Ask the GRU: Multi-task Learning for Deep Text Recommendations

Accepted by RecSys'16, 文章链接：https://dl.acm.org/citation.cfm?id=2959180

摘要

从日常生活的需求上来说，学术论文、电影简介、新闻博客等的推荐，都与文本有关。从文本中提取特征，融合到推荐系统中最常用的分解模型中是一种很自然的想法，这也使得那些冷启动的物品具有较好的特征表示。之前的工作主要有使用 **主题模型** 或者是相关文本的 **平均词向量** 来表示物品特征向量，本文提出的方法则是通过深度 RNN （GRU）网络把物品相关的文本映射为特征向量。在学术论文推荐的实验中具有更高的准确率；在冷启动场景下，能够缓解协同过滤模型的数据稀疏问题，也超过了 state-of-the-art。作者认为这得益于多任务学习，因为处理文本的网络的学习过程是 **物品推荐** 和子问题 **分类问题** 的结合。

简介

文本推荐是近年来 NLP 相关应用的热点，可以细分为博客推荐、社区帖子推荐、新闻、商品（使用简介、评论）、学术论文等。既然是推荐系统的问题，就离不开协同过滤（Collaborative Filtering）、基于内容的过滤（Content-based）和混合过滤模型（Hybrid Models）。然而这些模型并不能很好的利用推荐物品的文本特征，仅仅是简单的使用基于统计的词袋模型作为补充特征。从文本中提取特征，是一种非监督学习任务，如此学习到的特征很难与推荐系统有机结合。（个人理解：非监督学习得到的文本特征，可能是注重文本的语法特征，而对于推荐系统来说，更关注语义信息；另一方面，推荐系统可能仅仅关注文本中的几个词，而不是完整的文本。这主要是因为非监督的学习，难以保证特征学习的质量，学习出的特征空间，很难去适合推荐系统需要的特征空间。）

本文提出的方法是，在协同过滤模型中，使用 RNN，通过 **显式监督学习**，对物品的文本进行表示。使用 RNN，是因为它在 NLP 领域中独占鳌头：

- 对词语的顺序敏感
- 不需要人工标注特征
- 适合处理大的未标注语料数据
- 可并行学习与计算处理
- 作为一个特征编码器，可以对有文本信息的冷启动物品抽取特征

深度网络模型的学习能力很强，导致容易在训练时过拟合。传统做法是在非监督学习的过程中，对参数加入正则项，这里则是把学习到的特征作为 RNN 网络的输入而不适用。（疑问：加一个惩罚项可以吗？）通过 **多任务学习**，作者增加了一个分类子任务——预测物品的某些 meta-data （标签等）防止过拟合。

背景与相关工作

问题定义

本文的任务是，利用物品所附的文本信息进行推荐。每一个物品都具有一段文本，并且该物品有它的几个标签。分类的子任务就是预测物品的标签，推荐的主要任务则是从用户未表达过喜好的备选物品中选出他可能感兴趣的。

隐变量分解模型

推荐系统可以建模为打分预测问题，用户 i 对物品 j 预测的打分 $\hat{r}_{ij} = b_i + b_j + \tilde{u}_i^T \tilde{v}_j$ ，这里的 b_i, b_j 分别表示用户和物品的偏置，$\tilde{u}_i, \tilde{v}_j$ 分别是用户和物品的隐变量，而模型需要学习的就是这四个参数。学习（优化）过程有两种：

- 基于显式反馈通常只使用有用户、物品打分记录的数据，优化所有显式反馈的均方误差加上参数的正则化损失
- 基于隐式反馈，即用户点击与否的情况，优化的目标通常是为所有的用户、物品商品设置权重，观察到的显式反馈行为设置很大的权重，未观察到的行为设置很小的权重，最后也加上正则化参数损失。当然，有的也会用 rank 损失（BPR 等）作为优化目标

