一种基于文档的移动平台间UI控件对应方法

徐同同[1] 刘曲涛[1] 郑晓梅[2] 潘敏学[1] 张 天[1]
(南京大学计算机软件新技术国家重点实验室 南京 210023)[1]
(南京中医药大学信息技术学院 南京 210023)[2]

摘 要 多平台开发是移动应用软件开发的一个重要特点,同时还具有版本演化快和开发周期短的要求,这给移动开发带来了巨大的挑战。由于目前主流的移动平台大多采用MVC架构模式,并且在开发上体现出了UI驱动和事件驱动的特点,因此不同平台(如iOS和Android等)之间的UI控件具有较强的对应性,这给移动应用的开发人员在多平台开发时提供了重要的参考。提出了一种基于文档来理解不同平台之间UI控件对应性的方法,该方法以iOS和Android两种移动平台为研究对象,通过自然语言处理技术来分析从官方文档中抓取的UI控件描述文字,基于空间向量模型计算控件之间的相似度,并针对移动应用的特点设计了同义词集来保证匹配的准确性。基于所提方法,对iOS和Android平台上的典型UI控件进行了具体实验,结果表明,在单控件对应性方面,该方法能找到大部分的控件对应性,具有较高的准确度。

关键词 自然语言处理,移动开发,UI控件
中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.11.015

Document Based Matching Method for Mobile UI Components

XU Tong-tong[1] LIU Qu-tao[1] ZHENG Xiao-mei[2] PAN Min-xue[1] ZHANG Tian[1]
(Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China)[1]
(School of Information Technology, Nanjing University of Chinese Medicine, Nanjing 210023, China)[2]

Abstract Development for multi-platform is an important requirement of developing mobile applications. At the same time, the developments should be flexible for short cycle and rapid evolution. These are challenges for mobile developments. Fortunately, most mobile platforms are designed as MVC-based and event/UI-driven. Hence the UI components are similar between different platforms, which is very helpful when developing Apps from one platform to another. This paper provided a document based matching method for uncovering the similarities of UI components between different platforms. The iOS and Android are selected, and the documentations for their UI components are extracted from the official websites. Then the NLP techniques are used to build the vector space model so as to compute the similarities of UI components between two platforms. To increase the accuracy, the sets of synonymous words were presented according to the UI features of components. The experiments were performed on a set of typical iOS and Android UI components. The results illustrate that the accuracy of the method is acceptable for the most UI components especially for those of one-one-matching.

Keywords NLP, Mobile development, UI components

1 引言

移动开发是软件开发的一个类别,是当今软件开发市场的热点。移动开发的一大特点是平台多、版本多。目前世界上流行最广的平台有iOS开发平台、Android开发平台和windowsphone开发平台,此外还有很多小众的开发平台,如黑莓、索尼;版本繁多、更新频繁是移动开发的另一个特点,如Android在2008年9月至2012年1月短短3年多的时间内推出了多达几十款不同版本的系统,并且不同版本的系统都有较大比例的手机植入。平台多、版本多的特点给软件开发

到稿日期:2016-10-11 返修日期:2016-12-25 本文受基于MDE的异构数据建模及转换研究(61472180),基于场景规约的中断驱动系统的建模与验证技术研究(61502228),基于SysML和MARTE的异构数据模型转换方法研究(BK20141322),中断驱动系统的建模与分析(BK20150589)资助。

徐同同(1993-),男,主要研究方向为软件工程;刘曲涛 男,硕士,主要研究方向为软件工程;郑晓梅(1978-),女,硕士,副教授,主要研究方向为软件工程;潘敏学(1983-),男,博士,助理研究员,主要研究方向为形式化方法和软件工程,E-mail:mxp@nju.edu.cn;张 天(1978-),男,博士,副教授,主要研究方向为软件工程。

