基于KG的推荐

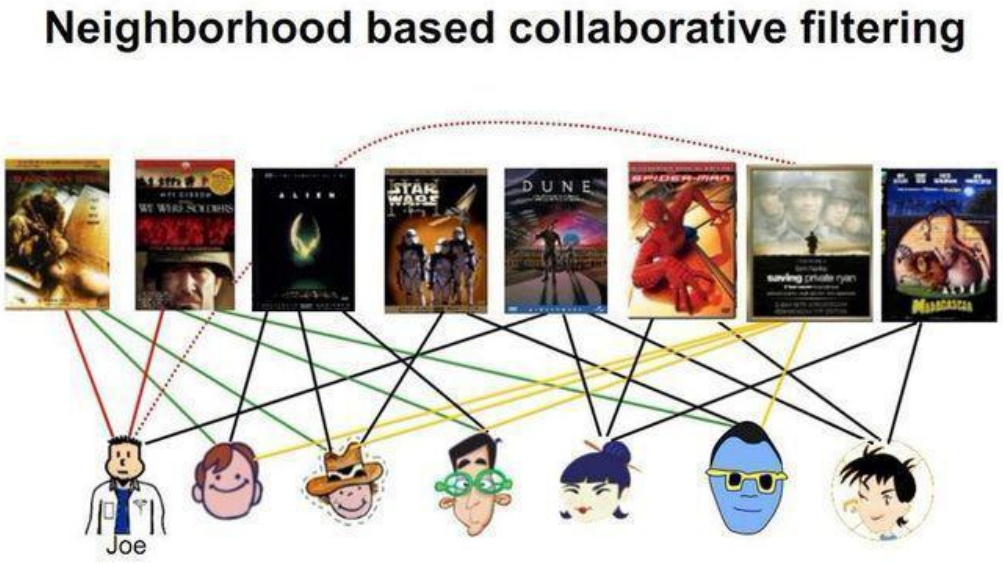
许明司

華東師範大學
EAST CHINA NORMAL UNIVERSITY
上海市高可信计算重点实验室
Shanghai Key Laboratory of Trustworthy Computing

Recommendation

- 一个推荐系统的目标: 无论是评分预测还是点击率预测都可以归结为对用户与信息/ 物品 进行一个匹配。

- 匹配的依据往往是对于用户过往的历史行为, 最有代表性的是基于人以群分,物以类聚原则的协同过滤算法。



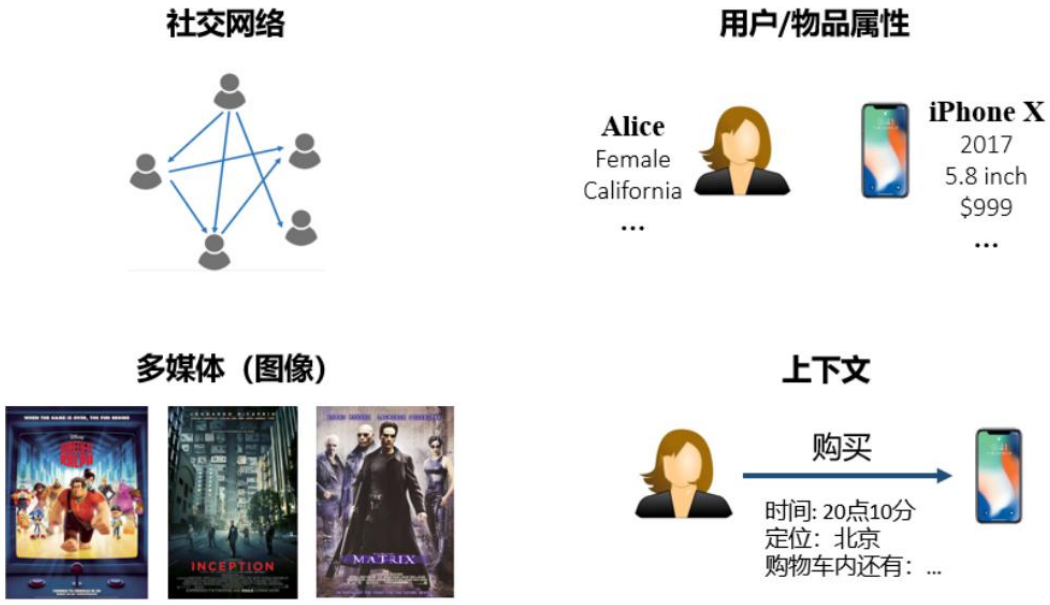
Recommendation

- 传统的推荐算法往往面临两个问题:

- 1. **数据稀疏性**问题: 一个大规模的推荐系统中往往有大量的用户和大量的物品, 相关对而言用户和物品的交互信息往往是非常**sparse**的.例如一个电商平台中可能有上亿个商品, 有上千万的用户, 一个用户可能所购买过的商品只有几十份。用少量的已知信息去预测大量的未知信息，会极大增加模型**过拟合**的风险。
- 2. **冷启动问题**: 对于新加入的用户或者物品, 系统中没有用户对于相应物品的行为, 因此无法进行精准地建模与推荐。常见于**新闻推荐**领域, 系统给用户推荐的往往是三天内的新闻, 新闻缺乏用户的行为信息。

Recommendation

- 一个常见的解决办法是**引入辅助信息**,去丰富物品,用户的信息。例如社交网络(一个用户对某个物品感兴趣，他的朋友可能也会对该物品感兴趣),用户/物品属性(拥有同种属性的用户可能会对同一类物品感兴趣)
- 图像/视频/音频/文本等多媒体信息, 上下文(用户-物品交互的时间、地点、当前会话信息)

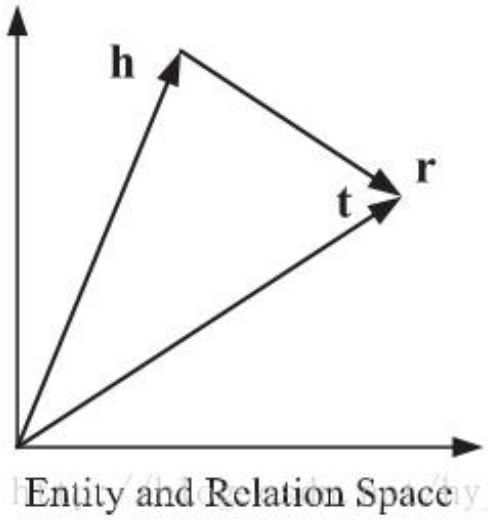


Knowledge Graph

- 在各种辅助信息中，**知识图谱**是一种能够捕获语义网络的新兴类型的辅助信息, 其结点代表**实体**(entity), 边代表实体之间的各种**语义关系**(relation)。 一个知识图谱由若干个三元组(h、 r、 t)组成, 其中h和t代表一条关系的头结点和尾节点，r 代表关系。

Knowledge Graph

- 知识图谱特征学习中的翻译距离模型:
- 1. transE
- $\vec{h}$ + $\vec{r}$ 与$\vec{t}$ 之间的距离 $f_r(h,t) = -||\vec{h} + \vec{r} - \vec{t}||_{1/2}$



Entity and Relation Space

训练过程中，最小化hinge loss

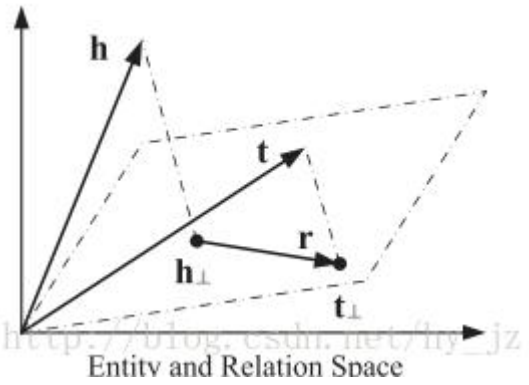
$$\mathcal{L} = \sum_{(h,r,t)\in S} \sum_{(h',r,t')\in S'_{(h,r,t)}} [\gamma + f_r(h,t) - f_r(h',t')]_+$$

[x]+表示max{0,x} , γ是一种容忍程度, 表示的是负样本的分数最多比正样本高γ，再大就没有奖励了

缺点: h向量和t向量的差向量是固定好的,不可能同时表示两个实体之间的多种关系, 比如可以同时是夫妻关系和同事关系.

