Screen2Vec: Semantic Embedding of GUI Screens and GUI Components

Screen2Vec:GUI 屏幕与组件的语义嵌入

Toby Jia-Jun Li*

Toby Jia-Jun Li*

tobyli@cs.cmu.edu
Carnegie Mellon University

卡内基梅隆大学

Pittsburgh, PA

匹兹堡，宾夕法尼亚

Lindsay Popowski*

Lindsay Popowski*

lpopowski@g.hmc.edu
Harvey Mudd College

哈维穆德学院

Claremont, CA

克莱蒙特，加利福尼亚

Tom M. Mitchell

Tom M. Mitchell

tom.mitchell@cs.cmu.edu
Carnegie Mellon University

卡内基梅隆大学

Pittsburgh, PA

匹兹堡，宾夕法尼亚

Brad A. Myers

Brad A. Myers

bam@cs.cmu.edu

Carnegie Mellon University

卡内基梅隆大学

Pittsburgh, PA

匹兹堡，宾夕法尼亚

ABSTRACT

摘要

Representing the semantics of GUI screens and components is crucial to data-driven computational methods for modeling user-GUI interactions and mining GUI designs. Existing GUI semantic representations are limited to encoding either the textual content, the visual design and layout patterns, or the app contexts. Many representation techniques also require significant manual data annotation efforts. This paper presents Screen2Vec, a new self-supervised technique for generating representations in embedding vectors of GUI screens and components that encode all of the above GUI features without requiring manual annotation using the context of user interaction traces. Screen2Vec is inspired by the word embedding method Word2Vec, but uses a new two-layer pipeline informed by the structure of GUIs and interaction traces and incorporates screen- and app-specific metadata. Through several sample downstream tasks, we demonstrate Screen2Vec's key useful properties: representing between-screen similarity through nearest neighbors, composability, and capability to represent user tasks.

表示 GUI 屏幕和组件的语义对于以数据为驱动的用户–GUI 交互建模与 GUI 设计挖掘方法至关重要。现有的 GUI 语义表示仅能编码文本内容、视觉设计与布局模式，或应用上下文中的某一类特征。许多表示技术还需要大量人工数据标注。本文提出 Screen2Vec，一种新的自监督技术，用于生成 GUI 屏幕和组件的嵌入向量表示，能在不需人工标注的情况下通过用户交互轨迹的上下文编码上述所有 GUI 特征。Screen2Vec 受词嵌入方法 Word2Vec 启发，但采用了基于 GUI 结构与交互轨迹的二层新型流水线，并融合了屏幕与应用特定的元数据。通过若干示例下游任务，我们展示了 Screen2Vec 的关键有用属性: 通过最近邻表示屏幕间相似性、可组合性，以及表征用户任务的能力。

CCS CONCEPTS

CCS 概念

- Human-centered computing → Smartphones; User interface design; Graphical user interfaces; - Computing methodologies → Neural networks.

- 以人为中心的计算 → 智能手机；用户界面设计；图形用户界面；- 计算方法学 → 神经网络。

KEYWORDS

关键词

GUI embedding, interaction mining, screen semantics

GUI 嵌入，交互挖掘，屏幕语义

ACM Reference Format:

ACM 参考格式:

Toby Jia-Jun Li, Lindsay Popowski, Tom M. Mitchell, and Brad A. Myers. 2021. Screen2Vec: Semantic Embedding of GUI Screens and GUI Components. In CHI Conference on Human Factors in Computing Systems (CHI '21), May 8-13, 2021, Yokohama, Japan. ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3411764.34450

Toby Jia-Jun Li, Lindsay Popowski, Tom M. Mitchell, and Brad A. Myers. 2021. Screen2Vec: Semantic Embedding of GUI Screens and GUI Components. In CHI Conference on Human Factors in Computing Systems (CHI '21), May 8-13, 2021, Yokohama, Japan. ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3411764.3445049

1 INTRODUCTION

1 引言

With the rise of data-driven computational methods for modeling user interactions with graphical user interfaces (GUIs), the GUI screens have become not only interfaces for human users to interact with the underlying computing services, but also valuable data sources that encode the underlying task flow, the supported user interactions, and the design patterns of the corresponding apps, which have proven useful for AI-powered applications. For example, programming-by-demonstration (PBD) intelligent agents such as [20, 25, 40] use task-relevant entities and hierarchical structures extracted from GUIs to parameterize, disambiguate, and handle errors in user-demonstrated task automation scripts. ERICA [10] mines a large repository of mobile app GUIs to enable user interface (UI) designers to search for example design patterns to inform their own design. Kite [26] extracts task flows from mobile app GUIs to bootstrap conversational agents.

随着用于建模人们与图形用户界面 (GUI) 交互的数据驱动计算方法的兴起，GUI 屏幕不仅是人类用户与底层计算服务交互的界面，还是编码底层任务流程、支持的用户交互和相应应用设计模式的宝贵数据源，这些已被证明对 AI 驱动的应用非常有用。例如，编程即示范 (PBD) 智能代理如 [20, 25, 40] 使用从 GUI 提取的与任务相关的实体和层次结构来参数化、消歧并处理用户示范的任务自动化脚本中的错误。ERICA [10] 挖掘大型移动应用 GUI 仓库，使用户界面 (UI) 设计师能够搜索示例设计模式以指导其设计。Kite [26] 从移动应用 GUI 提取任务流程以引导会话式代理。

Semantic representations of GUI screens and components, where each screen and component is encoded as a vector (known as the embedding), are highly useful in these applications. The representations of GUI

screens and components can be used to also represent other entities of interest. For example, a task in an app can be modeled as a sequence of GUI actions, where each action can be represented as a GUI screen, a type of interaction (e.g., click), and the component that is interacted with on the screen. An app can be modeled as a collection of all its screens, or a large collection of user interaction traces of using the app. Voice shortcuts in mobile app deep links [2] can be modeled as matching the user's intent expressed in natural language to the target GUI screens. The representation of the screen that the user is viewing or has previously viewed can also be used as the context to help infer the user's intents and activities in predictive intelligent interfaces. The semantic embedding approach represents GUI screens and components in a distributed form [4] (i.e., an item is represented across multiple dimensions) as continuous-valued vectors, making it especially suitable for use in popular machine learning models.