人员带来了很大的困难。

为了帮助移动开发人员解决这些问题，提高移动开发的效率，本文研究了移动开发的特点与方式。通过研究发现，不同移动开发的平台或版本具有本质的共同点，即都是基于 UI 驱动、事件驱动的，开发者需要学习控件的功能和使用方法。并且 UI 控件在不同的平台架构上都具有 mvc 设计模式的特点。从 UI 控件设计的角度出发，UI 控件需要体现出移动交互的特点，如点击触发事件、压力感应、可视化等；从设备硬件能力的角度出发，不同设备拥有类似的硬件条件，如陀螺仪、gps 接收器，因此在不同平台的设计上其具有相同的设计需求，故控件具有一定的对应性。软件的设计开发人员可以利用这种对应性，以提高跨平台应用开发的效率。本文主要聚焦于研究 iOS 和 Android 两种主流移动平台给定版本情况下的控件的对应性，通过理解两种平台上 UI 控件的相似性，给出从一种平台到另一种平台界面控件的对应关系，这种对应关系可以帮助开发人员更好地进行移动应用的设计与开发。

一般开发者开发移动应用通常需要同时对两个平台具有一定程度的理解，考虑到平台控件较多且版本演化非常快速，开发者同时掌握几种平台的开发方法较为困难，且无法适应移动开发短平快的特点。因此我们希望实现不同平台与版本中控件对应的自动化，帮助一般开发者快速理解两个平台开发的特点，提高开发效率；观察到 Android 与 iOS 都会提供随版本更新的控件文档信息，因此可以利用这些控件文档，通过合适的算法自动化地得到不同平台版本的控件对应性。本文提出了一种基于自然语言处理的方法，通过定量计算出控件文档的相似性，推导出控件间的对应关系。我们的方法在形式上分 3 步：1）找到合适的可供使用的 iOS 与 Android 平台的控件文档，在本文实验中，使用两个平台的官网控件文档[5-6]，通过观察分析文档结构，选取了能够体现控件特点功能的部分文档内容；2）设计自然语言处理算法，对各个控件文档过滤无关词汇，同时提取合适的关键词并优化其形式，综合考虑单词间自然语言层面的同义性，对于两两控件，给出足够精细的特征向量；3）将空间向量模型作用于描述控件文档的特征向量，给出两两控件的相关性数值，继而构筑控件的相关矩阵，以此为依据实现控件的对应性判定。我们的程序最终可以实现不同平台控件对应的自动化，在以 Android 控件为查询关键词的情况下查询得到 iOS 平台控件，两者的对应关系的正确率达到了 81.8%，考虑到一些控件实际上并无其他平台控件与之对应等问题，这样的正确率可以接受。

2　背景介绍

近年来，搜索引擎的发展非常迅猛，涌现出了很多高效、易于实现的信息检索算法，包括布尔模型、空间向量模型、基于机器学习的搜索模型等。本文程序借鉴了搜索引擎算法的相关思想，为了对 iOS 控件与 Android 控件之间进行相关性的定量描述，选择使用空间向量模型进行定量描述，一方面，控件文档的专业性较好且篇幅短小，易于空间向量模型进行关键词提取；另一方面，空间向量模型本身易于实现，且性能优良。对于一组待定控件，首先通过自然语言处理优化出关键词集合，以此为维度构造特征向量。需要注意的是，对于不同控件，其对应的关键词集合是不同的，特征向量的长度由关键词集合的关键词数目决定。一旦特征向量构建完成，便可运用空间向量模型计算它们的归一化余弦，其被称为文档的相关度，归一化余弦公式如式(1)所示，其中，a_i 是第 i 个控件的特征向量，a_j 是第 j 个控件的特征向量，如式(1)所示。空间向量模型是对两个向量相关性的描述，本文将控件文档之间的相关性近似看成控件间的相似性，即不同平台控件之间的对应性。

$$Sim(i,j)=\frac{\vec{a_i}\cdot\vec{a_j}}{|a_i|\times|a_j|} \tag{1}$$

近年来自然语言处理技术发展较快，本文使用了一些现有的成熟的自然语言处理工具。其中包括 apache 基金会下的 lucence[1]，这是一款集成了很多自然语言处理技术的开源软件，使用方便且高效。我们使用 lucence 提供的一些自然语言处理技术辅助优化程序设计。此外，还使用了斯坦福大学开发的 stanford corenlp[2] 提供的一些自然语言处理技术（包括停词过滤、词形还原等）来优化程序。在程序中使用了 PorterStemmer[3] 提供的包含数万个专业词汇与相应词根形式的词根表，通过词根提取技术，可以使程序得到进一步优化。

