Item2vec理论与实践

Item2vec是微软2016年发表在RecSys上论文Item2Vec：Neural Item Embedding for Collaborative Filtering。

这篇文章比较朴素，基本是参照了google的word2vec方法，把item2vec应用到推荐场景的i2i相似度计算中，从实际效果看还不错。主要思想是：把item看做word2vec中的word，用户的行为序列看做一个集合（句子），item间的共现为正样本，并按照item的频率分布进行负样本采样。因此，item2vec可以看作word2vec的推广。Item2vec的本质就是Word2vec中的skip-gram + Negative sampling。

# 1背景

推荐系统中，传统的CF算法都是利用item2item关系计算商品间相似性。i2i数据在业界的推荐系统中起着非常重要的作用。传统的i2i的主要计算方法分两类，memory-based和model-based。

作者受nlp中运用embedding算法学习word的latent representation的启发，特别是参考了google发布的的word2vec（Skip-gram with Negative Sampling，SGNS），利用item-based CF学习item在低维latent space的embedding representation，优化i2i的计算。

这篇论文是微软将word2vec应用于推荐领域的一篇实用性很强的文章。该文的方法简单易用，可以说极大拓展了word2vec的应用范围，使其从NLP领域直接扩展到推荐、广告、搜索等任何可以生成sequence的领域。

# 2相关知识

## 2.1 What is Embedding

简单来说，embedding 就是用一个低维的向量表示一个物体，可以是一个词，或是一个商品，或是一个电影等等。这个embedding向量的性质是能使距离相近的向量对应的物体有相近的含义，比如 Embedding(复仇者联盟)和 Embedding(钢铁侠)之间的距离就会很接近，但Embedding(复仇者联盟)和Embedding(乱世佳人)的距离就会远一些。

除此之外Embedding甚至还具有数学运算的关系，比如Embedding（马德里） - Embedding（西班牙）+ Embedding(法国) ≈ Embedding(巴黎)

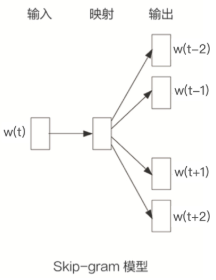
## 2.2 Why Embedding

Embedding 能够用低维向量对物体进行编码还能保留其含义的特点非常适合深度学习。在传统机器学习模型构建过程中，我们经常使用one hot encoding对离散特征，特别是id类特征进行编码，但由于one hot encoding的维度等于物体的总数，比如阿里的商品one hot encoding的维度就至少是千万量级的。这样的编码方式对于商品来说是极端稀疏的，甚至用multi hot encoding对用户浏览历史的编码也会是一个非常稀疏的向量。而深度学习的特点以及工程方面的原因使其不利于稀疏特征向量的处理。因此如果能把物体编码为一个低维稠密向量再喂给DNN，自然是一个高效的基本操作。因此，可以认为embedding是深度学习的基本操作。

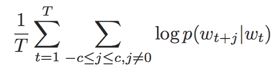
# 3 Word2vec回顾

Word2vec把words and phrases映射到一个低维语义和句法的向量空间中，用CBOW和Skip-gram两种方式。这里只是简单描述下Word2vec中Skip-gram相关的知识，因为这部分与item2vec基本一致。如果想更清楚的了解Word2vec，可以阅读**https://mp.weixin.qq.com/s?\_\_biz=Mzg4MDAzNjMyMg==&mid=2247484575&idx=1&sn=93126d100e2a7a1cd201f37e427ed522&chksm=cf7a1412f80d9d0467e3b67f44be26ed674fc4fb141641f74c77408c503505b627573daead76&token=842391589&lang=zh\_CN#rd**

