Joint Deep Modeling of Users and Items Using Reviews for Recommendation

Github 前阵子出了个 Repository Recommendation 的功能，偶然看到这篇文章及其扩展的一个开源实现，本周阅读这篇基础文章，下一阶段再精读改进模型的文章。

Accepted by WSDM'2017，论文链接：https://arxiv.org/abs/1701.04783

本文摘要

用户的评论中通常蕴含大量有用的信息，而这些信息在推荐系统中的使用程度很低，但却是一定程度上能够缓解冷启动问题，提升推荐性能的一部分。本文提出了一个深度模型，从评论中联合学习物品的属性和用户的行为。本文提出的 DeepCoNN 模型，有两个平行的神经网络，其中一个对于用户所写的评论分别学习用户的行为数据，另一个则是对物品的所有评论中习得物品的属性信息，最后通过一个“共享层”把这两部分网络特征融合在一起。从而，用户和物品的各自的表示能够以一种类似于分解模型的方式交互。在多个实验数据中，DeepCoNN 均打败了其他方法和模型。

本文简介

众所周知，推荐系统能够为用户“万里挑一”，推荐出最感兴趣的内容。如何推荐才能让用户更为满意，是所有网络平台都关注的问题，这里面涉及到的因素非常多：用户的历史记录、用户个人的喜好、用户当前需求、地理位置、生活环境等。许多主流的做法都是协同过滤及其衍生的模型，而又以分解模型最为典型，通过计算和交互用户、物品的隐藏特征进行推荐预测。尽管如此，实际中还是有很多困境，最大的一个就是冷启动问题，通常由数据过于稀疏导致，记录比较少的或者是新的用户，很难或者根本没有办法学习一个用户的表示。换句话说，整个系统内，没有十足的把握去推荐那些很少被打分的物品；对于那些打分记录少的用户，也很难对他们进行推荐。

一个比较好的解决方法就是引入其他外部辅助信息，如评论信息。评论信息往往会解释用户为什么给出好评或差评的理由，但是现有的协同过滤框架很难把这部分信息添加进去，因为协同过滤模型仅仅考虑用户的打分信息，即数值信息。相关工作表明，引入评论信息，能够提高推荐系统的打分预测准确性，尤其是那些冷启动用户和物品。

本文提出了一种基于神经网络的深度模型，Deep **Co**operative **N**eural **N**etworks（DeepCoNN），使用评论信息，联合对用户和物品建模，提供一个更好的打分预测性能。DeepCoNN 使用了两个对称的神经网络联合学习用户和物品的隐藏表示，从而最大化打分预测的准确性。一个网络用来从用户写过的所有评论中获取用户的表示，另一个网络从物品相关的所有评论中学习物品的表示。与矩阵分解技术类似，得到这两个表示后，后接一个全连接层进行特征交互，得到一个预测打分。本文的贡献总结为以下三点：

- 首次提出了利用神经网络的方式构建出 DeepCoNN 模型，在推荐系统中融合评论信息，联合学习用户和物品的表示。从评论中学到的用户表示，最终的优化目标是最小化推荐系统的打分误差，与传统的直接融合评论信息的做法，有一个更为明确的任务导向；
- 评论文本使用预训练的模型，以 word-embedding 方式作文本表示，可以保留文本的语义信息，比 BOW、LDA 等都更有竞争力；
- DeepCoNN 不仅缓解了冷启动的问题，对于一些热启动的用户也有性能的提升。