GUI 屏幕和组件的语义表示，即将每个屏幕和组件编码为向量 (称为嵌入)，在这些应用中非常有用。GUI 屏幕和组件的表示也可以用于表示其他感兴趣的实体。例如，应用中的一个任务可建模为 GUI 操作序列，其中每个操作可表示为一个 GUI 屏幕、一种交互类型 (例如点击) 以及在该屏幕上被交互的组件。一个应用可建模为其所有屏幕的集合，或是大量使用该应用的用户交互轨迹集合。移动应用深度链接中的语音快捷方式 [2] 可建模为将用户以自然语言表达的意图与目标 GUI 屏幕匹配。用户正在查看或先前查看的屏幕表示也可作为上下文，用以帮助在预测智能界面中推断用户的意图和活动。语义嵌入方法以分布式形式 [4](即一个条目在多个维度上表示) 将 GUI 屏幕和组件表示为连续值向量，使其特别适合用于流行的机器学习模型。

However, existing approaches of representing GUI screens and components are limited. One type of approach solely focuses on capturing the text on the screen, treating the screen as a bag of words or phrases. For example, Sugilite [20] uses exact matches of text labels on the screen to generalize the user demonstrated tasks. SOVITE [22] uses the average of individual word embedding vectors for all the text labels on the screen to represent the screen for retrieving relevant task intents. This approach can capture the semantics of the screen's textual content, but misses out on using the information encoded in the layout and the design pattern of the screen and the task context encoded in the interactivity and meta-data of the screen components.

然而, 现有的 GUI 屏幕和组件表示方法存在局限。一类方法仅关注捕捉屏幕上的文本, 将屏幕视为词袋或短语集合。例如, Sugilite [20] 使用屏幕中文本标签的精确匹配来泛化用户示范的任务。SOVITE [22] 使用屏幕上所有文本标签的各个词嵌入向量的平均值来表示屏幕以检索相关任务意图。这种方法能捕捉屏幕文本内容的语义, 但忽略了布局和屏幕的设计模式以及编码在交互性和屏幕组件元数据中的任务语境所包含的信息。

---

*Both authors contributed equally.

* 两位作者贡献相同。

Another type of approach focuses on the visual design patterns and GUI layouts. ERICA [10] uses an unsupervised clustering method to create semantic clusters of visually similar GUI components. Liu et al.'s approach [30] leverages the hierarchical GUI structures, the class names of GUI components, and the visual classifications of graphical icons to annotate the design semantics of GUIs. This type of approach has been shown to be able to determine the category of a GUI component (e.g., list items, tab labels, navigation buttons),

the ”UX concept” semantics of buttons (e.g., ”back”, ”delete”, ”save”, and ”share”), and the overall type of task flow of screens (e.g., ”searching”, ”promoting”, and ”onboarding”). However, it does not capture the content in the GUIs-two structurally and visually similar screens with different content (e.g., the search results screen in a restaurant app and a hotel booking app) will yield similar results.

另一类方法侧重于视觉设计模式和 GUI 布局。ERICA [10] 使用无监督聚类方法创建视觉上相似的 GUI 组件的语义簇。Liu 等人的方法 [30] 利用层次化的 GUI 结构、GUI 组件的类名以及图形图标的视觉分类来注释 GUI 的设计语义。这类方法已被证明能够确定 GUI 组件的类别 (例如列表项、选项卡标签、导航按钮)、按钮的“用户体验概念”语义 (例如“返回”、“删除”、“保存”和“分享”) 以及屏幕的整体任务流程类型 (例如“搜索”、“促销”和“引导”)。然而，它未能捕捉 GUI 中的内容——两个结构和视觉上相似但内容不同的屏幕 (例如餐厅应用和酒店预订应用中的搜索结果屏幕) 会产生相似的结果。

There have been prior approaches that combine the textual content and the visual design patterns [28, 36]. However, these approaches use supervised learning with large datasets for very specific task objectives. Therefore they require significant task-specific manual data labeling efforts, and their resulting models cannot be used in different downstream tasks. For example, Pasupat et al. [36] create a embedding-based model that can map the user’s natural language commands to web GUI elements based on the text content, attributes, and spatial context of the GUI elements. Li et al.’s work [28] describes a model that predicts sequences of mobile GUI action sequences based on step-by-step natural language descriptions of actions. Both models are trained using large manually-annotated corpora of natural language utterances and the corresponding GUI actions.

此前已有将文本内容与视觉设计模式结合的方法 [28, 36]。然而，这些方法对非常具体的任务目标采用有监督学习并使用大型数据集，因此需要大量任务特定的人工标注工作，且其所得模型无法用于不同的下游任务。例如，Pasupat 等人 [36] 构建了一个基于嵌入的模型，能够根据 GUI 元素的文本内容、属性和空间上下文将用户的自然语言指令映射到网页 GUI 元素。Li 等人 [28] 的工作描述了一个根据逐步自然语言动作描述预测移动 GUI 操作序列的模型。这两种模型都使用大量人工注释的自然语言话语及相应 GUI 操作的语料进行训练。

We present a new self-supervised technique (i.e., the type of machine learning approach that trains a model without human-labeled data by withholding some part of the data, and tasking the network with predicting it) Screen2Vec for generating more comprehensive semantic representations of GUI screens and components. Screen2Vec uses the screens’ textual content, visual design and layout patterns, and app context meta-data. Screen2Vec’s approach is inspired by the popular word embedding method Word2Vec [32], where the embedding vector representations of GUI screens and components are generated through the process of training a prediction model. However, unlike Word2Vec, Screen2Vec uses a two-layer pipeline informed by the structures of GUIs and GUI interaction traces and incorporates screen- and app-specific metadata.