Skip-gram的模型架构如下图所示：

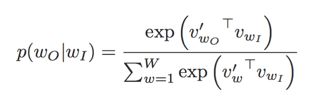


Skip-gram是利用当前词预测其上下文词。给定一个训练序列w1,w2,...,wT，模型的目标函数是最大化平均的log概率：

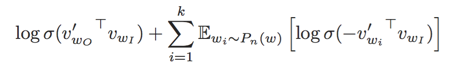


目标函数中c指context的大小。c越大，训练样本也就越大，准确率也越高，同时训练时间也会变长。

在skip-gram中，利用softmax函数定义如下：

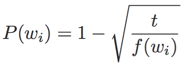


W是整个语料库的大小。上式的梯度的计算量正比于W，W通常非常大，直接计算上式是不现实的。为了解决这个问题，google提出了两个方法，一个是hierarchical softmax，另一个方法是negative sample。negative sample的思想本身源自于对Noise Contrastive Estimation的一个简化，具体的，把目标函数修正为：



是噪声分布( noise distribution )。即训练目标是使用Logistic regression区分出目标词和噪音词。具体的Pn(w)方面有些trick，google使用的是unigram的3/4方，即，好于unigram，uniform distribution。

另外，由于自然语言中很多高频词出现频率极高，但包含的信息量非常小（如'is' 'a' 'the'）。为了balance低频词和高频词，利用简单的概率丢弃词wi：



其中f(wi)是wi的词频，t的确定比较trick，启发式获得。实际中t大约在10的-5次方附近。

# 4 Item2vec算法原理

Item2vec，顾名思义，就是为item做embedding的方法。item2vec将用户的行为序列转化成item组成的句子，模仿word2vec训练word embedding将item embedding。基本思想是把原来高维稀疏的表示方式(one hot)映射到低维稠密的向量空间中，这样我们就可以用这个低维向量来表示该项目(电影)，进而通过计算两个低维向量之间的相似度来衡量两个项目之间的相似性。

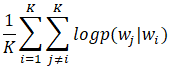
类似于Word2vec，item2vec有两种方式：CBOW和Skip-gram模型。

CBOW使用的是词袋模型，模型的训练输入是某一个特征词的上下文相关的词对应的词向量，而输出就是这特定的一个词的词向量。Skip-Gram模型和CBOW的思路是反着来的，即输入是特定的一个词的词向量，而输出是特定词对应的上下文词向量。

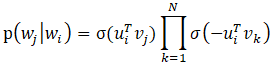
Item2vec流程：

* 从log中抽取用户行为序列
* 将行为序列当成预料训练word2Vec得到item embedding
* 得到item sim关系用于推荐

Item2vec中把用户浏览的商品集合等价于word2vec中的word的序列，即句子（忽略了商品序列空间信息spatial information）。出现在同一个集合的商品对视为正例，否则为负例。同一集合可以根据具体场景定义，例如：用户同一订单下的商品。对于集合w1,w2,…,wK目标函数：

