1. 首先从https://github.com/wsdream/WS-DREAM下载代码，配置好环境，需要在ubantu环境下配置代码，window比较麻烦。（PS: 或者<https://github.com/wsdream/WS-DREAM/tarball/master>）

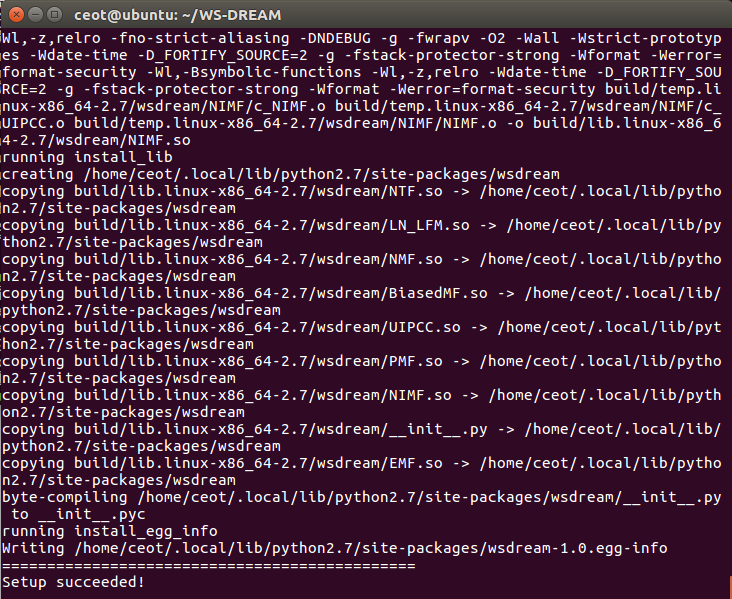
Python 2.7 (https://www.python.org)

Cython 0.20.1 (http://cython.org)

numpy 1.8.1 (http://www.scipy.org)

scipy 0.13.3 (http://www.scipy.org)

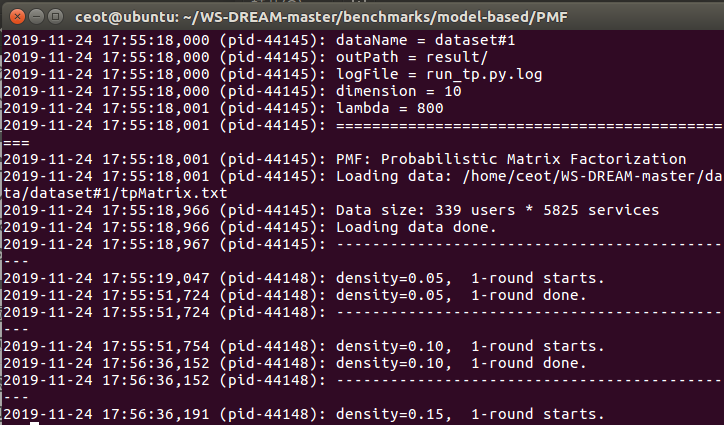
1. 进到~/WS-DREAM目录下运行编译指令python setup.py install –user，会出现一堆warning,不用管出现setup succeeded即可。



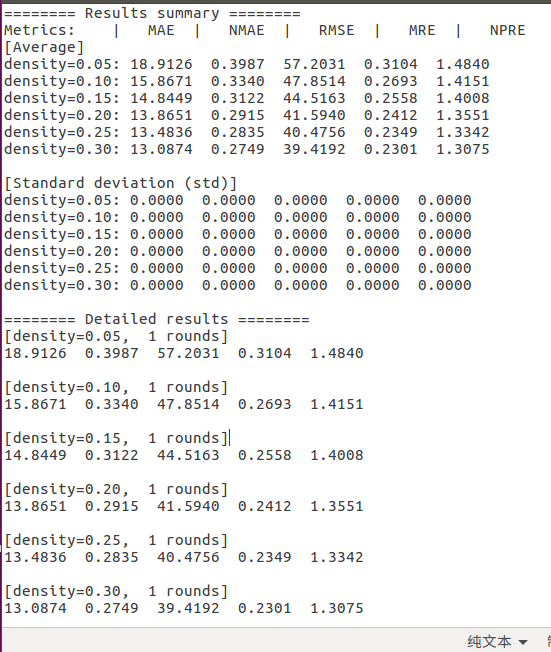
1. 运行时候发现数据集没有下，直接手动下，运行python文件需要科学工具。
2. **PMF方法**

