

推荐系统从零单排系列(七)--Airbnb实时个性化推荐之Embedding真的好用吗？

原创 可爱又迷人的反派角色宁宁 机器学习荐货情报局 2019-06-19



李宁宁 NingLee

读完需要

19

分钟

速读仅需7分钟

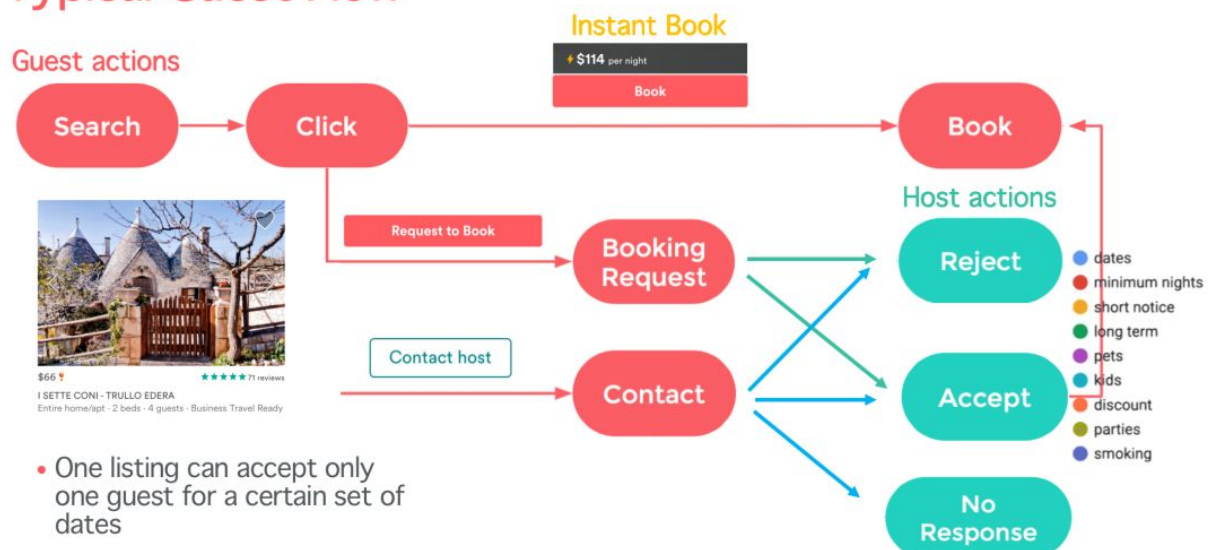
【导读】是的，Embedding真的很好用！Airbnb论文《Real-time Personalization using Embeddings for Search Ranking at Airbnb》拿下了KDD 2018年的best paper。文章干货满满，无论是理论还是工程都有很多值得借鉴的地方。好了，跟着宁宁一起来看看到底怎么回事吧~

1

Airbnb业务分析

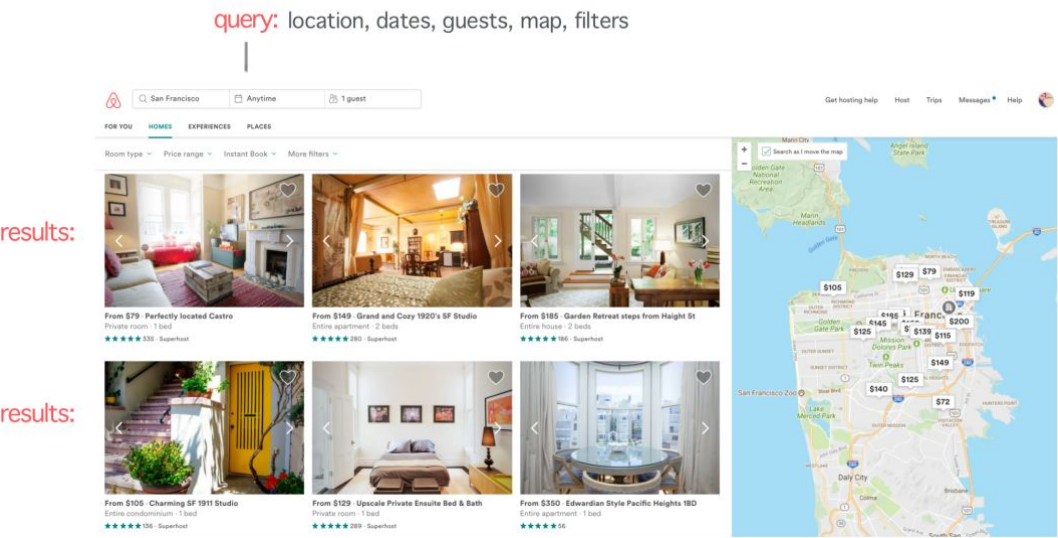
作为全球最大的短租平台，每天都有上百万的用户通过Airbnb找到自己心仪的短租屋。一个典型的用户操作流程如下：