我们提出了一种新的自监督技术 (即通过隐藏部分数据并让网络预测它来训练模型、无需人工标注的数据驱动方法)Screen2Vec，用于生成更全面的 GUI 屏幕及组件语义表示。Screen2Vec 利用屏幕的文本内容、视觉设计与布局模式以及应用上下文元数据。Screen2Vec 的方法受流行词嵌入方法 Word2Vec [32] 启发，通过训练预测模型来生成 GUI 屏幕与组件的嵌入向量表示。然而，不同于 Word2Vec，Screen2Vec 使用了受 GUI 结构和 GUI 交互轨迹启发的两层管道，并融合了屏幕与应用特定的元数据。

The embedding vector representations produced by Screen2Vec can be used in a variety of useful downstream tasks such as nearest neighbor retrieval, composability-based retrieval, and representing mobile tasks. The self-supervised nature of Screen2Vec allows its model to be trained without any manual data labeling efforts-it can be trained with a large collection of GUI screens and the user interaction traces on these screens such as the RICO [9] dataset.

Screen2Vec 产生的嵌入向量表示可用于多种有用的下游任务，例如最近邻检索、基于可组合性的检索以及表示移动任务。Screen2Vec 的自监督特性使其模型可以在无需任何人工标注的情况下训练——它可以使用大量 GUI 屏幕集合和这些屏幕上的用户交互轨迹 (例如 RICO [9] 数据集) 进行训练。

Along with this paper, we also release the open-source [1] code of Screen2Vec as well as a pre-computed Screen2Vec model trained on the RICO dataset [9] (more in Section 2.1). The pre-computed model can encode the GUI screens of Android apps into embedding vectors off-the-shelf. The open-source code can be used to train models for other platforms given the appropriate dataset of user interaction traces.

随本文一起, 我们还发布了 Screen2Vec 的开源 [1] 代码以及在 RICO 数据集 [9] 上预训练的 Screen2Vec 模型 (详见第 2.1 节)。该预训练模型可开箱即用地将 Android 应用的 GUI 屏幕编码为嵌入向量。开源代码可用于在获得相应用户交互轨迹数据集的情况下为其他平台训练模型。

Screen2Vec addresses an important gap in prior work about computational HCI research. The lack of comprehensive semantic representations of GUI screens and components has been identified as a major limitation in prior work in GUI-based interactive task learning (e.g., [25, 40]), intelligent suggestive interfaces (e.g., [7]), assistive tools (e.g., [5]), and GUI design aids (e.g., [17, 41]). Screen2Vec embeddings can encode the semantics, contexts, layouts, and patterns of GUIs, providing representations of these types of information in a form that can be easily and effectively incorporated into popular modern machine learning models.

Screen2Vec 弥补了以往计算型人机交互研究中的重要空白。缺乏对 GUI 屏幕与组件的全面语义表示已被认为是以往 GUI 驱动交互任务学习 (例如 [25, 40])、智能建议界面 (例如 [7])、辅助工具 (例如 [5]) 和 GUI 设计辅助 (例如 [17, 41]) 等工作的主要限制。Screen2Vec 嵌入能够编码 GUI 的语义、上下文、布局和模式，以一种可被现代机器学习模型轻松且有效整合的形式表示这些信息。

This paper makes the following contributions:

本文做出如下贡献:

(1) Screen2Vec: a new self-supervised technique for generating more comprehensive semantic embeddings of GUI screens and components using their textual content, visual design and layout patterns, and app meta-data.

(1) Screen2Vec: 一种新的自监督技术，利用文本内容、视觉设计与布局模式及应用元数据生成更全面的 GUI 屏幕与组件语义嵌入。

(2) An open-sourced GUI embedding model trained using the Screen2Vec technique on the RICO [9] dataset that can be used off-the-shelf.

(2) 一个基于 Screen2Vec 技术并在 RICO [9] 数据集上训练的开源 GUI 嵌入模型，可开箱即用。

(3) Several sample downstream tasks that showcase the model's usefulness.

(3) 若干示例下游任务，展示该模型的实用性。

2 OUR APPROACH

2 我们的方法

Figure 1 illustrates the architecture of Screen2Vec. Overall, the pipeline of Screen2Vec consists of two levels: the GUI component level (shown in the gray shade) and the GUI screen level. We will first describe the approach at a high-level here, and then explain the details in Section 2.2.

图 1 展示了 Screen2Vec 的架构。总体上，Screen2Vec 的管道由两层组成:GUI 组件层 (灰色阴影所示) 和 GUI 屏幕层。我们将先在此给出高层次的描述，然后在第 2.2 节中解释详细内容。

The GUI component level model encodes the textual content and the class type of a GUI component into a 768-dimensional [2] embedding vector to represent the GUI component (e.g., a button, a textbox, a list entry etc.). This GUI component embedding vector is computed with two inputs: (1) a 768-dimensional embedding vector of the text label of the GUI component, encoded using a pre-trained Sentence-BERT [39] model; and (2) a 6-dimensional class embedding vector that represents the class type of the GUI component, which we will discuss in detail later in Section 2.2. The two embedding vectors are combined using a linear layer, resulting in the 768-dimensional GUI component embedding vector that represents the GUI component. The class embeddings in the class type embedder and the weights in the linear layer are optimized through training a Continuous Bag-of-Words (CBOW) prediction task: for each GUI component on each screen, the task predicts the current GUI component using its context (i.e., all the other GUI components on the same screen). The training process optimizes the weights in the class embeddings and the weights in the linear layer for combining the text embedding and the class embedding.

