哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

服务计算-论文鉴赏

学号：1140310606

姓名：张茗帅

哈尔滨工业大学计算机科学与技术学院

2017 年 5 月 3 日星期三

目录

论文 1

1.1 Title

Freelancer influence evaluation and Gig service quality prediction in Fiverr

1.2 题目

在 Fiverr 网站中用户影响评估以及 gig 服务质量的预测（注：Fiverr 网站提供的服务叫做 gig 服务）

1.3 文章摘要

服务技术和众包运动已经做出了一系列成功的推动人类服务系统快速发展的努力，在这个系统中，大量全球分布的自由职业者在 web 网络上挑战和发布一系列问题。这些众包服务通过较低的价格和短暂的反应时间来给客户提供便利，然而，这种便利并不能够掩盖由人类相关因素所造成的不稳定性，例如不可定义的某一个特定账户声誉，不确定的产品质量等问题。本文则通过全面的数据驱动的调查来分析指定用户服务的质量以及其他相关信息。调查内容包括能够评估用户影响的关键的因素，同时本文提出了 GSRC 模型来预测服务质量。GSPC 模型包括（Gig 服务属性+卖方影响属性+客户评价+语义内容），这是第一次将服务语义信息作为预测的依据，同时也是第一次整合了这四个因素至于同一预测模型中。

1.4 文章目的

在寻找影响 gig 服务流行度和 seller 的声誉度的因子时，除了要考虑传统的客观因素如：用户的等级，排名，服务的评论数目，服务提供者对于评论的平均回复时间，gig 服务的创建时间等等。（Sell level 是官方的评估标准，但文章后来证明了它存在这很多缺陷，于是作者又提出了一种新的评估标准（Sell score）），同时也要考虑主观因素如用户的市场行为和 gig 服务的相关描述信息等。于是，本文综合了主观因素和客观因素提出了以下三点从而达到对卖家信誉和 gig 服务质量进行更好的分析和预测。

1 重新定义卖家评分(Sell level -> Sell score)和 gig 服务质量两个指标，以评估卖家的影响力（这里面所说的影响可以类比于卖家的信誉评分，下同）及其卖家提供的 gig 服务的市场表现。

2 通过统计方法和相关分析，确定卖家的影响力和 gig 服务质量有很大影响的主要特征。这些特征分析的结果可以帮助卖家（特别是新秀卖家 2）进行自我推广和服务质量管理。

3 基于关键特征和机器学习算法构建新的测量模型，以预测卖家评分和服务质量。 特别是，本文开发了一个名为 GSRC（Gig Property 即 gig 服务相关属性 + Seller Impact 卖家影响+ Customer Review 客户评论反馈+ Semantic Content 相关语义信息）的新模型来预测服务质量。

1.5 相关名词的定义

Seller Score 代替传统的 Seller Level 用来衡量卖家的信誉。

曾经的 Seller Level 方式对卖家的区分度太低，极少数为顶级信誉卖家，其余的大多数均为 Level 2 的卖家。而新的权量值 Seller Score 则考虑的更多方面的元素，如卖家信誉排名的占比，以及卖家对用户相关的评论而进行回复数量。其计算表达式如下：

Ur/Umr 代表卖家信誉评分占最高评分比例

Urc 代表卖家回复评论的数量

$$u_s = \log_{10}\left(u_{rc} * \frac{u_r}{U_{MR}}\right)$$

Gig 服务质量，

曾经的 Fiverr 网站的排名系统存在很多缺点，如下：

1 Fiverr 对产品质量评分是不连续的，严重不平衡。 超过 90%的产品具有较高的评级（> 4.5）。

2 Fiverr 不会将销售信息公开。 我们只能通过用户回复进行粗略的估计。

3 Fiverr 只给出某一特定产品的总销售额，但是不给出每月的销售额或近期销售。

本文综合了产品的用户评论数、gig 服务的寿命、以及价格来衡量 gig 服务的质量，下面的式子就可以计算 gig 服务的总和市场表现：

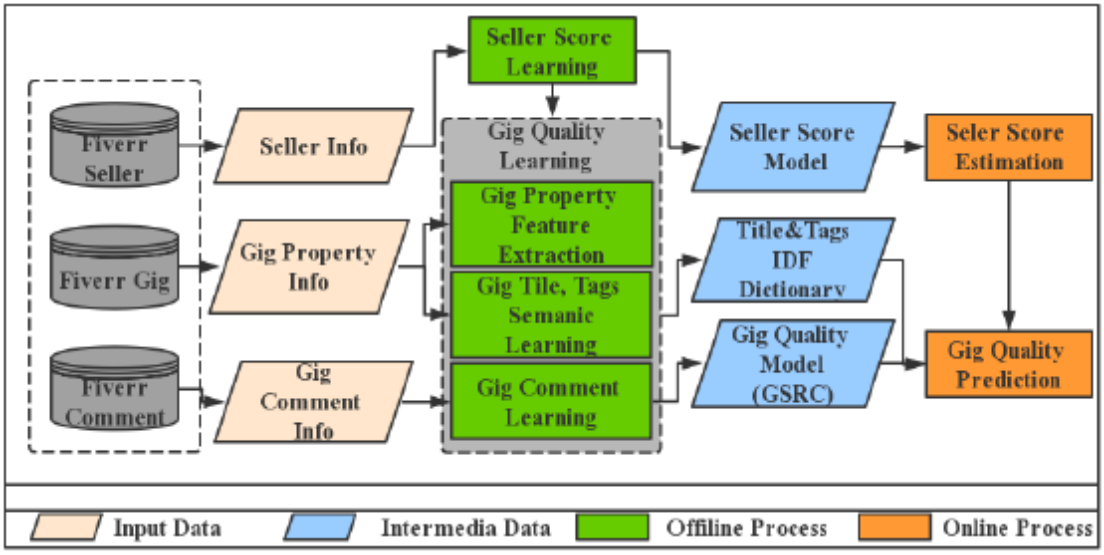
$$g_q = \log_{10}\left(\left(g_{rc} * \frac{g_r}{G_{MR}} * g_p\right) \Big/ g_m + 1\right)$$