Typical Guest Flow



用户输入地点、日期、人数进行搜索，系统会根据你的需求推荐你可能会感兴趣的房源，这就是Airbnb的第一大业务：搜索推荐。

Search At Airbnb



浏览滑动选择一个你感兴趣的，点击查看，新页面中除了该短租屋具体信息，在页面底部会有一个更多房源，这就是Airbnb的第二大推荐业务场景：相似房源推荐。



与其他公司业务不同的地方在于，host可以拒绝user的预定。论文就是从这两大类业务场景出发，通过Embedding技术来提高推荐质量。具体包括：

- 使用Embedding来建模用户long-term偏好，提高搜索推荐的质量
- 使用Embedding来学习listing的低维表达，提高相似房源推荐的质量

2

Listing Embedding

Listing Embedding是用来做 Similar Listing Recommendation 。用户点击了搜索页面中的某个短租屋，查看信息之后在页面底部系统会推荐跟点击房源类似的其他短租屋，这就是相似房源推荐。和电商领域不同，Airbnb的短租业务主要面临的挑战如下：

1. 用户访问非常少，绝大部分用户每年也才旅行2-3次
2. 用户每次访问时，偏好都会有所不同
3. 用户几乎不会去同一个地方两次其实这几个问题是非常难的，先别急着往下看，仔细思考一下如果让你来做，你有什么好的idea吗？你的用户一年才用一两次你的系统，而且每次访问偏好都会不同，而且之前去过的地方他基本都不会再去。

爱彼迎的算法工程师想出的idea是 Real Time Personalization 或者叫 In-session personalization 。首先，记住你最近点击过的 listings；然后，根据你 liked、wishlistd、long clicked、contacted的listing，推荐给你相似的listing。那如何判断哪些listing是相似的那，就是利用 Listing Embedding 啦。文中的listing就是指一个短租屋，既然人家这么叫，咱也只能跟着叫了。（其实直接说是item会不会更直观一些）

Listing Embeddings

How to represent a listing?

Listing

Casa Vacanze in Rome

sparse

entire home
rome
2 guests
cheap

1	1	0	1	0	...	0	1
---	---	---	---	---	-----	---	---

dense

0.9	2.6	3.1	0.1	2.2
-----	-----	-----	-----	-----

embedding

- Represent listings as numeric vectors
- Vectors need to be learned using training data (search sessions)
- We want listings with similar contexts to have similar vectors
- Popular tool for training: word2vec

Listing Embedding的学习使用的是经典的Word2Vec中Skip-gram Model，并没有特别高深的技巧，但是也根据实际的业务做了一些改进。对Word2Vec不熟悉的同学就不要往下看了，先去把基础知识复习一下吧。参考公众号中的三篇Word2Vec文章。

微软在2016年提出的Item2Vec，参考《Item2Vec-Neural Item Embedding for Collaborative Filtering》，大大扩展了Word2Vec适用的范围：凡是能组成sequence的东西都是可以利用Word2Vec来学习其Embedding表示的。Word之间的时序关系在某些场景下即使忽略也照样能带来不错的效果。Airbnb利用了这一思想，将用户过去一个月内的Click按照时间顺序进行排列组成 Click Session，并过滤掉其中停留时间小于30s的click，以过滤噪声。如果两次Click之间时间超过30分钟，则此Session结束，后面的

Click重新组合Session，这样来保证每个 Click Session的连续性，Session中的listing应该是相似的。Airbnb整理出了8亿条Click Session用于训练。

Actions by single user (listing clicks, inquiries, bookings) ordered in time

S1 748612 4160766

S2 5473823 2582727 5473823 2582727 5473823 5473823

S3 6251934 9257649

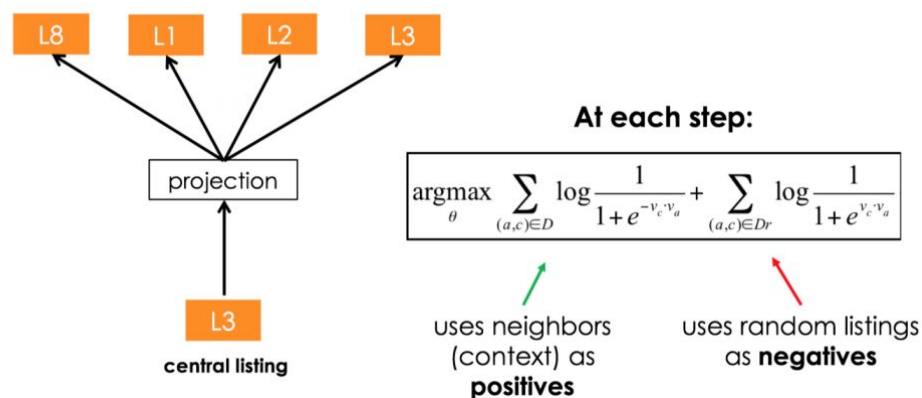
S4 7924193 10116733 8998529 9075420 4113166