冷启动问题

新的物品、用户或者是系统学习过程中未见过的都是冷启动的对象，是隐变量分解（协同过滤）无法处理的，因为打分公式中的 $\tilde{u}_i$ 或 $\tilde{v}_j$ 是无法学习的。为此，引入了混合式模型，利用物品、用户的外部信息，假设某个新物品的有相关的文本数据：$X_j = \{w_1, w_2, \cdots, w_{n_j}\}$，就可以学习到适用于打分预测公式的物品特征 $f(X_j)$，此时 $\hat{r}_{ij} = b_i + b_j + \tilde{u}_i^T f(X_j)$。这里的 f 完成了从词序列到特征向量的映射（特征编码器）。

有的做法是从物品的类别、用户的地理信息、性别等信息，手动设计特征，通过线性回归计算预测打分。通过深度学习可以省去这部分特征工程，并且改良线性回归欠拟合的问题。CTR （协同主题回归）模型则是在物品-词矩阵上利用概率分解模型（LDA 方法）、用户-物品矩阵上使用隐变量分解模型学习特征。这就是本文任务对比的 state-of-the-art。

加入多任务学习的正则化

推荐系统的许多数据集都是比较稀疏的，协同过滤的性能大打折扣。通过利用外部信息，设计子任务学习特征，接着共享的用户或物品的特征表示可以作为补充。设计多标签分类的子任务，即通过物品的文本信息，预测相关的标签来学习物品特征，虽然可能不可靠、不完整，但可以一定程度上缓解主要的推荐预测的过拟合。由此带来的另一个好处是，可以缓解冷启动的问题。

深度学习

本文的核心任务是设计学习特征映射函数 f ，深度学习模型具有优秀的学习能力。从卷积神经网络（CNN）到循环神经网络（RNN），都在 NLP 任务上展现出强大的能力。RNN 在处理序列化的文本更是翘楚，尽管存在过拟合的问题。

本文模型

模型的关键在于学习一个从物品 j 的文本序列信息 X_j，转化为特征向量的映射。不考虑冷启动情况，通过协同过滤等隐因子分解模型可以得到物品的一个表示 $\tilde{v}_j$， 此处定义 $f(X_j) = g(X_j) + \tilde{v}_j$ ，冷启动物品则设置 $\tilde{v}_j = \vec{0}$，f 仅和 g 有关了。

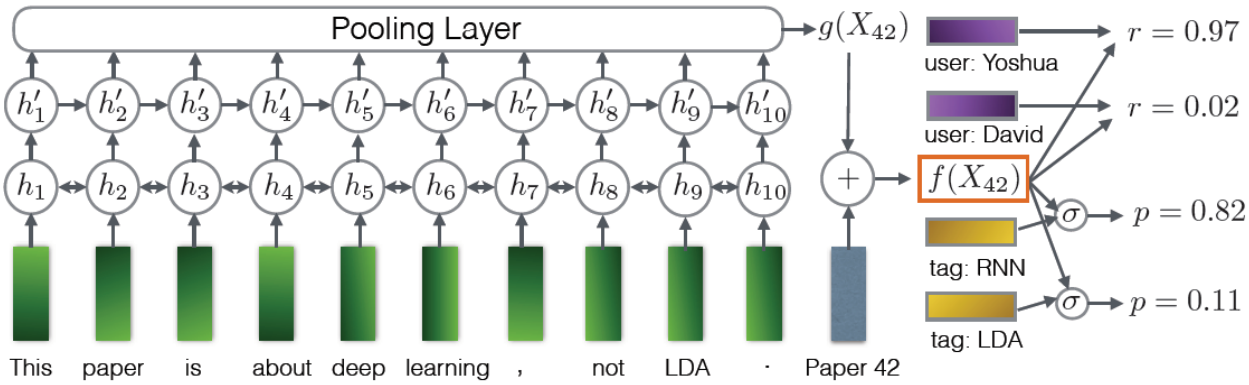

Figure 1: Proposed architecture for text item recommendation. Rectangular boxes represent embeddings. Two layers of RNN with GRU are used, where the first layer is a bi-directional RNN. The output of all the hidden units at the second layer is pooled to produce a text encoding which is combined with an item-specific embedding to produce the final representation $f(X)$. Users and tags are also represented by embeddings, which are combined with the item representation to do tag prediction and recommendation.