3　基于自然语言处理的控件对应性分析

3.1　方法框架

本文方法的结构流程图如图 1 所示。其中，iOS 文档与 Android 文档是程序中使用的控件描述文本，之后的 NLP 模块以此为输入提取出合适的关键词集合。NLP 代表自然语言处理模块，是整个结构流程图的核心，NLP 模块以 iOS 与 Android 控件文档为输入，经过 NLP 内部文本处理、优化算法，输出关键单词与相似词集集合，作为空间向量模型的特征向量维度。空间向量模型是搜索引擎处理文档信息的常用模型，具有很好的实际使用效果。

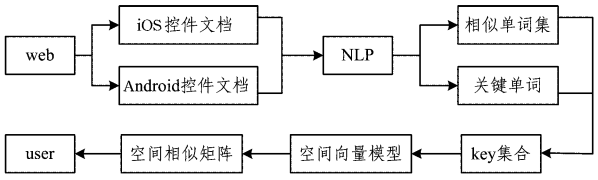

图 1　程序结构图

iOS 和 Android 控件文档是从相应平台的官网上爬取的，考虑到网页结构特征，爬取能够体现控件功能性的文档部分，例如对于 iOS 平台的控件，仅爬取 overview 与 purpose 部分的文档；对于 Android 平台的文档，仅爬取 overview 以及 class overview 的部分。实验显示，按此方式爬取的实验结果较好。

NLP 模块包含很多自然语言处理的功能模块，包括单词的提取、无语意停词的过滤、词形的还原、词根的提取、相似词集的构造。通过这些技术的处理、优化之后，最终构造出合理的关键词集合以供空间向量模型使用，关于 NLP 模块的具体实现见 3.3 节。

空间向量模型模块接受特定的 iOS 与 Android 平台的控件文档作为特征向量，接受 NLP 处理输出的关键词集合作为特征维度，以此计算特定控件间的相关度，即对应性，再遍历所有待选控件文档便能形成控件间的对应性矩阵，以此为依据判定控件的对应性关系。

3.2 控件文档的选取

表 1 是一组控件示例 Android textview 以及 iOS textview 的文档描述，是从各个平台的官网 User GuideLines 上爬取下来的，本文对算法的说明均以这组控件为示例。这些文档经过 NLP 模块处理后被优化为一组高质量的关键词集合(见表 2)，以这组关键词集合为空间向量模型的特征维度，构造 Android textview 以及 iOS textview 的特征向量，通过矢量相乘并归一化，得到这组控件的相关性。为了提供对比性，本文加入了另外一组控件 Android webview 以及 iOS webview 作为参考(实验样本共包括 20 个控件，为了书写清晰，仅展示其中两组的数值结果)，得到的相似性见 3.3 节。为了表达清晰，本节所有讨论都基于以 Android textview 以及 iOS textview 和 Android webview 以及 iOS webview 构建的空间向量模型，更全面的 UI 控件实验结果见第 4 节。

表 1 文档示例 textview(iOS，android)

Android_Textview：
Displays text to the user and optionally allows them to edit it. A TextView is a complete text editor，however the basic class is configured to not allow editing；see EditText for a subclass that configures the text view for editing. To allow users to copy some or all of the TextView's value and paste it somewhere else，set the XML attribute android：textIsSelectable to "true" or call setTextIsSelectable(true). The textIsSelectable flag allows users to make selection gestures in the TextView，which in turn triggers the system's built-in copy/paste controls. XML attributesSeeTextView Attributes，View Attributes
iOS_textview：
A text view accepts and displays multiple lines of text. Text views support scrolling and text editing. You typically use a text view to display a large amount of text，such as the body of an email message. Purpose. Text views allow users to： Input user content into an app

表 2 给定控件处理得到的关键词表

text	see
textview	copi
class	past
subclass	set
view	call
valu	trigger
xml	amount
attribut	email
android	messag
flag	purpos
select	input
gestur	content
system	accept
control	support
displai	Scroll
allow	
edit	messag content
configure	allow accept

