**移动应用程序排名欺骗的发现**

**摘要**——移动应用市场中的排名欺骗指的是欺骗性或欺诈性的活动，它的目的是在流行列表中突出应用程序。事实上，应用程序开发人员使用可疑手段变得越来越频繁，比如：夸大他们的应用程序的销售或发布虚假的应用程序评级，进行排名欺骗。尽管防止等级欺骗的重要性已得到广泛认可，但是在这方面的理解和研究有限。为此，在本文中，我们对排名欺骗提供了一个整体的看法，并且提出一个用于移动应用程序的等级欺骗检测系统。具体而言，我们首先建议通过挖掘活跃期，即主要阶段，准确定位移动应用程序的排名欺骗行为。这种主要阶段可以用来检测移动应用程序排名的局部异常而不是全局异常。此外，通过统计假设测试对应用程序的排名、评级和评论行为进行建模，我们还研究了三种类型的证据，即基于排序的证据，基于等级的证据和基于评论的证据。另外，我们提出了基于聚合方法的优化来整合欺骗检测的所有证据。最后，我们对从iOS应用程序商店收集很长一段时间内的真实应用程序数据进行评估。在实验中，我们验证了该系统的有效性，并展示了检测算法的可扩展性以及排序欺骗活动的规律性。

**关键词**——移动应用程序，排名欺骗检测，证据聚合，历史排名记录，评级和评论

**1 前言**

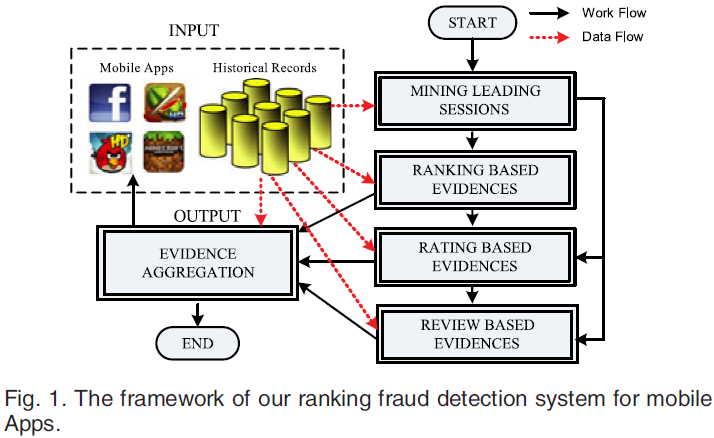
在过去的几年里，移动应用程序的数量以惊人的速度增长。例如，截至2013年4月底，在苹果应用商店和Google Play（前名为安卓市场），有超过160万个应用程序。为了刺激移动应用程序的发展，许多应用程序商店推出每日APP排行榜，其中显示最受欢迎的应用程序的图表排名。事实上，APP排行榜是推动移动应用程序发展的最重要的方式。排行榜上排名越高的通常会引起大量下载和百万美元的收益。因此，APP开发人员倾向于探索各种方式，例如广告活动来推销他们的应用程序，以便让这些应用程序在排行榜中的应用排名尽可能高。

然而，最近的一个趋势是，可疑的应用程序开发者不再依据传统的营销方案，而是采取欺骗手段，故意地提高他们应用程序的排名，最终在APP Store操纵排行榜。这通常是通过使用所谓的“机器人农场”或“人类水兵”来在短时间内对应用程序的下载、评级和评论进行膨胀来实现。

例如，来自互联网产业新闻博客的一篇报道说，当一个应用程序借助排名操纵提升了排名，它可以从苹果最高免费排行榜上的1,800位推动到前25名，在几天内可以获得超过50,000-100,000名新用户。事实上，这种排名欺骗引起了移动应用行业的极大关注。例如，苹果警告说，要严惩那些在苹果应用商店中犯排名欺骗的程序开发人员。

在文献中，虽然有一些相关的工作，如网页排名作弊检测[ 22 ]、[ 25 ]、[ 30 ]，在线评论垃圾邮件检测[ 19 ]、[ 27 ]、[ 28 ]，和移动应用程序的建议[ 24 ]、[ 29 ]、[ 31 [ 32 ]，检测移动应用程序的排名欺骗的问题仍在探索中。为了填补这一至关重要的空隙，在本文中，我们提出了开发移动应用程序排名的欺骗检测系统。沿着这条线，我们确定了几个重要的挑战。首先，排名欺骗并不总是发生在应用程序的整个生命周期中，所以我们需要检测欺骗发生的时间。这种挑战可以被视为检测本地异常，而不是移动应用程序的全局异常。第二，由于移动应用程序数量巨大，很难手动为每个应用程序标记排名欺骗，有一个不使用任何基准信息的可扩展方式，来自动检测排名欺骗是很重要的。最后，由于排行榜的动态性， 识别并确认与排名欺骗挂钩的证据是不容易的，这促使我们发现一些移动应用程序的隐含欺骗模式作为证据。

事实上，我们仔细观察发现，移动应用程序并不总是排在排行榜中，但只有在一些重大事件中，形成不同的主要阶段。请注意，我们将在以后详细介绍主要事件和主要阶段。换句话说，排名欺骗通常发生在这些主要阶段。因此，检测移动应用程序的排名欺骗实际上是在移动应用的主要阶段中检测排名欺骗。具体而言，我们首先提出一种简单而有效的算法，根据其历史排名记录来确定每个应用程序的主要阶段。然后，通过对应用程序排名的行为的分析，我们发现欺骗程序与普通的应用程序比较，在每个主要阶段的排名方式不同。因此，我们根据应用程序的历史排名记录，描述了一些欺骗证据，并且开发了三个功能提取基于欺骗证据的排名。尽管如此，排名为基础的证据可以受APP开发者声誉和合法性的营销活动所影响，如“限时折扣”。这样的结果是，仅仅使用排名为基础的证据是不充足的。因此，我们进一步提出了两种类型的欺骗的证据——基于应用程序的评级以及应用程序的评论，这就从应用程序的历史评级和评论记录方面反应了一些异常现象。此外，我们还开发了一个无监督证据聚合方法来整合这三种类型的证据去评估移动应用程序的主要阶段的信誉。图1显示了我们的移动应用程序的排名欺骗检测系统的框架。



值得注意的是，所有的证据都是通过对应用程序的排名、评级和评论行为建模，通过统计假设检验来提取的。该框架是可扩展的，并且可以延伸到其他领域的排名欺骗检测。最后，我们对从iOS应用程序商店收集长期（大于两年）的真实应用程序数据进行评估。实验的结果表明了该系统的有效性、检测算法的可扩展性以及欺骗行为的排序的规律性。

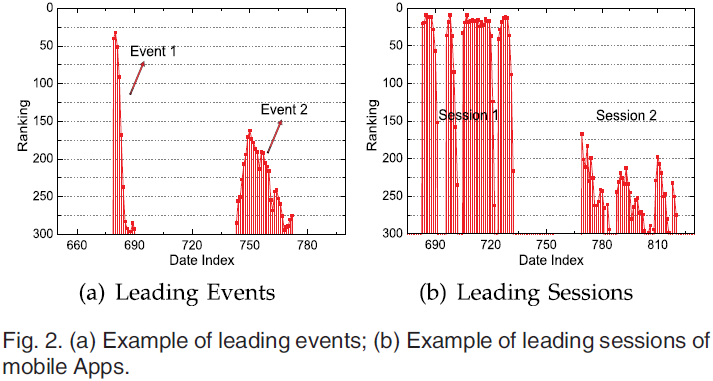
概述：本文的其余部分安排如下。在第2部分中，我们将介绍一些准备工作，以及如何为移动应用程序挖掘主要阶段。第3部分，提出如何提取基于排序、评级和评论的证据，并将它们结合起来排序欺骗检测。在第4部分，我们对提出的方法进行了进一步的讨论。在第5部分，我们在两个长期真实世界的数据集的基础上对实验结果做了报告。第6部分，提供了相关工作简要回顾。最后，在第7部分，我们总结全文并提出未来的研究方向。