Knowledge Graph

- 知识图谱特征学习中的翻译距离模型:
- 2. transH
- TransH 依然把实体 h,t 表示为向量，但是把关系r表示成两个向量: 超平面的法向量 $\vec{W}$ 和关系r在超平面内的向量表示 $\vec{d}$



头结点h和尾节点t映射到超平面的表示为:

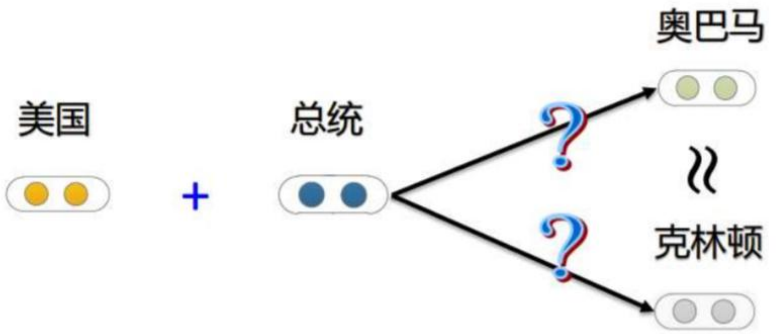
$$\vec{h}_\perp = \vec{h} - \vec{W}_r^T \vec{h} \vec{W}_r \quad \vec{t}_\perp = \vec{t} - \vec{W}_r^T \vec{t} \vec{W}_r$$

对于不同的关系, 节点可能在关系ri中相似，但可能在关系rj中不相似。

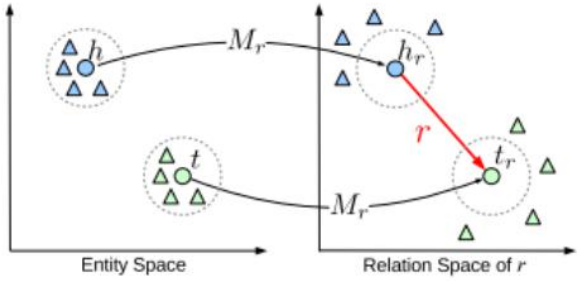
构造负例时, 对于1-N 做了如下处理，以更大的概率替换头节点。对于N-1以更大的概率替换尾节点。

Knowledge Graph

实际上, 实体向量加上关系向量可能得到多种不同的向量表示



- 3. transR
- 在TransE和TransH这两个模型中,实体和关系向量被映射到同一个向量空间, 而TansR把关系向量映射到不同的空间, **一劳永逸解决了所有多关系问题**.既把实体向量嵌入到线性空间,把关系嵌入到矩阵空间当中



$$\vec{h}_r = \vec{h}M_r \quad \vec{t}_r = \vec{t}M_r$$

$$f_r(\vec{h},\vec{t}) = ||\vec{h}_r + \vec{r} - \vec{t}_r||_2^2$$

$$||\vec{h}||_2 \le 1, ||\vec{t}||_2 \le 1, ||\vec{r}||_2 \le 1, ||\vec{h}_r||_2 \le 1, ||\vec{t}_r||_2 \le 1$$

TransR 模型虽然效果比现有的模型好，但是模型参数较多，相比TransE、TransH模型而言计算量更大。

Knowledge Graph

- 知识图谱特征学习中还有一种是基于语义的匹配模型:
- *Analogical Inference for Multi-relational Embeddings* ***ICML 2017***
- 在ripple net中用到其中相关结论
- 作者认为, 用矩阵定义的线性映射表达能力比用向量定义的加法映射更强。于是**将关系r视为线性映射**，即给定三元组(s,r,o)，作者希望对于所有有效的三元组

$$v_s^\top W_r \approx v_o^\top$$

满足的程度就用一个score function表示，模型的目标就是学到恰当的v和W，来让这个score function给有效的三元组高分，无效的三元组低分。

$$\phi(s,r,o) = \langle v_s^\top W_r, v_o \rangle = v_s^\top W_r v_o$$

KG & Recommendation

- 目前，将知识图谱特征学习应用到推荐系统中主要通过三种方式——依次学习、联合学习、以及交替学习。
  - 依次学习：把知识图谱特征学习和推荐系统当做两个独立的部分

  - 联合学习:将知识图谱的特征和目标函数结合, 去端到端的联合学习.

  - 交替学习: 将KG和推荐系统视为两个分离又相关的任务.使用多任务学习(multi-task)这种框架进行交替学习

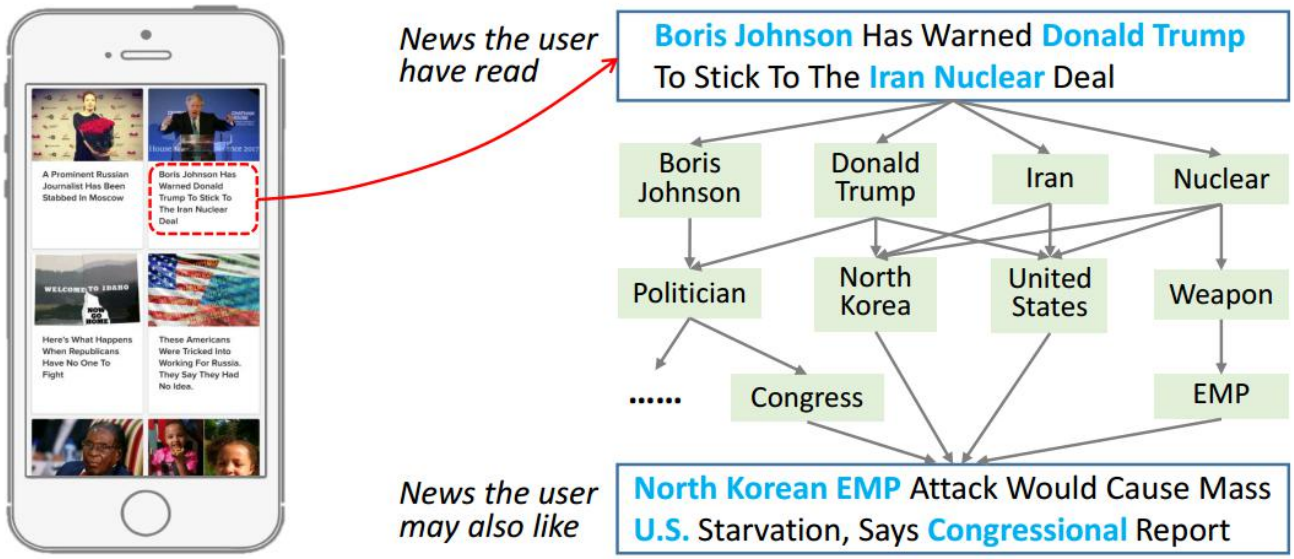
依次学习

新闻推荐场景(缺少用户行为数据,典型的冷启动问题)：

- *DKN: Deep knowledge-aware network for news recommendation* ***www'18***
  - 新闻具有**高度时效性**和**话题敏感性**，且用户关注话题一般有针对性；
    - **Attention network**
  - 新闻的语言高度浓缩，往往包含很多**常识知识**，而目前基于词汇共现的模型，很难发现这些潜在的知识。
    - **结合KG挖掘新闻信息**

依次学习

新闻推荐场景:

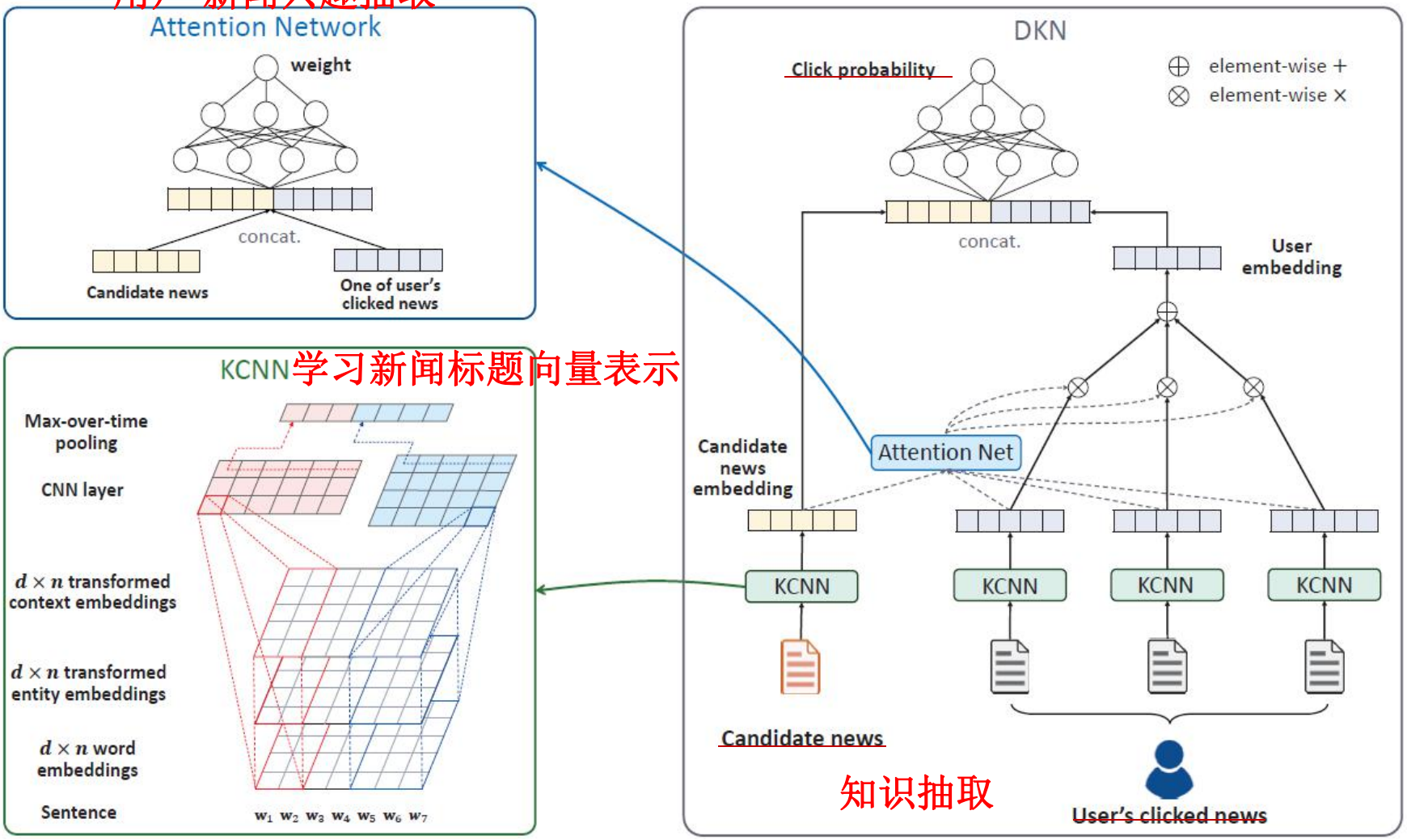


新闻标题和正文中通常存在大量的实体，实体间的语义关系可以去挖掘用户的潜在兴趣。 但是判断这样两个新闻标题的相似性需要上下文知识和常识推理, 传统方法（话题模型、词向量）很难去发掘**知识级别的关联**。

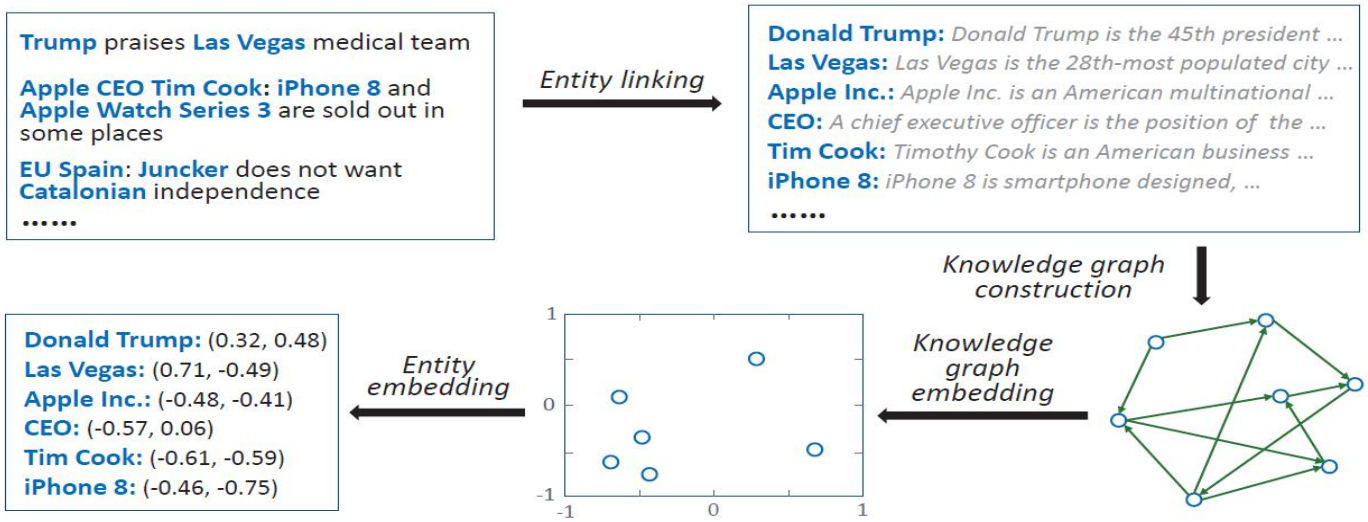
DKN模型

DKN 模型主要分成三部分:知识蒸馏(Knowledge Distillation)、知识感知卷积神经网络(KCNN)、用于抽取用户兴趣的注意力网络(Attention Network)

用户-新闻兴趣抽取



DKN- 知识蒸馏

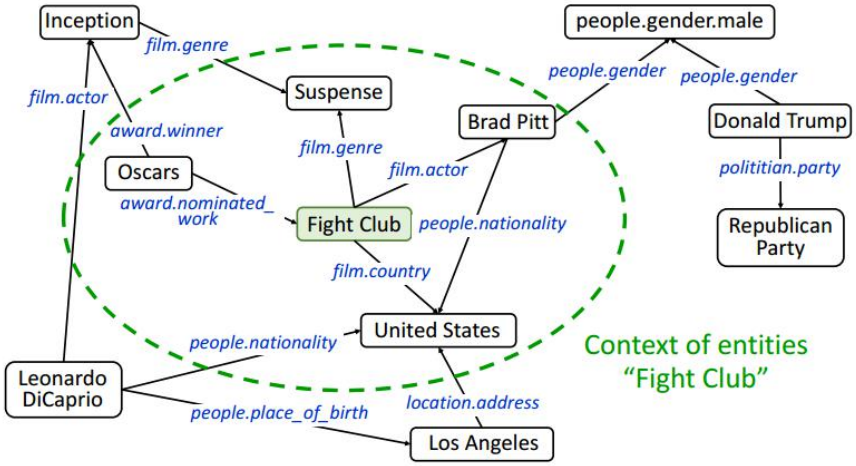


1.识别出文本中的知识实体并利用实体链接技术进行实体消歧.

2.新闻文本中的实体与关系就构成了一个KG的一个子图, 把所有与文中的实体的链接在一个跳之内的所有实体都扩展到该子图中来。

3.构建好知识子图以后，利用特征学习得到每个实体的向量

DKN- 知识蒸馏



由于构造子图的方式有一定的信息损失, 为了更好地利用一个实体在原知识图谱的位置信息, 利用一个实体e在原上下文中的邻居去表示这个实体。

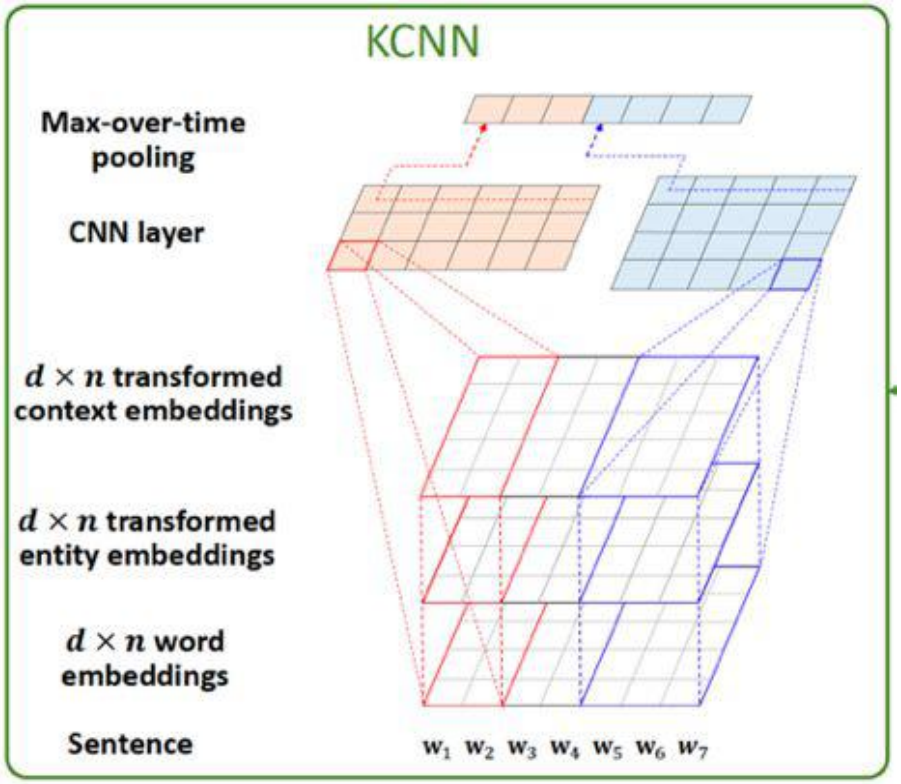
$$context(e) = \{e_i \mid (e, r, e_i) \in \mathcal{G} \text{ or } (e_i, r, e) \in \mathcal{G}\},$$

$$\bar{\mathbf{e}} = \frac{1}{|context(e)|} \sum_{e_i \in context(e)} \mathbf{e}_i,$$

G是构建的知识图谱子图,ei由特征学习的方式得到.