3.3 空间向量模型

本文工作借鉴了搜索引擎算法的相关设计思想。在搜索查询领域，比较文档相似性的常用方法有布尔模型、空间向量模型等，本文使用空间向量模型表征控件间的相似性。首先介绍空间向量模型的相关工作，有助于对特定实例相似性进行描述。空间向量模型简单易于实现，且控件描述文本大多专业性强，利于空间模型使用。空间向量模型以选取文档单词的个数作为维度，在不做任何优化处理的情况下，维度可能成千上万。一个给定单词在空间向量模型中的特征权值一般由词频、逆文档频率相乘决定。对于一组待测文档，设置 m 个特征，根据一定规则，计算每个控件的文档在各个特征分量上的对应数值，即为每个文档赋予一个 m 维的向量，空间向量模型便是对 m 维的向量归一化后取余弦，所得的数值度量了两个文档的相关度。在本文实验中，相关度即是相似度，m 为向量长度，取值为关键词集合的元素数目，不同控件组间的特征维度不同，因为不同控件组所取的关键词集合不相同。空间向量模型公式如下所示，其中，a_i 是第 i 个控件的特征向量，a_j 是第 j 个控件的特征向量。

$$Sim(i,j)=\frac{\vec{a_i}\cdot\vec{a_j}}{|a_i|\times|a_j|}$$

首先选出针对不同控件的关键词作为特征维度，主要使用了自然语言处理知识领域的分词、原形抽取、去除停词、词根提取等方法，此外使用同义词归并为单维度词集的方式优化结果。一旦构建了针对特定两控件的特征基矢，便可将这两个矢量归一化取余弦，得出相关性，我们将这种相关性近似理解为控件之间定量的相似性。

考虑到实验目的在于找到 iSO 和 Android 控件间的相似性，我们构造了一个矩阵，其行代表 iOS 控件，列代表 Android 控件，每个矩阵元代表对应 iOS 与 Android 控件的相似性度量。每个矩阵元的值需要另外计算，我们选取关键词集合作为空间向量模型的特征维度，在保证文本向量归一化的前提下，以此计算两控件的相似性，并将其存放在对应矩阵元中。实验中，当需要考察某个控件与其他控件的相似性时，仅需要考察相应的行或列即可。

最后，对于用户查询的特定控件，选取相似性数值最大的控件作为另一平台的对应控件，在后续方法描述中，我们会频繁使用空间向量模型对实例进行分析论证。

表 3 相似矩阵

	iOS testview	iOS webview
Android testview	**0.3837927490**	0.1796181129
Android webview	0.3103662455	**0.4628665424**

表 3 列出了 3.2 节的两组控件通过空间向量模型计算的相似性矩阵，可以看到，对角元数值较非对角元数值更大，符合算法设计的初衷，关于数值结果的进一步讨论见 3.4 节。

3.4 基于 NLP 的关键词维度的确定

本节内容是整个程序的关键，实验结果的好坏完全由关键词集合选取的合理程度决定。可以从很多角度出发来评价 iOS 与 Android 控件的相似程度，例如实例代码、类继承关系与语法结构，本实验决定使用从官网爬取的控件描述文本作为整个实验的原料，一方面，可以从自然语言的角度理解控件的相似性，同时可以利用很多经验性的研究结果并合理增添文本描述；另一方面，与比较研究编译运行级别的代码相似性相比，本文方法更加简单可行。考虑到表述性文本中存在大量的无用词汇，以及同一单词有多种表达形式、不同单词间具有相似的语义，我们需要合理地处理这些差异性，去除冗余维度，尽量选取正交的特征维度，使实验结果尽量客观合理。我们使用一组 20 个控件组成的实验样本中的两组相似控件 testview 和 webview 作为比较说明，以表达不同优化下的性能的好坏。

3.4.1　停留词处理

一般的人类语言包含大量的介词、助动词、数词，例如 the，in，on，between 等，而这些词作为描述文本内容的维度是不合理的。在实验中，我们通过借助 apache 的软件包 lucence[1] 能方便快速地解决该问题。lucence 使用一个停词过滤文件类，可轻易地过滤掉冗余的停词。

3.4.2　词性处理

一段文本所含单词中有许多词性，如动词、名词、形容词、副词等，由于控件描述文本的专业性很强，动词与名词具有更好的描述能力；相对地，形容词和副词的描述能力较差，比如动词 display 和 edit，名词 web 和 text 等都能很好地描述文档特点，而副词、形容词 complete 和 basic 等就无法很好地体现控件功能。为了更加严谨，我们进行了对照实验，一组实验只使用动词、名词作为关键词备选项（见表 4），另一组实验不仅使用动词、名词，还使用了副词、形容词作为备选项（见表 5）。由结果可知，仅使用动词名词的实验组的相似性矩阵的表现更好。考虑到一些动词过于常用，表达的专业性含义不足，例如 be，do 等，不将这样的单词加入关键词集合。对比表 4、表 5 可以明显发现，仅包含动词、名词作为备选项的实验组的对角线矩阵元数值更大，对应控件的相似值比例更高，更能定量表现实际对应控件间的相似性，这说明了我们的程序仅以名词、动词作为备选项是合理的。