S5 5503478 4986017 5503478 5503478 5879904 4140396

S6 10237904 8680483 8718513 11691507 4831342 8004575 7866901

有了Click Session就相当于有了sentence，每一个listing就是一个word。原始的使用Negative Sampling Skip-gram模型的objective如下：

Skip-gram model (word2vec)



然后，Airbnb在此基础上，针对objective进行了两项优化：

1. 优化一：Booked Listing as Global Context
2. 优化二：Adapting Training for Congregated Search

2.1

Booked Listing as Global Context

仔细研究发现click session可以分为两类。一类是用户click，最后book（预定）了短租房，称为booked session；一类是用户只是看看，click的最后并没有预定，只是上来看

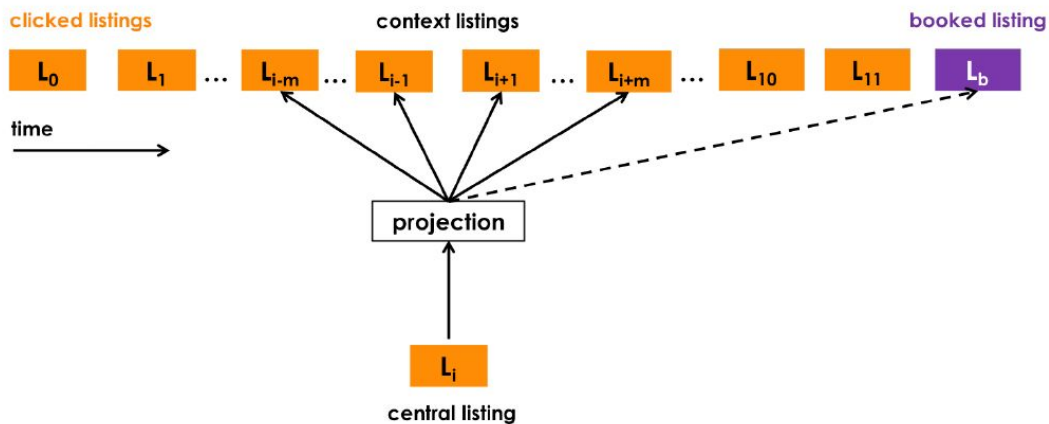
看，称为exploratory session。预定Booked是一个很强的信号，毕竟是真金白银，用户在Booked之前，肯定会click多个非常相似的listing进行对比，查看。

所以我们将Booked Listing作为一个正样本，作为Global Context加入到模型训练中。也就是说无论当前训练中心词周围有没有Booked Listing，我们都认为Booked Listing在它的上下文范围中。相应的objective就修改为如下形式：

$$\operatorname{argmax}_{\theta} \sum_{(l,c) \in \mathcal{D}_p} \log \frac{1}{1 + e^{-v'_c v_l}} + \sum_{(l,c) \in \mathcal{D}_n} \log \frac{1}{1 + e^{v'_c v_l}} + \log \frac{1}{1 + e^{-v'_l v_{l_b}}}$$

\mathcal{D}_p 表示当前center listing的context正样本； \mathcal{D}_n 表示Negative Sampling采样得到的负样本；最后 l_b 表示最后Booked Listing。Objective非常直观：针对当前的center listing, l ，作为输入，我们期望其上下文中listing \mathcal{D}_p ，以及最终Booked Listing, l_b 出现的概率尽可能的高；希望不是上下文中的listing, \mathcal{D}_n 出现的概率尽可能的低。注意上面的负号，为什么正样本前面有负号，负样本前面没有负号那，留给同学们思考？提示一下： $1 - \text{Sigmoid}(x) = \text{Sigmoid}(-x)$ 。

相应的一次模型训练step示意图如下：



第一项优化就讲完了，理解起来有困难的同学请回去继续翻Skip-gram的论文。

2.2

Adapting Training for Congregated Search

第二项优化是针对objective中的负样本选取的改进。Airbnb的用户在搜索时多半是固定一个market的，比如用户搜索一个固定的地点成都，那负样本也应该是同一个market成都的。但是listing上百万，Negative Sampling采样得到的大部分listing都满足条件。改进也很简单，在objective中加入对同一个market的负样本采样：

$$\begin{aligned} \operatorname{argmax}_{\theta} \sum_{(l,c) \in \mathcal{D}_p} \log \frac{1}{1 + e^{-v'_c v_l}} + \sum_{(l,c) \in \mathcal{D}_n} \log \frac{1}{1 + e^{v'_c v_l}} \\ + \log \frac{1}{1 + e^{-v'_l v_l}} + \sum_{(l,m_n) \in \mathcal{D}_{m_n}} \log \frac{1}{1 + e^{v'_m v_l}}. \end{aligned}$$

其中 \mathcal{D}_{m_n} 就是同一个market中采样的负样本。注意只有 l_b 的前面是没有求和号的，因为每个训练Step都只有一个Booked Listing。

2.3

Cold start listing embeddings

这对新的listing冷启动问题，Airbnb给出的解决方案非常简单：找到其方圆10英里之内的3个最相似的listing，然后对其Listing Embedding取平均即可。论文提到，利用这一简单的方法，可以解决98%的新listing冷启动问题。