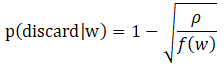


同word2vec，利用负采样，将定义为：



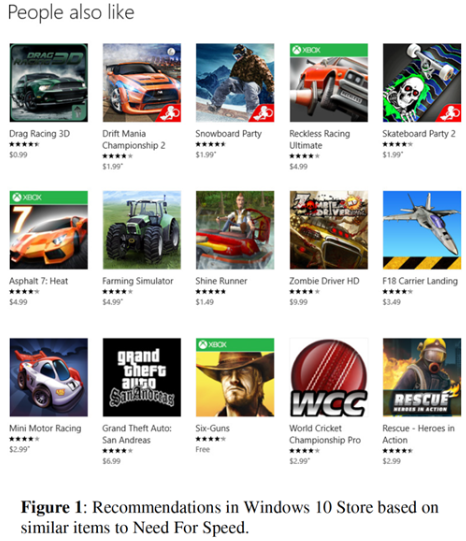


subsample的方式也是同word2vec：



最终，利用SGD方法学习的目标函数max，得到每个商品的embedding representation，商品之间两两计算cosine相似度即为商品的相似度。

paper中提到的业务问题，是在windows 10应用商店的推荐问题：在“用户也喜欢”该怎么做推荐？如下图所示。



在这样一个明确的推荐场景下，基于具体商品的推荐是更有优势的，下面根据上图进行说明：

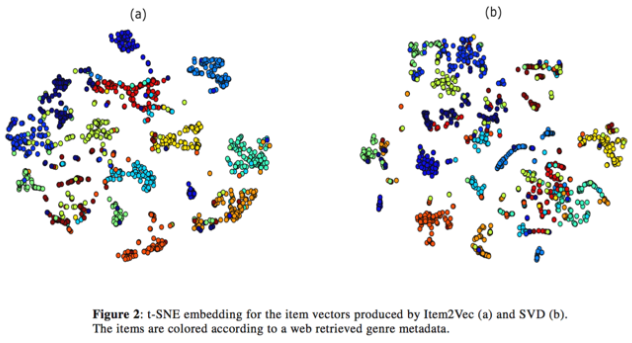
已经得知用户的兴趣点是由Need For Speed引导出来的，那么在上下文关系中，由该游戏为基础进行的协同过滤更符合逻辑。此处可以类比于电商场景下商品详情页的推荐。

在应用商店中，user的数量远大于item的数量。学习item之间相关关系的计算压力要远小于学习item与user的关系。（user和user就更不可能了，又不是社交场景）

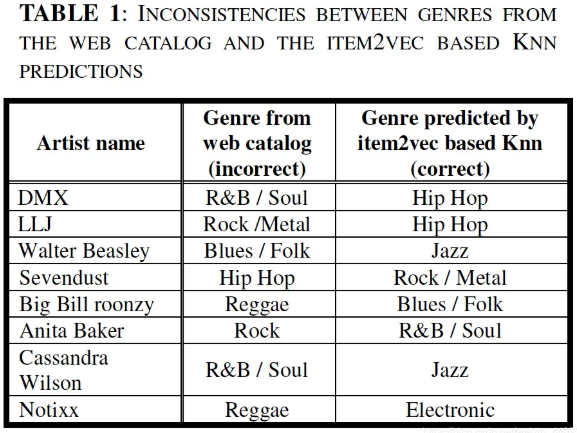
既然问题定位到学习item之间的相关关系，这里使用skip-gram的做法就是Item2Vec的灵魂所在了。Item2vec并没有加入对时间信息的考量，因为序列数据产生的时间并不重要。继而，序列内item的相对位置也不重要，只要是在一个序列内，所有item组成的pair对都是可以作为正样本的，也就是skip-gram的window size是自适应的，是和序列长相等。

# 5 Item2vec效果

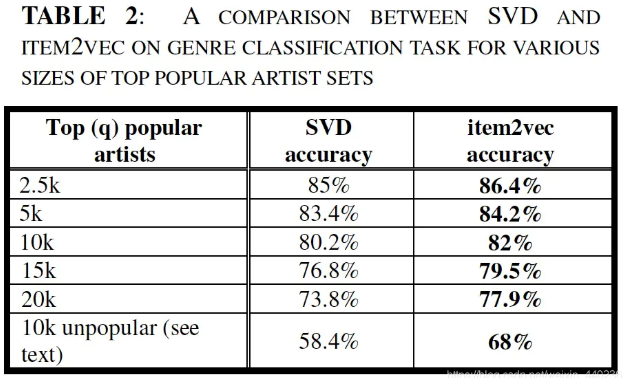
对比的baseline方法是基于SVD方法的用户embedding得到的相似度，SVD分解的维度和item2vec的向量维度都取40。数据是应用在music领域的，作者利用web上音乐人的类别进行聚类，同一个颜色的节点表示相同类型的音乐人，结果对比如下：