表 4　考虑动词、名词、形容词

	iOS testview	iOS webview
Android testview	**0.38379274**	0.17961811
Android webview	0.31036624	**0.46286654**
相似值比例/%	67.9	**59.4**

表 5　仅考虑动词、名词

	iOS testview	iOS webview
Android testview	**0.3971458**	0.18459184
Android webview	0.3300541	**0.48884999**
相似值比例/%	68.3	**59.8**

3.4.3 词形处理

英文词汇中，一个单词存在多种形式，如第三人称单数、过去式、现在进行时等，而这些不同的形式表达的含义相同，因此，有必要对这一类单词进行合并，如将 taking，takes，take 设置为 3 个独立的维度是错误的。本实验中使用 stanford 的 corenlp 包[2] 对不同词形进行还原，特征向量的构造需要还原每个维度的词形，还原后的文档有利于进一步的使用。

3.4.4　词根提取

实验采用 PorterStemmer[3] 提供的词根表将待测文本词汇词根化，进一步压缩冗余维度，例如，accept，acceptance 词形原形并不相同，然而语义是相同的，其通过词根提取变形为相同的词根 accept 形式，有助于实验结果的进一步优化。需要注意的是，这一步处理是基于我们的主观经验，在实验中有较好性能。

3.4.5　同义词归并处理

在文档描述中，使用相似的同义词是常用的现象，但在自然语言处理中，如果不考虑同义词问题，空间向量中同义词不会提供任何相似度。对于这个问题，我们设立相似词集存储同义词，相似词集作为特征向量的一个维度，借此在空间向量模型中体现对同义词的处理优化。使用 wordnet[4] 提供的 api 来判定同义词，wordnet 是一个能够定量描述单词与单词之间相似性的工具，由于其仅支持同词型单词的比较，若其因此实验中将所有词汇分为动词、名词分别处理，给定一个阈值，将通过 wordnet 计算出来的相似数值与之比较，若其大于该阈值，则认为这两个单词是同义词，并将其归为一个词集。对词集的数目也需要仔细考虑，很明显，阈值和词集数目的确定决定了优化的效果，接下来主要阐述对阈值和词集大小的确定。

(1)阈值的设置决定了判定两个词为相似的难度，若阈值过大，一些意义相近的单词会被筛掉；若阈值过小，一些意义无关的单词会被判定为相似同义，只有合适地设置阈值大小，才能将应该被归为相似的词集归并在一起。本实验仅仅考察动词、名词的相关性，通过设置不同的阈值，观察多个中间文件生成的相似词集，判定出阈值在哪个范围内时得到的相似词集更接近人类语言的理解。最终，将阈值设为0.95。

(2)词集大小的设置既要保证不同词集的差异性，又要保证同一词集的元集具有较大差异。如果不考虑差异性，即允许大量单词归并到同一词集中，对于给定平台控件，其与对应平台的所有控件的相似度都会以上升，导致优化效果较差。例如，考虑极端情况，即不设置词集大小上限，只要 wordnet 给出的相似度大于阈值，便将这些单词全部放在一个词集里，有时这样的词集包含 4～5 个单词。在这种模式下，我们得出的实验结果并没有过多优化，主要是因为给太多单词提供了新的特征维度，从而失去了差异性。在考虑差异性的前提下再考虑同义性，设置单词上限为 N，经过程序逻辑后会得到一组不设上限的相似词集，对包含单词数目大于 N 的词集进行缩减，具体的操作是：对该词集的所有单词遍历包含 3 个单词的词组元，对于每组的 3 个单词，两两计算其 wordnet 相似度并求和，最后选取求和最大的 3 个单词作为合格词集，被淘汰的该词集的其他单词会再次作为文档备选单词进行该过程，直到所有词集的单词数目小于或等于 N，经过该过程，我们获得了最终的相似词集特征。实验发现，N 为 3 时实验结果最好。