Gr/Gmr 代表 gig 服务的排名占最高排名的比例（从 0-5，以 0.5 数值进行分割）；

Grc 代表用用户评论数；Gm 代表 gig 服务的寿命；Gp 代表 gig 服务的价格

1.6 整体框架

下图的是用于进行用户影响分析以及服务质量预测的整体框架图：



主要包括 4 个部分：输入数据、离线的相关步骤、中间数据、在线的相关步骤。

输入数据：

1 卖家信息：包括卖家的描述，语言，国家，技能，认证，提供的服务（产品），等等。

2 Gig 服务信息：包括服务的标题，描述，价格，标签，评级，评论数，收藏计数等。

3 客户评论：包括客户评论和购买后的评级，卖家对客户评论的回复。

离线步骤：

该过程主要有两个任务**卖家评分学习**和 **gig 服务质量学习**。

Seller Score Learning：从输入数据中找到（学习到）和 Seller Score 关系最大的一些因素（在文章中称为 key features），从而为接下来的验证和预测分析做准备。

Gig Quality Learning：通过 gig 的相关输入数据来训练 GSRC 模型，通过这个模型我们才可以获知哪些因素与 gig 服务质量的关系最大，并且我们可以通过他们对未来 gig 的服务质量进行预测。

中间数据：

包括“离线步骤”中生成的 seller score 模型和 gig 服务质量模型，已经包含了 gig 服务标题、标签的 IDF 字典，这个字典通过用 TF-IDF 进行计算。

在线步骤：

主要包含两步：

1 卖家评分估测：通过“离线步骤”学习到的模型来进行相关估计。

2 gig 服务质量估测：通过 gig 服务模型、卖家评分估测结果、IDF 字典进行 gig 服务质量的预测。

该过程可以可用于评估新卖家的市场影响力和他 gig 服务产品的市场潜力，此外，它可以用于推荐高质量的 gig 服务给客户。

1.7 卖家影响的估测方法

影响卖家评分 Seller Score 的主要因素包含：

卖家的经营时间（正比关系）；

卖家的自我描述长度（包括国家、语言、技能等等）（正比关系）；

卖家收到公众表扬的程度（通常通过卖家收到的用户评论长度来衡量、研究发现，用户评论长度越短、代表用户对该产品越满意，从而 Seller Score 分数越高）

服务的描述长度（正比关系）

利用数学中的线性回归和树模型对这些因素进行分析，得到各因素的权重关系如下（F 值越大 P 值越小代表该因素越重要）：

TABLE I. PEARSON CORRELATION COEFFICIENT OF KEY FEATURES

Feature	F value	p-value
online_time	494.6955	8.55E-95
gig_num	56.2127	1.11E-13
reply_ratio	49.1405	3.61E-12
avg_desp	42.4339	9.99E-11
comment_length	36.7661	1.69E-09
area	27.1281	2.17E-07
language	21.8358	3.24E-06
sub_area	18.6486	1.68E-05
avg_title	8.2355	0.004166217
education	4.6544	0.031134694
desp_length	3.8495	0.049947072
reply_length	3.1253	0.07729197

通过这个表我们可以看到卖家经营时间、卖家售卖的 gig 服务产品个数、回复比例、以及 gig 服务描述的完整性对 Seller Score 影响最大。

1.8 Gig 服务质量预测-GSRC 模型

GSRC 模型：

G：gig 服务属性。包括 gig 服务名、描述、标签、和其他语义描述、寿命、完成时间、指导性的视频、用户收藏数、市场表现等等。通过统计数据发现，某一特定 gig 服务的用户收藏数、描述文字的长度、包装的完备性（基本包装 or 包含专业包装等多包装系统）与该 gig 服务的质量成正比关系。

S：卖家影响。就是刚刚预测得到的卖家评分及 Seller Score 运用于此，它与 gig 服务质量成正比关系。

R：用户的评论。主要考虑三个特性：用户评论的平均长度、卖家回复的平均长度、卖家对用户评论的回复比例，这些因素都和 gig 服务质量呈现弱的正比关系。

C：语义信息：计算 gig 服务名称和标签的 TF-IDF 值（TF-IDF：评估一字词对于一个文件集或一个语料库中的其中一份文件的重要程度），这个值可以反应这个名称的标签对于整个语料库的记录中的重要性，从而可以通过它加强对 gig 服务质量的预测。

类似的，利用线性回归和树模型进行分析得到下表：

TABLE II. PEARSON CORRELATION COEFFICIENT OF TOP 15 FEATURES

Feature	F value	p-value
favorite	599.7965	1.54E-119
packages	275.1786	7.25E-59
seller score	262.1601	2.73E-56
featured	113.477	5.67E-26
reply ratio	91.0194	3.11E-21
max quantity	71.8012	3.90E-17
title TF-IDF	70.8413	6.26E-17
online time	68.0049	1.26E-17
description length	67.1346	3.91E-16
seller level	55.8384	1.06E-13
comment length	51.5788	8.90E-13
duration	29.9076	4.96E-08
category=graphics-design	24.8507	6.60E-07
category=lifestyle	22.6999	2.00E-06
tags TF-IDF	19.6254	9.80E-06

通过这个表可以看出，用户收藏、（G 中）Gig 服务的包丰富性、（S 中）卖家评分、（R）中的回复比例、和（C 中）TF-IDF 值与 gig 服务质量有着很大的关联。它证明了 GSRC 这个模型的正确性。

1.9 实验测试

利用爬虫获取半年内的 Fiverr 网站上的相关数据。用 RMSE 作为 Seller Score 以及 gig 服务质量的的评估指标，用 Accuracy 作为 gig 服务预测的评估指标（这两个值都是越小证明预测的越好）

这两个公式如下：

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(g_{q,i} - \hat{g}_{q,i}\right)^2}$$

$$Accuracy = \frac{1}{N}\sum_{i=1}^{N} I\left(\left|g_{q,i} - \hat{g}_{q,i}\right| < 0.5\right)$$