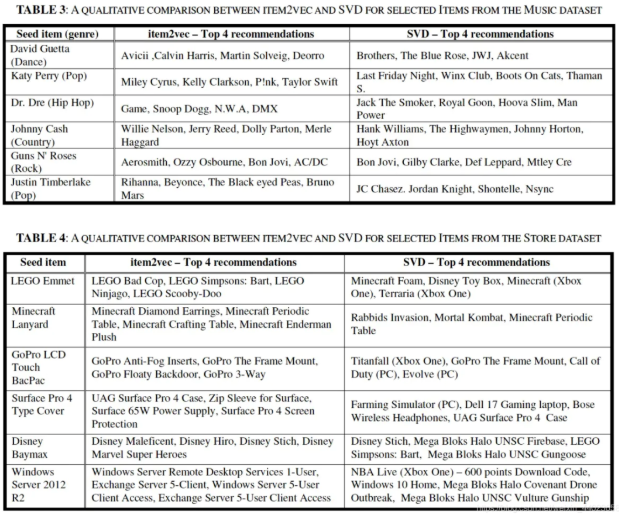
同种类型的artists用相同颜色进行标识，结果显示，图a是item2vec的聚合效果，图b是SVD分解的聚合效果，看起来item2vec的聚合效果更好些。



**对topq的artists进行分类预测（KNN），准确率较高，且对历史数据较少的artists表示效果更好。**



item2vec和SVD直观效果比较，表中列除了seed item以及与其最相似item



可以得出结论：

1）item2vec可以探索出除了“类型”之外其他的相似性；

2）item2vec提供列更相关的item，且对于信息较少的情况下，表现较佳

# 6 item2vec与MF的区别

首先，二者都应用了隐向量来表征实体特征，不同的是，传统的MF通常是user-item矩阵，而Item2Vec通过滑动窗口样本生成的方式构造出的则更像是item-item矩阵；另外，二者得到隐向量的方式也不同，MF利用均方差损失，使预测得分与已有得分之间的误差尽可能地小，而Item2Vec则是利用空间信息并借助了最大似然估计的思想，使用对数损失，使上下文关系或者共现关系构造出的正样本的item Pair出现的概率可能地大；此外训练 Item2Vec的时候还要引入负样本，这也是与MF不同的地方。

对于二者在推荐效果上的差异，一个经验是传统MF推荐会让热门内容经常性排在前面，而Item2vec能更好的学到中频内容的相似性。Iterm2Vec加上较短的时间窗口，相似推荐会比MF好很多。

# 7 通俗版本Item2vec

把场景转换到一个新闻媒体如A公司。

在A公司的多个页面中，电商公司B有他们的一个主页，专门介绍他们公司一些产品促销，抢购和发布会什么的。

公司A目前有很多用户的浏览数据，如用户u浏览了公司A的页面a1，a2，a3等。

把这些数据处理一下，整合成word2vec能处理的数据，如下

U1 a1,a2,a3……

U2 a2,a3,a5,……

U3 a1,a3,a6,……

其中u1，u2，u3表示不同的用户，后面的一串表示这些用户的浏览记录，如U1 a1,a2,a3表示用户u1先浏览了页面a1，再浏览a2，然后浏览了a3,……

这些数据还不符合word2vec的输入数据格式，把第一列去掉，变成下面的样子

a1,a2,a3……

a2,a3,a5,……

a1,a3,a6,……

这些数据就可以作为word2vec的输入数据了。

就把这些数据作为word2vec的训练数据，词向量维度为3，进行训练，完成后得到下面的输出

A1 (0.3,-0.5,0.1)

A2 (0.1,0.4,0.2)

A3 (-0.3,0.7,0.8)

……

An (0.7,-0.1,0.3)

就得到了每个页面的向量。

这些向量有啥意义呢？其实单个向量的意义不大，只是用这些向量可以计算一个东西——距离，这个距离是页面之间的距离，如页面a1和a2可以用欧式距离或者cos距离计算公式来计算一个距离，这个距离是有意义的，表示的是两个网页在用户浏览的过程中的相似程度（也可以认为是这两个页面的距离越近，被同一个人浏览的概率越大）。注意这个距离的绝对值本身也是没有意义的，但是这个距离的相对大小是有意义的，意思就是说，假设页面a1跟a2、a3、a4的距离分别是0.3、0.4、0.5，这0.3、0.4、0.5没啥意义，但是相对来说，页面a2与a1的相似程度就要比a3和a4要大。