词序不敏感编码器

对 X_j 中的每个词计算平均词向量作为 $g(X_j)$

词序敏感编码器

词袋模型的不足就是忽略了句子中的词序，导致两句完全相反意思的话具有相同的表示。RNN 编码器则按照词语出现的顺序进行编码，考虑了更为全面的语义。本文与其他任务不同之处在于没有定义一个非监督学习的目标（如 LDA 中类似的某个词在某个主题下的似然概率），而是直接作为子任务的特征输入来学习这个编码器的参数。本文中采用的是 GRU 这种变体，所以文章标题叫做 Ask GRU...

多任务学习

我认为这是本文最重要的亮点，其他地方真的非常 naive。引入多任务学习，从文本中用 GRU 学习到的物品特征作为一个多标签分类的子任务的输入。如模型图中，得到的 $f(X_j)$ 与标签的 embedding $\tilde{t}_l$ 经过 sigmoid 函数作用，得到一个概率值 $p_{jl} = \sigma(f(X_j)^T \tilde{t}_l)$，这个子任务的优化目标定义为

$$C_T(\theta) = \frac{1}{|T|} \sum_j \sum_l \{t_{jl} \log p_{jl} + c'_{jl}(1 - t_{jl}) \log(1 - p_{jl})\}$$

t_{jl} 为 0/1 值，表示物品 j 是否具有标签 l，c'_{jl} 是对未观测到数据（即物品不具有某个标签）的权重，通常设置的很小。

最终的优化目标

以上是关于子任务的优化目标，对于推荐主任务来说，定义的损失是基于隐式反馈（0/1 数据）

$$C_R(\theta) = \frac{1}{|R|} \sum_{(i,j) \in R} c_{ij}(\hat{r}_{ij} - r_{ij})^2 + \Omega(\theta)$$

c_{ij} 是一个调节观测到和未观测到数据的权重，对于正样本（观测到的）设置很大，对于负样本（未观测到的）设置的很小。最终的优化目标：

$$C(\theta) = \lambda C_R(\theta) + (1 - \lambda) C_T(\theta)$$

个人对模型的点评

没有什么惊喜的模型，很中规中矩，我觉得还是属于 Collective Matrix Factorization 的范畴。能够学习到的地方在于，对于未观测到的样本不是进行完全的否定，而是设置一个较小的权重，是比较符合实际情况的。融合子任务的多任务学习也不是很高明，意料之中的把两个任务的 loss 加和，因为用到了同样的一些参数，可以共享，进而优化整个模型在两个任务上的处理能力。

实验

与 CTR 模型（2011年，目前的 state-of-the-art）做对比，相同的学术论文推荐数据集：CiteULike-a 和 CiteULike-t。待推荐的物品即为学术论文，相关的文本数据仅仅使用论文的摘要部分。实验分为两部分：

1. 仅仅使用 RNN/GRU 做推荐这一个任务
2. 融合了标签预测的多任务， 对 CTR 模型进行了改造：基于 Collective Matrix Factorization 和 CTR 的概率分解模型，用一个 Collective 平方误差为优化目标（*文中仅仅描述了一下，具体怎么做的不清楚*）

热启动实验采用了 5-fold 验证，那些出现次数少于 5 次的文章保证只出现在训练集（作者的解释是这些数据很难评价，其实我认为就是学习效果不好）；而关于冷启动的文章进行的实验验证：把某些文章在训练集中的记录删除。评价指标使用的是 Recall@M。

结果对比

Table 1: % Recall@50 for all the methods (higher is better).