1.10 预测算法

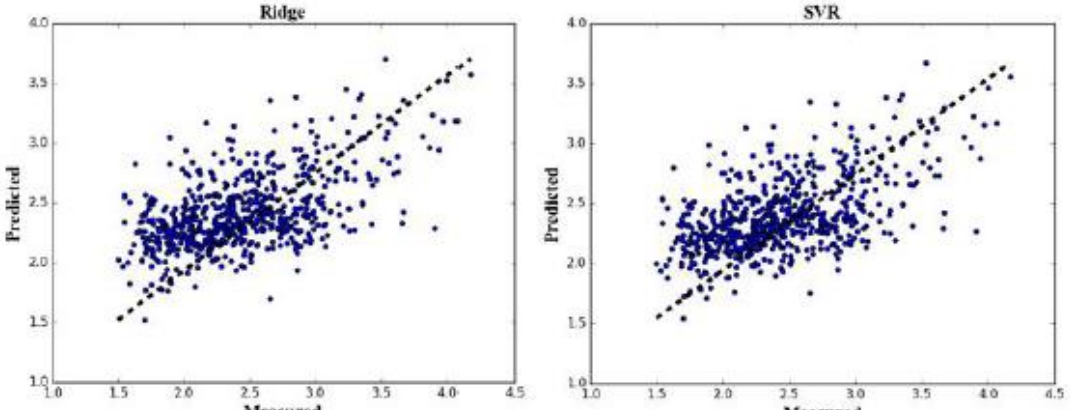
主要采用 4 个特殊的回归算法 ridge regression、lasso regression、GBRT (Gradient Boost Regression Tree)、Random Forest、and SVR。通常数据被分为训练集、验证集、测试集。

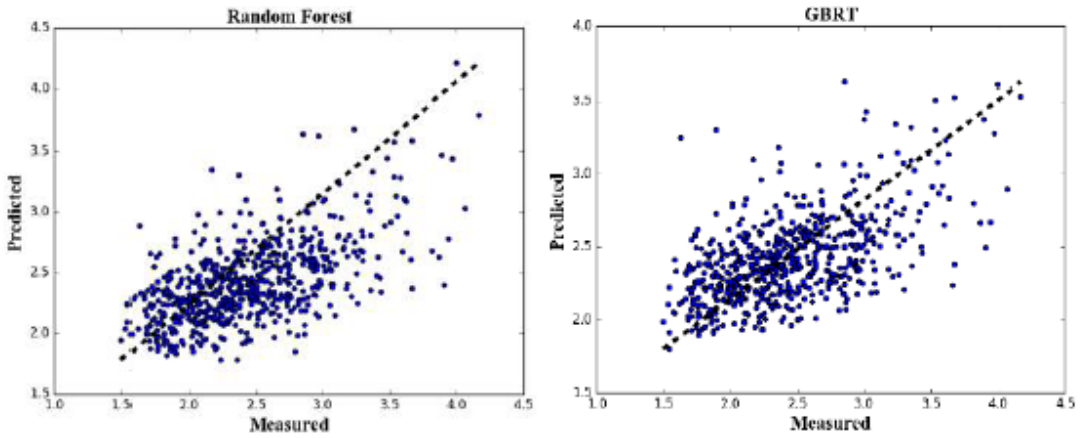
1.11 Seller Score 预测结果

用上述 4 中方法进行测试，得到 GBRT 的分析性能最优，不过大体上区别不大，其中 GBRT 预测分析下的关键因素如下：

Feature	Feature Importance	Comments
avg_desp	0.1286	Average length of description
online_time	0.1203	Seller longevity
reply_ratio	0.1171	Reply ratio
avg_title	0.0877	Average length of title
desp_length	0.0873	Self-description length
comment_length	0.0865	Comment length
reply_length	0.069	Reply length
avg_resp_time	0.0659	Average response time
skills	0.0587	Skills
gig_num	0.053	# of gigs
language	0.0498	Language
area	0.0344	Area

各预测算法分析后与实际测量对比如下（各个算法性能相似）：





1.12 Gig 服务质量预测

分别利用对 GSRC 的每一个部分进行预测分析，得到性能表如下：

TABLE VI. GIG QUALITY RESULT

Method	RMSE			
	G	GS	GSR	GSRC
Ridge	0.4075	0.3984	0.3973	**0.3923**
Lasso	0.4064	**0.3906**	0.3967	0.3956
SVR	0.4036	0.3883	0.3872	**0.3855**
RF	0.3338	0.3305	0.3303	**0.3267**
GBRT	0.3282	0.3180	0.3203	**0.3171**

Method	Accuracy			
	G	GS	GSR	GSRC
Ridge	0.7778	0.7823	0.7857	**0.7937**
Lasso	0.7789	**0.7971**	0.7880	0.7902
SVR	0.7981	0.8073	0.8095	**0.8174**
RF	0.8968	0.8968	0.8990	**0.9048**
GBRT	0.8889	0.9138	0.9138	**0.9172**