2.4

效果如何

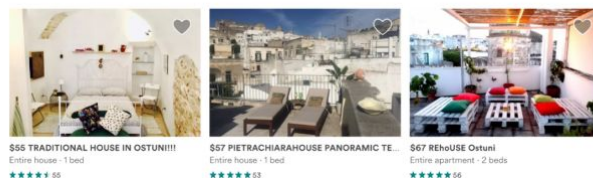
现在有了Listing Embedding，那么根据简单的余弦相似度就可以计算出listing之间的相似度。然后就可以找到跟用户当前点击的listing最相似的Top K个listing进行推荐了。看看下面的对比试验，你就能感受到效果的明显提升：

Applications

Similar Listings



Similar listings



Before

Similar Listings



After

没有利用图片信息，却找到了房屋类型结构非常相似的短租屋，可以看到在用户的click session中真的隐藏着非常多的有用信息。但是只有通过仔细分析，用心深入思考才能发现

这些信息。除此之外，Airbnb还做了一个用来验证效果的工具，输入listing id，系统给出最相似的10个listing。效果看起来还不错：

Embedding Evaluation Tool

Search

Query Type


Listing ID

Listing ID

🏠 13253

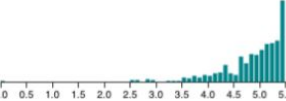
Search

I'm Feeling Lucky



\$268 Live in an historic English Castle!
★★★★★61 reviews
Location: Cumbria, United Kingdom
Description: Choose from 15 grand rooms to call home with delicious breakfast in our fabulous relaxed castle in glorious Northern England which is our family home....

Score Histogram




*Includes scores for up to the 500 nearest listings


Other options

Number of listings


Nearest listings (10)




\$268 Live in an historic English Castle!
★★★★★61 reviews
KNN: /admin/embedding_evaluation/13253
Score: 0.00
Location: Cumbria, United Kingdom
Description: Choose from 15 grand rooms to call home with delicious breakfast in our fabulous relaxed castle in glorious Northern England which is our family home....




\$736 Dairsie Castle (historic Scotland)
★★★★★11 reviews
KNN: /admin/embedding_evaluation/330808
Score: 2.51
Location: Fife, United Kingdom
Description:




\$8434 A whole castle to call your own
KNN: /admin/embedding_evaluation/9651657
Score: 2.59
Location: Cumbria, United Kingdom
Description: Hire our family castle exclusively for your own party. We will look after you as our house guests and cater for all your meals. The castle sits in fif...



\$268 A four poster bed in a castle
2 reviews
KNN: /admin/embedding_evaluation/9448372
Score: 2.62
Location: Cumbria, United Kingdom
Description: Sleep in a grand bedroom and wake to a delicious breakfast in our fabulously relaxed yet wonderful 15 bedroom small country house which is also our fa...



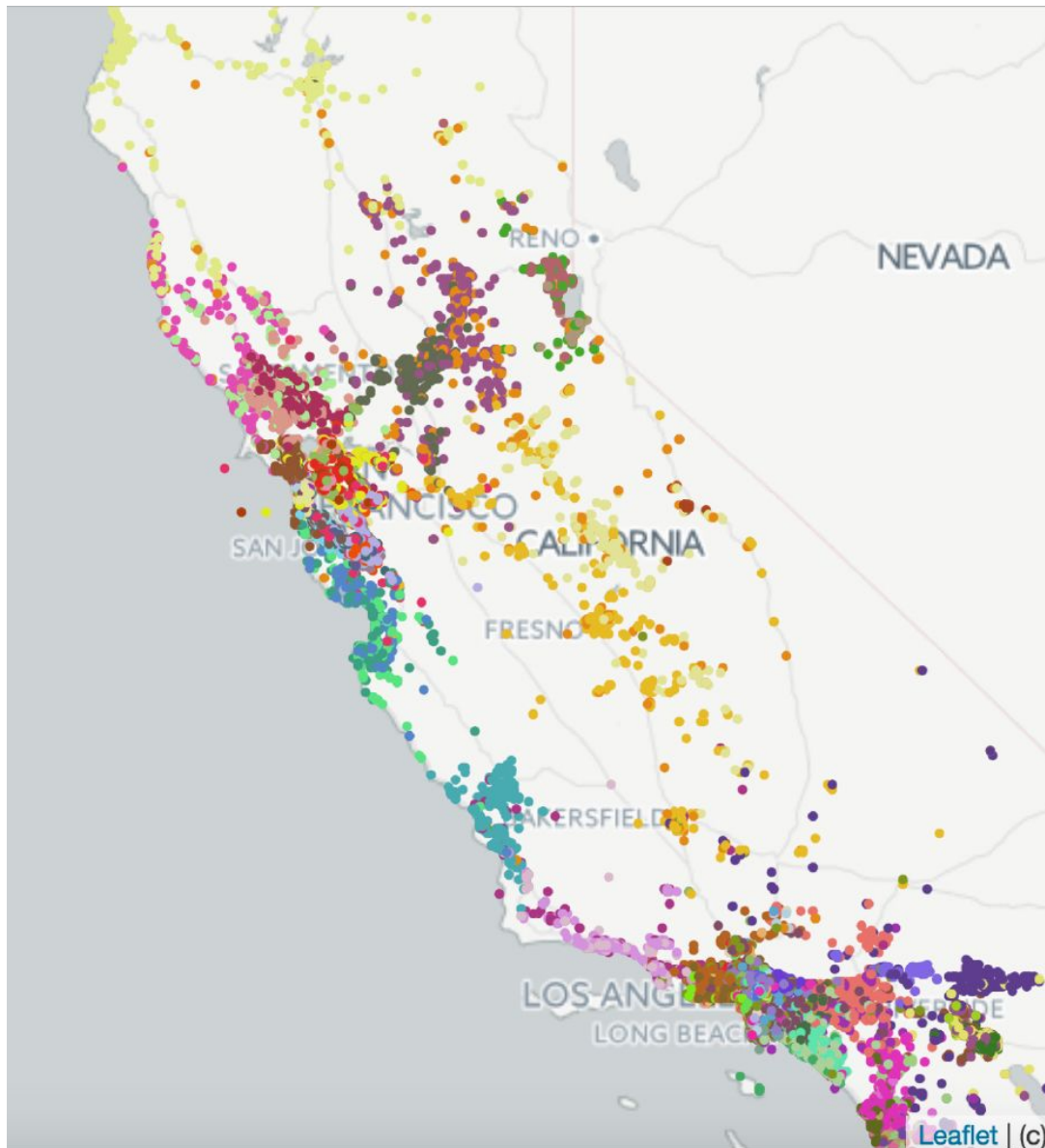
\$133 The Gatehouse to Ayton Castle
★★★★★167 reviews