**2 确定移动应用程序的主要阶段**

在这个部分，我们首先介绍一些准备工作，然后展示如何从历史排名记录中挖掘移动应用的主要阶段。

2.1 准备工作

APP排行榜表明对于不同的类别的前K个流行的应用程序，如“热门免费应用程序”和“热门付费软件”。此外，排行榜通常是定期更新（例如， 日更）。因此，每个移动应用程序a有许多历史排名记录，这些记录可以表示为时间序列，这里的是a在时间戳ti的排名。正无穷代表了a不在排名前K名的列表之中。n表示所有等级记录的数字。值得注意的是，值越小，应用程序获得的排名越高。



通过分析移动应用排名的历史记录，我们观察到应用程序并不总是在排行榜排名很高，而是仅仅在一些主要的事件中。例如，图2（a）显示了移动应用程序的主要事件的一个例子。形式上，我们定义了的主要事件如下：

定义1（主要事件）。给定一个排名阈值，APP a的主要事件e 包含一个时间范围,以及a的相应排名,满足,，另外，，我们都有。

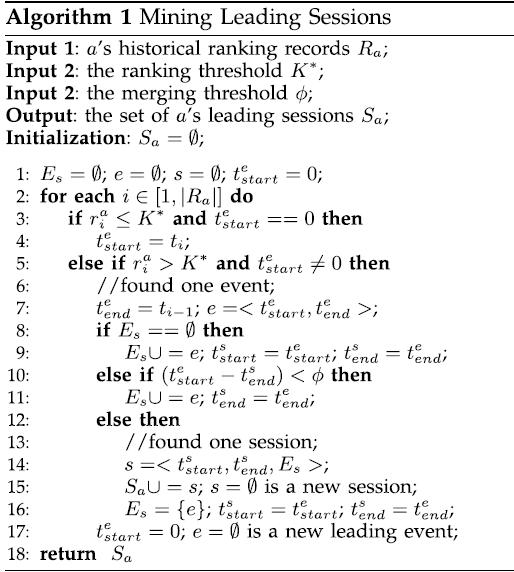
请注意，我们应用排名阈值K\*，它通常小于K，因为K可能非常大（例如，超过1000），并且，超过K\*（例如：300）对于检测排名操作不太有用。此外，我们还发现，一些应用程序有几个相邻的主要事件，彼此接近，形成一个主要阶段。例如，图2（b）显示了一个给定的手机移动应用程序的相邻主要事件的例子，形成了两个主要阶段。一个没有其他邻近主要事件的主要活动也可以被视为特别的主要阶段。正式的主要阶段的定义如下：

定义2（主要阶段）。APP a的主要阶段s包含一个时间段，，和n个邻近的主要事件{e1,e2,…en}，满足，，并且没有其他的主要阶段s\*使得，同时，，我们都有，这里∅是合并主要事件的预定义时间阈值。

直观地说，一个移动应用程序的主要阶段表示它的所有时期的普及程度，所以排名操作只会发生在这些主要阶段。因此，检测排名欺骗的问题就是去检测欺骗主要阶段。在这条线上，第一个任务是如何从历史排名记录中挖掘移动应用程序的主要阶段。

2.2 挖掘主要阶段

挖掘主要阶段有两个主要步骤。首先，我们需要从应用程序的历史记录发现主要事件。其次，我们需要整合相邻的主要事件，从而形成主要阶段。具体来说，算法1演示了给定应用程序a的挖掘主要阶段的伪代码。



在算法1中，我们表示了每一个主要事件e，主要阶段s分别作为组和中的元素，在这两个组中，Es是阶段s的主要事件集。

具体来说，我们首先从开始的时间提取给定应用程序a的单个主要事件e。对于每个提取出来的主要事件e，我们检查e与当前主要阶段s之间的时间间隔，从而去决定基于定义2，它们是否属于相同的主要阶段。具体来说，如果，那么e将会被认为是一个新的主要阶段（例如：第8到16步）。因此，该算法可以通过只扫描a的历史排名记录一次，确定主要事件和主要阶段。

**3 排名欺骗检测的证据提取**

在这个部分，我们研究了如何提取和组合欺骗证据进行等级欺骗检测。

3.1 基于排序的证据

根据第2部分中介绍的定义，一个主要阶段由几个主要事件组成。因此，为了提取欺骗证据，我们应该首先分析主要事件的基本特征。

通过分析应用程序的历史排名记录，我们发现，应用程序的排名行为在一个主要事件中，总是满足特定的排名模式，其中包括三个不同的排名阶段，即上升阶段、维持阶段和衰退阶段。具体而言，在每一个主要事件中，一个应用程序的排名首先会在排行榜中上升到一个峰值位置（即上升阶段），然后保持一段时间这样的位置（即维持阶段），最后减少直至事件的结束（即衰退阶段）。图3展示出了主要事件的不同排序阶段的一个示例。事实上，这样的排序模式显示了对于主要事件的很重要的理解。下面，我们正式定义一个主要事件的三个阶段。

定义3（一个主要事件排序阶段）。给定一个APP a的主要事件e，时间范围是，在这里，a的最高排序的位置是，∈∆R。e的上升阶段在时间范围[，这里的，∈∆R，并且，都满足。e的维持阶段在时间范围[，这里的∈∆R，并且，都满足。衰落阶段在时间范围[，这里。

![C:\Users\Administrator\AppData\Roaming\Tencent\Users\2406753528\QQ\WinTemp\RichOle\[B98](_V}%U)M[CB3(EMY)0.png](data:image/png;base64,)

注意，在定义3中，是决定维持阶段的开始时间和结束时间的排序的范围。和是APP排名在中的开始时间和结束时间。这是因为一个应用程序，即使有排名操作，也不可能总是在排行榜中保持同一个峰值（例如：第一），而是仅仅保持在一个范围。（例如：前25）。如果一个应用程序a的主要阶段s有排名欺骗，那么a在s阶段的三个排序阶段的主要事件的排序行为和一个正常的主要阶段是不同的。事实上，我们发现每个有排名操作的应用程序总是有一个预期的排名目标（例如：在排行榜中保持一周的前25名），根据这种排名的期望，聘请的营销公司通常会收取相应的费用（例如：前25名的时候，每天1000美元）。因此，对于应用开发者和营销公司来说，排名期望越早得到满足，赚的钱就越多。此外，在达到和保持期望的排名一段时间之后，操作将会停止，恶意应用程序的排名将急剧下降。结果是，可疑的主要事件可能包含非常短的上升和衰退阶段。同时，由于应用商店的排名原则不清，应用程序开发人员之间竞争激烈，所以高期望排名的操控成本非常昂贵。 因此，欺骗性应用的主要事件往往在排名很高的位置具有很短的维持阶段。

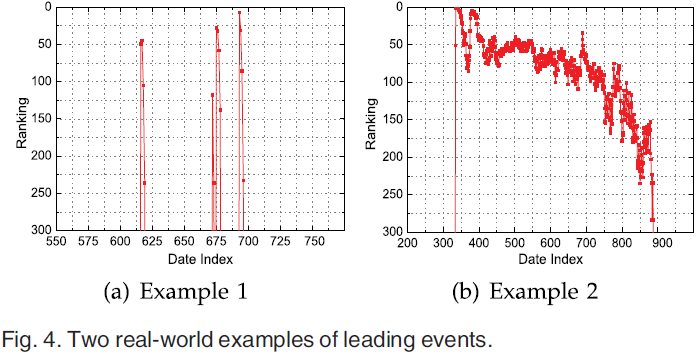
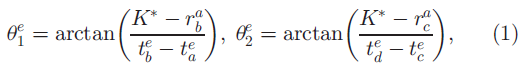
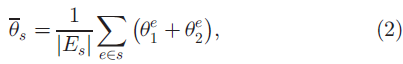


