**Web服务发现两篇文章总结**

**为什么看着两篇文章呢？**

主要是想参考针对语义Web的服务发现机制，形成自己针对物联网环境下的服务发现机制。因为，物联网包括感知功能、计算功能均可以打包成由WSDL、OWL-S或者是hRests描述的服务。我们只需要为上层应用暴露开发接口，上层应用调用这些服务去形成自己的功能逻辑，就可以实现对底层设备的屏蔽。同时基于语义的服务发现也可以实现上层应用对合作对象的屏蔽，也就是应用并不需要知道它具体使用了那些设备或者哪些计算服务，它只需要告诉中间件它想得到一个什么样子的总体服务，中间件为其打包实现所有的逻辑。

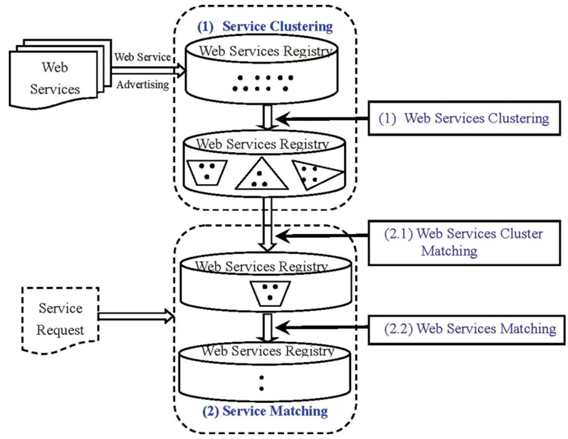
**这两篇文章的整体设计思路是一样的：**

first，定义服务的相似度评价模型

second，根据服务描述计算各个服务相似度，进行服务预聚类

third，客户的服务请求同服务类簇进行匹配

fourth，针对候选服务进行更为细致的模式匹配



**Paper 1： Web service discovery among large service pools utilising semantic similarity and clustering**

**1.对OWL-S格式服务描述进行针对服务发现的格式抽象**

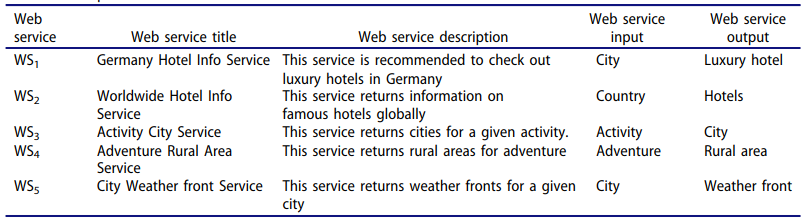
刨除掉对服务发现的冗余信息，也就是OWL-S格式的grounding属性，并将其余部分抽象为一个四元组<D, I, P, O>

D： 服务的描述属性

I：服务的输入输出接口

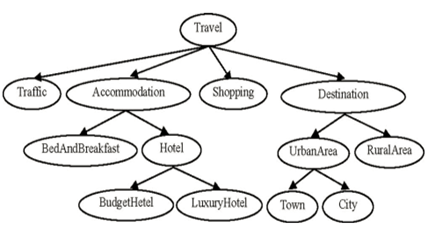
P：过程属性，也就是服务描述的自己每一步执行的逻辑

O：服务所涉及的ontology集合



其中，输入输出均为domain ontology concepts

**2.定义用于预聚类和具体服务匹配的服务相似度模型**



ontology concepts structure

因为无论是服务描述、输入输出参数还是过程属性都会涉及domain ontology concept，因此首先给出concept的相似度模型：

C:\Users\ZJBPOP~1\AppData\Local\Temp\enhtmlclip\Image(2).png

这是一个基于概率的相似度评价方法，LCA指的是Least Common Ancestor

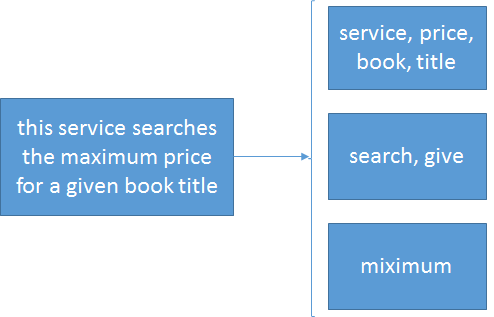
相较于前人的工作，本文主要是提出了后面的oppositeness这一个指标，也就是引入反义词指标，引入这个指标是很显然的，假设上面的UrbanArea和RuralArea是一对反义词，但它们由于具有相同的父节点，单纯利用前半部分的相似度评价的话实际是具有很高相似性的，但实际结果是两者表述的概念是相反的，尤其是对于形容词更甚。