\$344 Stay in the Historic Craster Tower
★★★★★34 reviews

题外话：看到中间那个城堡了么，8434美元！不知道你有什么感受，我是流下了来自底层贫穷人民的眼泪。哎，左手温暖下右手，擦擦眼泪活下去。

还可以根据Listing Embeddings结果对短租屋进行聚类，并指导优化运营策略，如下图所示：



2.5

小结

我们利用click session训练得到了listing embedding。在用户点击某个listing后，可以利用该embedding在页面底部推荐相似listing。那么还剩一个问题：Search Recommendation。用户输入搜索query，系统给出listing推荐列表。

3

User-type & Listing-type Embeddings

Listing Embeddings使用Click Session进行训练，擅长寻找相似的listing，适合short-term, in-session, same market的个性化推荐，目的是在用户当前的点击中，给出和点击listing相似的推荐。但是，除了in-session的个性化，用户long-term的兴趣偏好也需要考虑。比如用户当前session搜索的是洛杉矶的短租屋，但是用户之前在纽约、伦敦住过的短租屋也代表着用户的兴趣偏好。

为了捕捉用户long-term偏好，仿照上面的做法，我们可以使用Booked Listing组成Booked Session，然后训练一个User Embedding，来对用户偏好建模，是不是就可以了那？答案是否定的。主要原因如下：

1. Booking session数量相比于click session少太多了，embedding的训练需要大量的数据
2. 很多用户只预定过一次短租屋，booking session的长度只有1，这也是没有办法学习的
3. 用户的两次booking之间，可能会经历很长时间，在这期间用户的偏好已经发生变化

Booked listing实在是太稀疏了，这么多的用户，非常少的购买行为，典型的高维度稀疏问题。如果是你，你会怎么做那？

3.1

聚合user_id, listing_id成为user_type, listing_type

Airbnb的方案是，对user和listing按照规则进行聚合，组成user_type和listing_type。然后根据user_type和listing_type组合成Booked session进行训练。聚合规则也非常简单，先根据listing属性值进行分桶，然后对桶进行组合，如下图所示：

Table 3: Mappings of listing meta data to listing type buckets

Buckets	1	2	3	4	5	6	7	8
Country	US	CA	GB	FR	MX	AU	ES	...
Listing Type	Ent	Priv	Share					
\$ per Night	<40	40-55	56-69	70-83	84-100	101-129	130-189	190+
\$ per Guest	<21	21-27	28-34	35-42	43-52	53-75	76+	
Num Reviews	0	1	2-5	6-10	11-35	35+		
Listing 5 Star %	0-40	41-60	61-90	90+				
Capacity	1	2	3	4	5	6+		
Num Beds	1	2	3	4+				
Num Bedrooms	0	1	2	3	4+			
Num Bathroom	0	1	2	3+				
New Guest Acc %	<60	61-90	>91					

举个例子，一个在US的listing：Listing Type是Priv，对应Buckets id是2；\$ per Night是56-69，对应Buckets id是3；\$per Guest是<21，对应Buckets id是1；那么listing_type=US_lt2_pn3_pg1。user type也是类似的，不再详述。