1.首先测试model-based/PMF

2.Python run\_tp.py运行



3.运行结果：



4.查阅实现论文Personalized Reliability Prediction of Web Services，Web服务的个性化可靠性预测。论文中提出两种方法：两种个性化的Web服务可靠性预测方法，即基于邻域的方法和基于模型的方法。而代码是基于模型的实现方法的，具体实现过程如下：

考虑一个m×n用户项矩阵P，尝试使用一个l因子模型来找到两个矩阵W（m行和l列）和H（l行和n列），使得：



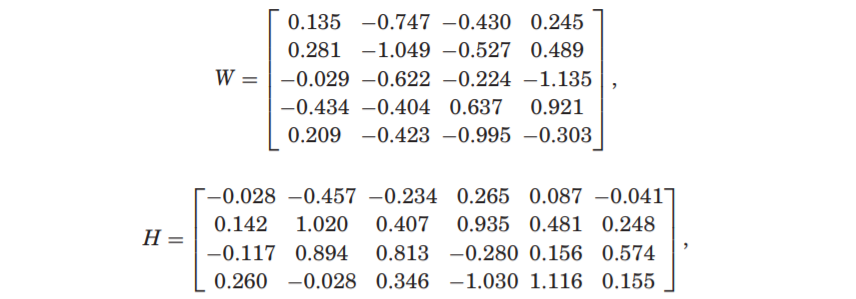
其中l是因子数。

选择W的列数和H的行数，以便W和H的乘积将成为P的近似值。

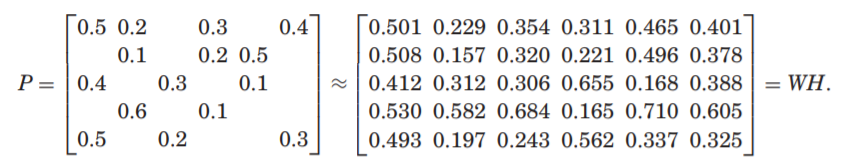
此矩阵分解过程（即，将用户项矩阵P分解为两个矩阵W和H）具有明确的物理含义：H的每一列都是一个因子向量，包括Web服务的l个因子的值，而每一行 W是用户的用户特定系数。

用户的Web服务使用经验与特定于用户的系数和Web服务因子向量的线性组合相对应。

例如，采用具有四个因子（l = 4）的矩阵分解技术，可以将图1中的用户项矩阵分解为两个矩阵W和H：

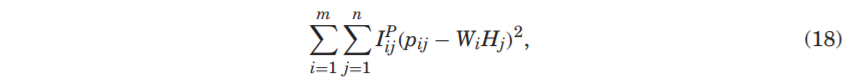


其中W的每一行包含一个用户的四个特定于用户的系数，H的每一列具有Web服务的四个因子的值。W和H的乘积是原始用户项目矩阵P的近似值：



从这个介绍中，可以看到矩阵W和H是未知的，需要使用用户项矩阵P中的可用值来估计。矩阵W和H通常是不唯一的，需要通过最小化WH和P之间的距离。由于仅部分观察到用户项矩阵P，如果可以找到矩阵W和H，则可以使用这两个矩阵的乘积来预测用户项矩阵P中未观察到的条目。

P和WH之间差异的最常见度量是平方误差，可以通过以下公式计算

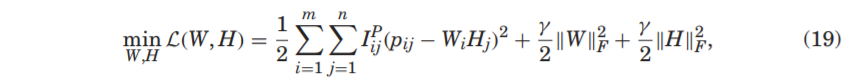


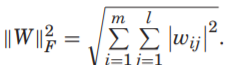
其中IPij是指示符函数，如果在用户项矩阵中有pij值可用（指示Web服务j先前已被用户i调用），则等于1，否则等于0，否则Wi是指示符函数的第i行 矩阵W（代表用户i的特定于用户的系数），Hj是矩阵H（代表Web服务j的因子向量）的第j列。

