偏差来初始化在线阶段的item。

对于用户参数的初始化，这里关注的是新来到系统的用户，他们一切都是新的，称为冷启动用户。

我们通过使用所有离线用户的特征和偏差平均值来初始化冷启动用户。

总体的推荐步骤：

（1）选择模型（绝对或者成对），然后选择偏好诱导机制：Abs 、 Abs Pos 、 Abs Pos & Neg 、Pairwise；

（2）使用离线数据初始化模型

（3）当一个新用户到来的时候，用几个问题来迭代：

a.系统选择一个问题来询问

b.用户回答问题

c.更新模型所有参数

d.问题--；

（4）系统给出最终推荐列表

7.问题选择策略

论文里面说问题选择的任务就是item选择的任务。（没太理解，那么前面提到的你有没有车这个问题怎么理解呢？）然而文中所有的讨论都是在研究item选择的策略。

这部分主要说的就是选择item的方式。这里列出的所有方法最终目的就是一个——选择item。只是他们的选择方法和标准不一样。

先看看最下面的几个：

Greedy: 每次只选择最高估计noisy affinity的item。用来利用。

Random：从1~N随机选择，用来探测。这是一种纯粹的随机选择

Min. Item Trait：与MaxItemTrait相反，选择信息量最小的。论文中假设它的效果最差，作为选择效果的下限。

主动学习(active learning)，指的是这样一种学习方法：

现实中，带类标的数据比较少而没有类标的数据是很多的，但是对数据进行人工标注代价高，这时候，学习算法可以主动地提出一些标注请求，将一些经过筛选的数据提交给专家进行标注。

这个筛选过程也就是主动学习主要研究的地方了，怎么样筛选数据才能使得请求标注的次数尽量少而最终的结果又尽量好。

主动学习的过程大致是：

通过有类标签的数据集找出没有类标签的数据集中的一部分，把这部分交给专家进行标注，标注完成的数据集纳入到有类标签的数据集中，这样一次次迭代。

就是说，主动学习的“主动”，指的是主动提出标注请求。主动学习是交互进行的。

最大方差：选择noisy affinity方差最大的item。 探测性选择。一种方差缩减策略。

最大item的特征向量：选择包含特征向量最大信息量的item。也就是最高的2范数。

Bandit learning：

所有的选择问题，都可以归纳为一种经典的模型——多臂赌博机问题（Multi-armed bandit problem）。

多臂赌博机问题描述：一个赌徒，到一个赌场，有若干台一模一样的赌博机，每个赌博机吐钱的概率不一样，也不知道每个赌博机吐钱的概率分布，那么想最大化收益，他得怎么整？

解决这类选择问题有专门的算法——bandit算法

怎么衡量不同bandit算法解决多臂问题的好坏？多臂问题里有一个概念叫做累计遗憾(regret)

每次选择后，老天爷爷都告诉你，和本该做的最佳选择差了多少，然后把每次差距累加起来就是总的遗憾。

是第i次试验时被选中臂的期望收益， 是所有臂中的最佳那个，如果老天爷爷提前告诉你，我们当然每次试验都选它，问题是老天爷爷不告诉你，所以我们就有了接下来的bandit算法。

完全朴素的方法：

先试几次，每个臂都有了均值之后，一直选均值最大那个臂。这个算法是我们人类在实际中最常采用的，不可否认，它还是比随机乱猜要好。

Thompson采样

假设每个臂是否产生收益，其背后有一个概率分布，产生收益的概率为p；

我们不断地试验，去估计出一个置信度较高的 概率p的概率分布 就能近似解决这个问题了。

怎么能估计概率p的概率分布呢？ 答案是假设概率p的概率分布符合beta(wins, lose)分布，它有两个参数: wins, lose。每个臂都维护一个beta分布的参数。每次试验后，选中一个臂，摇一下，有收益则该臂的wins增加1，否则该臂的lose增加1。

每次选择臂的方式是：用每个臂现有的beta分布产生一个随机数b，选择所有臂产生的随机数中最大的那个臂去摇。

Thompson采样用python就1行代码：

choice = numpy.argmax(pymc.rbeta(1 + self.wins, 1 + self.trials - self.wins))

beta分布可以看作一个概率的概率分布，当你不知道一个东西的具体概率是多少时，它可以给出了所有概率出现的可能性大小。

我们的x轴就表示各个击球率的取值，x对应的y值就是这个击球率所对应的概率。也就是说beta分布可以看作一个概率的概率分布。

UCB算法

UCB算法全称是Upper Confidence Bound(置信区间上界)