3.2

训练

用户的兴趣偏好可能会随着时间而改变，为此user_type和listing_type在同一个vector space中进行训练。这里我看了很久没看懂，用户的兴趣发生变化和在不在同一个向量空间训练有什么关系那？欢迎各位同学来找我讨论、发表意见，适当的讨论不仅可以解惑还可以帮助我们思考。

为了做到这一点，作者将user_type和listing_type组合起来，一起构成Booked Session。如下：

$$s_b = (u_{type_1}l_{type_1}, \dots, u_{type_M}l_{type_M}) \in S_b$$

一个Session中每一项都是一个Booking event，按照时间排列的(user_type, listing_type)的tuple组。每一个session是由同一个user_id的购买记录生成的，因为user_type会发生变化，所以Utype1和Utypen不一定是一样的。

这里也没看懂，原来主要问题是booking session太稀疏，很多user只有一次、两次的booking记录。现在依旧按照user_id来组合session的话，那该user的booking记录还是这么少，不是依旧存在这个问题吗？难道不是应该按照user_type来组合session吗？从user_type的维度来看，booking记录就非常多了。同样，希望看懂的同学加我交流、讨论，非常欢迎~

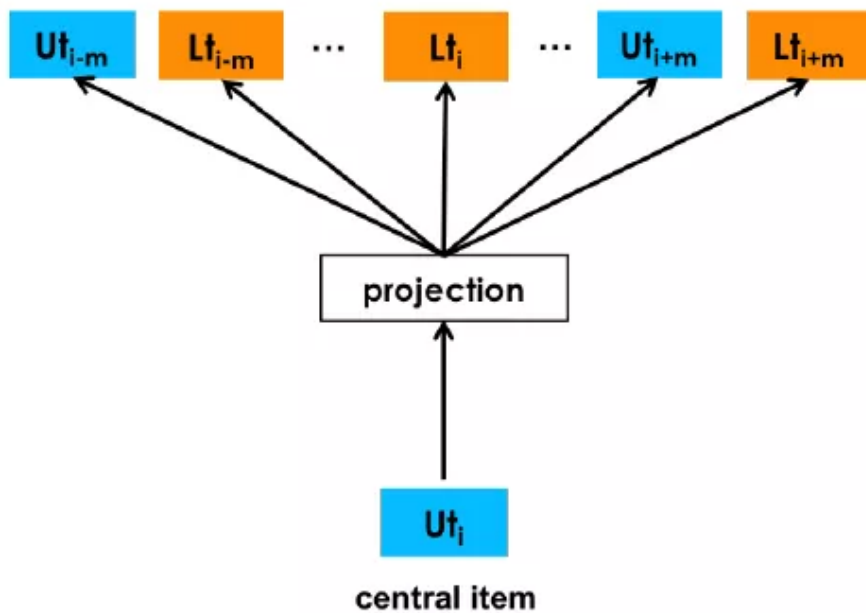
继续回到之前的问题来，为什么说将user_type,listing_type组成tuple就可以在同一个vector space进行训练了那？文中并没有细说，根据objective可以猜个八九不离十。如果当前中心词是user_type，那么objective是：

$$\operatorname{argmax}_{\theta} \sum_{(u_t, c) \in \mathcal{D}_{book}} \log \frac{1}{1 + e^{-v'_c v_{u_t}}} + \sum_{(u_t, c) \in \mathcal{D}_{neg}} \log \frac{1}{1 + e^{v'_c v_{u_t}}},$$

如果当前中心词是listing_type，那么objective是：

$$\operatorname{argmax}_{\theta} \sum_{(l_t, c) \in \mathcal{D}_{book}} \log \frac{1}{1 + e^{-v'_c v_{l_t}}} + \sum_{(l_t, c) \in \mathcal{D}_{neg}} \log \frac{1}{1 + e^{v'_c v_{l_t}}}.$$

可以看到几乎是一样的，其实就是将user_type和listing_type展平，context中不在进行区分，统一都认为是上下文。就相当于把user_type和listing_type当成一个字典中的不同单词，按照tuple+时间的顺序排列成了sentence然后进行学习。学完了之后再把词典分成两半，这边是user_type,那边是listing_type。一个step的模型训练图如下：