图4（a）展示了报道的欺骗应用例子之一的排名记录。我们可以看到，这个应用程序有几个排名很高的主要事件冲突。相反地，一个正常应用程序的主要事件的排名行为可能是完全不同的。例如，图4（b）展示了一个流行的应用程序“愤怒的小鸟：空间”的排名的记录，其中包含了很长一段时间的主要事件（例如：超过一年），尤其是对衰落阶段来说。事实上，一旦一个应用程序在排行榜上排名很高，它通常会有很多诚实的粉丝，而且会吸引越来越多的用户下载它。因此，该应用程序将会在排行榜上很长一段时间排名很高。基于以上的分析，我们提出基于主要事件指数的排序，去为排名欺骗检测构造证据。  
证据1。从图3可以看到，我们使用两个形状参数θ1和θ2来量化APP a的主要事件e的上升阶段和衰落阶段的排名模式，可以通过以下计算得到：



这里的K\*是定义1中的排名阈值。直观地说，一个很大的θ1可能表明了该应用程序已经在短时间内达到很高的排名，并且一个很大的θ2可能表明该应用程序在短时间内从一个很高的排名跌落到底端。因此，一个有很大θ1和θ2值的主要阶段，有很大的可能是排名欺骗。这里，我们为一个主要阶段定义一个欺骗指数，如下：

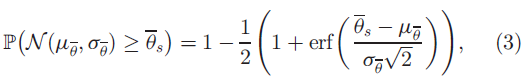


这里，是阶段s中的主要事件的个数。直观地说，如果排行榜上的应用程序的一个主要阶段s有和其他主要阶段相比明显有较高的，那么该主要阶段很大可能有排名欺骗。为了捕捉到这些，我们提出了应用统计假设检验对每个主要阶段计算其相应的指数。具体来说，我们定义了两个统计假设如下，并且计算了每个主要阶段的p值。

假设0：主要阶段s的指数对于检测排名欺骗无用。

假设1：主要阶段s的指数明显比预期的好。

这里，我们建议使用流行的高斯相似去计算在以上两个假设基础上的p值。具体来说， 我们假定遵循高斯分布,,这里的和可以在所有APP的历史主要阶段观察到，通过经典的最大似然估计方法学到。然后，我们可以使用



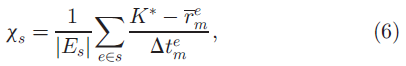
计算p值。这里erf(x)是高斯误差函数，如下：

IMG_256

直观地说，一个有较小的p值的**P**有更多的机会去拒绝假设0，接受假设1。这意味着它更有可能是排名欺骗。因此，我们定义的证据为：

IMG_256

证据2。根据上面的讨论，有排名欺骗的APP，在每个主要事件中，通常在高的排名位置都有一个很短的维持阶段。因此，如果我们表明一个主要事件e的维持阶段为=(-+1)，并且将在维持阶段的平均排名设定为，我们可以为每个主要阶段定义一个欺骗指数χs如下：



这里的K\*是定义1中的排名阈值。如果排行榜上的应用程序的一个主要阶段s有和其他主要阶段相比明显有较高的χs，那么该主要阶段很大可能有排名欺骗。为了捕捉到这些，我们定义了两个统计假设如下，用来计算每个主要阶段的χs的意义。

假设0：主要阶段s的指数χs对于检测排名欺骗无用。

假设1：主要阶段s的指数χs明显比预期的高。

这里，我们仍旧使用高斯相似去计算在以上两个假设基础上的p值。具体来说， 我们假定χs遵循高斯分布，，这里的和可以在所有APP的历史主要阶段观察到χs，通过经典的最大似然估计方法学到。然后，我们计算的证据为：

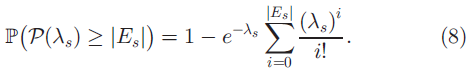
IMG_256

证据3。一个主要阶段的主要事件的个数，例如：，也是排名欺骗的一个重要指数。一个正常的APP，衰落阶段表明受欢迎程度的流失。因此，在一个主要事件之后，在很短的时间内出现另一个主要事件是不可能的，除非该APP更新了它的版本或者执行一些销售推广。因此，如果排行榜中的APP的一个主要阶段相较于其他主要阶段拥有更多的主要事件，那么该主要阶段很大可能有排名欺骗。为了捕捉到这些，我们定义了两个统计假设如下，用来计算每个主要阶段的的意义。

假设0：主要阶段s的指数对于检测排名欺骗无用。

假设1：主要阶段s的指数明显比预期的大。

由于总是离散的值，我们建议根据上面的假设，使用泊松近似计算p值。具体来说，我们假定遵循泊松分布，，这里的参数可以在所有APP的历史主要阶段观察到，通过最大似然估计方法学到。然后，我们可以计算p值如下：



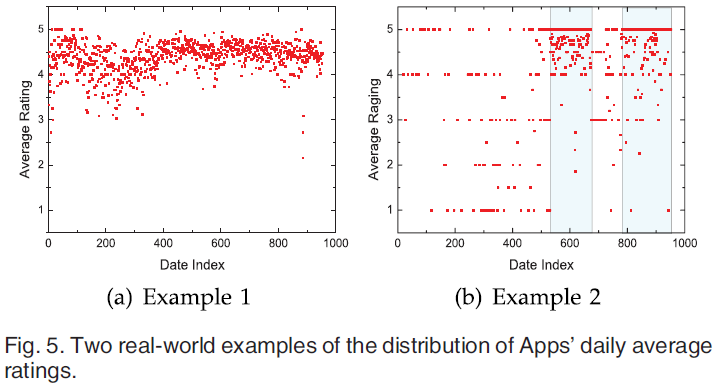
因此，我们计算的证据为：

IMG_256

直观地，以上三个证据、和的值都在[0,1]。同时，一个主要阶段的证据值越高，那么，这个主要阶段就更可能有排名欺骗的活动。

3.2 基于等级的证据

基于排序的证据对于排名欺骗检测是有用的。然而，有时，仅仅使用基于排序的证据是不充分的。例如，一些应用程序由出名的开发者创建，比如Gameloft公司，可能会因为开发者的可靠性和口碑广告效应导致一些主要事件拥有很大的θ1。然而，一些法律服务市场，比如“限时折扣”，可能也会造成重要基于排名的证据。为了解决这个问题，我们还研究了怎样从APP的历史等级记录提取欺骗的证据。



具体来说，当一个APP已经发布，它可以由下载它的任何用户评定。事实上，用户评级是APP广告最重要的特征之一。一个具有较高的评级的应用程序可能会吸引更多的用户下载，并且可以在排行榜中获得更高的排名。因此，评级操作也是排名欺骗中另一个很重要的观点。直观地说，如果一个应用程序在一个主要阶段有排名欺骗，那么s这个时间段的等级可能相较于历史的等级，会有异常模式，这些可以用来建立基于评级的证据。例如，图5（a）和5（b）分别显示了流行APP“WhatsAPP”、和通过我们的方法发现的一个可疑应用程序的每日平均评级的分布。我们可以观察到一个正常的应用程序每个获得平均等级总是相似的，然而一个欺骗性的应用在一些时间段的（例如：主要阶段）平均等级相比于其他时间段相对较高。因此，我们定义了基于用户评级行为的等级欺骗证据如下：

证据4。对于一个正常的应用程序，在一个具体的主要阶段的平均等级应该与所有历史等级的均值一致。相反地，一个有等级操作的应用程序在关于其历史等级的欺骗主要阶段可能会有惊人的高等级。这里，我们为每个主要阶段定义一个欺骗指数如下：

IMG_256

这里是主要阶段s中的平均等级，是APP a的平均历史等级。因此，如果排行榜上的应用程序的一个主要阶段s有和其他主要阶段相比明显有较高的，那么该主要阶段很大可能有排名欺骗。为了捕捉到这些，我们定义了两个统计假设如下，用来计算每个主要阶段的的意义。

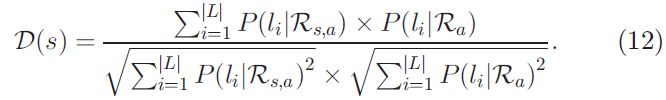
假设0：主要阶段s的指数对于检测排名欺骗无用。

假设1：主要阶段s的指数明显比预期的高。

这里，我们建议使用流行的高斯相似去计算在以上两个假设基础上的p值。具体来说，我们假定遵循高斯分布,，这里的和可以在所有APP的历史主要阶段观察到，通过最大似然估计方法学到。然后，我们计算的证据为：