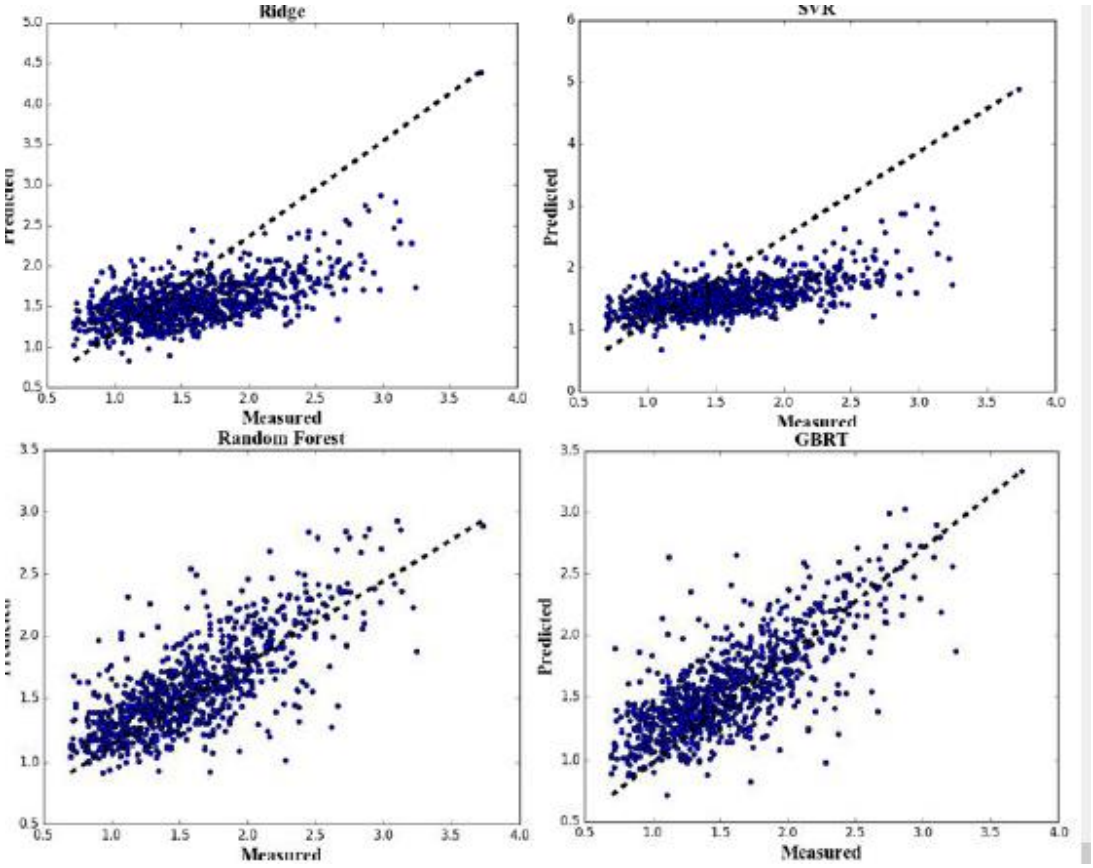
通过这两个分析表可以看出当考虑 GSRC 的每一个部分并采用 GBRT 作为预测分析算法的时候，我们可以得到最大的的准确率以及最小的 RMSE，即性能最优

Method	RMSE	Accuracy
GBRT + GSRC(Seller Level)	0.3309	0.9010
GBRT + GSRC(Seller Score)	**0.3171**	**0.9172**

通过上面这张表可以看出改进后的 Seller Score 比改进前的 Seller Level 具有更好的分析性能。

Method	RMSE	Accuracy
GBRT + GSRC(Length, count)	0.3203	0.9082
GBRT + GSRC(TF-IDF)	**0.3178**	**0.9138**

通过上面这张表可以看出，添加了 TF-IDF 值作为分析指标后，我们获得了更好的性能分析效果。



通过上面这张表可以看出，GBRT 为最适合进行预测分析的算法。

1.13 个人观点

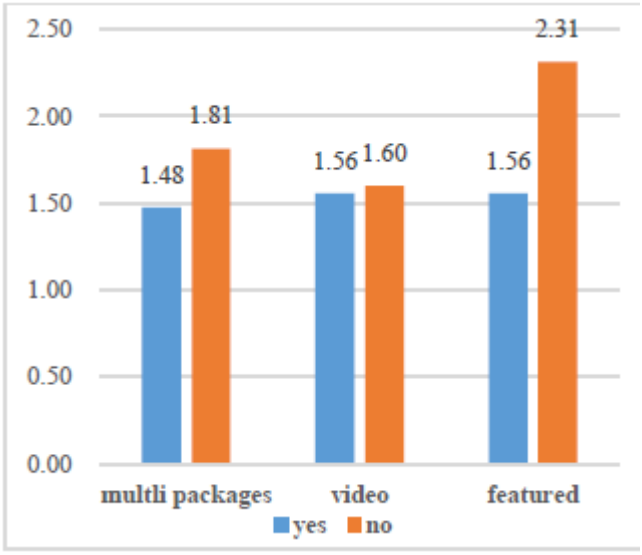
从整体的角度上看我十分支持该文章的观点，本文从实际生活中的网站发现了问题，并通过数据统计与研究十分科学地提出了相关解决策略，然后进行试验测试成功地证明了自己所提出观点的正确性。而且该研究十分有现实意义，可以为买家提供正确的卖家以及其产品质量的相关信息从而提高该网站信誉度，便会吸引更多的买家成为该网站的客户进行消费。不过在本论文中我发现了一些小的错误如下：

1 如下图，论文第四页中的这句话第四个小括号应当指的是（C）来代表 semantic content 从而来完美映射 GSRC 这四个字母。

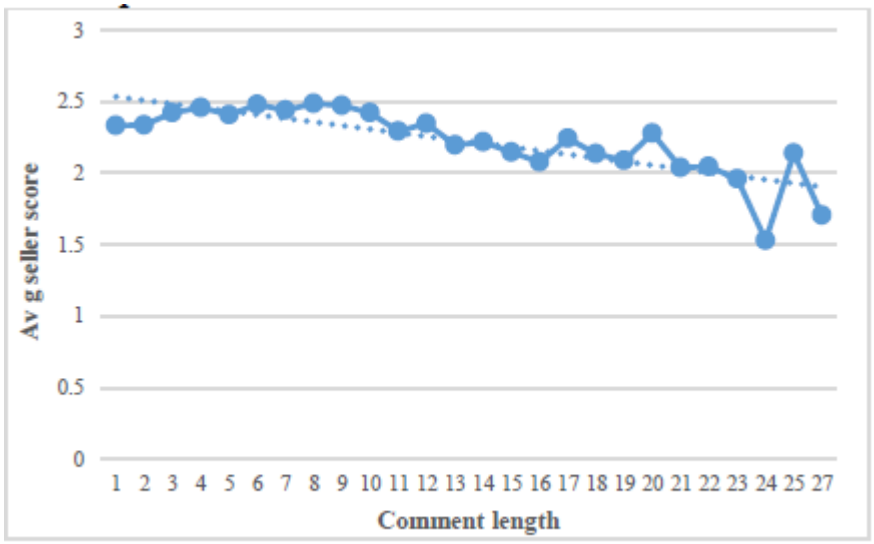
C. Gig Quality Prediction - GSRC Model

This study proposes a GSRC model that can be used to analyze and predict the gig's quality. This model consists of four elements: gig property (G), seller impact (S), customer review (R), and semantic content (S). This section will illustrate the detail of these elements.