分解问题可以表示为：给定部分观察到的用户项矩阵P，找到两个矩阵W和H，它们使和平方误差最小，可以通过等式2计算。

由于找到W和H的目的是预测矩阵中未知的缺失值，因此这两个矩阵应避免过拟合所观察的值，以使所得模型对未知值具有良好的泛化性能。

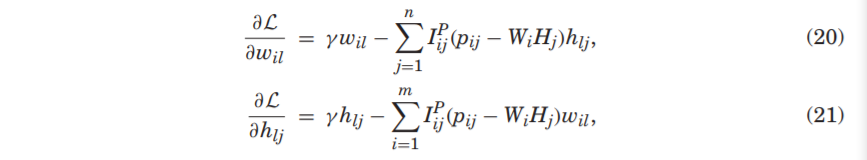
通过添加W和H规范的约束以惩罚W和H的大值，那么有以下优化问题：



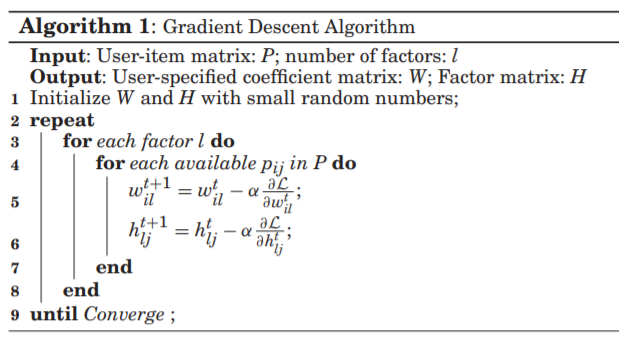
其中γ控制正则化的程度，以惩罚矩阵W和H中的大值以避免过拟合问题，而·2 F表示Frobenius范数[Golub and Loan 1996]，其定义为矩阵和的和的平方根矩阵中值的绝对平方。例如，可以通过以下公式计算||W||2 F（m行l列）：等式中的优化问题。（19）用二次正则项最小化平方和误差目标函数。

方程给出的目标函数的局部最小值，通过执行梯度下降可以找到（19）。梯度下降是一种用于最小化作为可微函数之和编写的目标函数的优化方法。梯度下降算法循环遍历用户项矩阵中的所有可用值，并逐个训练l个因子。一个梯度步骤旨在减小仅一个值的预测误差的平方。