3.3

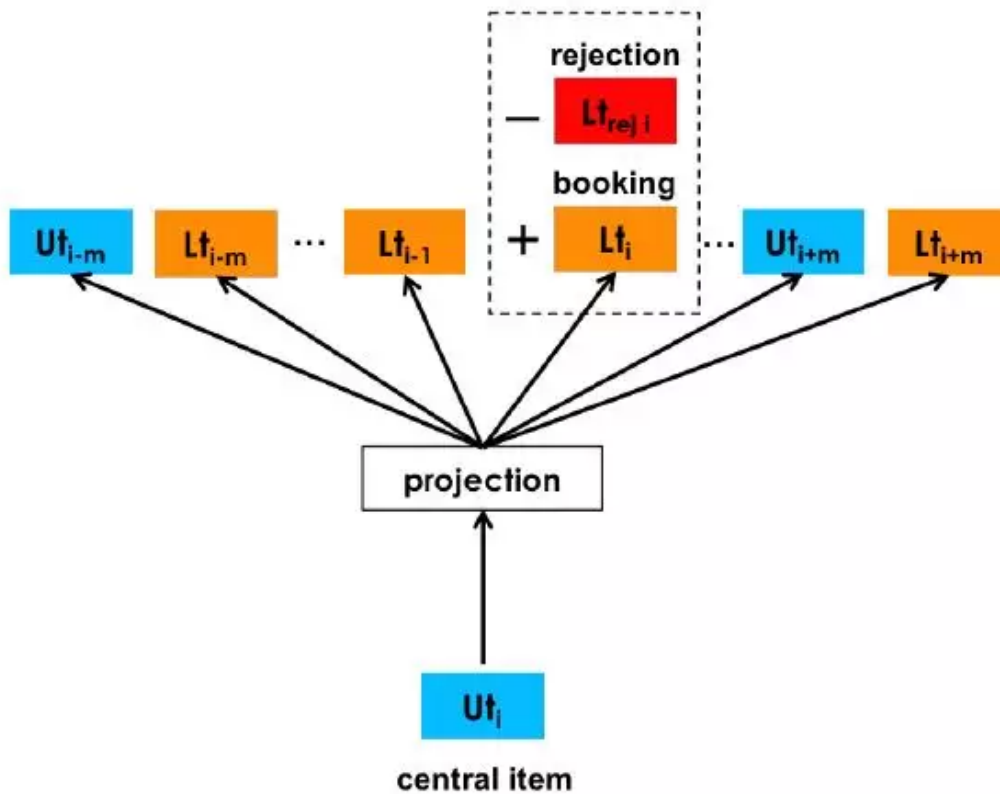
Explicit Negatives for Host Rejections

Airbnb业务特殊之处在于，user选定短租屋之后，host是可以拒绝的。可能是觉得user的过往评价不好等等之类的，是一个双向的选择过程。

那么在推荐的时候，我们需要尽可能的降低user被host拒绝的概率。所以我们明确的在模型负样本中增加user_id之前被拒绝的那部分listing，告诉模型不要给user推荐这种类型的listing，因为会被host reject掉。修改之后的User Type训练 objective 如下，Dreject代表该user之前被拒绝的listing集合。listing的类似，不在详述。

$$\begin{aligned} \operatorname{argmax}_{\theta} \quad & \sum_{(u_t, c) \in \mathcal{D}_{book}} \log \frac{1}{1 + \exp^{-v'_c v_{ut}}} + \sum_{(u_t, c) \in \mathcal{D}_{neg}} \log \frac{1}{1 + \exp^{v'_c v_{ut}}} \\ & + \sum_{(u_t, l_t) \in \mathcal{D}_{reject}} \log \frac{1}{1 + \exp^{v'_{l_t} v_{ut}}} . \end{aligned} \quad (8)$$

对应的一个step的训练网络图如下所示，在原来的基础上增加了rejection部分。



4

工程应用：Real time personalization in Search Ranking using Embeddings

接下来就是如何在Search Ranking Model中使用新特征了。我们关注Rank过程，认为召回已经是完成了的。根据前面得到的listing embeddings和user type、listing type embeddings，新增加的特征如下。可以看到并不是很多：

Table 6: Embedding Features for Search Ranking

Feature Name	Description
EmbClickSim	similarity to clicked listings in H_c
EmbSkipSim	similarity to skipped listings H_s
EmbLongClickSim	similarity to long clicked listings H_{lc}
EmbWishlistSim	similarity to wishlisted listings H_w
EmbInqSim	similarity to contacted listings H_i
EmbBookSim	similarity to booked listing H_b
EmbLastLongClickSim	similarity to last long clicked listing
UserTypeListingTypeSim	user type and listing type similarity

假设现在是用户user在搜索，Search Ranking Model已经有了召回集，针对召回集中的候选listing li 我们想得到其Rank。

4.1

Listing Embedding Features

这部分都是short-term特征。收集用户过去 两周 的历史数据，并 实时更新 。如下图所示：

- (1) H_c : **clicked listing_ids** - listings that user clicked on in last 2 weeks.
- (2) H_{lc} : **long-clicked listing_ids** - listing that user clicked and stayed on the listing page for longer than 60 sec.
- (3) H_s : **skipped listing_ids** - listings that user skipped in favor of a click on a lower positioned listing
- (4) H_w : **wishlisted listing_ids** - listings that user added to a wishlist in last 2 weeks.
- (5) H_i : **inquired listing_ids** - listings that user contacted in last 2 weeks but did not book.
- (6) H_b : **booked listing_ids** - listings that user booked in last 2 weeks.