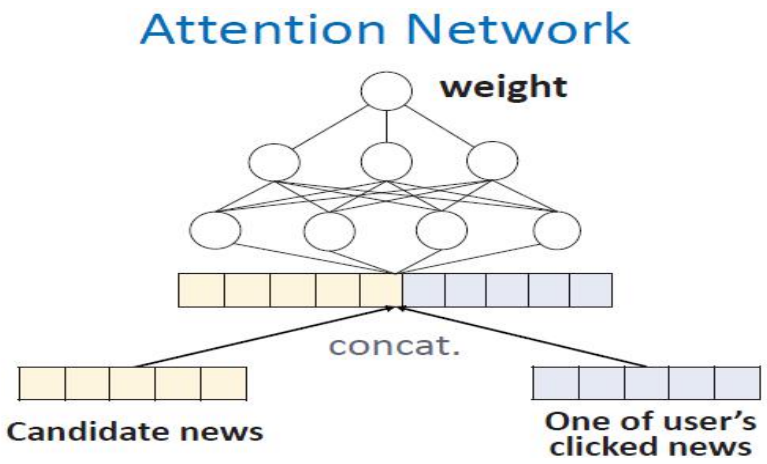
DKN- 知识感知卷积神经网络

- **问题：**
  - 三种不同的向量表示基于不同模型学习到的，直接放到同一个向量空间不合理。
- **解决方案：使用多通道&实体对齐的KCNN**
  - 先把链接实体、上下文实体的向量表示通过一个**非线性变换映射**到同一个向量空间；
  - 将词、链接实体、上下文实体的向量表示作为**CNN多通道**的输入,类似图像的RGB三个通道；



$$\mathbf{W} = \left[[\mathbf{w}_1\, g(\mathbf{e}_1)\, g(\bar{\mathbf{e}}_1)] \, [\mathbf{w}_2\, g(\mathbf{e}_2)\, \bar{g}(\mathbf{e}_2)] \ldots [\mathbf{e}_n\, g(\mathbf{e}_n)\, g(\bar{\mathbf{e}}_n)] \right] \in \mathbb{R}^{d \times n \times 3}$$

DKN- 注意力机制



- 输入：用户的**点击**历史新闻
  - 通过***KCNN***获得它们的向量表示。
  - 使用***DNN***作为注意力网络， **softmax**函数计算影响力权重。
- 计算候选文档对于用户每篇点击文档的attention

$$s_{t_k^i,t_j} = \text{softmax}\left(\mathcal{H}\left(\mathbf{e}(t_k^i),\mathbf{e}(t_j)\right)\right) = \frac{\exp\left(\mathcal{H}\left(\mathbf{e}(t_k^i),\mathbf{e}(t_j)\right)\right)}{\sum_{k=1}^{N_i}\exp\left(\mathcal{H}\left(\mathbf{e}(t_k^i),\mathbf{e}(t_j)\right)\right)}.$$

- 得到用户i 关于候选新闻的向量表示。

$$\mathbf{e}(i) = \sum_{k=1}^{N_i} s_{t_k^i,t_j}\mathbf{e}(t_k^i).$$

- 可用多种方法计算用户i点击新闻 tj的概率, 比如输入到另一个DNN中.

实验

因子分解机

深度语义分解模型

memorization+generalization

神经网络+因子分解机

深度矩阵分解

基于内容的推荐

协同过滤

Models*	F1	AUC	*p*-value**
DKN	**68.9 ± 1.5**	**65.9 ± 1.2**	–
LibFM	61.8 ± 2.1 (-10.3%)	59.7 ± 1.8 (-9.4%)	$< 10^{-3}$
LibFM(-)	61.1 ± 1.9 (-11.3%)	58.9 ± 1.7 (-10.6%)	$< 10^{-3}$
KPCNN	67.0 ± 1.6 (-2.8%)	64.2 ± 1.4 (-2.6%)	0.098
KPCNN(-)	65.8 ± 1.4 (-4.5%)	63.1 ± 1.5 (-4.2%)	0.036
DSSM	66.7 ± 1.8 (-3.2%)	63.6 ± 2.0 (-3.5%)	0.063
DSSM(-)	66.1 ± 1.6 (-4.1%)	63.2 ± 1.8 (-4.1%)	0.045
DeepWide	66.0 ±1.2 (-4.2%)	63.3 ± 1.5 (-3.9%)	0.039
DeepWide(-)	63.7 ± 0.9 (-7.5%)	61.5 ± 1.1 (-6.7%)	0.004
DeepFM	63.8 ± 1.5 (-7.4%)	61.2 ± 2.3 (-7.1%)	0.014
DeepFM(-)	64.0 ± 1.9 (-7.1%)	61.1 ± 1.8 (-7.3%)	0.007
YouTubeNet	65.5 ± 1.2 (-4.9%)	63.0 ± 1.4 (-4.4%)	0.025
YouTubeNet(-)	65.1 ± 0.7 (-5.5%)	62.1 ± 1.3 (-5.8%)	0.011
DMF	57.2 ± 1.2 (-17.0%)	55.3 ± 1.0 (-16.1%)	$< 10^{-3}$

* "(-)" denotes "without input of entity embeddings".

** *p*-value is the probability of no significant difference with DKN on AUC by *t*-test.

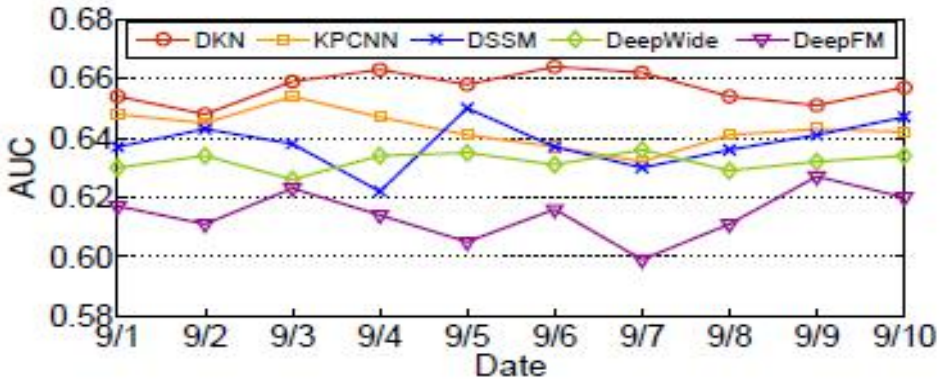


Figure 7: AUC score of DKN and baselines over ten days (Sep. 01-10, 2017).

实验

Table 3: Comparison among DKN variants.

Variants	F1	AUC
DKN with entity and context emd.	**68.8 ± 1.4**	**65.7 ± 1.1**
DKN with entity emd. only	67.2 ± 1.2	64.8 ± 1.0
DKN with context emd. only	66.5 ± 1.5	64.2 ± 1.3
DKN without entity nor context emd.	66.1 ±1.4	63.5 ± 1.1
DKN + TransE	67.6 ± 1.6	65.0 ± 1.3
DKN + TransH	67.3 ± 1.3	64.7 ± 1.2
DKN + TransR	67.9 ± 1.5	65.1 ± 1.5
DKN + TransD	**68.8 ± 1.3**	**65.8 ± 1.4**
DKN with non-linear mapping	**69.0 ± 1.7**	**66.1 ± 1.4**
DKN with linear mapping	67.1 ± 1.5	64.9 ± 1.3
DKN without mapping	66.7 ± 1.6	63.7 ± 1.6
DKN with attention	**68.7 ± 1.3**	**65.7 ± 1.2**
DKN without attention	67.0 ± 1.0	64.8 ± 0.8