IMG_256

证据5。在APP的等级记录中，每个等级可以被归类到|L|离散的评级水平，比如：1到5，这样可以表示应用程序的用户偏好。在一个正常的应用程序a的主要阶段s中，关于等级水平li的分布，，应该与a的历史等级记录的分布相一致，反之亦然。具体地，我们可以通过计算分布，这里的是s阶段的等级的数量，而且评级在水平li,是s阶段评级的总数。同时，我们可以以同样的方式计算。然后，我们用和之间的余弦相似性去估计差异如下：



因此，如果排行榜上的应用程序的一个主要阶段s有和其他主要阶段相比明显有较低的D(s)，那么该主要阶段很大可能有排名欺骗。为了捕捉到这些，我们定义了两个统计假设如下，用来计算每个主要阶段的D(s)的意义。

假设0：主要阶段s的指数D(s)对于检测排名欺骗无用。

假设1：主要阶段s的指数D(s)明显比预期的低。

这里，我们使用高斯相似去计算在以上两个假设基础上的p值。具体来说，我们假定D(s)遵循高斯分布, ，这里的和可以在所有APP的历史主要阶段观察到D(s)，通过最大似然估计方法学到。然后，我们计算的证据为：

IMG_256

两个证据的值都在[0,1]。同时，一个主要阶段的证据值越高，那么，这个主要阶段就更可能有排名欺骗的活动。

3.3 基于评论的证据

除了评级，大多数的应用程序商店还允许用户写一些文字评论作为应用程序评论。这样的评论可以反映对于特定的移动应用程序，现有用户的个人的看法和使用经验。事实上，评论操作是APP排名欺骗的最重要的视角之一。具体而言，在下载或购买一个新的移动应用程序之前，用户经常阅读它的历史评论从而缓解他们的决策，并且一个移动APP拥有更多正面的评论将吸引更多的用户下载。因此，骗子经常在主要阶段发布虚假评论，从而增加应用程序的下载，继而提高应用程序在排行榜中的位置。尽管一些之前的针对评论垃圾邮件检测的工作已经在近些年报导，但是在主要阶段检测局部异常，还有为了排名欺骗检测抓获它们的问题仍旧没有得到充分开发。为此，这里，我们为了检测排名欺骗，提出基于主要阶段中APP评论行为的两个欺骗证据。

证据6。事实上，大部分的评论操作都是由机器人农场实施的，因为人力资源很昂贵。因此，评论垃圾经常在同一个APP下发布多种重复或接近重复的评论从而刺激下载[ 19 ]， [ 21 ]。与此相反，正常的应用程序总是会有多样化的评论，因为用户有不同的个人观点和用户体验。基于上面的观察，这里我们定义了一个欺骗指标Sim（s）,它表明主要阶段s中评论的平均相互相似的程度。具体来说，这个欺骗指数可以通过以下步骤计算得到。

首先，对主要阶段s中的每个评论c,我们删除所有的停止词（例如：“of”,“the”），并且，规范动词和形容词（例如：“plays->play”,“better->good”）。

其次，我们为每一个评论c建立一个标准化的字矢量, dim[n]这里n表明在s所有评论中所有独特的标准化词的数量。具体而言，这里我们有，其中是c中第i个字的频率。

最后，我们可以通过余弦相似度，计算两个评论ci和cj之间的相似度。因此，该欺骗指数Sim（s）计算如下：

IMG_256

这里的Ns是主要阶段s时期的评论的数目。直观地，Sim（s）越高，表明s中重复或接近重复的评论越多。因此，如果排行榜上的应用程序的一个主要阶段s有和其他主要阶段相比明显有较高的Sim（s），那么该主要阶段很大可能有排名欺骗。为了捕捉到这些，我们定义了两个统计假设如下，用来计算每个主要阶段的Sim（s）的意义。

假设0：主要阶段s的指数Sim（s）对于检测排名欺骗无用。

假设1：主要阶段s的指数Sim（s）明显比预期的高。

这里，我们使用高斯相似去计算在以上两个假设基础上的p值。具体来说，我们假定Sim（s）遵循高斯分布,，这里的和可以在所有APP的历史主要阶段观察到Sim（s），通过最大似然估计方法学到。然后，我们计算的证据为：

IMG_256

证据7。从对现实世界的观察，我们发现每个评论c总是与特定的潜在的话题z相关联。例如：一些评论可能会涉及到的潜在话题“值得玩”，然而另一些可能与“很无聊”相关。同时，由于不同移动应用程序的用户有不同的个人喜好，每一个APP a 在它们的历史评论记录里可能有不同的话题分布。在一个应用程序a的一个正常的主要阶段，评论的话题分布，比如，P(z|s)，应该与a的所有历史记录的话题分布相一致，比如，P(z|a)。这是因为评论的话题是以用户个人的使用经验而不是移动应用的流行程度为基础的。相反地，如果s的评论被操作，那么这两个话题分布将会明显不同。例如，在主要阶段，可能会有许多积极的话题，比如“值得玩”和“流行”。

在本文中，我们提出利用话题模型提取评论的潜在话题。具体来说，这里，我们采用被广泛使用的隐含狄利克雷分配（LDA），从而研究话题的潜在语义。更具体地说，移动APP a的历史评论，例如，Ca，假定由以下步骤形成。首先，在Ca形成之前，给定的潜在话题的K优先分布条件是由优先Dirichletβ分布形成的。第二，对每个移动应用程序APP a ，之前的潜在话题分布是由优先Dirichletα分布形成的。然后，为了形成Ca中的第j个字表示为Wa,j，模型首先从形成一个潜在的话题z，然后从中形成Wa,j。LDA模型训练的过程是学习正确的潜在变量和，从而最大化评论观察的后验分布，例如，|。在本文中，使用一种名为吉布斯抽样的马尔可夫链蒙特卡罗方法来测试LDA模型。如果我们将a的主要阶段s的评论指定为Cs,a,我们可以使用KL散度估计在Ca和cs之间主题分布的差异。

IMG_256

这里和可以通过LDA训练过程获得。DKL(s||a)的值越大，表明Ca与Cs,a之间的话题分布差别越大。因此，如果排行榜上的应用程序的一个主要阶段s有和其他主要阶段相比明显有较高的DKL(s||a)，那么该主要阶段很大可能有排名欺骗。为了捕捉到这些，我们定义了两个统计假设如下，用来计算每个主要阶段的DKL(s||a)的意义。

假设0：主要阶段s的指数DKL(s||a)对于检测排名欺骗无用。

假设1：主要阶段s的指数DKL(s||a)明显比预期的高。

这里，我们仍旧使用高斯相似去计算在以上两个假设基础上的p值。具体来说，我们假定DKL(s||a)遵循高斯分布,，这里的和可以在所有APP的历史主要阶段观察到DKL(s||a)，通过最大似然估计方法学到。然后，我们计算的证据为：

IMG_256

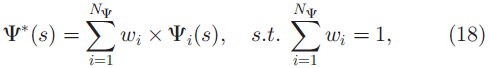
证据的值在[0,1]。同时，一个主要阶段的证据值越高，这个阶段就越有可能有排名欺骗的活动。

3.4 证据聚集

在提取三种欺骗证据之后，下一个挑战是如何整合它们从而进行排名欺骗检测。事实上，在文献里有许多排名和证据聚集整合的方法，例如，基于排列的模型[ 17 ]，[ 18 ]，评分模型[ 11 ]，[ 26 ]，Dempster-Shafer规则[ 10 ]，[ 23 ]。

然而，这些方法中的一些侧重于全局所有候选的应用程序的排名。这不是为了检测新的APP的排名欺骗的正确的方法。其他方法是基于监督学习技术，其依赖于标记的训练数据和，难以开发。我们提出了一种基于欺骗相似的无监督的方法去整合这些证据。