那么这里就有玄机了，如果页面a1是电商公司B的主页，页面a2、a3、a4与a1的距离在所有页面里面是最小的，其他都比这三个距离要大，那么就可以认为同一个用户u浏览a1的同时，浏览a2、a3、a4的概率也比较大，那么反过来，一个用户经常浏览a2、a3、a4，那么浏览a1的概率是不是也比较大呢？从实验看来可以这么认为的。同时还可以得到一个推论，就是用户可能会喜欢a1这个页面对应的广告主的广告。

这个在实验中实际上也出现过的。这里模拟一个例子吧，如a1是匹克体育用品公司在媒体公司A上的官网，a2是湖人队比赛数据页，a3是热火队的灌水讨论区，a4是小牛队的球员讨论区。这个结果看起来是相当激动人心的。

根据这样的一个结果，就可以在广告主下单的那个页面上增加一个条件——经常浏览的相似页面推荐，功能就是——在广告主过来选条件的时候，可以选择那些经常浏览跟自己主页相似的页面的用户。举个例子就是，当匹克体育用品公司来下单的时候，页面上给它推荐了几个经常浏览页面的粉丝：湖人队比赛数据页，热火队的灌水讨论区，小牛队的球员讨论区。意思是说，目标人群中包括了经常浏览这三个页面的人。

# 8 实用Tricks

## 8.1为什么选择的是SGNS而不是其它的组合

Item2vec为什么采用Skp-Gram + Negative Sampling这种组合呢？因为效果好，而在很多文章中也提到了SGNS这种组合下的实际业务提升要好一些（但并不能一刀切，只是说大多数业务场景下SGNS的效果好，但还是要视具体的情况而定）

## 8.2实验场景的选择

对于这种item相似的算法，如何选择合适的实验场景呢？Item2vec论文中提到的是使用Windows10 App Store的「看了又看」推荐场景，即某个App的相似App推荐场景，这种场景下，对Item相似类算法进行是很合适的。

但是比如把item sim items加入到典型的「recall -> rank」场景中，其实达到的效果并没有那么好，但不能说不合适，这取决于截断的数目，即每个item取多少相似的item。因为在召回中并不会区分item之间的顺序，比如top100，把100个item全部加到召回池中，并不会区分这100个item之间的顺序，这就会在一定程度上丢失掉这种相似的信息，极端情况下，假设我们的item总数为1000个，而召回时将item的相似item1000全部加入到召回池中，这种极端情况下就失去了个性化的意义。

因此选择一个合适的业务实验场景去评估我们的算法是极其重要的，否则得出的结论也没有什么说服力！

## 8.3负采样不代表是均匀的随机负采样

均匀的随机负采样就代表采样时对所有的负样本采样的概率是一样的，但其实这是不符合实际的数据分布概率的。

因此论文中也使用到了一种非均匀的随机负采样技术，其表达式为：



其中，f(w)表示的是item w的频次，表示是人为设定的参数，是一个经验值（论文中针对App Store数据集的设定值是10的-3次方，音乐数据集设定值是10的-5次方）

## 8.4 Item2vec基于SGNS的改进点

其改进点为修改window size为set的长度，即从原来的定长变成了变长，其它的保持不变。

Item2vec丢失了集合中的时间和空间信息。

## 8.5 Item2vec的等效表达方法

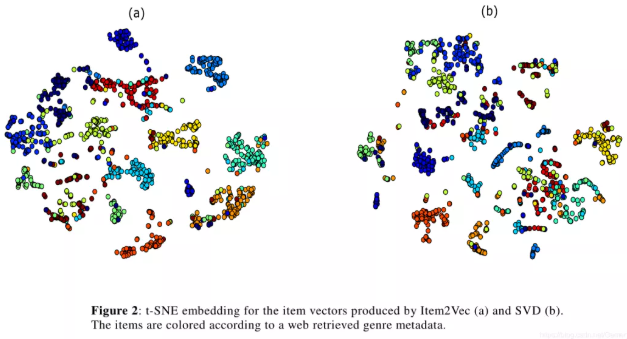
论文中特意指出采用保持word2vec的SGNS不变，仅将set集合中的item进行随机的排序可以达到和Item2vec同样的效果，相关的实验也对此进行了佐证。