GUI 组件级模型将 GUI 组件的文本内容和类类型编码为一个 768 维的 [2] 嵌入向量来表示该 GUI 组件 (例如按钮、文本框、列表条目等)。该 GUI 组件嵌入向量由两个输入计算得到:(1) 使用预训练的 Sentence-BERT [39] 模型编码的 GUI 组件文本标签的 768 维嵌入向量；(2) 表示 GUI 组件类类型的 6 维类嵌入向量，关于类嵌入我们将在第 2.2 节详述。这两个嵌入向量通过线性层组合，产生表示该 GUI 组件的 768 维嵌入向量。类类型嵌入器中的类嵌入以及线性层的权重通过训练一个连续词袋 (CBOW) 预测任务来优化: 对于每个屏幕上的每个 GUI 组件，该任务使用其上下文 (即同一屏幕上的所有其它 GUI 组件) 来预测当前 GUI 组件。训练过程优化类嵌入的权重以及用于组合文本嵌入和类嵌入的线性层权重。

---

[1] A pre-trained model and the Screen2Vec source code are available at: https://github.com/tobyli/screen2vec

[1] 一个预训练模型和 Screen2Vec 源代码可在以下地址获取: https://github.com/tobyli/screen2vec

[2] We decided to produce 768-dimensional vectors so that they can be directly used with the 768-dimensional vectors produced by the pre-trained Sentence-BERT model with its default settings [39]

[2] 我们决定生成 768 维向量，以便它们可以直接与预训练 Sentence-BERT 模型在默认设置下生成的 768 维向量兼容 [39]

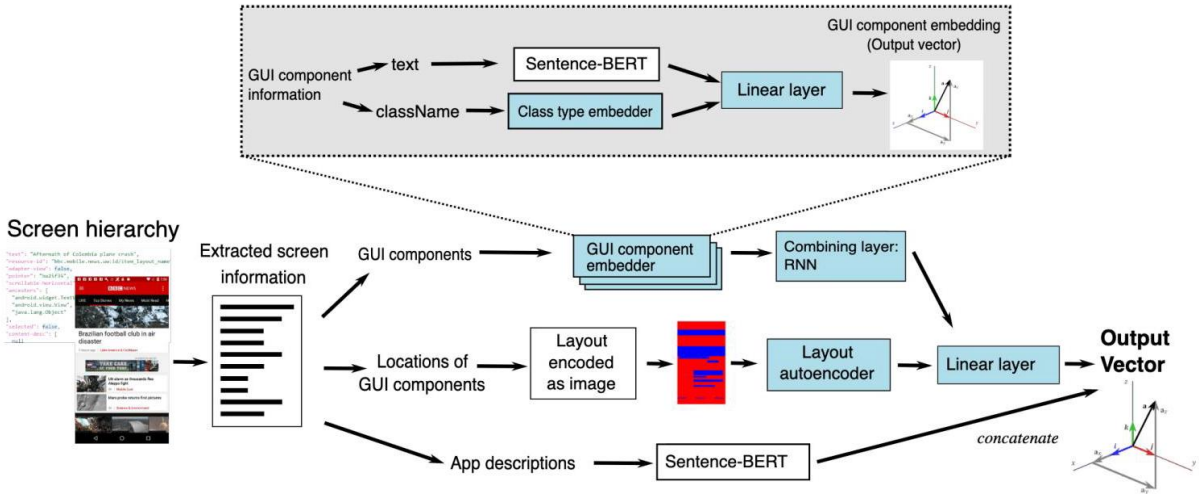

Figure 1: The two-level architecture of Screen2Vec for generating GUI component and screen embeddings. The weights for the steps in teal color are optimized during the training process.

图 1:Screen2Vec 用于生成 GUI 组件和屏幕嵌入的两级架构。以青绿色显示的步骤的权重在训练过程中被优化。

The GUI screen level model encodes the textual content, visual design and layout patterns, and app context of a GUI screen into an 1536-dimensional embedding vector. This GUI screen embedding vector is computed using three inputs: (1) the collection of the GUI component embedding vectors for all the GUI components on the screen (as described in the last paragraph), combined into a 768- dimension vector using a recurrent neural network model (RNN), which we will discuss more in Section 2.2; (2) a 64-dimensional layout embedding vector that encodes the screen's visual layout (details later in Section 2.2); and (3) a 768-dimensional embedding vector of the textual App Store description for the underlying app, encoded with a pre-trained Sentence-BERT [39] model. These GUI and layout vectors are combined using a linear layer, resulting in a 768-dimensional vector. After training, the description embedding vector is concatenated on, resulting in the 1536-dimensional GUI screen embedding vector (if included in the training, the description dominates the entire embedding, overshadowing information specific to that screen within the app). The weights in the RNN layer for combining GUI component embeddings and the weights in the linear layer for producing the final output vector are similarly trained on a CBOW prediction task on a large number of interaction traces (each represented as a sequence of screens). For each trace, a sliding window moves over the sequence of screens. The model tries to use the representation of the context (the surrounding screens) to predict the screen in the middle. See Section 2.2 for more details.

GUI 屏幕级模型将 GUI 屏幕的文本内容、视觉设计与布局模式以及应用上下文编码为一个 1536 维嵌入向量。该 GUI 屏幕嵌入向量由三个输入计算得到:(1) 屏幕上所有 GUI 组件的 GUI 组件嵌入向量集合 (如上段所述)，使用循环神经网络 (RNN) 组合为一个 768 维向量，关于此将在第 2.2 节详述；(2) 一个 64 维的布局嵌入向量，用于编码屏幕的视觉布局 (详见第 2.2 节)；(3) 使用预训练 Sentence-BERT [39] 模型编码的、表示底层应用的 App Store 描述的 768 维文本嵌入向量。这些 GUI 和布局向量通过线性层组合，产出一个 768 维向量。训练后，将描述嵌入向量拼接上去，得到 1536 维的 GUI 屏幕嵌入向量 (如果在训练中包含描述，描述会主导整个嵌入，掩盖同一应用中该屏幕特有的信息)。用于组合 GUI 组件嵌入的 RNN 层权重以及用于生成最终输出向量的线性层权重，同样通过在大量交互轨迹 (每条轨迹由一系列屏幕表示) 上的 CBOW 预测任务进行训练。对于每条轨迹，滑动窗口在屏幕序列上移动，模型尝试使用上下文 (周围屏幕) 的表示来预测中间的屏幕。更多细节见第 2.2 节。

However, unlike the GUI component level embedding model, the GUI screen level model is trained on a screen prediction task in the user interaction traces of using the apps. Within each trace, the training task tries to predict the current screen using other screens in the same trace.