可以通过以下方式计算梯度：

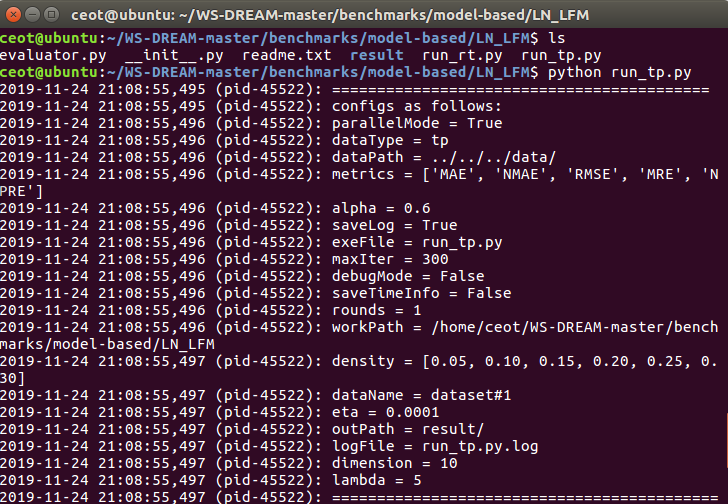


最后算法流程为：

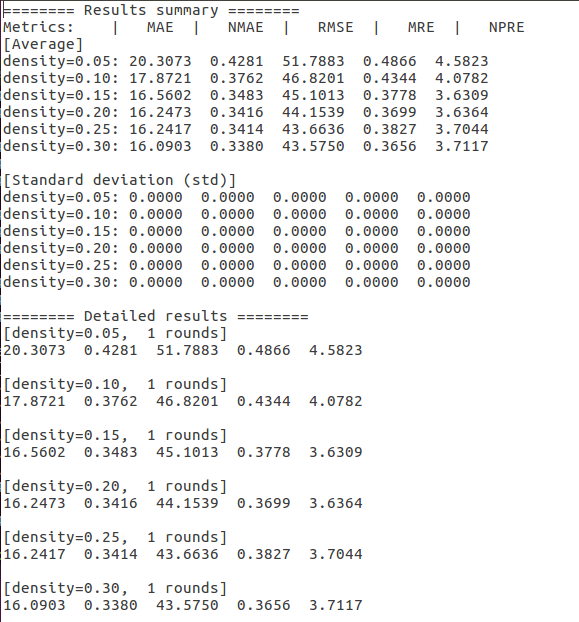


算法1显示了梯度下降算法的迭代过程，用于解决方程式中的优化问题。（19）。首先用小的随机值初始化矩阵W和H。然后，该算法以与梯度相反的方向与α成比例的幅度迭代更新矩阵W和H。wt il表示第t次迭代中wil的值。参数α称为学习率，它控制迭代的速度。当方程中给出目标函数的局部最小值时，该过程停止，（19）已达到。

1. **LN\_LMF**
2. 测试model-based/ LN\_LMF。
3. python run\_tp.py运行。



1. 运行结果为：

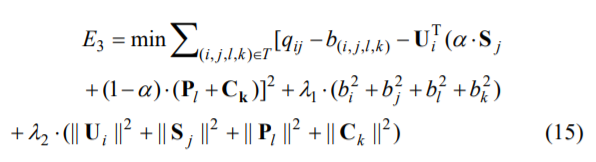


1. 论文为Personalized QoS Prediction for Web Services using Latent Factor Models，使用潜在因子模型的Web服务个性化QoS预测。在文中，作者集中于LFM方法之一的奇异值分解（SVD）方法，其中将用户和项目映射到相同的潜在因子空间以进行可行的比较。每个用户或项目都与一个潜在特征向量相关联，该潜在特征向量包含低维，以解释潜在因子与其自身之间的关系。更具体地，在QoS预测的问题中，对于用户服务QoS矩阵，在潜在空间试图自动解释从用户发布的服务调用过程中获取的QoS值。

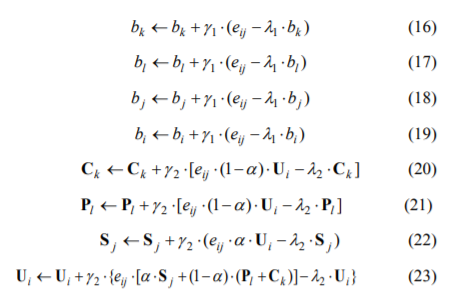
论文中最终模型由扩展基线估计模型，潜在因子模型和潜在邻居模型三者组成。作者提出名为LN-LFM的最终模型，该模型集成了所有三个模型：



其中参数α是操纵系数，α∈[0,1]。与以前的模型相同，利用适当的正则化项来避免对此模型的过度拟合，例如：



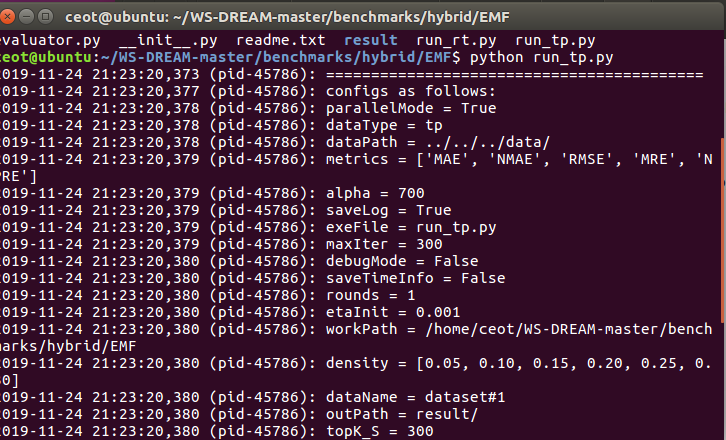
可以采用简单的随机梯度下降法来解决此优化问题。在随机梯度下降的学习过程中，对于每个训练样本，我们计算实际值与预测值之间的差，即eij =qij qij^。每次单次迭代结束时，我们都会通过以下方式更新模型参数：