输入信息

Trans模型

映射

是否用Attention

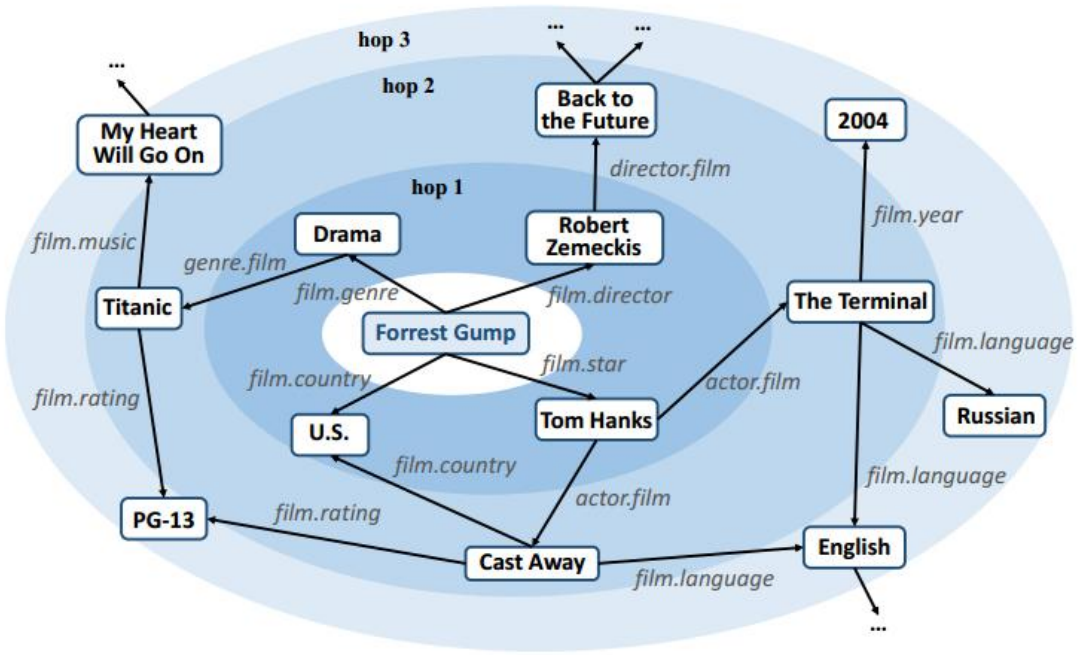
DKN小结

- 研究思路是近几年工作中比较常见的，利用知识提升深度神经网络的效果。
- DKN 的特点是融合了知识图谱与深度学习，从语义层面和知识两个层面对新闻进行表示，而且实体和单词的对齐机制融合了异构的信息源，能更好地捕捉新闻之间的隐含关系。
- 没有考虑怎样选择KG中上下文实体，**只简单把单跳范围内实体作为上下文**，对于某些需要用两跳及两跳以上才能完整表示的关系不考虑。

联合学习

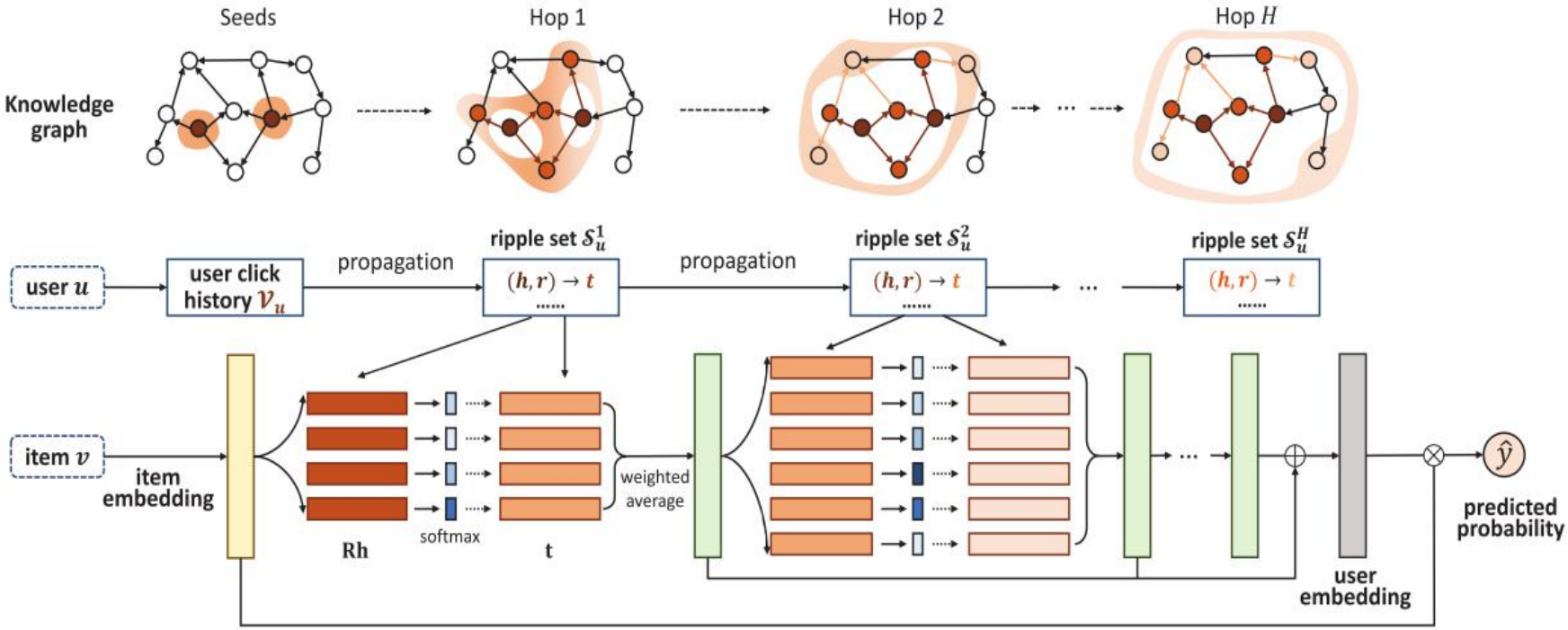
- *RippleNet: Propagating User Preferences on the Knowledge Graph for Recommender Systems.* ***CIKM'18***
- *为什么叫RippleNet(水波网络)?*
- *1. 模拟用户兴趣在知识图谱上传播的过程,就好像一滴水滴在水面上, 会在水面上荡起一层层的涟漪。*
- *2.用户u的偏好随着跳数k的增大而下降, 如同水波的层层递减*
- *3. 最大跳数H不能过大, 太远的实体带来的可能是噪声而不是信号。*

RippleNet



- 如果你看过Forrset Gump恰好因为你是Tom Hanks的粉丝。Tom Hanks又主演过Cast Away和The Terminal, 那么你就有可能对Cast Away和The Terminal也感兴趣。

RippleNet



已点击的商品/看过的电影作为一个种子集(seed), 对于知识图谱做一个知识的链接.找到当前的商品, 就以它为头结点, KG的关系是有方向性的,看KG的出度指向哪些结点, 把这些作为第一跳的三元组,直至第N跳的ripple set。
h和r分别和item embedding做运算.通过softmax求得一个概率表示每个h， r和item embedding的关系, 和尾结点做一个乘法. 加权后得到绿色embedding

RippleNet

- 通过递归可以得到以下相关定义:
- 1.相关实体:

$$\mathcal{E}_u^k = \{t \mid (h,r,t) \in \mathcal{G} \text{ and } h \in \mathcal{E}_u^{k-1}\}, \quad k = 1, 2, ..., H, \quad (2)$$

where $\mathcal{E}_u^0 = \mathcal{V}_u = \{v \mid y_{uv} = 1\}$ *is the set of user's clicked items in the past, which can be seen as the seed set of user* u *in KG.*

- h是k-1跳时的相关节点.拿h作为投节点去KG中找出三元组.三元组的尾结点就是相关节点。就是种子节点, 用户历史点击的item.
- 2.水波集

$$\mathcal{S}_u^k = \{(h,r,t) \mid (h,r,t) \in \mathcal{G} \text{ and } h \in \mathcal{E}_u^{k-1}\}, \quad k = 1, 2, ..., H.$$

- h是k-1跳时的相关节点, 所相关的三元组就是第k跳的水波集。

RippleNet

- 通过递归可以得到以下相关定义:
- 3.交互行为:

$$y_{uv} = \begin{cases} 1, & \text{if interaction } (u, v) \text{ is observed;} \\ 0, & \text{otherwise.} \end{cases}$$

- 对于一个u行,v的交互矩阵,如果用户u点击了商品v, 其元素置1.
- 4.物品向量和三元组向量的关系

$$p_i = \text{softmax}\left(\mathbf{v}^{\mathrm{T}}\mathbf{R}_i\mathbf{h}_i\right) = \frac{\exp\left(\mathbf{v}^{\mathrm{T}}\mathbf{R}_i\mathbf{h}_i\right)}{\sum_{(h,r,t)\in\mathcal{S}_u^1}\exp\left(\mathbf{v}^{\mathrm{T}}\mathbf{R}\mathbf{h}\right)},$$

- 对于每一跳ripple set的(h, r, t), 当前跳下item embedding乘上r和h.做softmax得到当前三元组和当前跳下item embedding的相关度.用p乘上每个t。

RippleNet

- 通过递归可以得到以下相关定义:
- 5.用户的向量表示:

$$\mathbf{o}_u^1 = \sum_{(h_i, r_i, t_i) \in \mathcal{S}_u^1} p_i \mathbf{t}_i,$$

- 得到用户在第一跳下的embedding.

$$\mathbf{u} = \alpha_1 \mathbf{o}_u^1 + \alpha_2 \mathbf{o}_u^2 + \ldots + \alpha_H \mathbf{o}_u^H, \qquad \sum_{i=1}^{H} \alpha_i = 1.$$

- 当经过了H跳,可以得到向量表示。
- 6.预测概率

$$\hat{y}_{uv} = \sigma(\mathbf{u}^{\mathrm{T}} \mathbf{v}),$$

where $\sigma(x) = \frac{1}{1+\exp(-x)}$ is the sigmoid function.