模型

DeepCoNN 模型分析评论信息，利用对称的深度网络，联合建模，获得用户和物品的隐藏表示，从而预测用户的打分。

模型结构

如下图所示，左边是用户网络 Net_u，右边是物品网络 Net_i，这两个对称网络的输入分别是和用户、物品相关的评论。

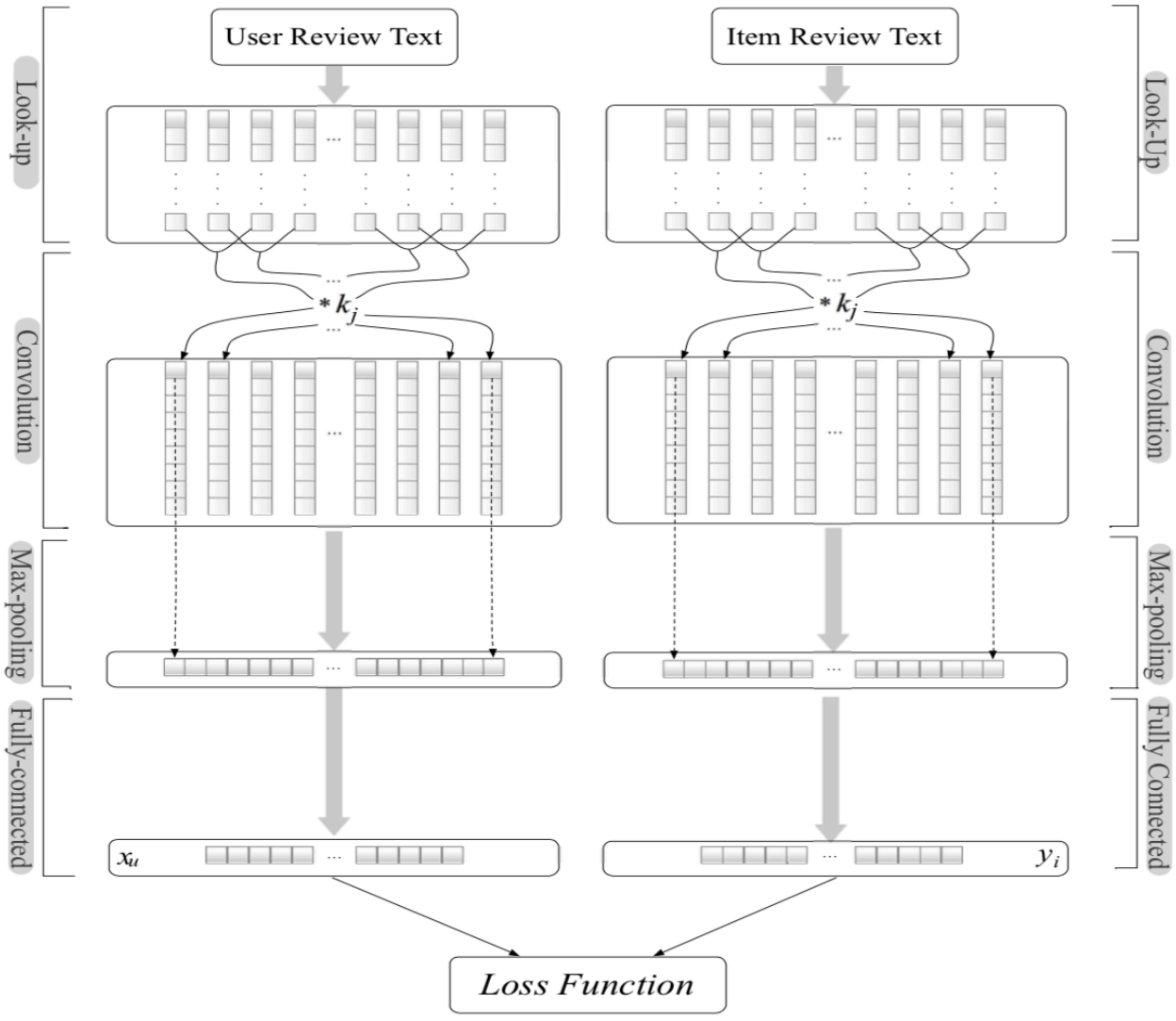


Figure 1: The architecture of the proposed model

- 第一层是 look-up 层，把用户、物品的相关评论通过 word-embedding，转换成一个 embedding 的矩阵，具体做法是利用 word2vec 等相关模型得到每个词的分布式表示，接着就通过拼接（stack）的方式构成长度为 n 的矩阵，用 $V_{1:n}$ 表示；
- 第二层是 CNN 网络层，使用经典的卷积神经网络结构和 ReLU 激活函数，获得多个卷积核的特征输出：卷积核为 K_j，偏置为 b_j，则对于卷积核 K_j 的输出即为 $z_j = ReLU(V_{1:n} * K_j + b_j)$
- 第三层为 max-pooling 操作，得到一个固定的长度的向量表示 $O = \{o_1, o_2, \cdots, o_{n_1}\}$，这里的 n_1 表示卷积核的总数；
- 最后一层是全连接层，以用户表示为例，$x_u = f(W \times O + g)$，从 n_1 维的输入映射到 n_2 维空间，便于和商品表示 y_i 交互。