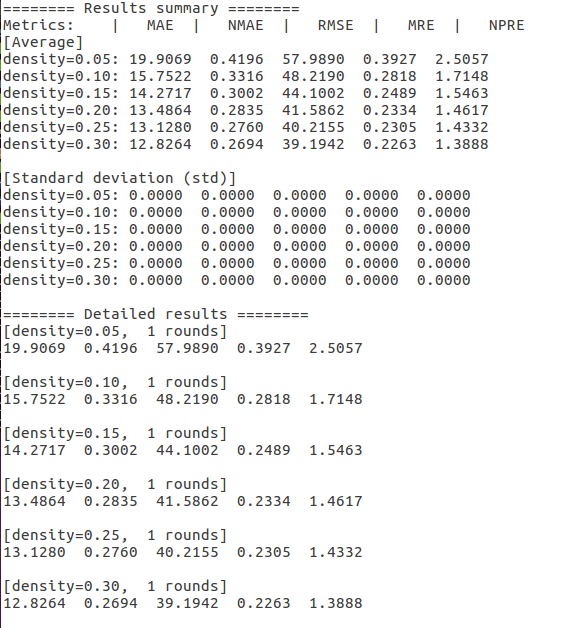


在此，参数1γ，2γ均表示学习率。

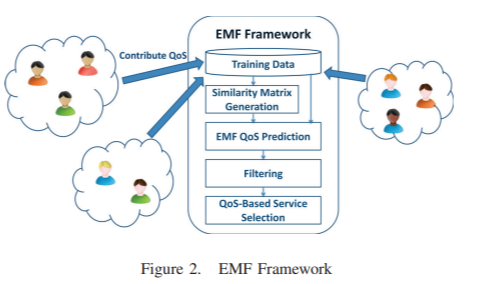
经过一定数量的迭代循环后，我们将目标函数最小化以解决平方问题，然后获得最终的预测模型。为了简化下面的实验，将参数λ1，λ2和γ1，γ2分别设置为相同的值，即，λ1＝λ2和γ1＝γ2。

1. **EMF**
2. 测试hybrid/ EMF。
3. python run\_tp.py运行。



1. 运行结果为：
2. 
3. 论文为An Extended Matrix Factorization Approach for QoS Prediction in Service Selection，服务选择中QoS预测的扩展矩阵分解方法。论文提出了一种具有关系正则化的扩展矩阵分解（EMF）框架，以进行丢失QoS值的预测。首先从一般角度阐述矩阵分解（MF）模型。为了准确地收集人群的智慧，我们在用户端和服务端采用了不同的相似性度量来识别邻居。然后，我们在邻域内系统设计两个新的关系正则化术语。最后，将两个术语组合到一个统一的MF框架中，以预测丢失的QoS值。

下图说明了作者提出的EMF框架的一般服务选择框架。首先，从异构数据源收集QoS信息。通过充分利用人群的智慧，和通过不同的计算技术生成相似度矩阵。然后，将源信息和相似关系输入到我们的EMF引擎中，从而预测丢失的QoS值。在观察到大量QoS信息之后，对一些不相关的服务进行了过滤，以减少用于基于QoS的选择的服务数量。最后一步将自动选择高质量的服务，并将其推荐给公众。



在现实世界中，有m个用户和n个Web服务。它们构成一个m×n用户服务矩阵R，每个条目rui代表一个QoS值，该值记录了用户u执行的Web服务i的特定使用信息。如第1节所述，R非常稀疏，因此包含许多丢失的QoS值。在本文中研究的问题是如何有效和高效地预测用户服务矩阵R的缺失QoS值。

为了解决这个问题，低秩矩阵分解[7]模型被广泛使用。MF将用户项矩阵分解，从而进行准确的预测。目的是将用户和项目都映射到低维d的联合潜在因子空间，以便用户与项目的交互可以被捕获为该空间中的内部产品。低维MF技术的前提是，只有少数几个因素会影响用户与项目的交互，而每个因素如何影响用户都会影响用户的交互体验。

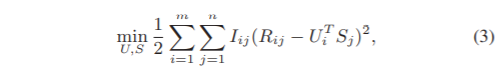
在本文中，我们关注于一个m×n用户-服务交互矩阵R。该矩阵可以大致分为U和S两部分，受d-rank因子约束：