- 最后可以预测用户u点击item v的一个概率。

RippleNet

- 学习算法:
- 给定知识图谱G,和用户的历史行为Y,希望最大化后验概率。

$$\max p(\Theta|\mathcal{G}, \mathbf{Y}),$$

- 后验概率展开

$$p(\Theta|\mathcal{G}, \mathbf{Y}) = \frac{p(\Theta, \mathcal{G}, \mathbf{Y})}{p(\mathcal{G}, \mathbf{Y})} \propto p(\Theta) \cdot p(\mathcal{G}|\Theta) \cdot p(\mathbf{Y}|\Theta, \mathcal{G})$$

$$p(\Theta) = \mathcal{N}(\mathbf{0}, \lambda_1^{-1}\mathbf{I}).$$

- 首先认为认为模型的参数满足0均值的正态分布.

$$\begin{aligned} p(\mathcal{G}|\Theta) &= \prod_{(h,r,t)\in\mathcal{E}\times\mathcal{R}\times\mathcal{E}} p\big((h,r,t)|\Theta\big) \\ &= \prod_{(h,r,t)\in\mathcal{E}\times\mathcal{R}\times\mathcal{E}} \mathcal{N}\big(I_{h,r,t} - \mathbf{h}^{\mathrm{T}}\mathbf{R}\mathbf{t},\ \lambda_2^{-1}\big), \end{aligned}$$

- 给定模型参数, 求知识图谱的概率。可以通过三维张量分解的方法得到似然函数。

RippleNet

- 学习算法:

$$p(\mathbf{Y}|\Theta,\mathcal{G}) = \prod\nolimits_{(u,v)\in\mathbf{Y}} \sigma(\mathbf{u}^{\mathrm{T}}\mathbf{v})^{y_{uv}} \cdot \left(1-\sigma(\mathbf{u}^{\mathrm{T}}\mathbf{v})\right)^{1-y_{uv}}$$

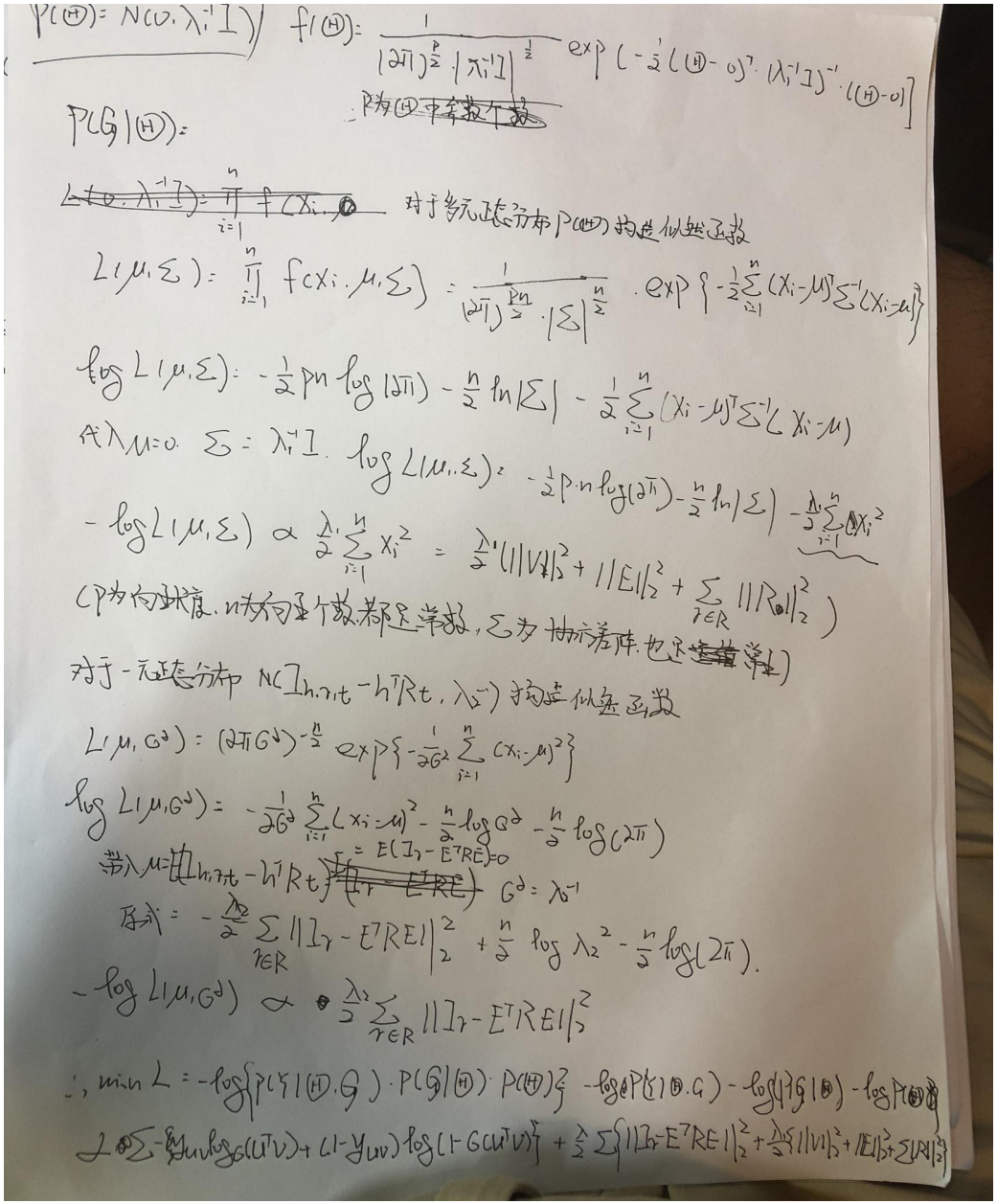
- 似然函数定义成伯努利分布乘积的形式。

$$\begin{aligned}\min \mathcal{L} &= -\log\left(p(\mathbf{Y}|\Theta,\mathcal{G})\cdot p(\mathcal{G}|\Theta)\cdot p(\Theta)\right)\\ &= \sum_{(u,v)\in\mathbf{Y}} -\Big(y_{uv}\log\sigma(\mathbf{u}^{\mathrm{T}}\mathbf{v}) + (1-y_{uv})\log\left(1-\sigma(\mathbf{u}^{\mathrm{T}}\mathbf{v})\right)\Big)\\ &\quad + \frac{\lambda_2}{2}\sum_{r\in\mathcal{R}}\|\mathbf{I}_r - \mathbf{E}^{\mathrm{T}}\mathbf{R}\mathbf{E}\|_2^2 + \frac{\lambda_1}{2}\Big(\|\mathbf{V}\|_2^2 + \|\mathbf{E}\|_2^2 + \sum_{r\in\mathcal{R}}\|\mathbf{R}\|_2^2\Big)\end{aligned}$$

- Loss=二元交叉熵loss+知识图谱的loss+L2正则化的loss

RippleNet

- 自己写的算法推导:



RippleNet

- 实现上的一些trick
- 1.存在沉没实体(sink entities): 只有入度没有出度, 中断传播, 保持user embedding的影响
- 2.每次扩张依据下一跳邻居个数,扩张固定数量的邻居, 保持推荐的多样性, 不用担心蔓延至整个图谱.
- 3.控制最大跳数, 实际上跳数也会做一个衡量效果的超参.
- 4.每一跳可以选择多种方式更新item embedding.
- 。 。 。