具体来说，我们定义最终的证据评分作为所有现有证据的方程的线性组合。注意，这里，我们建议使用线性组合，因为它已经被证明是有效的，并且在相关领域被广泛使用，比如，排名聚合[ 16 ]，[ 20 ]

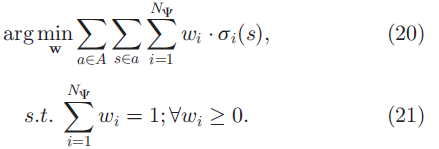


这里是证据的数目，权重是证据Ψi(s)的证据聚合参数。因此，证据的聚合问题就是怎样从测试的主导阶段学习正确的参数。

我们首先为我们的证据聚合方法提出了一个直观的假设为原则1。具体来说，我们假定每个主要阶段的有效的证据应该有相同的证据得分，而差的证据将其他的引起不同的得分。换句话说，倾向于与多个证据一致的证据将会给予较高的权重，而倾向于不和其他证据一致的证据将会给予较少的权重。为此，对于每一个证据的得分Ψi(s)，我们可以使用方差方法去测量它的一致性

IMG_256

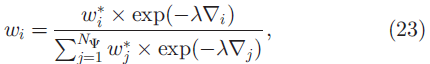
这里(s)是所有证据的主要阶段s的平均证据得分。如果(s)很小，相应的Ψi(s)应该被给予一个较大的权重，反之亦然。因此，给定带有它们的主要阶段集合{sj}的一个APP 的集合A={ai}，我们可以将证据聚集定义为一个在所有主要阶段最小化证据的加权差异的优化问题；即



在本文中，我们利用带有指数更新的基于梯度（幂梯度算法）的方法来解决这个问题。更具体地说，我们首先分配wi=作为初始值，然后对于每个s,我们计算其梯度如下,

IMG_256

然后，我们可以更新权重Wi

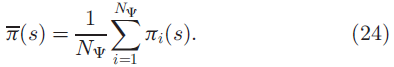


这里的是最后更新的权重Wi的值，λ是测试概率，在我们的实验中将会根据经验定为λ=10-2。

最后，我们可以利用方程（18）来估计每一个主要阶段的最终证据得分。此外，鉴于给定的s有一个预定义的阈值τ，我们可以确定如果，那么s有排名欺骗。

但是，有时仅用证据得分从而聚合证据是不适当的。这是因为不同的证据可能有不同的评分范围，然后评价主要阶段。例如，有些证据相较于平均证据得分，可能总是对主要阶段形成较高的评分，尽管它们可以检测欺骗主要阶段，并且以正确的方式对它们进行排序。

因此，这里，我们为我们的证据聚合方法提出另一种假设原则2。具体来说，我们假定有效的证据可以从一个相似的条件分布对主要阶段排序，而无效的证据将导致一个更均匀的随机排序分布。为此，对于给定的一组主要阶段，我们首先根据每个证据得分对它们进行排序，并且获得排名表。我们将(s)表示为从Ψi(s)返回的阶段s的排名，然后我们可以计算s的平均排名



然后，对于每个证据得分Ψi(s)，我们可以用方差方法去测量它的一致性。

IMG_256

如果(s)小，那么相应的Ψi(s)也应该被给予一个较大的权重，反之亦然。然后，我们可以在方程中用(s)替代(s)，并且利用基于上面介绍的方法的相似梯度去研究学习证据的权重。

**4 讨论**

这里，我们提出一些关于移动应用程序APP已提出的排名欺骗检测系统的一些讨论。

首先，下载信息对于检测排名欺骗是一个很重要的指标，因为排名操作是使用所谓的“僵尸农场”或“人水军队”去刺激应用程序在短时间内的下载和评级。但是，每个移动APP的即时下载信息通常不可用于分析。实际上，苹果和谷歌并不提供任何应用程序准确的下载信息。此外，应用程序开发人员自己由于各种原因，也不愿意公布他们的下载信息。因此，本文我们主要专注于从应用程序的历史排名、等级以及评论为排名欺骗检测提取证据。但是，我们的方法是可扩展的，用于集成其他证据（如果可用），比如，基于下载信息和APP开发者声誉的证据。

其次，已提出的方法可以检测APP的历史主要阶段的排名欺骗。但是，有时，我们需要从APP当前的排名观察中检测这样的排名欺骗。事实上，考虑到当前APP a的排名，我们可以在两个不同的情况下检测它的排名欺骗。第一，，这里的K\*是定义1中的排名阈值，我们相信a没有参与排名欺骗，因为它没有在一个主要阶段中。第二，如果，这就说明a在一个新的主要事件e中，我们把这种情况作为一个特殊的情况：和θ2=0。因此，这样的实时排名欺骗也可以被已提出的方法检测到。

最后，在为一个移动APP的每个主要阶段检测之后，剩下的问题就是怎样去估计这个应用程序的可信度。事实上，我们的方法可以发现移动应用程序的局部异常而不是全局异常。因此，在估计APP的可信度时，我们应该考虑这种局部特征。

具体来说，我们根据a的主要阶段包含的排名欺骗的个数定义一个APP的排名欺骗得分

IMG_256

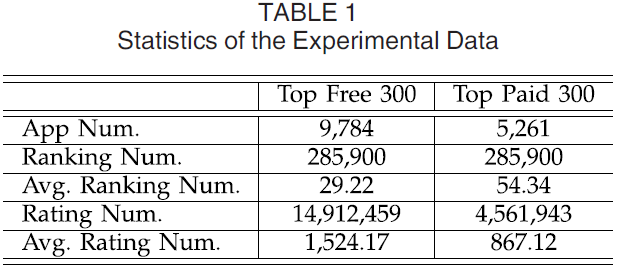
这里s∈a表明s是APP a的一个主要阶段，是可以由方程18计算得到的最终的主要阶段s的证据得分。特别是，我们定义了一个信号功能**[**x**]**，（比如，如果x是真，那么**[**x**]=1**，否则为0）和一个欺骗阈值τ去决定前K个欺骗主要阶段。此外，是s的时间范围，这表明排序欺骗的持续时间。直观地说，一个包含更多主要阶段的APP，它有着很高的欺骗证据得分和很长的持续时间，将会有更高的APP欺骗得分。

**5 实验结果**

在这个部分，我们利用真实世界的APP数据评估排序欺骗检测的性能。

5.1 实验数据

实验的数据集是在2010年2月2日到2012年9月17日在苹果应用排行榜的“前300免费的”和“前300付费的”中收集来的。数据集分别包含排名前300名的免费应用程序和前300名付费应用的每日排行榜排名。此外，每个数据集还包含用户的等级以及评论的信息。表1展示了我们的数据集的详细的数据信息。



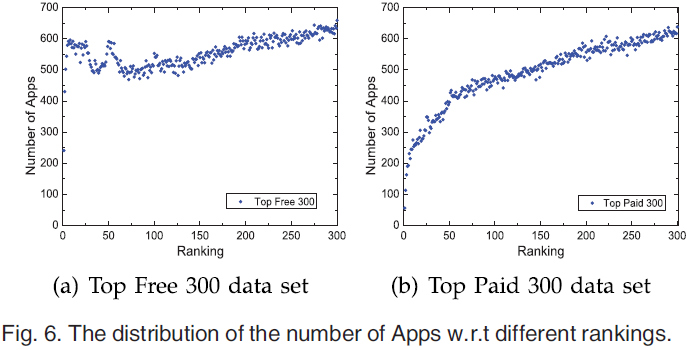
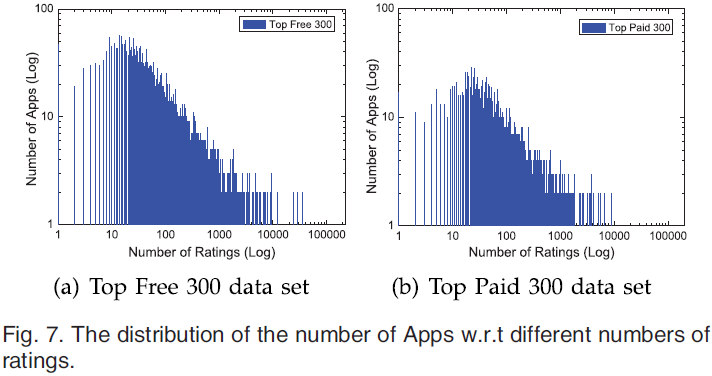
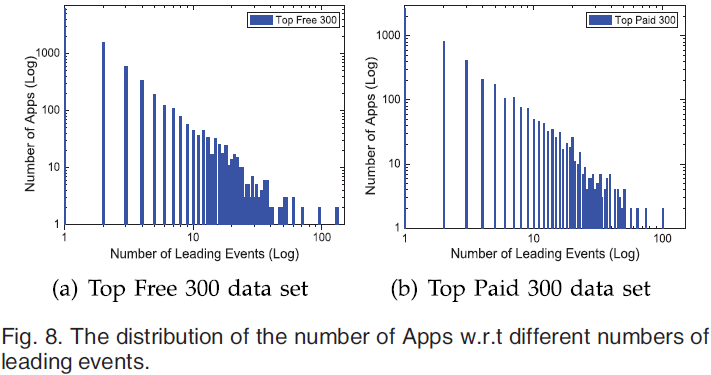