然而，与 GUI 组件级嵌入模型不同，GUI 屏幕级模型在应用使用的用户交互轨迹上的屏幕预测任务上进行训练。在每条轨迹内，训练任务尝试使用同一轨迹中的其它屏幕来预测当前屏幕。

2.1 Dataset

2.1 数据集

We trained Screen2Vec on the open-sourced RICO[3] dataset [9]. The RICO dataset contains interaction traces on 66,261 unique GUI screens from 9,384 free Android apps collected using a hybrid crowd-sourcing plus automated discovery approach. For each GUI screen, the RICO dataset includes a screenshot image (that we did not use in Screen2Vec), and the screen's "view hierarchy" in a JSON file. The view hierarchy is structurally similar to a DOM tree in HTML; it starts with a root view, and contains all its descents in a tree. The node for each view includes the class type of this GUI component, its textual content (if any), its location as the bounding box on the screen, and various other properties such as whether it is clickable, focused, or scrollable, etc. Each interaction trace is represented as a sequence of GUI screens, as well as information about which (x, y) screen location was clicked or swiped on to transit from the previous screen to the current screen.

我们在开源的 RICO[3] 数据集 [9] 上训练了 Screen2Vec。RICO 数据集包含通过混合众包与自动化发现方法收集的来自 9,384 个免费 Android 应用的 66,261 个独特 GUI 屏幕的交互轨迹。对于每个 GUI 屏幕，RICO 数据集包括一个屏幕截图 (在 Screen2Vec 中我们未使用) 和以 JSON 文件形式的屏幕"视图层次结构"。视图层次结构在结构上类似于 HTML 的 DOM 树；它以根视图开始，并包含其所有子孙节点。每个视图节点包括该 GUI 组件的类类型、其文本内容 (若存在)、在屏幕上的边界框位置，以及是否可点击、是否聚焦或是否可滚动等各种属性。每条交互轨迹表示为一系列 GUI 屏幕，以及关于从上一个屏幕到当前屏幕时点击或滑动的 (x, y) 屏幕位置的信息。

2.2 Models

2.2 模型

This section explains the implementation details of each key step in the pipeline shown in Figure 1.

本节解释图 1 中管道每个关键步骤的实现细节。

GUI Class Type Embeddings. To represent the class types of GUI components, we trained a class embedder to encode the class types into the vector space. We used a total of 26 class categories: the 22 categories that were present in [30], one layout category, list and drawer categories, and an ”Other” category. We classified the GUI component classes based on the classes of their className properties and, sometimes, other simple heuristic rules (see Table 1). For example, if a GUI component is an instance of EditText (i.e., its className property is either EditText, or a class that inherits EditText), then it is classified as an Input. There are two exceptions: the Drawer and the List Item categories look at the className of the parent of the current GUI component instead of the className of itself. A standard PyTorch embedder (torch.nn. Embedding [4]) maps each of these 26 discrete categories into a continuous 6-dimensional vector. The embedding vector value for each category is optimized during the training process for the GUI component prediction tasks so that GUI components categories that are semantically similar to each other are closer together in the vector space.

GUI 类别类型嵌入。为表示 GUI 组件的类别类型，我们训练了一个类别嵌入器，将类别类型编码到向量空间中。我们使用了共计 26 个类别: 文献 [30] 中出现的 22 个类别、一个布局类别、列表和抽屉类别，以及一个“Other”类别。我们根据组件的 className 属性的类 (有时结合其它简单的启发式规则，见表 1) 对 GUI 组件类进行分类。例如，如果某 GUI 组件是 EditText 的实例 (即其 className 属性为 EditText 或继承自 EditText 的类)，则将其归为 Input。有两个例外:Drawer 和 List Item 类别查看的是当前组件父节点的 className，而不是其自身的 className。一个标准的 PyTorch 嵌入器 (torch.nn.Embedding [4]) 将这 26 个离散类别中的每一个映射到一个连续的 6 维向量。每个类别的嵌入向量值在用于 GUI 组件预测任务的训练过程中被优化，使得语义相近的 GUI 组件类别在向量空间中更接近。

---

[3] Available at: http://interactionmining.org/rico

[3] 可在:http://interactionmining.org/rico

GUI Component Context. As discussed earlier, Screen2Vec uses a Continuous Bag-of-Words (CBOW) prediction task [32] for training the weights in the model, where for each GUI component, the model tries to predict it using its context. In Screen2Vec, we define the context of a GUI component as its 16 nearest components. The size 16 is chosen to balance the model performance and the computational cost.

GUI 组件上下文。如前所述，Screen2Vec 使用连续词袋 (CBOW) 预测任务 [32] 来训练模型权重，对于每个 GUI 组件，模型尝试用其上下文来预测它。在 Screen2Vec 中，我们将一个 GUI 组件的上下文定义为其 16 个最近的组件。大小 16 的选择旨在平衡模型性能与计算成本。

Inspired by prior work on the correlation between the semantic relatedness of entities and the spatial distance between them [27]. We tried using two different measures of screen distance for determining GUI component context in our model: EUCLIDEAN, which is the straight-line minimal distance on the screen (measured in pixels) between the bounding boxes of the two GUI components; and HIERARCHICAL, which is the distance between the two GUI components on the hierarchical GUI view tree. For example, a GUI component has a distance of 1 to its parent and children and a distance of 2 to its direct siblings.