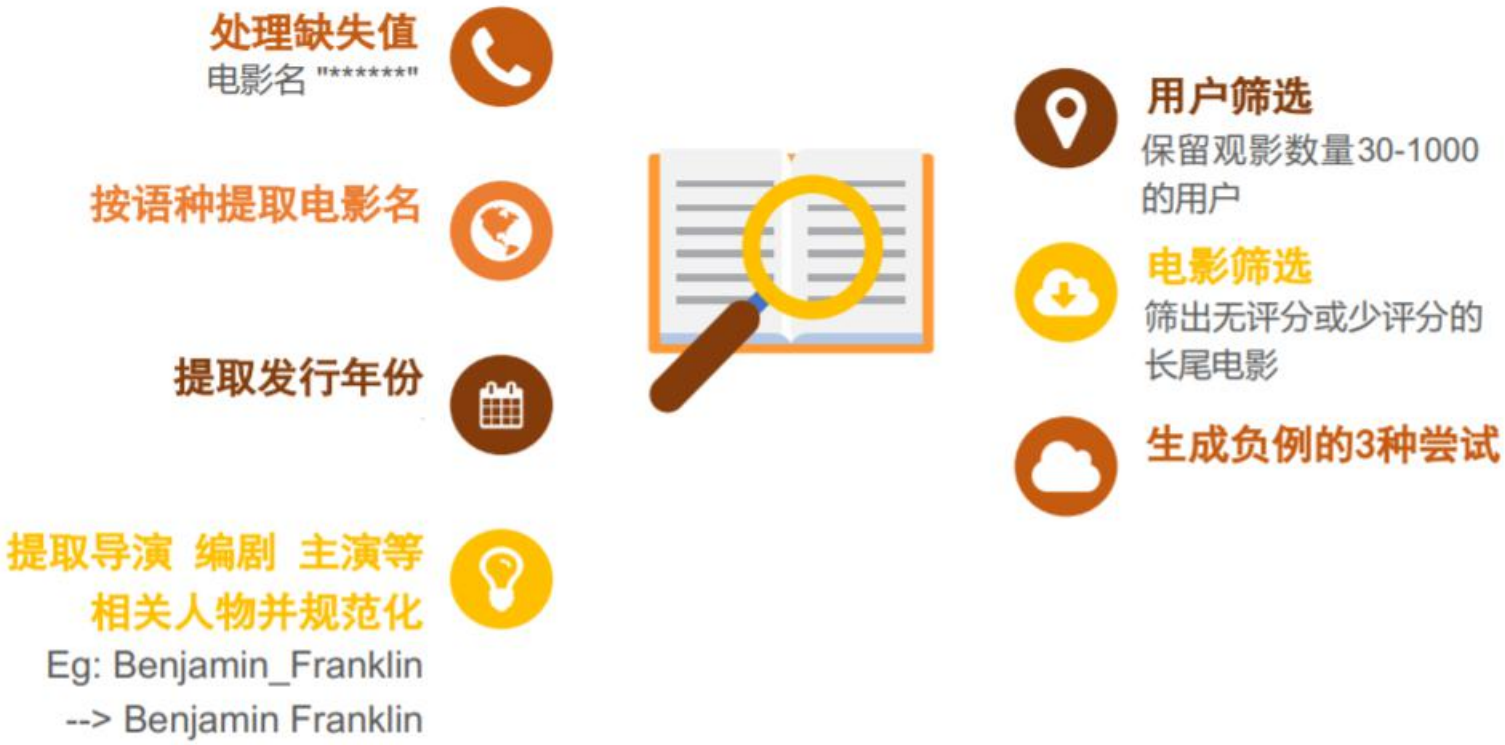
RippleNet

- item embedding的更新允许多种方式:

```
def update_item_embedding(self, item_embeddings, o):
    if self.item_update_mode == "replace":
        item_embeddings = o
    elif self.item_update_mode == "plus":
        item_embeddings = item_embeddings + o
    elif self.item_update_mode == "replace_transform":
        item_embeddings = tf.matmul(o, self.transform_matrix)
    elif self.item_update_mode == "plus_transform":
        item_embeddings = tf.matmul(item_embeddings + o, self.transform_matrix)
    else:
        raise Exception("Unknown item updating mode: " + self.item_update_mode)
    return item_embeddings
```

RippleNet

- 不同于基于路径的方法:



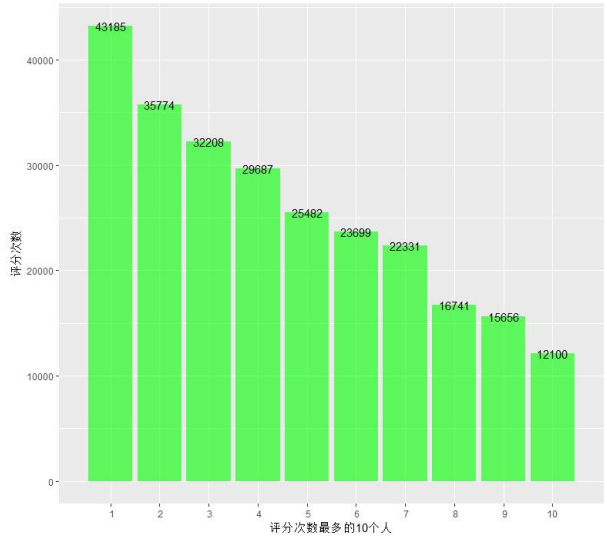
其他传统方法

- 传统上基于item-based CF的方法具有更好的效果。

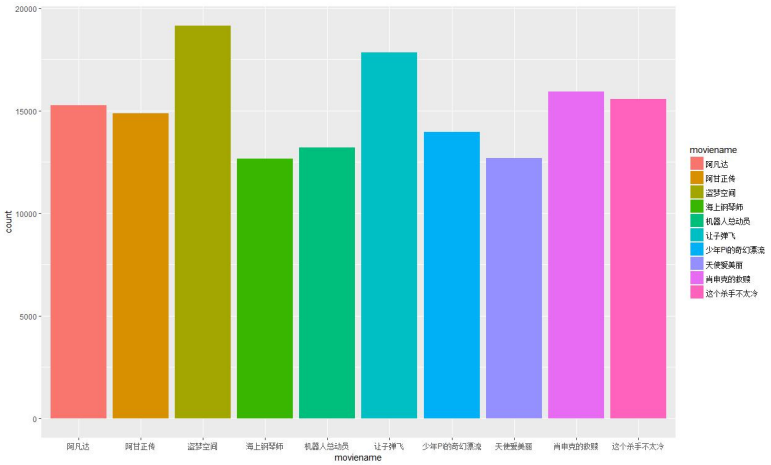
- 1. 电影的相似度比人偏好的相似更稳定一些
- 2. 数据稀疏性分析, 电影被评分的数量比人打分的数量要多。
- 3. 对于传统CF的改进: 在评估相似度的时候增加了一些惩罚项, 比如
- 1) user打分的多样性, 方差不能太小(全部打5分的用户直接删除)
- 2)item被评分的方差不能太大(如果每个人给同一部电影打分完全不一样, 认为是有问题的电影)
- 3) user打分的次数不要太少

- 4.认为数据本身存在一定的偏置.

数据分析

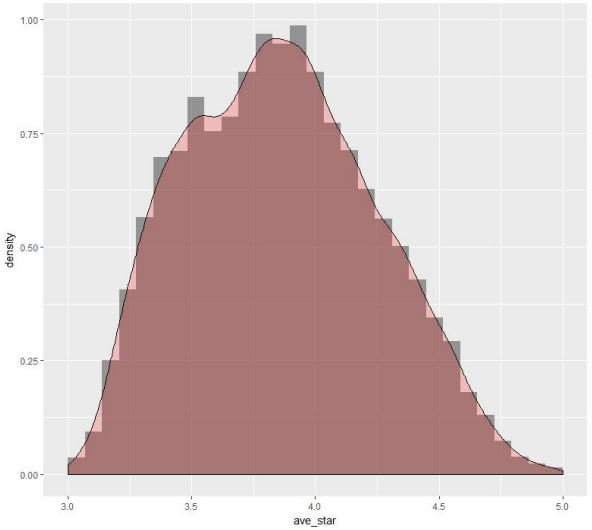


对于每一个user对电影的评分我们挑选出评分次数前10的电影

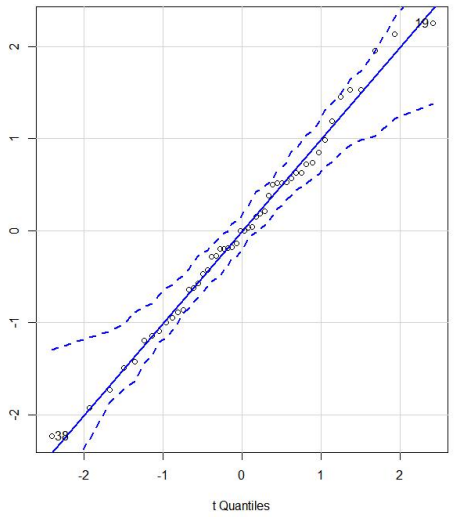


对于电影的评分人数,找出评分人数最高的10部电影.