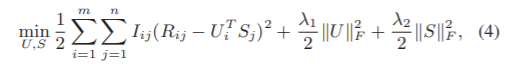
其中d <min（m，n）的U∈Rd×m和S∈Rd×n分别表示用户特征空间和服务特征空间。奇异值分解（SVD）[4]技术通过最小化以下项，用U和S逼近原始矩阵R：



其中·F表示F robenius范数。在现实情况下，原始矩阵R仅包含一些服务调用记录。这个稀疏的问题导致在实践中进行了以下修改：



其中Iij充当指标，当用户ui与服务sj交互时，Iij等于1，否则等于0。为避免模型过度拟合的问题，涉及与U和S有关的两个正则化术语如下：

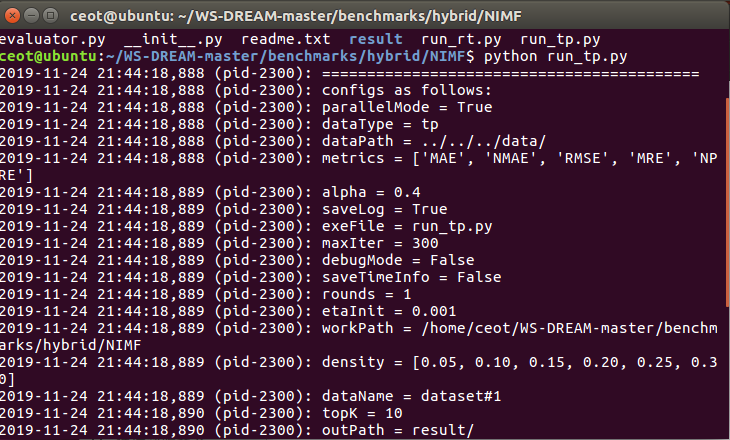


其中λ1和λ2是学习率。等式中的优化问题。（4）用二次正则项最小化平方误差和目标函数。上面的形式广泛用于推荐系统。最后合并关系正则化项以改造传统的矩阵分解模型。

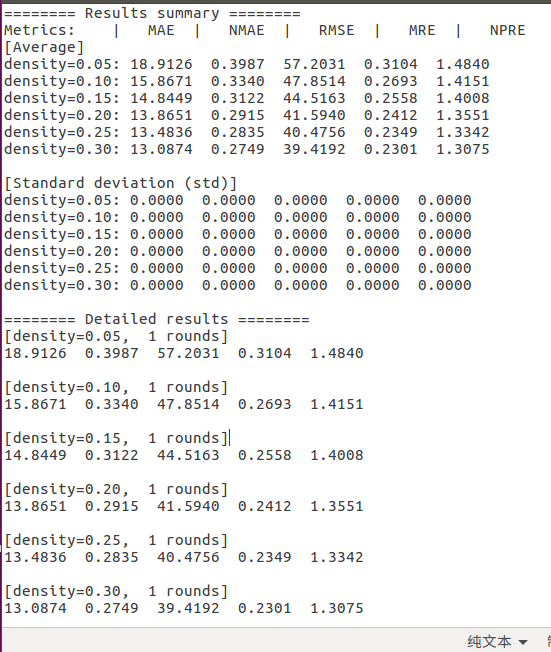
1. **NIMF**

1.测试hybrid/NIMF。

2.python run\_tp.py运行。



3.运行结果为：

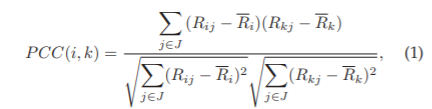


4.论文为：Collaborative Web Service QoS Prediction via Neighborhood Integrated Matrix Factorization，通过邻域集成矩阵分解的协作Web服务QoS预测。论文提出了一种利用Web服务用户过去的Web Service使用经验的协作式Web服务质量（QoS）预测方法。首先将用户协作的概念应用于Web服务QoS信息共享。然后基于收集到的QoS数据，设计了一种邻域集成方法来进行个性化Web服务QoS值预测。