因此在很大程度上方便了 Item2vec的等效表达，即采样Word2vec的算法代码同样可以得到Item2vec中的Item的embedding表示。

## 8.6基于Item Embedding计算相似性的实验思路

### 8.6.1聚类数据可视化

依据embedding进行item聚类，并在二维空间中进行数据展示，观察效果。比如论文中使用的t-SNE进行可视化，如下图所示，(a) 为基于Item2vec产出的item embedding进行的聚类可视化，(b)为基于SVD产出的item vector进行的聚类可视化，肉眼可见(a)要比(b)的效果好很多。

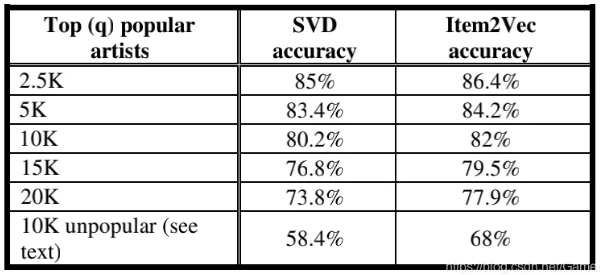


### 8.6.2 top N item的类型一致性检验

首先基于KNN选取每个item的相似items，top N个，然后根据类型一致性检验统计top N中的一致性占比，继而进行对比分析。

在进行类型一致性检验时，这里选择了一些比较受欢迎的item，论文中关于「受欢迎」和「不受欢迎」的item定义方法为：如果一个item被用户交互次数少于15，则被认为是不受欢迎的。

实验结果如下图所示：



## 8.7聚类可视化的分析和带给我的思考（精华）

在上边提到了使用聚类可视化进行item2vec的效果表达，论文中指出上图(a)中那些同色的簇类中被其它颜色簇类污染的item中，有很多是标签打错或者类型交叉引起的。

那么这里是否可以通过可视化和分析对作品的标签或者其它属性进行反馈和修正呢？比如说上标签打错的item反馈给运营，让运营进行评估和修正，然后继续使用类似的方法进行评估，再反馈修正，以此达到一个良性的循环，从而解决运营打标签，人工再校验的复杂流程和容易出错的问题。

## 8.8实验参数的学习

论文中提到了了Item2vec进行实验时的参数值，这也为我们在实际业务中进行相关尝试，提供了初始的、经验上的模型参数值。

迭代次数：20

负采样数：1:15

embedding维度：App数据集为40，Music数据集为100

人工设定值：App数据集为10的-3次方，Music的数据集为10的-5次方

# 9实践

参考：<https://github.com/jpegbert/NLP_Coding/tree/master/item2vec>

包含python实现item2vec、gensim实现item2vec以及基于MovieLens数据集得到的item2vec实现相似电影的推荐功能。

# 10总结

Item2Vec舍弃时间信息，对skip-gram做出一定的改造，是一种描述item间关系的新思路。实践证明，embedding技术对于工业场景来说有着很大的价值和应用前景。Item2vec的缺点是相似度的计算还只是利用到了item共现信息，忽略了user行为序列信息，没有建模用户对不同item的喜欢程度高低。

参考：

<https://zhuanlan.zhihu.com/p/24339183?refer=deeplearning-surfing>

<https://mp.weixin.qq.com/s/lnGvdLc4U6b0kniyFfPbbw>

<https://mp.weixin.qq.com/s/xXYkySG7JHuCux862jXwcQ>

<https://www.cnblogs.com/hellojamest/p/11766401.html>

<https://mp.weixin.qq.com/s/pzQYYm11G-3DpKuWxe0rJQ>

<https://mp.weixin.qq.com/s/ci99SQuYNtmalLnwMmBb1g>

<https://mp.weixin.qq.com/s/GaqC8oggHc2GlsTIvCanvw>

<https://mp.weixin.qq.com/s/CQaG7a1E9xGhUCygR9dEhA>