此时，测得的控件相似正确率较大。由于同义词优化有一定的主观性质，因此再进行一组实验，与不加同义词优化的实验结果进行对比（见表 6 和表 7）。对于表 6 和表 7，明显可以看出优化后的实验数值（加粗部分）更加凸显，对应控件的相似值比例得到很大提高，其他部分受到抑制，说明本文的同义词优化是合理且正确的。

表 6　同义词优化

	iOS testview	iOS webview
Android testview	**0.43392390**	0.204379961
Android webview	0.34562945	**0.567523978**
相似值比例/%	69.8	**62.1**

表 7　无同义词优化

	iOS testview	iOS webview
Android testview	**0.39714587**	0.18459184
Android webview	0.33005415	**0.48884999**
相似值比例/%	68.3	**59.8**

图 2 给出同义词优化的程序伪代码，iOS_Kit 是 iOS 平台对应的控件，Android_Kit 是 Android 平台对应的控件。

Step1：第 2－3 行使用 NLP 模块提供的算法将 iOS，Android 控件文本提取成关键单词形式。NLP 是通过各种自然语言处理得出某一控件符合条件的待选关键词的函数过程的集合，输入 iOS 与 Android 控件文本，输出的 wordlist 为符合词形条件的关键单词集合。之后，使用 wordnet 提供的 api 计算两两关键词之间的相似值，使用 WordNetCompare 比较这一数值是否大于阈值 threshold，若大于则将之放入同一相似词集，经过去重处理，可得到相似词集的集合 SimilarWordList-Union。

Step2：第 4－13 行，对每一个在 SimilarWordListUnion 集合中的相似词集 SimilarWordList 进行优化，使得每一个包含多于 3 个词汇的词集进行合理拆分，最终形成全都包含小于或等于 3 个词汇的相似词词集。Wordnet. Lin（word1，word2）是 wordnet 提供的一个称为 Lin 的 api，参数为两个单词，输出为这两个单词的相似性数值；之后通过比较一组三元关键词的总相似值，选择总相似值最大的 3 个单词组成相似词集，即第 8 行中的使 temp 最大的单词三元组（word1，word2，word3）为最新的 SimilarWordList。对于留下的单词，即 11 行中的原 SimilarWordList 去除（word1，word2，word3）所形成的 leftword 集合，将其附加到 SimilarWordListUnion 中继续循环过程，直至所有单词数大于 3 的词集全部被分解。易知，这一过程之后所有的相似词集的大小都不大于 3。

```
For each (los kit, Android kit):
WordList NLP(los kit, Android kit);
SimilarWordListUnion=WordNetCompare(WordList, threshold);
Foreach(SimilarWordList in SimilarWordListUnion):
if(length(SimilarWordList) >3):
MaxTemp = 0
For each((word1,word2,word3) in SimilarWordList):
Temp = Lin(word1,word2) + Lin(word1,word3) + Lin(word2,word3);
MaxTemp = Temp > MaxTemp? Temp: MaxTemp;
(word1, word2,word3) = Word(MaxTemp);
LeftWord = SimilarWordList − (word1,word2,word3);
SimilarWordList = (word1,word2,word3);
SimilarWordListUnion. append(LeftWord);
```

图 2 同义词优化算法的伪代码示例

4 实验结果与分析

4.1 控件相似实例

选取较有代表性的 Android buttontextview 和 switch 作为实例展出，找到 iOS 对应的相似控件，列出具体的实验结果。

表 8 中，在给点安卓控件 button 的情况下，程序按照相似度给出了前 3 个 iOS 对应的相似控件，很明显，实际相似的控件是 iOS 的 button，而程序给出的 iOS button 的相似度远远大于第 2 名、第 3 名相似控件的相似性，因此 button 控件的程序效果很好。本文程序对于大部分控件的处理结果都同表 8 一样。

表 8 Android_button 相似控件

Andriod_buttons	相似度	相似与否
iOS_Buttons	0.559438	Yes
iOS_ActionSheets	0.355296	No
iOS_Switches	0.329253	No

表 9 中的结果较差，iOS_Switches 与 Android 对应的 iOS switch 排在相似度的第 4 名，无关的几个控件的相似性却很高。我们查看了控件文档，发现 Android switch 的文档非常冗长，一部分交代了 switch 本身通过 toggle 二选一的功能，另一部分举例描述了 switch 在使用上的一些实例，其使用了大量与 switch 并不直接相关的词汇，包括 display，text，control 等与其他 iOS 控件匹配度很高的词汇，这直接提高了类似 iOS label，iOS toolbars 等无关控件的相似性。对于这种情况，下一步的实验考虑引入专家系统，通过对控件功能特点的预定义排除一些无关文档词汇的影响。我们的程序针对很小一部分控件给出的相似结果不太理想，包括本身不存在相似控件的情况，如 Android checkbox 在 iOS 中没有 checkbox 与之对应。