数据分析



大部分的电影被打分在3.8分左右。



利用Q-Q图做了对被评分电影做分布检验和方差齐次性检验.各评分数据可视为一致分布。

数据分析

处理缺失值
电影名 "*******"

按语种提取电影名

提取发行年份

提取导演 编剧 主演等相关人物并规范化
Eg: Benjamin_Franklin
--> Benjamin Franklin

用户筛选
保留观影数量30-1000的用户

电影筛选
筛出无评分或少评分的长尾电影

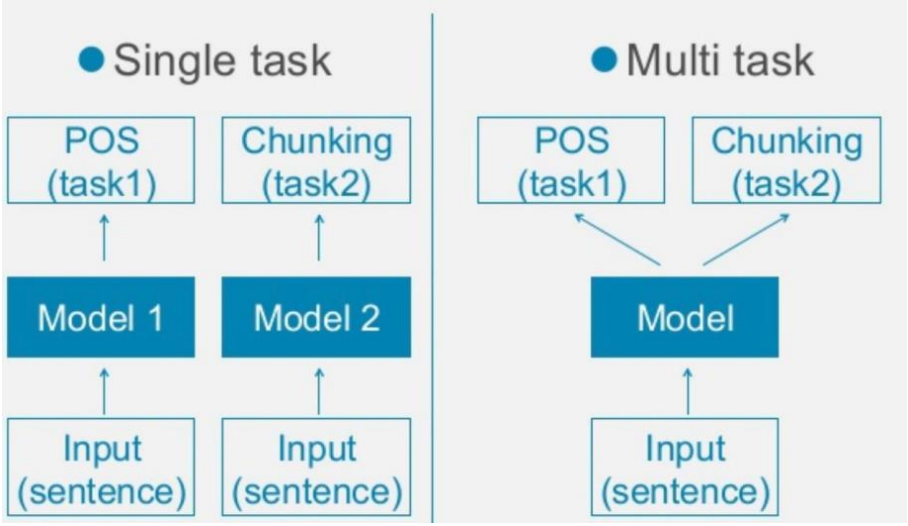
生成负例的3种尝试

数据分析


03
02
01
随机抽取和正例等数量的负实例
基于"用户喜欢看的电影占电影总数量占比很小"的原则
在热门电影里抽取用户未看过的电影当作负实例
基于"热门电影大多数人都看过
若某一用户未看，那么很可能不喜欢"的先验知识
利用SGD的方式进行SVD矩阵分解，学习得到负实例
问题：由于ratings文件中评分数据都是3-5之间，
生成的uid-mid评分也主要在3-5之间

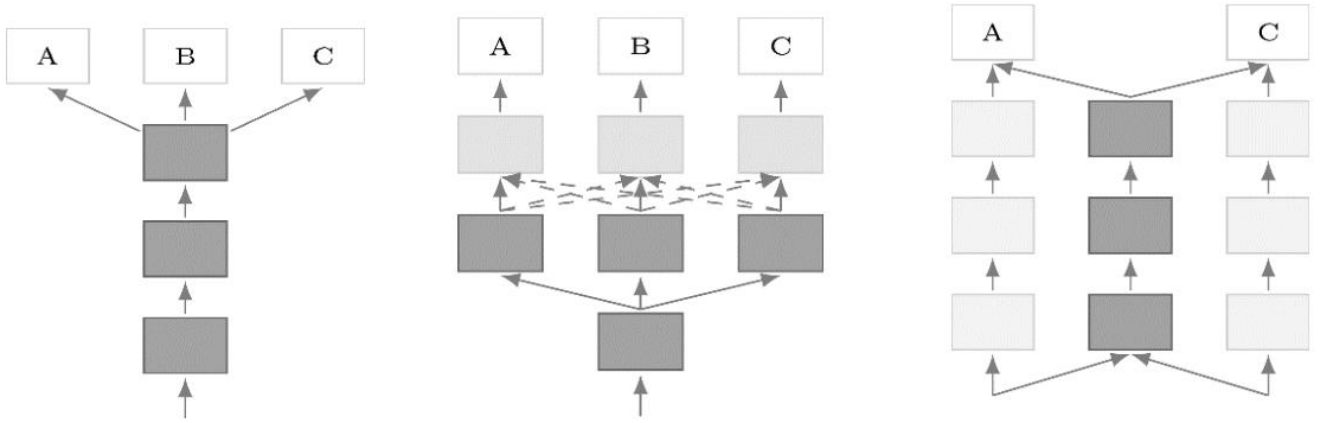
交替学习的思路

- 交替学习是一种较为创新和前沿的思路，研究方向一般是如何设计两个相关任务以及两个任务如何关联起来。从实际运用和时间开销上来说，**交替学习介于依次学习和联合学习中间。**
- 学界主流的方法是采用多任务学习。



- 对于任务1训练一个模型1，对于任务2训练一个模型2，多任务就是将两个任务放在一起用一个模型来处理。将模型学得的表示称为**共享表示**.一个简单的NLP例子就是使用做一个模型去做词性标注和组块分析.

多任务学习的几种常见模型



(a) 硬共享模式　(b) 软共享模式　(c) 共享-私有模式

硬共享: 在下面层共享，上层根据自己不同的任务做不同的设计

- 软共享: 通过不同的权重配比,私有层可以接受到不同的共享层表示
- 共享私有模式: 每个任务都可以接收到共享的表示,也可以接收到私有的表示

交替训练的思路

- Joint Training: The training is achieved in a stochastic manner by looping over the tasks:
  - 1. Select a random task.
  - 2. Select a random training example from this task.
  - 3. Update the parameters for this task by taking a gradient step with respect to this example.
  - 4. Go to 1.

- Fine Tuning: After the joint learning phase, we can use a fine tuning strategy to further optimize the performance for each task.

多任务学习中, 每个任务有自己的数据集。每一次训练的batch是从所有的数据集里面随机采样的,各种任务随机训练.
情况是有很多的任务, 每个任务有自己的数据集。每一次训练的batch是从所有的数据集里面随机采样的,各种任务随机训练.

交替训练的思路

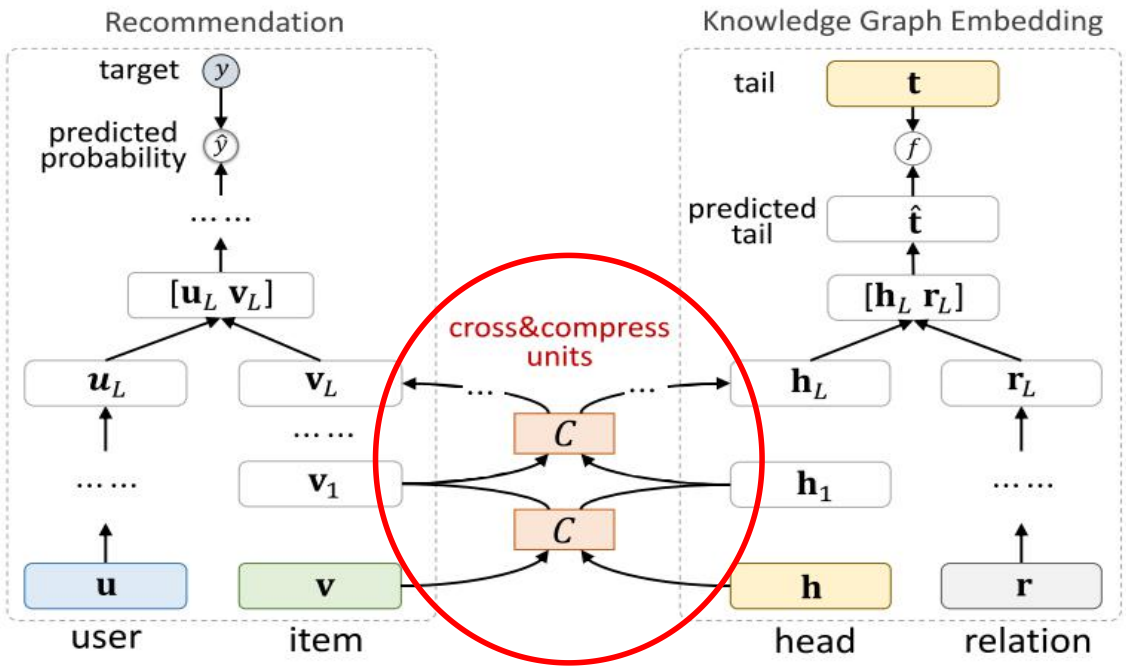
- Joint Training: The training is achieved in a stochastic manner by looping over the tasks:
  - 1. Select a random task.
  - 2. Select a random training example from this task.
  - 3. Update the parameters for this task by taking a gradient step with respect to this example.
  - 4. Go to 1.

- Fine Tuning: After the joint learning phase, we can use a fine tuning strategy to further optimize the performance for each task.

在推荐任务中，我们采用交替训练的方式：固定推荐系统模块的参数，训练知识图谱特征学习模块的参数；然后固定知识图谱特征学习模块的参数，训练推荐系统模块的参数。

模型

Multi-Task Feature Learning for Knowledge Graph Enhanced Recommendation ***WWW'19***



(a) Framework of MKR

由于推荐系统中的物品和知识图谱中的实体存在重合，所以两个任务并非相互独立。可以设计交叉特征共享单元,让两个任务交换信息的模块.他们之间的信息交叉共享可以让两者都获得来自对方的额外信息，从而弥补了自身的信息稀疏性的不足。

Thanks !