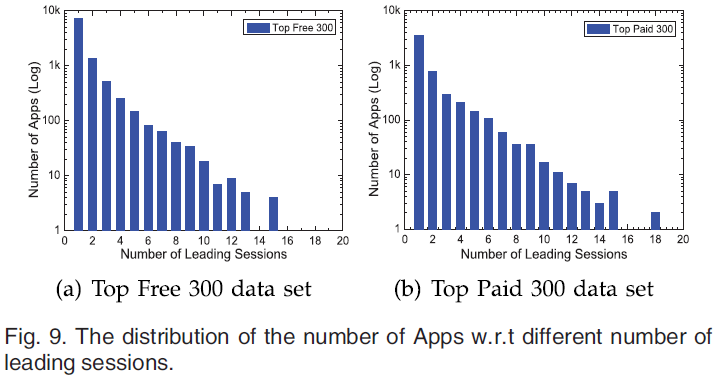
图6（a）和6（b）显示了这些数据集中不同排名的应用数量的分布。在这些数据中，我们可以看到排名较低的应用程序数量高于排名较高的应用程序数量。此外，免费的应用程序之间的竞争高于付费程序之间的竞争，特别是在高排名阶段（例如，前25）。图7a和7b展示了这些数据集中不同等级的应用程序的分布。在这些数据中，我们可以看到应用程序的等级分布不均匀，这就说明了只有一小部分的应用程序非常受欢迎。

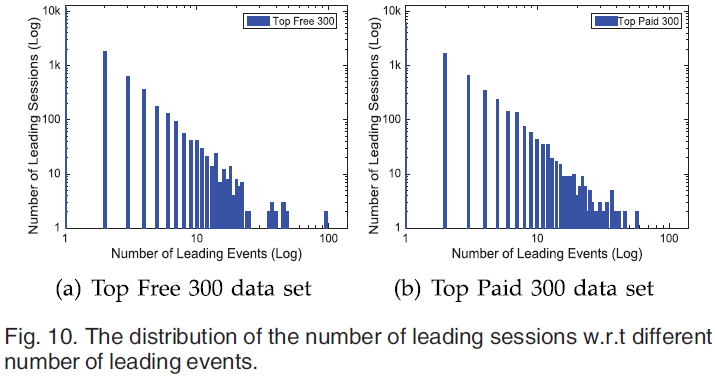


5.2 挖掘主要阶段

这里，我们展示了在两个数据集中的挖掘主要阶段的结果。具体来说，在算法1中，我们设定排名阈值K\*=300和阈值∅=7。这表明如果两个相邻的主要事件彼此都发生在一周之内，那么他们可以分段成同一个主要阶段。图8和图9展示了两个数据集中关于APP所包含的主要事件和主要阶段的不同的数量的APP数量的分布。从这些数据中，我们可以看到只有很少的APP有许多的主要事件和主要阶段。对于免费的APP软件，主要事件和主要阶段的平均数量为2.69和1.57，对于付费的，平均数量为4.20和1.86。此外，图10a和10b展示了在两个数据集中，关于主要阶段所包含的不同得主要事件的数量的主要阶段的个数的分布。在这些数据中，我们可以看到只有很少的主要阶段包含很多的主要事件。这也证实了证据。实际上，在每个主要阶段中，对于免费的APP来说，包含的平均主要事件的数目是1.7，对于付费的APP来说，是2.26。







5.3 基于人类判断的评估

据我们所知，没有现成的基准去决定哪一个主要阶段或者APP包含排名欺骗。因此，我们开发了四个基线，并且邀请了五个评价的人来证实我们的基于证据聚合的排名欺骗检测的方法的有效性。特别地，我们分别将拥有基于证据分数的集合的我们的方法（即原则1）作为EA-RFD-1，拥有基于排序的集合（即原则2）作为EA-RFD-2。

5.3.1 基线

第一个基线Ranking-RFD代表了基于排名欺骗检测的排序证据，它只通过使用基于排序的证据来评估每个主要阶段的排名欺骗（例如，到）。这三个证据是通过我们的聚合方法来整合的。

第二个基线Rating-RFD代表了基于排名欺骗检测的等级证据，它只通过使用基于等级的证据来评估每个主要阶段的排名欺骗（例如，和）。这两个证据是通过我们的聚合方法来整合的。

第三个基线Review-RFD代表了基于排名欺骗检测的评论证据，它只通过使用基于评论的证据来评估每个主要阶段的排名欺骗（例如，和）。这两个证据是通过我们的聚合方法来整合的。

特别地，在上述基线中，这里我们只使用基于排序的聚合方法（即原则2）,来整合证据。这是因为这些基线主要是用来评价不同种类的基线的有效性，我们的实验初步证实了有原则2的基线总是胜过有原则1的基线。

最后一个基线E-RFD基于排名欺骗检测的证据，它通过基于排名，评级和审查而没有证据聚合的证据证据来估计每个主要阶段的排名欺骗。具体地说，它按方程（18）——每个Wi都被设置成一样的1/7来排列主要阶段。这个基线被用来估计我们的排名聚合方法的有效性。

注意，根据定义3，我们需要在为了EA-RFD-1，EA-RFD-2，Rank-RFD和E-RFD提取基于排名的证据之前定义一些排名范围。在我们的实验中，我们将排名分为五个不同的范围，例如[1,10],[11,25], [26,50],

[51,100],[101,300],这些通常用于APP排行榜。此外，我们使用在3.3节介绍的LDA模型来提取评论主题。特别地，我们首先通过停止词删除[6]和Porter Stemmer[7]来标准化每个评论。然后，根据基于评估方法[8],[31]的困惑度设置潜在话题Kz为20。根据[13]，为了测试LDA，两个参数α和β被设置为50/K和0.1。