表 9 Android_Switch 相似控件

Android_Switch	相似度	相似与否
iOS_Segmented Controls	0.674801	No
iOS_Labels	0.571240	No
iOS_Toolbars	0.401461	No
iOS_Switches	0.387777	Yes

表 10 列出了 Android textview 以及程序找到的相似 iOS 控件的情况，可以看到 iOS Text View 作为实际对应的相似组件所得的相似度是最高的，然而，iOS Table View 的相似度也很高，排在第二名，与 iOS textview 的相似度相差无几。这种情况下的控件在程序中也占一小部分，这种情况在引入专家系统后会得到较大的优化。

表 10 Android_Textview 相似控件

AndroidTextView	相似度	相似与否
iOS_Text Views	0.433923	Yes
iOS_Table Views	0.378212	No
iOS_Text Fields	0.320310	No
iOS_Switches	0.308201	No

4.2 所有 UI 组件的结果展示

表 11 列出了 30 组 AndroidUI 控件的实验结果，结果显示，81.8%的控件可以直接获得正确结果，9.1%的结果有较大的改进空间，还有 26.7%的控件无法直接找到另一平台对应的控件，需要引入专家系统实现多对多的对应，这将是我们进一步的工作内容。

总体来说，实验结果可以接受，正确率达到 81.8%，对应错误的控件组合也可以通过类似对应多候选控件的方式进行优化以便工业使用（比如将相似度排名第 1～3 的控件都推荐给使用者）。然而，实验中存在误差的现象还需要通过合适的优化进行改善。4.1 节列举了一些控件实验结果，一部分实验结果很好，这一部分不需要优化；一部分结果（比如真实对应控件相似度排在第二）可以通过一些自然语言层面的技术进行优化，需要注意的是有 8 个 Android 的 UI 控件找不到一对一的 iOS 对应控件，例如 Andriod_ratingbar，虽然没有直接的 iOS 控件与之对应，但是 ratingbar 的父类 progressbar 在 iOS 中有直接的 iOS 控件与之对应，这一部分的控件在下一步实验中通过合理的 MVC 构造可以实现对应。此外，还有一部分的 Android 控件的功能过于具体，在 iOS 中实现同样

的功能时需要调用多个控件，对于这部分控件，下一步实验将引入专家系统、文档结构解析等，以寻找这些组件多对多的对应状态。

表 11　相似控件结果

控件分类(以 Android 为查询项)	数目	比例/%
相似度第一且相似度数值优势明显	13	59.1
相似度第一但相似度数值优势不明显	5	22.7
相似度非第一但相似度劣势不明显	2	9.1
相似度非第一且相似度劣势明显	2	9.1
无对应控件	8	Null

5　相关工作

本文实验首先需要获取不同平台的控件文档，以此为基础找到不同平台间的控件对应性。文献[11]对于特定领域，在 Web 上挖掘产品描述信息，基于聚类与 k 近邻的机器学习算法给出符合用户查询条件的特征产品。我们的实验与之类似，从 Web 上有针对性地获取需要的控件文档信息，但是我们的程序基于自然语言处理获取文档相关性，并不需要聚类等机器学习算法，实现起来更加容易，需要的数据量较小，结果也很好。文献[8]提出一种挖掘 app store 应用评分与用户评论的方法，以此寻找移动应用各种信息之间的关系，得到了不错的结果。考虑到移动应用不同于 pc 端程序，用户在 Web 上会留下很多非代码的有用信息，本文采用类似的思想从 Web 上寻找能够描述控件对应性的有用信息，最终得到的有效正确率高达81.8%，这种基于挖掘 Web 相关非代码信息的方式有很好的发展前景。