受先前关于实体语义相关性与其空间距离之间相关性的工作启发 [27]。我们尝试了两种不同的屏幕距离度量来确定模型中的 GUI 组件上下文:EUCLIDEAN，即屏幕上两个 GUI 组件边界框之间的直线最小距离 (以像素为单位)；以及 HIERARCHICAL，即在层次化 GUI 视图树中两个 GUI 组件之间的距离。例如，一个组件到其父节点和子节点的距离为 1，到其直接兄弟节点的距离为 2。

Linear Layers. At the end of each of the two levels in the pipeline, a linear layer is used to combine multiple vectors and shrink the combined vector into a lower-dimension vector that contains the relevant semantic content of each input. For example, in the GUI component embedding process, the model first concatenates the 768-dimensional text embedding with the 6-dimensional class embedding. The linear layer then shrinks the GUI component embedding back down to 768 dimensions. The linear layer works by creating 774×768 weights: one per pair of input dimension and output dimension. These weights are optimized along with other parameters during the training process, so as to minimize the overall total loss (loss function detail in Section 2.3).

线性层。在流水线的每个层级末端，使用线性层来组合多个向量并将组合后的向量压缩为包含相关语义信息的低维向量。例如，在 GUI 组件嵌入过程中，模型首先将 768 维的文本嵌入与 6 维的类别嵌入拼接。线性层随后将 GUI 组件嵌入缩回到 768 维。线性层通过创建 774×768 权重来工作: 针对每一对输入维度与输出维度各有一个权重。这些权重与其它参数一同在训练过程中被优化，以最小化整体总损失 (损失函数细节见第 2.3 节)。

In the screen embedding process, a linear layer is similarly used for combining the 768-dimensional layout embedding vector with the 64-dimensional GUI content embedding vector to produce a new 768-dimensional embedding vector that encodes both the screen content and the screen layout.

在屏幕嵌入过程中，线性层同样用于将 768 维的布局嵌入向量与 64 维的 GUI 内容嵌入向量结合，生成一个新的 768 维嵌入向量，该向量同时编码屏幕内容与屏幕布局。

Text Embeddings. We use a pre-trained Sentence-BERT language model [39] to encode the text labels on each GUI component and the Google Play store description for each app into 768-dimensional embedding vectors. This Sentence-BERT model, which is a modified BERT network [11], was pre-trained on the SNLI [6] dataset and the Multi-Genre NLI [43] dataset with a mean-pooling strategy, as described in [39]. This pre-trained model has been shown to perform well in deriving semantically meaningful sentence and phrase embeddings where semantically similar sentences and phrases are close to each other in the vector space [39].

文本嵌入。我们使用预训练的 Sentence-BERT 语言模型 [39] 将每个 GUI 组件上的文本标签以及每个应用在 Google Play 商店的描述编码为 768 维嵌入向量。该 Sentence-BERT 模型是一个经修改的 BERT 网络 [11]，按文献 [39] 所述采用均值池化策略在 SNLI [6] 数据集和 Multi-Genre NLI [43] 数据集上预训练。已证明该预训练模型在生成语义上有意义的句子和短语嵌入方面表现良好，使语义相近的句子与短语在向量空间中靠得更近 [39]。

Layout Embeddings. Another important step in the pipeline is to encode the visual layout pattern of each screen. We use the layout embedding technique from [9], where we first extract the layout of a screen from its screenshot using the bounding boxes of all the leaf GUI components in the hierarchical GUI tree, differentiating between text and non-text GUI components using different colors (Figure 2). This layout image represents the layout of the GUI screen while abstracting away its content and visual specifics. We then use an image autoencoder to encode each image into a 64-dimensional embedding vector. The autoencoder is trained using a typical encoder-decoder architecture, that is, the weights of the network are optimized to produce the 64-dimensional vector from the original input image that can produce the best reconstructed image when decoded.

布局嵌入。流水线中的另一个重要步骤是对每个屏幕的视觉布局模式进行编码。我们使用来自文献 [9] 的布局嵌入技术，首先从屏幕截图中提取屏幕布局，使用层次化 GUI 树中所有叶子 GUI 组件的边界框并用不同颜色区分文本与非文本 GUI 组件 (见图 2)。该布局图像在抽象掉内容和视觉细节的同时表示了 GUI 屏幕的布局。然后我们使用图像自编码器将每张图像编码为 64 维嵌入向量。该自编码器采用典型的编码器-解码器架构训练，即网络的权重被优化，使从原始输入图像生成的 64 维向量在解码后能产出最佳重构图像。

The encoder has input dimension of 11,200, and then two hidden layers of size 2,048 and 256, with output dimension of size 64; this means three linear layers of sizes $11,200 \rightarrow 2,048, 2,048 \rightarrow 256$, and $256 \rightarrow 64$. These layers have the Rectified Linear Unit (ReLU) [34] applied, so the output of each linear layer is put through an activation function which transforms any negative input to 0 . The decoder has the reverse architecture (three linear layers with ReLU $64 \rightarrow 256, 256 \rightarrow 2,048$, and $2,048 \rightarrow 11,200$) . The layout autoencoder is trained on the process of reconstructing the input image when it is run through the encoder and the decoder; the loss is determined by the mean squared error (MSE) between the input of the encoder and the output of the decoder.