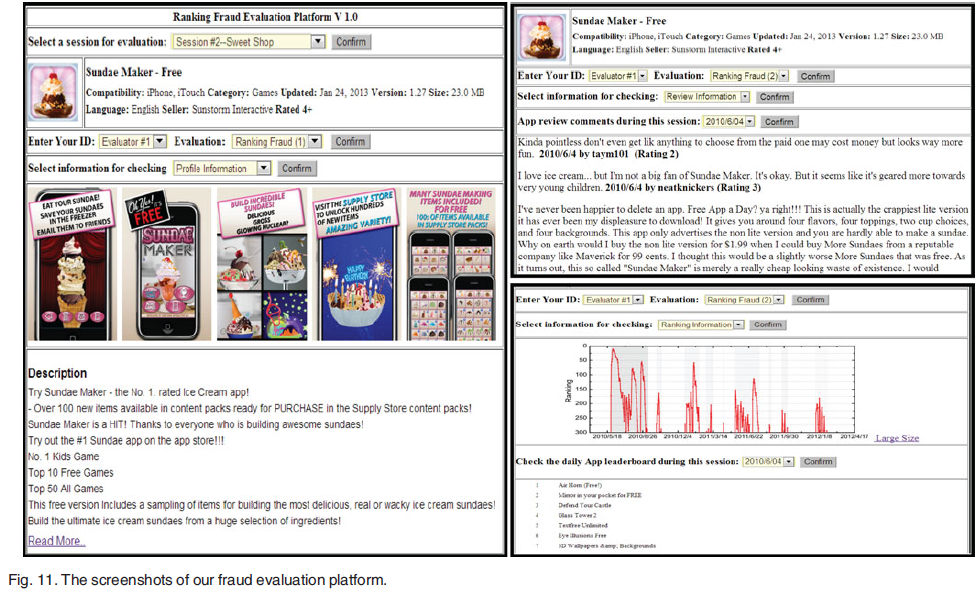
5.3.2 实验设置

为了通过每个方法研究排名欺骗检测，我们设定了如下的评估。

首先，对于每个方法，我们从每个数据集中选择了排名前50的主要阶段（例如，大部分最可疑的阶段），排名中间50的主要阶段（例如，大部分不确定的阶段）还有排名后50的主要阶段（例如，大部分正常的阶段）。然后，我们将所有选定的阶段合并到一个池中，其中包含来自“顶级免费300”数据集中的281个独特应用程序的587个独特阶段，以及“Top Paid 300”数据集中的213个独特应用程序的541个独特阶段。

其次，我们邀请了五位熟悉苹果APP Store和移动应用程序的人力资源评估人员，以2分（即欺骗），1（即不确定）和0（即非欺骗）手动标记所选的主要阶段。

具体来说，对于每个选定的主要阶段，通过全面考虑APP的简档信息（例如描述，截图），本次主要阶段的排名趋势，本次主要阶段期间的APP排行榜信息，本次主要阶段期间的评分趋势以及此次主要阶段期间的评论，每个评估者给出了适当的分数。此外，他们还可以下载并尝试相应的应用程序来获取用户体验。特别地，为了便于评估，我们开发了一个排名欺骗评估平台，确保评估者能够轻松浏览所有信息。此外，该平台还以随机次序的方式展示了主要阶段，这确保了主要阶段的排序与欺骗分数之间没有关系。图11展示了平台的截图。左侧面板显示主菜单，右上图显示给定阶段的评论，右下图显示给定阶段的排名相关信息。经过人力资源评估，每一个主要阶段s被分配一个欺骗分数f(s)∈[0,10]。因此，所有五位评估人员同意了在前300免费的数据集中，有86个欺骗阶段和113个非欺骗阶段。请注意，其中11个标签的欺骗阶段来自外部报告的可疑应用程序[4]，[5]，其验证了我们的人为判断的有效性。

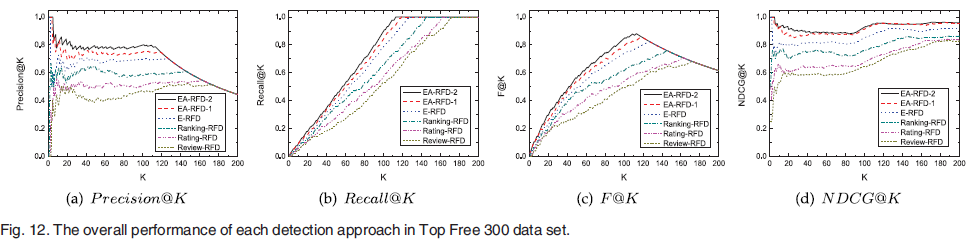


同样，所有五名评估人员同意在前300的付费的数据集中，有94个欺骗阶段和119个非欺骗阶段。此外，我们计算了每对评估者之间的Cohen’s KAPPa系数[1]，以估计评估者之间的一致性。Cohen的KAPPa系数的值在用户评估中在0:66到0:72之间。这表明了很一致[19]。

最后，我们通过每种方法关于他们的欺骗分数，进一步对主要阶段排序，并获得了六个排名的主要阶段名单。特别地，如果我们对待普遍被认同是欺骗的阶段（例如，前300免费的数据集中的89个阶段，前300付费的数据集中的94个阶段）作为实例，我们可以用三种广泛使用的度量来评估每种方法，即Precision @ K，Recall @ K，F @ K [2]。此外，我们可以利用度量标准化贴现累积增益（NDCG）来确定每种方法的排名表现。具体来说，给予一个截止等级的贴现累积增益K，可以由计算得到，这里的是人类标记的欺骗评分。NDCG @ K是由IDCG @ K标准化的DCG @ K，这是返回结果的理想排名列表的DCG @ K值，例如，我们有NDCG@K=。NDCG @ K表明通过截止等级K的方法返回的给定阶段的排序顺序。NDCG @ K值越大，排名欺骗检测的性能越好。

5.3.3 总体表现

在本节中，我们介绍了不同评估指标，即Precision @ K，Recall @ K，F @ K和NDCG @ K的每个排名欺骗检测方法的整体表现。特别地，在这里，我们将最大K设置为200，所有的实验都是在2.8 GHZ2四核CPU，4G主内存PC上进行的。



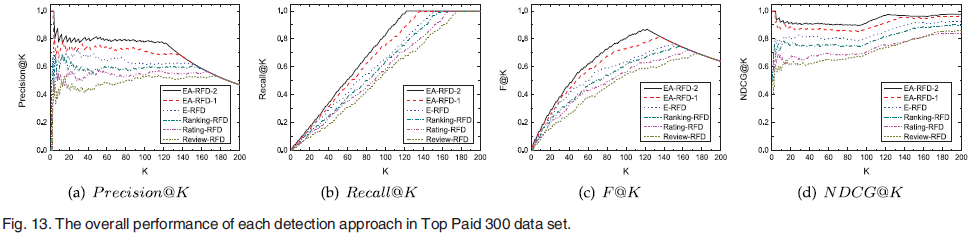
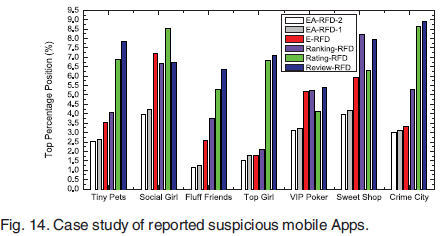


图12和13显示了两种数据集中每种检测方法的评估性能。从这些数据可以看出，两个数据集的评估结果是一致的。事实上，通过分析评估结果，我们可以得到一些有见地的观察。具体来说，首先，我们发现我们的方法，即EA-RFD-2 / EA-RFD-1，一贯优于其他基线，并且对于较小的K（例如，K <100），改进更显著。这个结果清楚地证明了我们基于证据聚合的框架检测排名欺骗的有效性。其次，EA-RFD-2在所有评估指标方面都远胜EA-RFD-1，这表明基于排序的聚合（即原则2）比基于分数的聚合（即原则1）用于整合欺诈证据更有效。第三，我们的方法始终优于E-RFD，这证明了证据分级检测排名欺骗的有效性。第四，E-RFD具有比Ranking-RFD，Rating-RFD和Review-RFD更好的检测性能。这表明即使没有证据聚合，利用三种证据比仅使用一种证据更有效。最后，通过比较Ranking-RFD，Rating-RFD和Review-RFD，我们可以观察到基于排名的证据比基于评级和评论更有效。这是因为评级和评论操纵只是排名操纵的补充。特别地，我们观察到，Review-RFD可能无法在两个数据集上的所有评估指标方面导致良好的表现。这种现象背后的一个可能原因是评论操纵（即，假正面评论）不会直接影响应用程序的排行榜排名，但可能会增加应用程序下载量和评级的可能性。因此，由于APP Store中未知的排名原则，评论操作不一定会导致排名欺骗。然而，所提出的基于评论的证据可以有助于对欺骗检测进行补充。实际上，在我们的初步实验中，我们发现基于评论的证据总是随着与其他证据一起使用而提高检测性能。这显然证实了基于评论的证据的有效性。