以EmbClickSim为例说明，针对候选listing l_i 所在market将 H 分为两部分，一部分在同一个market中，一部分不在该market中。然后分别对 H 中的listing embeddings向量取平均。候选listing l_i 的EmbClickSim取这两个里面较大的一个：

$$EmbClickSim(l_i, H_c) = \max_{m \in M} \cos(\mathbf{v}_{l_i}, \sum_{l_h \in m, l_h \in H_c} \mathbf{v}_{l_h}),$$

前6个特征都是类似的，不在详述。

除了用到用户过去一段时间内的某个Action的所有embedding来计算特征之外，还单独用了一个上一次LongClick的Sim，也是用的余弦距离，不再详述。

4.2

User-type & Listing-type Embedding Features

前面是 short-term 特征，针对 long-term 特征，只计算了一个 UserTypeListingTypeSim，将当前user_id转换成user_type, 候选listing_id转换成listing_type，利用余弦计算出一个相似度。特征重要度丢出来大家看一下：

Table 7: Embedding Features Coverage and Importances

Feature Name	Coverage	Feature Importance
EmbClickSim	76.16%	5/104
EmbSkipSim	78.64%	8/104
EmbLongClickSim	51.05%	20/104
EmbWishlistSim	36.50%	47/104
EmbInqSim	20.61%	12/104
EmbBookSim	8.06%	46/104
EmbLastLongClickSim	48.28%	11/104
UserTypeListingTypeSim	86.11%	22/104

线上的效果：

Table 8: Offline Experiment Results

Metrics	Percentage Lift
DCU -0.4 (rejections)	+0.31%
DCU 0.01 (clicks)	+1.48%
DCU 0.25 (contacts)	+1.95%
DCU 1 (bookings)	+2.58%
NDCU	+2.27%

5

总结

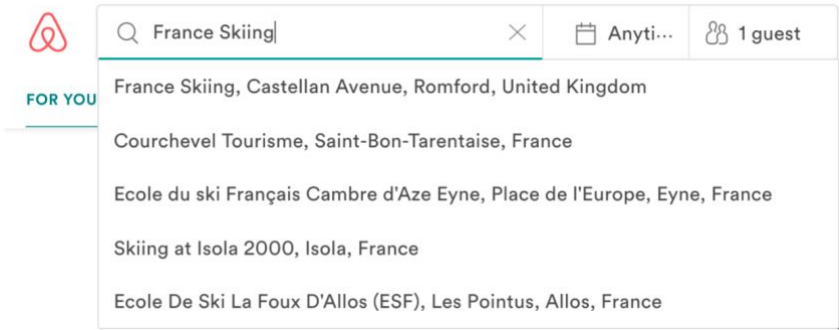
5.1

论文总结

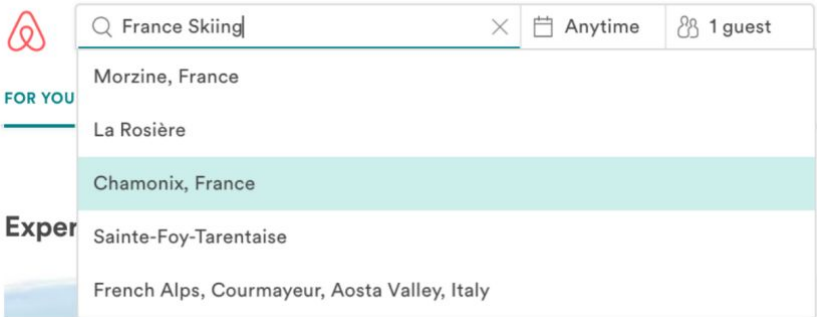
Airbnb利用click session学习了listing embedding，捕获short-term特征，很好的提升了Similar Listing推荐效果；利用booked session捕获long-term特征，并利用聚合user_id listing_id很好的解决了高维度稀疏问题，非常值得借鉴；最后利用学习到的两种embedding，反过来构造新特征，提升了Search Ranking Model的性能。

还有一个Query Embeddings in Search论文中没有提到，通过对Query进行embedding，不再仅仅是简单的匹配规则，而是学习到了更深层次的语义。感兴趣的见参考的slide。

Before



After



5.2

个人思考

简单之处，更见功力。

总体来说，我认为Airbnb的算法工程师提出的方案是非常优雅的，把很基础的技术根据业务进行优化；无论是训练集的生成，特征的选取都比较适当。没有用非常非常大的训练集、也没有恐怖到上亿维的特征，以一个相对简单、精巧的方式改进了模型性能，提升了业务体验。但是这样看似简单的改进，工作其实是需要非常深厚的功力的，背后一定是经过深入思考、不断尝试、不断迭代才能带来性能的微小提升。

相比之下，国内好多公司全量数据+全量特征，上亿的数据上亿的特征维度，然后就是神经网络一把梭！这样的开发模式可以说是非常的快糙猛了。正所谓是梭哈一时爽，优化火葬场。不经过深入的思考，结果只能是被短期kpi绑架。希望我们可以静下心来，追根溯源，深入思考。与诸君共勉~

6