至于这个oppositeness怎么得来= =，作者自己形容是a combination of thesaurus- and antonym-based algorithm，实际呢就是，作者首先收集了630万的反义词对，包含WordNet反义词对，以及通过dis之类修饰的反义词对。这就是oppositeness不为0的集合，然后对自己的本体集合中所有这些反义词对，让三个研究生给出一个0到1之间的数值，然后取平均作为最后的oppositeness。**我只想说，这尼玛也太low了一点吧，可以加以改进= =**

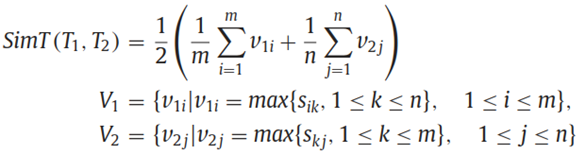
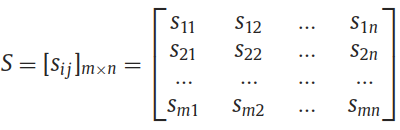
不管low不low，也是有改进，现在有了concept的相似度，我们可以得到服务描述的相似度也就是D的相似度，文章又是瞎他妈乱写，给的是这么个公式：

C:\Users\ZJBPOP~1\AppData\Local\Temp\enhtmlclip\Image(3).png

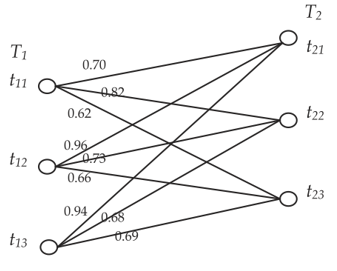
显然是不对的呀，神能保证两个服务能提取出一样数量的concepts？就算是你按照precondition或者effect标签取，textdescription这个最重要的部分不要了？看了下作者的另一篇文章，粗略的翻了一下，服务描述的相似度应该是这个样子的：



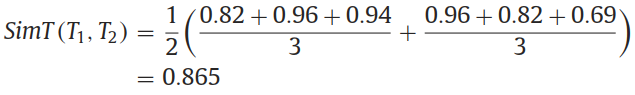
首先通过自然语言处理，把服务描述的concepts提取出来，然后的到两个服务的相似度矩阵



然后，两个服务的描述相似度定义为如上，取每个concept最大的相似度然后取平均。这里服务用T来表示了。



上图的两个服务描述的相似度就为：



现在得到了服务描述的相似度，对于输入输出接口的相似度，以输入相似度为例，作者给出的方法是通过将两个服务的所有input参数也就是ontology concept转换成一个字符，这样就串成了一个字符串，通过Edit Distance来比较两个字符串也就是Input接口的相似度，输入输出相似度模型如下：

C:\Users\ZJBPOP~1\AppData\Local\Temp\enhtmlclip\Image(9).pngC:\Users\ZJBPOP~1\AppData\Local\Temp\enhtmlclip\Image(10).png

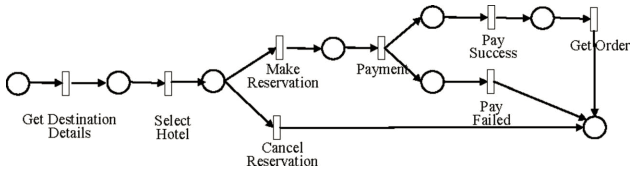
然后，服务描述和输入输出可以用来表明一个服务所代表的功能，因此，定义功能相似度如下：

C:\Users\ZJBPOP~1\AppData\Local\Temp\enhtmlclip\Image(11).png

**3.定义用于服务匹配的过程相似度模型**

另外，相较于功能相似度，还需要定义一个过程相似度，对应OWL-S中的composite process所描述的服务逻辑。来比较请求与服务之间过程逻辑的相似性。并不是所有的OWL-S都会有，仅限于Composite process