2 第五页的这个图表错误，yes 和 no 所代表的柱状图标反了



3 看如下两个图



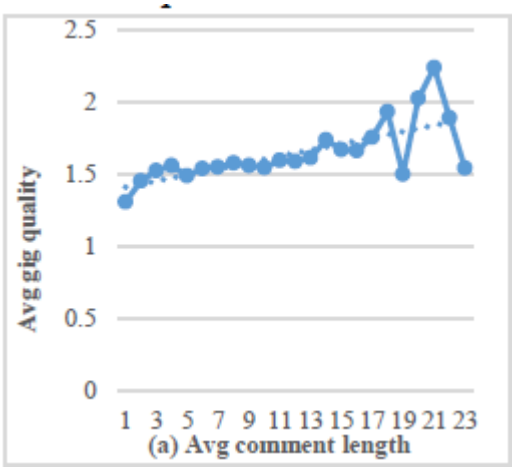


Fig 6. Customer comment length impact on seller score

左图为在评估 Seller Score 时得出的结论，认为用户评论长度越长，就可能是用户的抱怨文字，从而会使得 Seller Score 变低

而右图为评估 GSRC 中的 R 对 gig 服务质量影响的时候，认为用户评论长度越长，gig 服务的质量越高。

尽管在第二幅图片上作者指出用户评论长度和 gig 服务质量是弱的 (week) 正比关系，但我认为这两幅图还是产生了一定的矛盾，我们从后来的分析知道 Seller Score 越高，gig 的质量越好，按照上面的逻辑，用户评论文字越多，代表 Seller Score 越高，必会导致 gig 的质量越好，这种结论与第二幅图相反。再仔细观察第二幅图，可以发现当用户评论长度较长时（17 到 23 之间），这一段波动很大，不具备一般代表性，我认为此时这种关系可以不加以考虑。（如果这两个 comment 一个代表对卖家的 comment，一个代表对 gig 服务的 comment 那么则可以说的通，但文中未详细区别，结合现实来看通常我们购物只对产品进行评论，故我认为这两个 comment 都指的是对 gig 服务的评论）

注：以上仅代表个人观点，极有可能是我错了

论文 2

2.1 Title

Freelancer influence evaluation and Gig service quality prediction in Fiverr

2.2 题目

在 Fiverr 网站中用户影响评估以及 gig 服务质量的预测（注：Fiverr 网站提供的服务叫做 gig 服务）

2.3 文章摘要

当前的 Web 服务技术正在逐渐向简单的方法来定义 Web 服务的 API 进行进化。这种进化会挑战现有的为描述 Web 服务组成的语言，可能会导致曾经的语言和最新的 Web 服务不兼容。RESTful Web 服务引入了一种新的抽象：资源。这不与固有的 Web 服务描述的范例语言（WSDL）兼容。因此，RESTful Web 服务很难使用业务流程执行语言（WS-BPEL）进行描述，这是由于 WS-BPEL 与 WSDL 的紧密耦合关系。 为 REST 拓展的 BPEL 语言在文中具有双重层面上的含义。首先，我们的目标是这种新的 BPEL 语言可以同时作用于 RESTful Web 服务和传统 Web 服务。其次，我们展示如何发布 RESTful Web 服务的 BPEL 进程，通过使用 REST 交互原语暴露它执行状态的所选中的状态。我们提供一个为 REST 给出 BPEL 过程设计的详细示例，它可以应用于协调一个 RESTful 电子商务场景，并讨论如何提出扩展 BPEL 影响进程执行引擎的体系结构的。