给定一个m×n用户项矩阵R，它由m个服务用户和n个Web服务组成，此矩阵Rij中的每个条目代表服务用户i观察到的Web服务j的某个客户端QoS属性的值。如果用户i之前未调用Web服务j，则Rij = null。在useritem矩阵中使用可用的Web服务QoS值，这些值是从不同的服务收集的对于用户，可以通过Pearson相关系数（PCC）或矢量空间相似度（VSS）计算不同服务用户之间的相似度。PCC和VSS是两种常用的相似度计算方法。

PCC通常可以实现比VSS更高的性能，因为前者考虑了用户价值风格的差异并可以实现高精度。因此，在我们的方法中，我们采用PCC进行相似度计算。

使用PCC，可以根据两个用户i和k在共同调用的Web服务上观察到的QoS值，使用以下公式来计算它们之间的相似度：



其中J是用户i和用户k都调用的Web服务的子集，Rij是服务用户i观察到的Web服务j的QoS值，Ri和Rk是服务观察到的不同Web服务的平均QoS值 用户i和k。

根据此定义，两个服务用户i和k的相似性PCC（i，k）在[-1,1]的区间内，其中较大的PCC值表示较高的用户相似性。

在计算了当前用户与其他用户之间的相似度之后，可以基于PCC值来标识一组Top-K相似用户。实际上，服务用户可能具有有限数量的相似用户。传统的Top-K算法忽略了这个问题，仍然包括PCC值为负的异类用户，这将极大地影响预测准确性。

在我们的方法中，我们排除了具有负相关性（负PCC值）的不同服务用户。因此，对于服务用户i，可以通过以下等式来标识一组相似的用户T（i）：

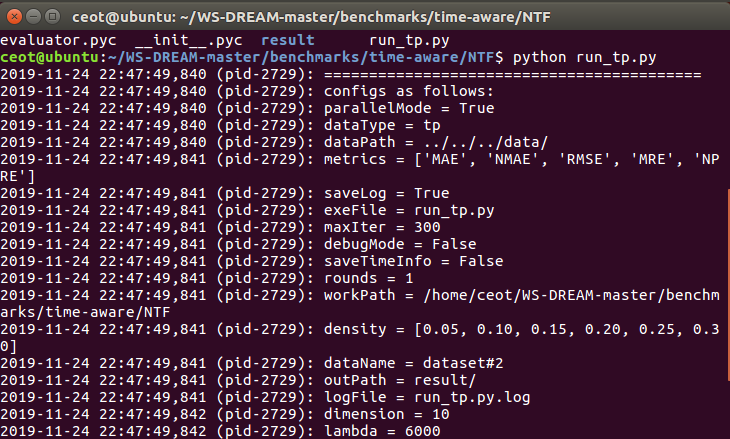


其中，Top-K（i）是当前用户i的Top-K个相似用户的集合，而PCC（i，k）是用户i和用户k之间的PCC相似度值，可以通过等式计算得出。（1）。注意，Top-K关系不是对称的。用户k在用户i的Top-K邻居中，不必指示用户i也在用户k的Top-K邻居中。有了邻域信息，现在可以设计用于QoS值预测的邻域集成​​矩阵分解模型。

1. **NTF**

1.测试time-aware/NTF。

2.python run\_tp.py运行。

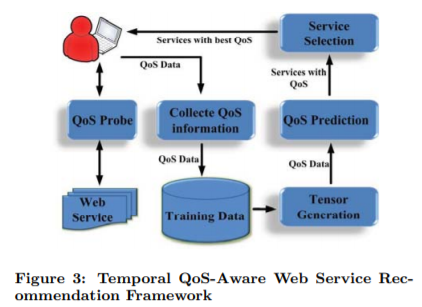


3.运行结果：(采用database#2数据集过大，虚拟机跑不出结果，内存超出错误)

4.论文为Temporal QoS-Aware Web Service Recommendation via Non-negative Tensor Factorization，通过非负张量分解的时间QoS感知Web服务建议。论文通过考虑第三动态上下文信息，提出了一个时间QoS感知Web服务推荐框架，以预测各种时间上下文下丢失的QoS值。此外，将此问题形式化为广义张量因子分解模型，并提出了一种能够处理用户服务时间模型的三元关系的非负张量因子分解（NTF）算法。

对于Web服务用户，服务QoS属性不如服务提供者声明的那么准确。为了获得每个服务用户的准确Web服务QoS值，论文采用了基于时间QoS的Web服务推荐框架来预测QoS值。如图所示，QoS预测框架收集来自不同服务用户的Web服务QoS信息。