首先引入Petri net来描述服务的process，如下是一个OWL-S定义的酒店预订的过程:



然后两种过程相似度的评价模型：

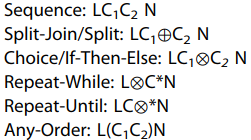
first，基于图的过程相似度：

C:\Users\ZJBPOP~1\AppData\Local\Temp\enhtmlclip\Image(13).png

其中E1和E2代表服务1和2对应的图，E0表示E1和E2重合的边所组成的图；

second，基于LCS也就是最长公共子序列的相似度：

首先，用于描述process的控制逻辑有如下几种，将它们转化成序列的形式;



然后相似度如下：

C:\Users\ZJBPOP~1\AppData\Local\Temp\enhtmlclip\Image(15).png

**4.利用DBSCAN算法基于功能相似度对服务进行聚类**

输入：初始数据集合、邻域半径(radius)和密度域值(minPts)

建立聚类集合：分别以每个对象为考察对象判断其是否为核心对象，如果是核心对象则建立聚类集合

合并集合：根据密度相连的原则合并聚类集合

输出：输出整理合并达到密度域值要求的集合

其中：

核心点：在半径eps内含有超过minPts数目的点。

边界点：在半径eps内点的数量小于minPts ，但是落在核心点的邻域内，也就是说该点不是核心点，但是与其他核心点的距离小于eps。

噪音点：既不是核心点也不是边界点的点，该类点的周围数据点非常少。

note that，两个服务之间的距离即为服务的功能相似度。

**5.利用功能相似度为服务请求匹配合适的服务cluster**

note that，服务请求也是基于OWL-S格式的

**6.最后，利用功能相似度与过程相似度在候选的服务中匹配精确地匹配服务**

C:\Users\ZJBPOP~1\AppData\Local\Temp\enhtmlclip\Image(16).png

**Paper 2：An Approach to Computation of Similarity, Inter-Cluster Distance and Selection of Threshold for Service Discovery Using Clusters**

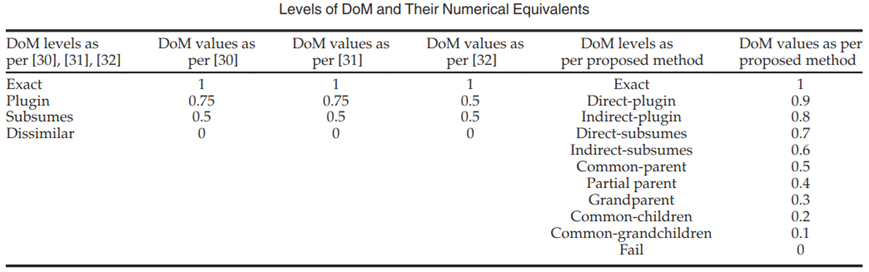
由于hRESTs在数据上较之OWL-S格式的数据描述的内容较为轻量，所以本文的相似度模型没有考虑process相似度和Description相似度。

与第一篇论文的思路相似，在这里主要介绍一下它在相似度模型上的不同点。

本文的观点是，输出参数的相似度才是预聚类的主要依据，它相较于，输入相似度或者是输入输出的相似度，更能排除各种服务功能的无关性。因此，本文在预聚类中采用输出相似度作为聚类的指标。

concept的相似度通过DoM（Degree of March）匹配程度的不同level进行计算和描述。

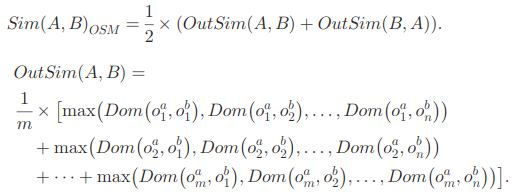
前人的DoM与本文的DoM的比较：



较之前人，本文基于ontology的concept结构提出了更加细粒度的DoM level及其相应的评价指数，通过该value进行输入输出相似度的计算

计算公式如下：

输出相似度模型：



总体相似度：

C:\Users\ZJBPOP~1\AppData\Local\Temp\enhtmlclip\Image(20).png

预聚类采用OSM，精确地服务匹配采用TSM。