共享层

虽然最终得到的 x_u 和 y_i 可以直接作为两者的特征表示，但是他们通常是属于不同向量空间的特征表示，因此引入一个共享层来完成映射。把 x_u 和 y_i 拼接起来，得到 $\hat{z} = (x_u, y_i)$，利用一个分解机模型来预测最终的得分：

$$J = \hat{w}_0 + \sum_{i=1}^{|\hat{z}|} \hat{w}_i \hat{z}_i + \sum_{i=1}^{|\hat{z}|} \sum_{j=i+1}^{|\hat{z}|} \langle \hat{v}_i, \hat{v}_j \rangle \hat{z}_i \hat{z}_j$$

这里的 J 一部分由 $\hat{W}^T \hat{z}$ 给出，这是一阶交互，强度由 $\hat{w}_*$ 控制，二阶交互则是由 $\hat{v}_*$ 控制 $\hat{z}$ 的每个特征维度。具体这里的 $\hat{v}_*$ 是什么含义，文中没有给出解释，两种猜测：一是正交向量组，二是模型超参，需要学习。

网络训练

根据每一个 batch 的梯度方向优化参数，具体采用了 RMSprop 方式替代了传统梯度下降方法。此外，为了防止过拟合，加入了 dropout。

模型分析

第一，相较于 BoW 方式处理评论，保持了原评论中的语序信息，先表示成一个矩阵，后通过 CNN 得到特征表示，是保留了语序信息的。（问题：why not RNN？）

第二，推荐系统注重“在线学习”，应尽可能支持。神经网络的状态可以被保存和恢复，所以可以在新的数据产生以后，对模型各种参数进行微调。

实验

本文在三个数据集 Yelp、Amazon 和 Beer 上进行了实验，由于是 rating prediction 任务，所以采用了 MSE 作为评价指标，数据集的情况如下：

Table 2: **The Statistics of the datasets**

Class	#users	#items	#review	#words	#reviews per user	#words per review
Yelp	366,715	60,785	1,569,264	198M	4.3	126.41
Amazon	6,643,669	2,441,053	34,686,770	4.053B	5.2	116.67
Beer	40,213	110,419	2,924,127	154M	72.7	52.67

基线实验设置

为了全面评价 DeepCoNN 的性能，基线模型可以分为三种类型：

- 只使用打分信息的，如矩阵分解 MF 和 PMF，以验证融合评论信息的作用；
- 引入主题模型的方法，如 LDA、CTR、HFT
- 深度学习模型，CDL

大多数模型都在验证集上采用 Grid Search 方法确定了超参。本文的 DeepCoNN 模型使用了预训练的 Word2Vec 模型得到每个词的表示。

实验结果

Table 3: **MSE Comparison with baselines. Best results are indicated in bold.**

Dataset	MF	PMF	LDA	CTR	HFT-10	HFT-50	CDL	DeepCoNN	Improvement of DeepCoNN (%)
Yelp	1.792	1.783	1.788	1.612	1.583	1.587	1.574	**1.441**	8.5%
Amazon	1.471	1.460	1.459	1.418	1.378	1.383	1.372	**1.268**	7.6%
Beer	0.612	0.527	0.306	0.305	0.303	0.302	0.299	**0.273**	8.7%
Average on all datasets	1.292	1.256	1.184	1.112	1.088	1.09	1.081	**0.994**	8.3%

首先，PMF 尽管比 MF 要优秀，但是大多数情况下相比融合评论文本的方法都要差一些，证明了评论文本的正向增益的假设。其次，LDA 模型是无监督的方法，缺少明确的训练指导，所以比其他的也要差一些。此外，CDL 模型比其他融合评论的做法都要更优，但是通过联合学习用户、物品的特征的 DeepCoNN 方法，还有提升的空间。

模型分析

为了回答

- 两个平行深度网络是否真的从评论文本中联合学习了?
- 对于评论文本从 word-embedding 到 CNN 提取特征是否保留了更好的语义信息?
- 最后的共享层起到了什么样的作用?