编码器的输入维度为 11,200，随后有两个隐藏层，大小为 2,048 和 256，输出维度为 64；这意味着有三层线性层，大小为 $11,200 \rightarrow 2,048, 2,048 \rightarrow 256$, and $256 \rightarrow 64$。对这些层应用了线性整流单元 (ReLU)[34]，因此每个线性层的输出都经过激活函数，将任何负值变为 0。解码器的架构相反 (三个带 ReLU 的线性层 $64 \rightarrow 256, 256 \rightarrow 2,048$, and $2,048 \rightarrow 11,200$) 。布局自编码器通过将输入图像分别经编码器和解码器重建的过程进行训练；损失由编码器输入与解码器输出之间的均方误差 (MSE) 决定。

GUI Embedding Combining Layer. To combine the embedding vectors of multiple GUI components on a screen into a single fixed-length embedding vector, we use an Recurrent Neural Network (RNN): The RNN operates similarly to the linear layer mentioned earlier, except it deals with sequential data (thus the ”recurrent” in the name). The RNN we used was a sequence of linear layers with the additional input of a hidden state. The GUI component embeddings are fed into the RNN in the pre-order traversal order of the GUI hierarchy tree. For the first input of GUI component embedding, the hidden state was all zeros, but for

the second input, the output from the first serves as the hidden state, and so on, so that the n^{th} input is fed into a linear layer along with $(n-1)^{th}$ output. The overall output is the output for the final GUI component in the sequence, which encodes parts of all of the GUI components, since the hidden states could pass on that information. This allows screens with different numbers of GUI components to have vector representations that both take all GUI components into account and are of the same size. This RNN is trained along with all other parameters in the screen embedding model, optimizing for the loss function (detail in Section 2.3) in the GUI screen prediction task.

GUI 嵌入合并层。为将屏幕上多个 GUI 组件的嵌入向量合并为单个定长嵌入向量，我们使用循环神经网络 (RNN):RNN 的工作方式类似于前述线性层，但它处理的是序列数据 (因此称为“循环”)。我们使用的 RNN 是一系列带有隐藏状态输入的线性层。GUI 组件嵌入按照 GUI 层次树的先序遍历顺序输入 RNN。对于第一个组件嵌入，隐藏状态全为零；对于第二个输入，第一个的输出作为隐藏状态，依此类推，因此 n^{th} 输入与 $(n-1)^{th}$ 输出一起被送入线性层。总体输出为序列中最后一个 GUI 组件的输出，它编码了所有 GUI 组件的部分信息，因为隐藏状态可以传递这些信息。这样，不同数量 GUI 组件的屏幕也能得到既考虑所有组件又具有相同尺寸的向量表示。该 RNN 与屏幕嵌入模型中的所有其他参数一同训练，在 GUI 屏幕预测任务中针对损失函数进行优化 (细节见第 2.3 节)。

2.3 Training Configurations

2.3 训练配置

In the training process, we use 90% of the data for training and save the other 10% for validation. The models are trained on a cross entropy loss function with an Adam optimizer [15], which is an adaptive learning gradient-based optimization algorithm of stochastic objective functions. For both stages, we use an initial learning rate of 0.001 and a batch size of 256.

在训练过程中，我们使用 90% 的数据进行训练，另 10% 用于验证。模型在交叉熵损失下用 Adam 优化器 [15] 训练，Adam 是一种用于随机目标函数的自适应梯度优化算法。两个阶段均使用初始学习率 0.001 和批量大小 256。

---

[4] https://pytorch.org/docs/stable/generated/torch.nn.Embedding.html

[4] https://pytorch.org/docs/stable/generated/torch.nn.Embedding.html

---

GUI Component	Associated Class Type	GUI Component	Associated Class Type
Advertisement	AdView, HtmlBannerWebView, AdContainer	Layouts	LinearLayout, AppBarLayout, FrameLayout, RelativeLayout, TableLayout
Bottom Navigation	BottomTabGroupView, BottomBar	Button Bar	ButtonBar
Card	CardView	CheckBox	CheckBox, CheckedTextView
Drawer (Parent)	DrawyerLayout	Date Picker	DatePicker
Image	ImageView	Image Button	ImageButton, GlyphView, AppCompatButton, AppCompatImageButton, ActionMenuItemView, ActionMenuItemPresenter
Input	EditText, SearchBoxView, AppCompatAutoCompleteTextView, TextView [a]	List Item (Parent)	ListView, RecyclerView, ListPopupWindow, TabItem, GridView
Map View	MapView	Multi-Tab	SlidingTab
Number Stepper	NumberPicker	On/Off Switch	Switch
Pager Indicator	ViewPagerIndicatorDots, PageIndicator, CircileIndicator, PagerIndicator	RadioButton	RadioButton, CheckedTextView
Slider	SeekBar	TextButton	Button [b] , TextView [c]
Tool Bar	ToolBar, TitleBar, ActionBar	Video	VideoView
Web View	WebView	Drawer Item	Others category and ancestor is Drawer (Parent)
List Item	Others category and ancestor is List(Parent)	Others	...

GUI 组件	关联类类型	GUI 组件	关联类类型
广告	AdView, HtmlBannerWebView, AdContainer	布局	LinearLayout, AppBarLayout, FrameLayout, RelativeLayout, TableLayout
底部导航	BottomTabGroupView, BottomBar	按钮栏	ButtonBar
卡片	CardView	复选框	CheckBox, CheckedTextView
抽屉 (父级)	DrawyerLayout	日期选择器	DatePicker
图片	ImageView	图像按钮	ImageButton, GlyphView, AppCompatButton, AppCompatImageButton, ActionMenuItemView, ActionMenuItemPresenter
输入	EditText, SearchBoxView, AppCompatAutoCompleteTextView, TextView [a]	列表项 (父级)	ListView, RecyclerView, ListPopupWindow, TabItem, GridView
地图视图	MapView	多标签	SlidingTab
数值微调器	NumberPicker	开关	Switch
分页指示器	ViewPagerIndicatorDots, PageIndicator, CircileIndicator, PagerIndicator	单选按钮	RadioButton, CheckedTextView
滑块	SeekBar	文本按钮	Button [b], TextView [c]
工具栏	ToolBar, TitleBar, ActionBar	视频	VideoView
网页视图	WebView	抽屉项	其他类别，且祖先为 Drawer(父级)
列表项	其他类别，且祖先为 List(父级)	其他	...

[a] The property editable needs to be TRUE.