如果Web服务用户的服务QoS信息贡献超过阈值，则可以通过我们的预测框架获得服务QoS值预测。



服务QoS信息贡献越大，可以实现更高的QoS值预测精度。在收集了大量的QoS信息后，我们对训练数据过滤了一些劣质的QoS信息，并使用预测引擎生成了用于预测丢失的QoS值。

遵循以下步骤：给定具有用户服务交互作用的时间信息的Web服务QoS数据集，请在给定的时间上下文下向每个用户推荐最佳服务列表。

为了说明这些概念，给出以下示例。

一个玩具示例：考虑在特定时间上下文中向用户推荐服务的实例，该实例在本文中分配给了服务调用时间。

然后，用户，服务，时间三元组具有以下属性：•用户：向其推荐Web服务的所有服务用户的集合； 它定义为UserID。

•服务：可以推荐的所有Web服务的集合； 它定义为ServiceID。

•时间：用户调用服务时Web服务的调用时间； 它定义为TimeID。

然后，分配给用户的服务调用的服务QoS值还取决于调用服务的位置和时间。

例如，向特定位置的用户推荐特定服务，这在很大程度上取决于他们计划何时调用它。每个QoS值均根据用户ID，服务ID和时间ID进行三维描述。因此，QoS值表示为三维空间中的点，每个点的坐标对应于三元组userID，serviceID，timeID的索引。

捕获三元组用户，服务，时间之间的三维交互的一种直接方法是将这些关系建模为张量。

I用户在K个时间间隔从J服务调用的Web服务的QoS值表示为张量Y∈RI×J×K，即三维张量，其中I×J×K项表示为Yijk ：（1）Yijk = Rating表示用户i在上下文类型k下调用服务j的丢失QoS值，Rating是该服务QoS值； （2）Yijk = 0表示尚未调用该服务。

即使我们的系统收集的数据集的密度仅为30％，实际的服务QoS值数据集也非常稀疏。为了获得用户服务时间张量中缺少的QoS值，可以使用其他服务用户观察到的Web服务QoS来预测当前用户的Web服务。

一旦获得了这些初始Web服务QoS值，我们的推荐系统将尝试使用QoS值函数T来估算尚未为用户，服务，时间三联获得的Web服务QoS值：用户ID×ServiceID×T imeID→ Rating，其中UserID，ServiceID和T imeID分别是用户，服务和时间段的索引，Rating是与三维索引相对应的QoS值。从该示例和其他情况可以看出，需要一种算法来估计QoS值函数T。

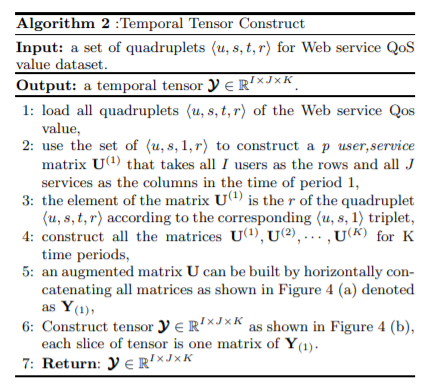
本文采用CP分解模型重建时域三维用户服务时间张量。我们的方法设计为两个阶段的过程。首先，构造由观测到的QoS值组成的时间QoS值张量。然后，我们提出了一种非负张量分解方法来预测张量中丢失的QoS值。

当服务用户调用Web服务时，我们的推荐系统将收集QoS属性的性能。运行一段时间后，推荐器会累积Web服务QoS属性数据的集合，这些数据可以由一组四组用户ID，ServicID，TimeID，Rating（或简称为u，s，t，r）表示。

使用QoS值数据，可以构建时间三维张量Y∈RI×J×K，其中I，J，K分别是用户数，服务数和时间段。

张量的每个条目表示在时间段k的u，s对的QoS值。

算法2中给出了三维时间张量构造算法：输入是一组Web服务QoS值，输出是构造的时间张量Y∈RI×J×K。张量Y中的每个额叶切片对应于每个时间间隔的u对s QoS值矩阵。



最后算法步骤如上所示。后面公式推导太复杂略。