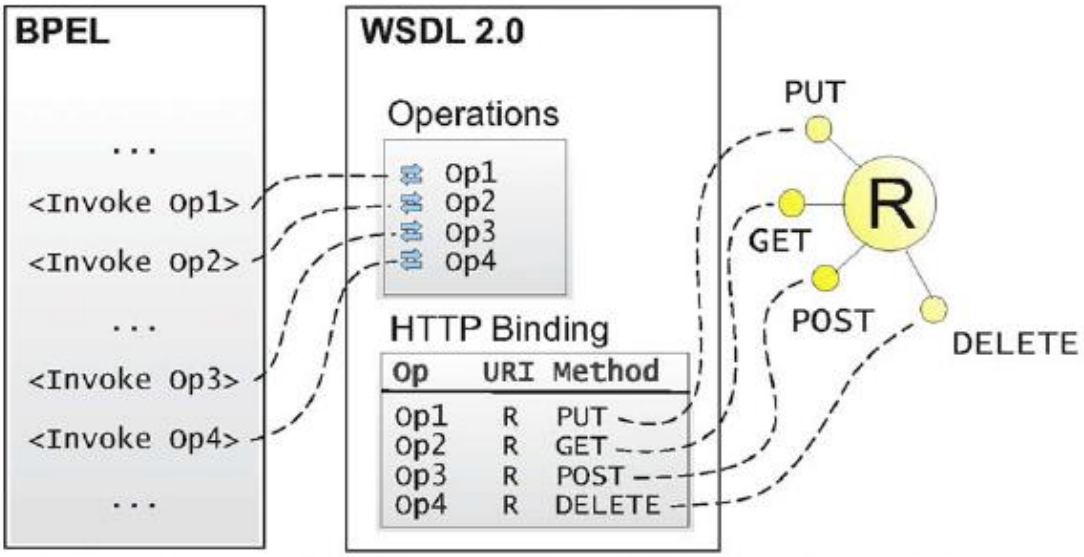
2.4 文章目的

1 提出一个拓展性的 BPEL 语言，这种新的 BPEL 语言可以同时作用于 RESTful Web 服务和传统 Web 服务。

2 我们通过一个具体的实例展示如何发布 RESTful Web 服务的 BPEL 进程，讨论扩展 BPEL 进程对执行引擎的体系结构的影响。

2.5 文章动机

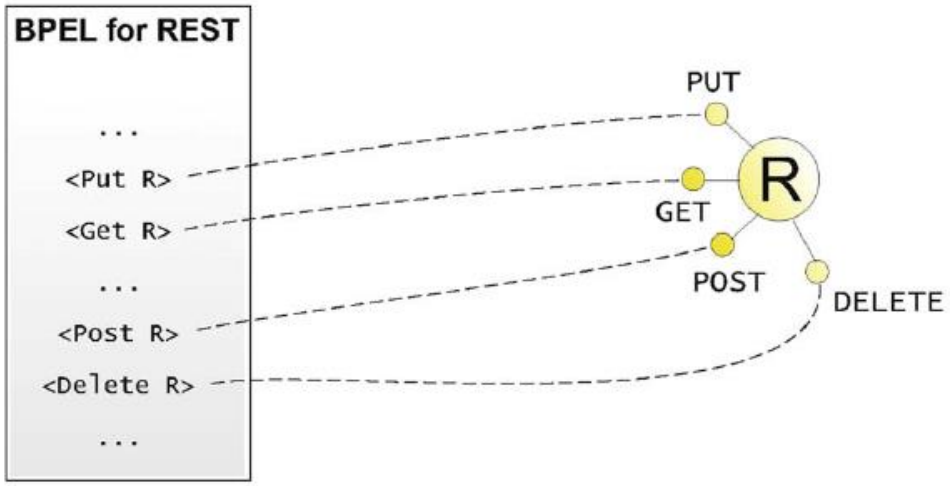
在本文中我们提出另一个使 BPEL 本身支持 RESTful Web 服务组合的扩展。现有努力一直将将重心放在了普通 BPEL（无扩展）实现 RESTful Web 服务通过 WSDL 2.0 HTTP 绑定技术。本文反对这种方法，理由是 WS-BPEL 2.0 尚未支持 WSDL 2.0 的参数，而是针对其前身 WSDL 1.1 进行设计。下图示出了使用新引入的 HTTP 绑定，



(a) Wrapping RESTful Web services through the WSDL 2.0 HTTP Binding

由图知，这种绑定可以在 WSDL 文档背后包装 RESTful Web 服务。此绑定使 BPEL 流程能够通过 HTTP 协议发送和接收数据，而不使用 SOAP 消息格式。因此，从与 BPEL 的支持这个角度来看，似乎使用 BPEL 来组合 RESTful Web 服务的问题已经解决了。然而，这个解决方案并不令人满意。

从理论的角度来看，这种方法隐藏了面向服务的抽象背后的资源抽象和相应的 RESTful 交互原语。由 WSDL 提供的组合机制（即同步和异步消息交换）与“GET”，“PUT”，“POST”或“DELETE”在资源 URI 上执行的请求的语义不匹配。因此，我们认为明确控制用于调用服务以及用于发布 BPEL 流程状态作为进程中的资源的本机支持的 RESTful 交互原语将是十分有益处的。如下图：



(b) Direct invocation of RESTful Web services using the BPEL for REST extensions

实际上，WSDL 2.0 还没有得到广泛的部署，特别是描述现有的 RESTful Web 服务，而且还有很多，并且很少有证据表明这将在将来发生改变。因此，重新构建包含 RESTful Web 服务的 WSDL 描述的负担从服务提供者转移到 BPEL 开发人员身上。无论何时更新 RESTful Web 服务，所有客户端都必须更新其相应 WSDL 文档的副本。这与现有的最佳做法相矛盾，（最佳做法：描述服务接口的合同应由服务提供商维护，而不是由服务的消费者维护）

2.6 RESTful 服务组成

在这里简单提一下

- 通过 URI 来安置资源（需要 BPEL 支持在一次运行环境下与 URI 地址进行动态绑定）
- 统一的用户接口（使用 HTTP 协议内置方法）

  GET、PUT、POST、DELETE（因此，在为 REST 服务的拓展性 BPEL 中，需要将他们添加成新进程的标准实例化机制。
- 自我描述性信息（BPEL 需要能够控制 REST 使用的 meta-data 媒体信息，这些媒体信息用于控制 RESTful 服务内部互相控制的属性，且 BPEL 支持适用于每个客户端的上下文的替代格式）。
- 超媒体作为应用状态的引擎（拓展 BPEL 应为进程提供一种生成新资源 URI 并将其发送回客户端的机制）。

2.7 拓展 BPEL 组成

主要包含三方面的拓展，如下：

1 拓展性 BPEL 进程应该支持对 RESTful Web 服务的直接唤醒，为了实现这个目标，我们提议增加以下四种活动：<get>、<post>、<put>、<delete>

```
<get uri="" response="" response_headers=""?>
  <header name="">*value</header>
  <catch code="">*...</catch>
  <catchAll>?...</catchAll>
</get>

<post uri="" request="" response="" response_headers=""?>
...
</post>

<put uri="" request="" response=""? response_headers=""?>
...
</put>

<delete uri="" response=""? response_headers=""?>
...
</delete>
```

Fig. 3. BPEL for REST extensions to invoke a RESTful Web service.

2 BPEL 进程的执行状态应当被看作为其发布的一个资源，因此我们通过<resource>容器来实现，这个结构可以依据相关的声明是否通过 BPEL 进程执行中正确的送达而可以动态地发布资源给客户。

```
<resource uri="">
 <variable>*
 <onGet>?                        ... </onGet>
 <onPut>?                        ... </onPut>
 <onDelete>?                     ... </onDelete>
 <onPost isolated="false"?>?  ... </onPost>
</resource>

<respond code=""?>
 <header name="">*value</header>
  payload
</respond>
```

Fig. 4. BPEL for REST extensions to declare resources within a process.