为了进一步验证实验结果，我们还进行了一系列0.95置信水平的配对T检验，显示了我们的方法的改进，即EA-RFD-2 / EA-RFD-1，与其他基线相比，所有评估指标的不同 K具有统计学意义。

5.4 案例研究：评估应用程序可信度

如第4节所述，我们的方法可用于通过方程（26）评估应用程序的可信度。在这里，我们研究基于现有报告的现有知识来评估应用程序可信度的性能。具体来说，如IBTimes [5]所报道，有8种免费的应用可能涉及排名欺骗。在本文中，我们使用其中的七个在我们的数据集（小宠物，社会女孩，绒毛朋友，犯罪之城，VIP扑克，Sweet Shop，Top Girl）进行评估。事实上，我们尝试研究每种方法是否可以找到具有较高排名的这些可疑应用程序，因为良好的排名欺骗检测系统应该具有捕获这些可疑应用程序的能力。特别地，除了设定等式（26）中的固定欺骗阈值τ之外，我们将排名前10的顶级会话视为可疑会话，以计算每个应用的可信度。



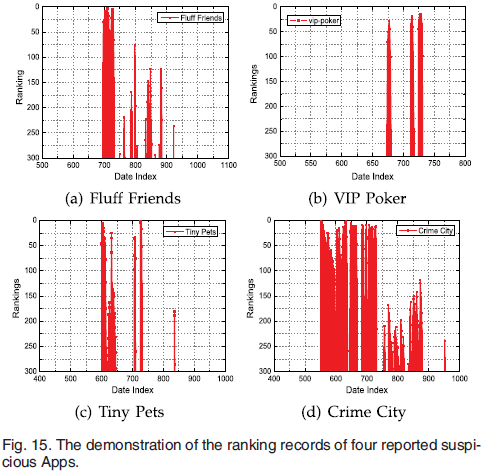
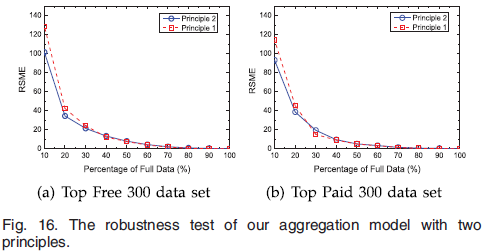


图14显示了每种方法返回的排名列表中每个APP的最高百分比位置。我们可以看到，我们的方法，即EA-RFD-2和EA-RFD-1，可以将这些可疑的应用程序排在比其他基线方法更高的位置。与第5.3.3节中的结果类似，只有利用单一类型的欺骗检测证据才能获得良好的表现，即在高位找到这样的可疑应用程序。

图15显示了上述应用程序的排名记录（受空间限制，我们只显示其中的四个）。在这个图中，我们发现所有这些应用都具有明确的基于排名的欺骗证据。例如，一些应用程序具有很高的排名（即证据1和2）的主要阶段，并且一些应用程序具有许多主要事件（即证据3）的主要阶段。这些观察结果清楚地证实了我们的方法的有效性。

5.5 我们的方法的效率和鲁棒性



我们的方法的计算成本主要来自于给定主要阶段提取三种欺骗证据的任务。事实上，这个任务的主要过程可以提前离线计算。例如，LDA模型可以离线测试，并且可以提前挖掘现有主要阶段的欺骗指数并将其存储在服务器中。在这种情况下，为每个主要阶段提取证据的过程将非常快（在我们的实验中平均小于100毫秒）。同时，需要一个学习过程来进行证据聚合。 在历史数据集上学习聚合模型后，每个新的测试应用程序可以重用该模型来检测排名欺骗。然而，仍然不清楚需要多少学习数据。 为了研究这个问题并验证我们的方法的鲁棒性，我们首先通过从整个数据集中得到的权重参数进行建模来对所有主要阶段进行排名。然后，我们还通过对从整个数据集的不同分段得到权重参数（即，10％，100％）进行建模来对所有主要阶段进行排名。最后，我们测试不同结果之间的主要阶段排名的均方根误差（RMSE）。图16显示了两个数据集的鲁棒测试结果。 我们可以发现聚合模型不需要大量的研究数据，因此我们的方法的鲁棒性是合理的。

**6 相关工作**

一般来说，本研究的相关工作可分为三类。

第一类是关于网站排名的垃圾邮件检测。 具体来说，垃圾邮件排名是指对选定网页带来无理有利的相关性或重要性的任何故意的行为[30]。例如，Ntoulas 等人[22]研究了网络上基于内容的垃圾邮件的各个方面，并提出了一些用于检测基于内容的垃圾邮件的启发式方法。Zhou等人 [30]研究了无监督的网页排名垃圾邮件检测的问题。具体来说，他们提出了一个有效的在线链接垃圾邮件和使用垃圾邮件的垃圾邮件检测方法。最近，Spirin和Han [25]报道了一个关于网络垃圾邮件检测的调查，全面介绍了文献中的原理和算法。实际上，网页排名垃圾邮件检测的工作主要是基于搜索引擎排名原则的分析，如PageRanK和查询词频。这与移动应用的欺骗检测排名不同。

第二类专注于检测在线评论垃圾邮件。例如，Lim 等人[19]已经确定了垃圾邮件发送者评论的几个代表性行为，并模拟这些行为来检测垃圾邮件发送者。吴等人[27]研究了对等级数据进行混合先令攻击的问题。所提出的方法是基于半监督学习，可用于可靠的产品推荐。谢等人[28]研究了单独评论垃圾邮件检测的问题。具体来说，他们通过检测基于多个评论的时间序列中的共异常模式来解决这个问题。虽然上述一些方法可以用于历史评级和评论记录中的异常检测，但是它们不能在给定时间段（即，主要阶段）中提取欺骗证据。

最后，第三类包括移动应用推荐的研究。例如，Yan和Chen [29]开发了一个名为APPjoy的移动应用推荐系统，该系统基于用户的应用程序使用记录来构建偏好矩阵，而不是使用明确的用户评级。另外，为了解决应用程序使用记录的稀疏性问题，Shi和Ali [24]研究了几个推荐模型，并提出了一个基于内容的协同过滤模型，名为EigenAPP，用于在其网站Getjar中推荐应用程序。此外，一些研究人员研究了利用移动应用推荐的丰富语境信息的问题。例如，Zhu等[32]提出了个性化上下文感知建议的统一框架，可以整合上下文独立性和依赖性假设。然而，据我们所知，以前的作品都没有研究过针对移动应用程序的欺诈检测排名的问题。

**7 结束语**

在本文中，我们开发了移动应用程序的排名欺骗检测系统。 具体来说，我们首先表明，排名欺骗发生在主要阶段上，并提供了从其历史排名记录中挖掘每个应用程序的主要阶段的方法。然后，我们确定了基于排名的证据，基于评级的证据和基于评论的证据来检测排名欺骗。 此外，我们提出了一种基于优化的聚合方法来整合用于评估移动应用程序的主要阶段可信度的所有证据。这种方法的独特之处在于，所有的证据都可以通过统计假设检验进行建模，因此很容易将其他领域知识证据扩展到检测排名欺骗中。最后，我们通过对从APPle APP Store收集的现实应用数据进行广泛的实验来验证所提出的系统。 实验结果表明该已提出方法的有效性。

未来，我们计划研究更有效的欺骗证据，分析评级、评论和排名之间的潜在关系。 此外，我们将扩展我们的排名欺骗检测方法到其他移动应用程序相关服务，如移动应用程序推荐，以增强用户体验。

致谢

这项工作部分得到了中国国家杰出青年科学基金（拨款号61325010），中国自然科学基金（NSFC，授权号：71329201），安徽省科技进步奖（赠予号 1301022064）的支持。这项工作也得到了美国国家科学基金会（NSF，授权号CCF-1018151和IIS-1256016）以及2014-2015年UNC夏洛特教授研究资助的部分支持。这是一个大幅度扩展和修订的[33]版本，出现在第22届信息与知识管理会（CIKM2013）的会议记录中。