先对每一个臂都试一遍，之后，每次选择以下值最大的那个臂

其中加号前面是这个臂到目前的收益均值，后面的叫做bonus，本质上是均值的标准差，t是目前的试验次数，是这个臂被试次数。

这个公式反映：均值越大，标准差越小，被选中的概率会越来越大，起到了exploit的作用；同时那些被选次数较少的臂也会得到试验机会，起到了explore的作用。

8.在线更新

在向用户提出问题后，需要将观察到的响应并入推荐器，以便继续学习。

用户的响应成为一个新的观测值，系统用这个观测值来更新新来的用户i和所涉及的item j相关的所有潜在模型参数

Item更新了， 模型参数也更新了。

9.绝对模型，绝对问题

这个是本文提出的。

这种模型利用了前面说的Thompson采样，选择noisy affinity最高的item

这个不多说

10.相对问题反馈诱导

人们平时更善于给出成对儿的比较，而不是绝对的判断。所以后面就有了相对问题和相对模型。看起来这样就可以解决问题了，但是，新的问题来了：

如果让用户在两个都不喜欢的item做出选择的话，这样就会出现一个错误的模型。

解决这一问题的方法在后面说。

本文提出了绝对模型，相对问题来生成问题"A vs. B"给用户i。

11.绝对模型，相对问题

实际上，这种机制就是让用户去给两个不相似的item给出偏好。两种好处：

（1）可以更高效的学习用户的偏好

（2）用户不必强制的选择非常相似的item。

Abs Pos和Abs Pos & Neg.都用相同的机制来生成问题"A vs. B"给用户i。

（1）使用Abs来选择itemA，这里的Abs就是前面说到的“绝对模型，绝对问题”，PPT前翻两页

（2）虚拟观测值：假设用户i不喜欢A。

（3）用元组（i, A, 0）并入绝对模型，计算模型的所有参数，导出后验。用纳入了虚拟观测值模型的后验来更新虚拟先验。前面提到了后验和先验的概念，这里不赘述。

（4）再次使用Abs来选择itemB。这次用的是虚拟先验作为先验。

Abs Pos和Abs Pos & Neg的不同之处在于：

Abs Pos只包含正面的信息。

Abs Pos&Neg包含正反信息。

为了解决前面的问题（上一页PPT），本文通过以下的方式来搞定：

（i）在Abs Pos＆Neg中并入两个不喜欢的内容，（ii）省略Abs Pos和Pairwise中的更新。

12.成对模型

人们平时更善于给出成对儿的比较，而不是绝对的判断。

当用户i喜欢项目j超过项目h（j和h都是J上的索引）时，这样的偏好产生（用户i，项目j，项目h）形式的元组。

这个成对模型与前面的绝对模型很相似，只是在后面使用noisy affinity difference来描述差异。

13.成对模型，相对问题

（1）先选择两个item：使用Abs选择j\*,然后选择B，其实也是利用Abs选择的，只不过选择的是maximum noisy difference.（就是上页PPT提到的成对模型的noisy affinity difference）

（2）直接向用户询问这两个item，你喜欢谁？

14.实验

15.实验分为两个阶段：离线阶段和在线阶段

离线阶段：M个用户与N个item之间的联系，通过这些来获得离线嵌入。

在线阶段：模型与冷启动用户联系，询问关于N个item的问题。

域：餐厅推荐

这里使用Ap@k这种标准来评估结果，评价推荐的准确性的。

Ap@k的值在0~1之间，其越接近于1，说明效果越好。

16.试验步骤

在搜索引擎上的搜索数据来得出离线嵌入：

搜索14年12月26号到15年4月26号的搜索记录和单击日志：3549个cookies，其中有289个餐厅

该数据来自 餐厅点评的网站 中评论剑桥的餐厅页面中的数据。过滤下来 来自搜索引擎查询和单击日志，来查找某一个个人PC访问这些餐厅页面的cookie。每一个cookie被视为不同的用户，餐厅评论页面上的所有访问都被视为用户喜欢的餐厅。

从2015年3月26日至4月26日期间，我们确定了3,549个Cookie（用户）访问了评估服务提供商确定的512个不同剑桥餐厅评论页面中的至少一个。记录了512间剑桥餐厅的289间独特餐厅的互动。用户 - 餐厅互动总数为9330。

离线数据由M = 3549个用户，N = 289个餐厅和9330个正例组成。

用户调研作为在线评估的依据

在进行的用户调研中，每个参与者在英国剑桥的餐厅中填写了一份关于他们偏好的匿名调查问卷。“你会考虑下星期五晚上在餐厅X就餐吗？”使用是/否作为标记。

精选10间餐厅，找到28个人进行问卷。

获取真实值

对每一个冷启动用户：

（1）随机抽取28名参与者之一

（2）观察被选中的参与者的调查问卷中关于询问的10家餐厅的标签

（3）推测用户的特征加入到先验嵌入中。

（4）设定用户的先验为刚刚取样的用户的值

（5）推测r\_i的分布

17.相对问题的那种方法更好？

如图所示

18.绝对问题还是相对问题更好？

绝对问题好

19.离线初始化有用吗？

有用啊！

20.哪一个问题选择策略最牛逼？

基于bandit的选择策略最牛逼

21. 结论

推荐系统与新用户交流而获得新用户的偏好。

完全的在线学习方法应用在推荐领域——使用绝对和相对反馈。

提出了各种各样的问题选择策略

绝对问题能产生最佳性能。

通过相对反馈是可以高效学习的

离线初始化提高了性能

基于bandit 的问题选择策略是非常有效的。