3 一些 BPEL 的结构的执行状态下的语义需要被修改来对应上 REST 的设计原理，首先修改拓展性的语义空间 namespace 如下：

```
<extensions>
  <extension namespace="http://jopera.org/bpel/rest/2008"
             mustUnderstand="yes" />
</extensions>
```

Fig. 5. BPEL for REST extensions namespace declaration.

鉴于 RESTful Web 服务没有明确定义的界面描述，以及对要交换的数据缺乏强类型约束，且 REST 的 BPEL 是一种动态类型的语言。因此<variable>声明的静态类型变为可选项。不使用属性 messageType（该属性直接依赖于 WSDL），而在在 RESTful Web 服务的 XML 模式描述的情况中，仍可使用这些类型或元素属性。

2.8 架构

拓展性 BPEL 引擎的结构图示如下：

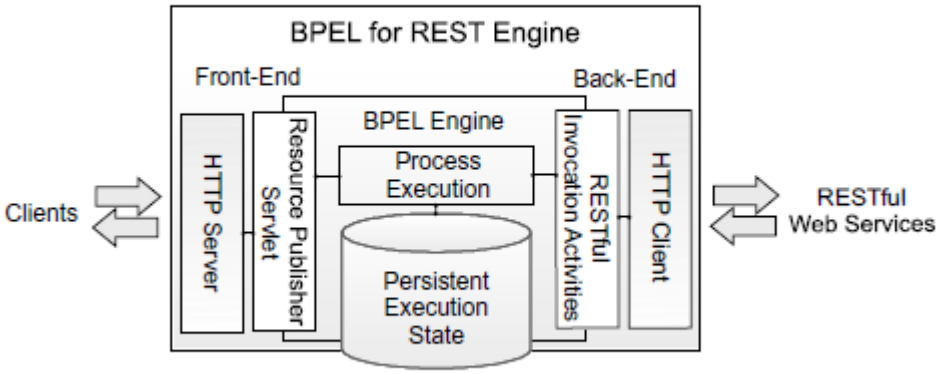


Fig. 6. Reference architecture for the BPEL for REST extensions.

2.9 实例

本部分我们给定一个实例显示拓展性 BPEL 是如何在实际中应用与 RESTful Web 服务中的。具体实例的场景为一个商店的服务，如下图：

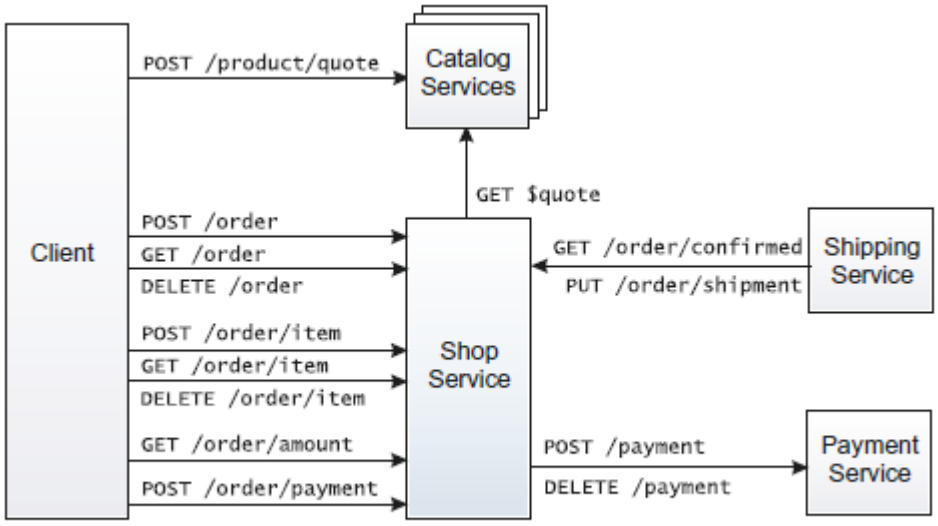


Fig. 7. RESTful e-Commerce Scenario.

由图可知，用户可以通过目录 catalog 并且对指定服务请求报价，如果商家给出的价格符合用户的要求，他们便会向商店 shop 下订单，一旦订单被创建后，用户便可以在该订单中添加或删除条目，以及检查订单的总金额。用户可以通过支付信息来确认一个订单，一旦订单被用户确认，用户便不可以更改这个订单。然后商家通过运输服务来给客户运送商品，用户可以获知商品的物流实时信息，如果商品丢失或发生了其他意外情况以至于没有被用户正确地收到，商家将会退还给客户之前支付的金额。

上图中 RESTful 服务的 API 总结：

Verb	URI	Parameters	Description
Shop Service			
GET	`/order`		Get the list of orders.
GET	`/order/confirmed`		Get a list of confirmed orders.
GET	`/order`	*oid*	Read information about the order *oid*.
GET	`/order/amount`	*oid*	Retrieve the total amount for order *oid*.
POST	`/order`	*cid*	Create a new order for customer *cid*
DELETE	`/order`	*oid*	Cancel order *oid*
GET	`/order/item`	*oid*	Get the list of line items for order *oid*.
GET	`/order/item`	*iid*	Read information about the line item *iid*.
POST	`/order/item`	*oid*	Create a new line item for order *oid*
DELETE	`/order/item`	*iid*	Remove line item *iid* from its order
POST	`/order/payment`	*oid*	Update order *oid* with payment information and proceed to checkout.
PUT	`/order/shipment`	*oid*	Update order *oid* with the shipment information once its items have been shipped.
Catalog Service			
GET	`/product`		Get the list of products of the catalog service.
GET	`/product`	*pid*	Read description of product *pid*.
POST	`/product/quote`	*pid*, *qty*	Request a quote for ordering a certain quantity *qty* of product *pid*.
GET	`/quote`	*qid*	Read the product *pid*, *price*, and *qty* associated with a certain quote *qid*.
Payment Service			
POST	`/payment`		Perform a payment.
DELETE	`/payment`	*vid*	Refund a previously charged *vid* payment.