这三个问题，作者设计了一系列对比实验，分别是：

- DeepCoNN-User、DeepCoNN-Item 使用一个矩阵随机初始化的矩阵分别替代 Net_u 和 Net_i，以验证 CNN 处理的有效性
- DeepCoNN-TFIDF，使用 TF-IDF 作为 word-embedding 的替代；DeepCoNN-Random，不使用 word2vec，而是随机初始化每个词的特征向量，以验证 word-embedding 的有效性
- DeepCoNn-DP，不使用共享层，直接采用向量内积，以验证共享层的有效性

实验结果如下表所示：

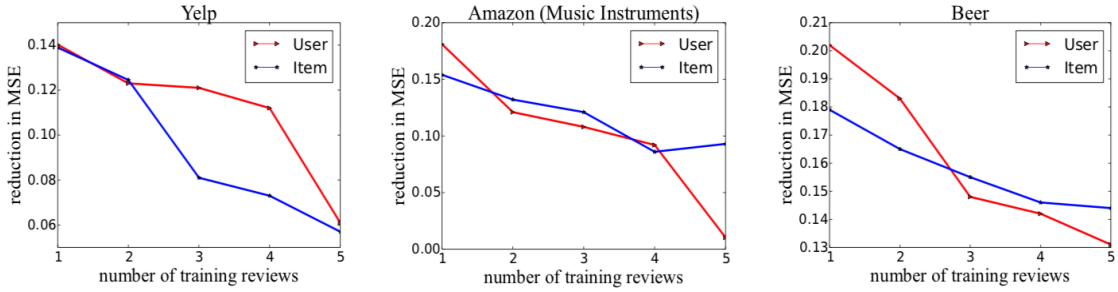
Table 4: **Comparing variants of the proposed model. Best results are indicated in bold.**

Model	Yelp	Amazon Music Instruments	Beer
DeepCoNN-User	1.577	1.373	0.292
DeepCoNN-Item	1.578	1.372	0.296
DeepCoNN-TFIDF	1.713	1.469	0.589
DeepCoNN-Random	1.799	1.517	0.627
DeepCoNN-DP	1.491	1.253	0.278
DeepCoNN	**1.441**	**1.233**	**0.273**

从对比试验的结果上，分别给予 DeepCoNN 肯定的回答。

数据集相关分析

由于冷启动是推荐系统中十分困难的问题，作者分别对冷启动的用户和物品进行了测试，下面的图给出了 DeepCoNN 相比于 MF 方法，对冷启动的用户和物品在 MSE 指标上降低的程度。



可见，越是冷的用户或物品，性能的提升越明显，即 MSE 降低的更多。

本文总结与点评

本文提出了 DeepCoNN 模型，在推荐系统任务中，引入了用户和商品相关的评论，充分考虑其中的语义信息，利用了两个平行深度网络，对两者的特征进行联合学习，并且加以用户打分这一监督信息，使得模型的预测打分更为准确，尤其是冷启动用户的改善效果更为明显。

DeepCoNN 的类似想法我也做过，当时我首先尝试了用 RNN、CNN 来学习评论的表示，但是没有联合学习，只是单纯地把这个评论和打分信息作为一个“情感分类”问题来学习这个评论的表示，然后把这个评论的信息通过拼接的方式，作为一个多层感知器的输入，作为辅助信息来帮助原始模型（NCF）预测打分。基本上没有效果，大概是因为融合方式过于粗暴，没有联合模型那种互相促进的作用。

我认为 MF 这样的经典模型还是很有用的，DeepCoNN 似乎是完全否定了这一点，如果把 MF 的隐藏表示也作为共享层的输入呢？作者是否曾经尝试过呢？