	Citeulike-a			Citeulike-t		
	Warm Start	Cold Start	Tag Prediction	Warm Start	Cold Start	Tag Prediction
GRU-MTL	**38.33**	**49.76**	**60.52**	45.60	**51.22**	**62.32**
GRU	36.87	46.16	—	42.59	47.59	—
CTR-MTL	35.51	39.87	48.95	**46.82**	34.98	46.66
CTR	31.10	39.00	—	40.44	33.74	—
Embed-MTL	36.64	41.71	60.36	43.02	38.16	62.29
Embed	33.95	38.53	—	37.98	35.85	—

“-MTL” 表示使用了多任务优化目标，Embed 是使用平均词向量的模型。

直观地，融合多任务学习的方式，推荐的准确率都要更高；从子任务的标签分类问题上来说，使用 GRU 关注序列化信息，并且有监督的学习效果也有微小的提升。

两个略显奇怪的结果的解释，猜测

1. GRU-MTL 在第二个数据集上 Warm Start 的表现要比 CTR 差，作者用了个 Interestingly，说在参数规模比较小的时候 GRU 还是优于 CTR 的。我认为，可能和数据质量与结构有关，LDA 和 GRU Encoder 得到的特征虽然处于不同的特征空间，但是 LDA 在这里得到的主题特征分布更适合这个任务，因为主题一定程度上体现出了文本的内容，读者对于学术论文，其中的关键词还是是否感兴趣的重要依据。

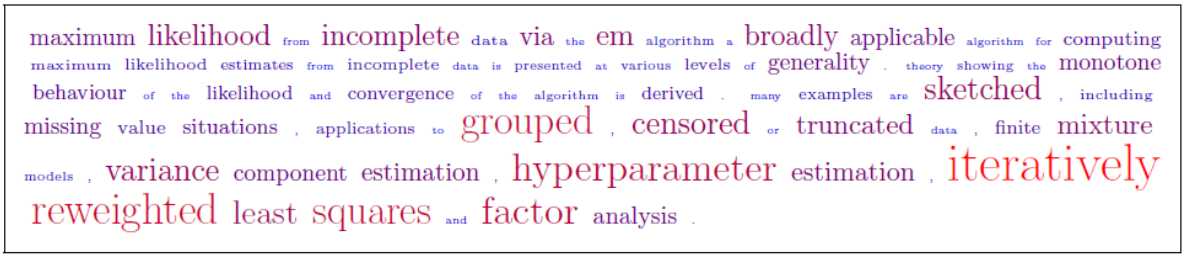

Figure 2: Saliency of each word in the abstract of the EM paper [53]. Size and color of the words indicate their leverage on the final rating. The model learns that chunks of word phrases are important, such as "maximum likelihood" and "iteratively reweighted least squares", and ignores punctutations and stop words.

这里作者给出的一个 case study，想要说明论文的摘要中每个词对于推荐系统预测打分的贡献，我觉得可能反而体现出了 LDA 性能有时候也不差的原因，这些词完全体现出了这个文章的主题。

2. 注意到 Embed-MTL 和 GRU-MTL 在标签预测的性能上基本不分伯仲，CTR-MTL 则差很多。作者给出的解释是 CTR 学出的特征不适合标签预测，更为适合用来预测推荐打分。在改造 CTR 的过程中定义的分解平方误差函数可能与传统的分类优化目标有较大的出入。我认为作者在 CTR-MTL 的优化目标中没有定义与多标签预测相关的损失函数，像 LDA 或者是 CTR 学到的论文特征（或者称为主题分布）直接用来做分类是否合适也是需要其他实验来验证的，可能和数据集的特征有关，LDA 究竟适合什么样的任务是不是还缺少对应的研究？

总结

这篇文章的标题取得很好，RNN / GRU 代表了深度神经网络这一方法，Multi-task Learning 又是最近比较热的研究点，而且还是做文本推荐的，和 NLP 关系比较紧密，却让我"高估"了他的 idea 和质量，感觉没什么亮点，无非就是几种手段融合在一起，套了个多任务学习的盒子，实验结果好的地方可以解释为模型真的起了作用，但是和其他方法相比没有显著的提升、有的甚至下降的情况，只是给出了几个猜测，总觉得问题解决的不够完美。