商店服务发布的订单资源遵循下图所示的状态机。 状态转换由商店服务客户的具体请求触发。 该状态机可以通过使用本文中介绍的扩展的 BPEL 流程实现。

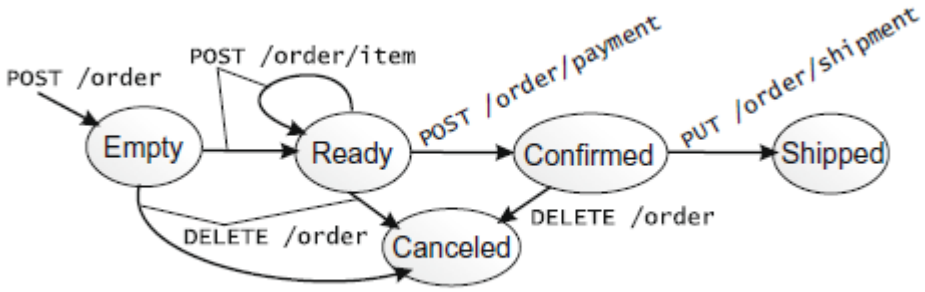


Fig. 8. State of an order managed by the shop service.

具体的实现代码在原论文的附录中，代码也在原文中进行了详细的解释说明，主要按照之前修改得到的拓展性 BPEL 来描述该 RESTful 服务。

2.10 讨论

用于 REST 扩展的 BPEL 的主要设计决策是使开发人员能够在 REST 定义的抽象级别组合服务。因此，语言通过反映 RESTful 交互原语的活动来进行拓展，并直接支持资源的组合。在这种情况下，重要的是讨论业务流程概念与资源抽象之间的关系。

使用 REST 而生的拓展性 BPEL，资源构造使得一个进程可以声明地发布一组任意的资

源。此外，可以使用业务流程来实现特定资源的状态转换逻辑，该业务流程定义资源应在哪些条件下改变状态以及如何应对客户端请求。因此，正如我们在示例中所示，使用拓展性 BPEL 可以定义 BPEL 流程实例的状态与其资源状态之间的关系。

BPEL 流程实例的生命周期从接收到触发执行具有 createInstance 属性集的实例接收或 pick 的消息开始。一旦一个进程被实例化，它的状态由分配给它的变量的值和一个“指令指针”组成，指示其活动的哪个子集当前是活动的。在执行期间，交换的所有消息都会根据其内容和过程中声明的相关集合与特定流程实例进行关联。流程实例的执行继续进行，直到达到退出活动，或者该流程完全用完可执行的活动。一旦执行完成，流程实例的状态通常被丢弃。

资源的生命周期以 POST / PUT 请求开始，以初始化其状态。资源创建完成后，客户端可以使用 GET 请求读取其当前状态，使用 PUT 请求更新其状态，并使用 DELETE 请求将其丢弃。

用于 REST 的 BPEL 不会对资源是否被实例化（作为进程执行的一部分），还是针对已发布资源的特定请求创建新的流程实例施加任何限制，这两种方法均会被支持。如果资源被声明为 BPEL 流程的顶级元素，则只要 BPEL 代码部署为执行，客户端就可以与其 URI 进行交互，无论流程实例是否尚未启动。如果从进程的本地范围内声明资源，则仅当进程实例的执行到达特定镜像时，才会发布其 URI。通过在顶级和本地资源声明之间引入这种区别，BPEL for REST 支持纯粹的面向资源的组合风格，BPEL 流程的结果是通过 REST 统一接口访问的资源，可以多次实例化。然而，BPEL 流程也可以使用标准兼容的启动活动来实例化，并在执行期间发布资源，将部分状态暴露给客户端作为 RESTful 网络服务。

在 REST 的 BPEL 中，使用其附加的请求处理程序来处理资源的状态。通过从<onPut>或<onPost>请求处理程序中的资源状态变量初始化创建一个新的资源实例。为了让客户端识别特定的资源实例，在最简单的情况下，可以由 BPEL 引擎自动生成 HTTP Cookie 来处理 PUT 请求。引擎可以拦截携带 HTTP 状态代码 201（创建）的响应，并添加具有唯一标识符的 cookie。客户端将发送 cookie 用于所有将来的交互（例如，GET，PUT 和 DELETE）与资源 URI，并且引擎将使用 cookie 将请求与资源实例的正确状态相关联。

基于 URI 重写的基于模板的资源标识符生成的无 Cookie 解决方案也是可能的。这将工作如下：给定顶级 URI 资源（例如/ order），如果使用 201（创建）状态代码应答 POST 请求，引擎将发出一个 Location：i 重定向头（我是一个唯一的新创建的实例的标识符），以便客户端可以进一步向特定资源实例 URI / order / i 指导 GET，PUT 和 DELETE 请求。嵌套资源 URI 仍然通过连接其 uri 属性来构建，该属性现在也应该交织资源实例标识符。对原始资源 URI /订单的 GET 请求将检索到由活动进程发布的所有活动资源实例的超链接。

与大多数现有 Web 开发语言和基础架构透明支持 HTTP 会话管理类似，这些（或其他）请求相关机制之间的选择不应影响语言的设计。这样，当部署进程执行时，可以选择最合适的关联机制。

2.11 个人观点

我十分欣赏这篇文章，本文解决了一个确确实实存在的问题。首先在理论方面提出如何对 WS-BPEL 进行拓展，使其兼容 RESTful Web 服务，这样源自 REST 统一接口原理的交互原语（GET、POST、PUT、DELETE）可以直接从 BPEL 流程中用作新的服务调用活动。接下来将该拓展性 BPEL 应用于一个实际的案例。对其进行详细的分析和解读，并讨论了它对执行引擎的体系结构的影响。作为读者的我深入理解了整个过程的每一个细节，收获颇丰。