当获取到可供使用的控件文档信息后，需要使用一些自然语言处理的技术对这些文档进行优化处理。我们的程序使用空间向量模型对控件文本进行相似性求值。一般情况下，空间向量模型以文档中出现的词项进行索引，文献[9]提出了一种用术语替代基于单纯词汇的处理方式。我们的程序与之类似，但并不使用单纯词汇，而是选取能够表征控件行为的名词和部分动词作为索引项，并提出了同义词以词集形式加入索引的优化方式。文献[7]提出了一种借助外部词典分析词项之间的语义相似度并构造词项相似度加权树，进而综合得出文本相似性的方法。本文实验中类似地借助了外部字典分析关键词相似度，然而我们构造了相似词词集并加入了关键词向量，以此体现词项相似度加权，所得结果得到了更好的优化。在我们的程序中，需要判断词汇与词汇之间的相似性，因此最终选择使用 wordnet 提供的词汇同义性评分。wordnet 是一种基于自然语义的词汇同义性理解工具，基于 Web 获取的控件文档是描述功能与行为的类自然语言，wordnet 对于这些控件文档中的词汇具有很好的同义识别能力。文献[12]肯定了 wordnet 对于自然语义层面同义词的处理能力，然而它对于具体程序代码 wordnet 并没有很好的处理效果；对于某种具体语言，如 Java 代码，需要构建特定的代码数据库来判定代码层次的语义相似性，我们在进一步的实验中将考虑代码层级。

文献[13-14]提出寻找两种语言之间 api 的对应关系，java 与 c# 这两种语言更多地用于传统 pc 端编程，api 层级的对应性有助于一般开发者更好地完成不同语言的开发任务。本文工作聚焦于移动开发，考虑到移动开发的特点是短平快并基于 UI 驱动的，快速地理解控件层级的对应性有助于一般开发者快速完成多平台开发任务，本文的目的即是得到不同平台下高正确率的控件对应关系。

结束语　现阶段我们的程序成功获得了不同平台间控件的对应关系。下一步的研究方向在于构造一个平台的基本模型，基于模型 MDE(Model Driving，Engine)研究的方式来抽取移动平台的 UI 控件模型，基于之前的研究，给出控件的特征，然后基于特征进行控件相似度的度量，并指导文档理解。

参 考 文 献

[1] lucence[OL]. https://lucene.apache.org.

[2] stanford corenlp[OL]. http://nlp.stanford.edu.

[3] PorterStemer[OL]. https://tartarus.org/martin/PorterStemmer.

[4] wordnet[OL]. http://wordnet.princeton.edu.

[5] ios_kit[OL]. https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/UIKitUICatalog/index.html.

[6] android_kit[OL]. http://www.android-doc.com/guide/topics/ui.

[7] HUANG C H，SUN Y S. A Text Similarity Measurement Combining Word Semantic Information with TF-IDF Method[J]. Chinese Journal of Computers，2011，34(5)：856-864.

[8] HARMAN M，JIA Y，ZHANG Y Y. App Store Mining and Analysis：MSR for App[C]// Working Conference on Mining Software Repositories. 2015：123-133.

[9] MILIOS E，ZHANG Y，HE B，et al. Automatic Term Extraction and Document Similarity in Special Text Corpora[C]// Proceedings of the sixth Conference of the Pacific Association for Computational Linguistics. 2003：275-284.

[10] LAKKARAJU P，GAUCH S，SPERETTA M. Document Similarity Based on Concept Tree Distance[C]// Nineteenth ACM Conference on Hypertext & Hypermedia. 2008：127-132.

[11] DUMITRU H，GIBIEC M，HARIRI N，et al. On-demand Feature Recommendations Derived from Mining Public Product Descriptions[J]. International Conference on Software Engineering，2011，111(2)：181-190.

[12] TIAN Y，LO D，LAWALL J. SEWordSim：Software-Specific Word Similarity Database[C]// ICSE Companion 2014 Companion Proceedings of the 36th International Conference on Software Engineering. 2014：568-571.

[13] ZHONG H，THUMMALAPENTA S，XIE T，et al. Mining API Mapping for Language[C]// Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering(ICSE'10). 2010：195-204.

[14] NGUYEN A T，NGUYEN H A，NGUYEN T T，et al. Statistical Learning Approach for Mining API Usage Mappings for Code Migration[C]// Proceedings of the 29th ACM/IEEE international conference on Automated software engineering(ASE'14). 2014：457-468.

[15] CHOW K，NOTKIN D. Semi-automatic update of applications in response to library changes. Software Maintenance[C]// International Conference on Date of Conference. 1996：359-368.