[a] 属性 editable 需要为 TRUE。

[b] The GUI component needs to have a non-empty text property.

[b] GUI 组件需要具有非空的 text 属性。

[c] The property clickable needs to be TRUE.

[c] 属性 clickable 需要为 TRUE。

Table 1: The 26 categories (including the ”Others” category) of GUI class types we used in Screen2Vec and their associated base class names. Some categories have additional heuristics, as shown in the notes. This categorization is adapted from [30].

表 1: 我们在 Screen2Vec 中使用的 26 类 GUI 类类型 (包括“其他”类) 及其关联的基类名。某些类别具有附加的启发式规则，见注释。本分类方法改编自 [30]。

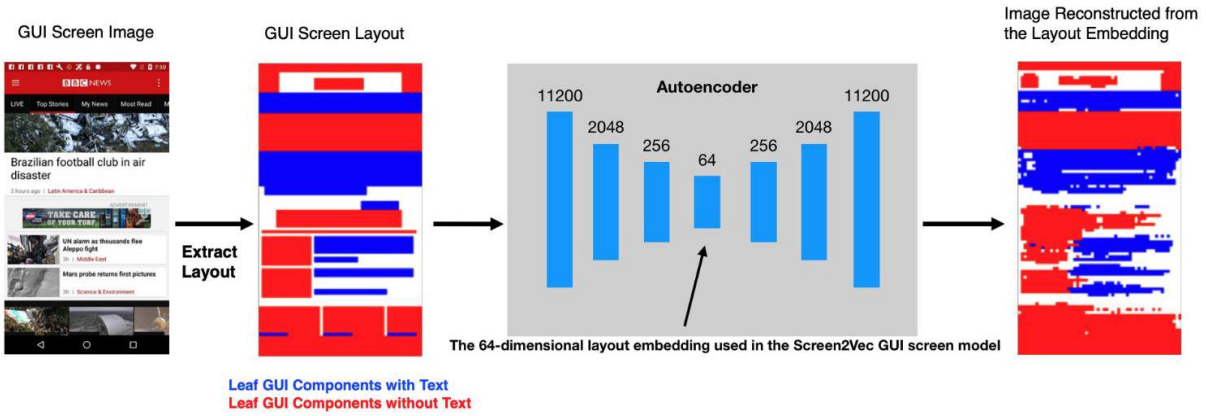

Figure 2: Screen2Vec extracts the layout of a GUI screen as a bitmap, and encodes this bitmap into a 64-dimensional vector using a standard autoencoder architecture where the autoencoder is trained on the loss of the output of the decoder [9].

图 2:Screen2Vec 将 GUI 屏幕的布局提取为位图，并使用标准自编码器架构将该位图编码为 64 维向量，其中自编码器通过解码器输出的损失进行训练 [9]。

The GUI component embedding model takes about 120 epochs to train, while the GUI screen embedding model takes 80-120 epochs depending on which version is being trained [5] . A virtual machine with 2 NVIDIA Tesla K80 GPUs can train the GUI component embedding model in about 72 hours, and train the GUI screen embedding model in about 6-8 hours.

GUI 组件嵌入模型大约需要训练 120 个 epoch，而 GUI 屏幕嵌入模型根据训练的版本需要 80-120 个 epoch [5] 。在配备 2 块 NVIDIA Tesla K80 GPU 的虚拟机上，GUI 组件嵌入模型大约需要 72 小时训练完成，GUI 屏幕嵌入模型约需要 6-8 小时。

We used PyTorch's implementation of the CrossEntropyLoss function [6] to calculate the prediction loss. The CrossEntropyLoss function combines negative log likelihood loss (NLL Loss) with the log softmax function:

我们使用 PyTorch 对 CrossEntropyLoss 函数的实现 [6] 来计算预测损失。CrossEntropyLoss 函数将负对数似然损失 (NLL Loss) 与 log softmax 函数结合起来:

$$\text{CrossEntropyLoss}\ (x,\ \text{class}\ ) = \ \text{NLL\_Loss}\ (\log \text{Softmax}\,(x)\,,\ \text{class}\ ))$$

$$= -\log\left(\frac{\exp\,(x\,[\ \text{class}\ ])}{\sum\limits_{c}\exp\,(x\,[c])}\right)$$

$$= -x\,[\ \text{class}\ ] + \log\sum_{c}\exp\,(x\,[c])$$

---

[5] The version without spatial information takes 80 epochs; and the one with spatial information takes 120.

[5] 不包含空间信息的版本需要 80 个 epoch；包含空间信息的版本需要 120 个。

[6] https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html

[6] https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html

---

In the case of the GUI component embedding model, the total loss is the sum of the cross entropy loss for the text prediction and the cross entropy loss for the class type prediction. In calculating the cross entropy loss, each text prediction was compared to every possible text embedding in the vocabulary, and each class prediction was compared to all possible class embeddings.

在 GUI 组件嵌入模型的情况下，总损失为文本预测的交叉熵损失与类别类型预测的交叉熵损失之和。在计算交叉熵损失时，每个文本预测都与词汇表中每个可能的文本嵌入进行比较，每个类别预测都与所有可能的类别嵌入进行比较。

In the case of the GUI screen embedding model, the loss is exclusively for screen predictions. However, the vector x does not contain the similarity between the correct prediction and every screen in the dataset. Instead we use negative sampling [31, 32] so that we do not have to recalculate and update every screen's embedding on every training iteration, which is computationally expensive and prone to over-fitting. In each iteration, the prediction is compared to the correct screen and a sample of negative data that consists of: a random sampling of size 128 of other screens, the other screens in the batch, and the screens in the same trace as the correct screen, used in the prediction task. We specifically include the screens in the same trace to promote screen-specific learning in this process: This way, we can disincentive screen embeddings that are based solely on the app [7] , and emphasize having the model learn to differentiate the different screens within the same app.

在 GUI 屏幕嵌入模型的情况下，损失仅用于屏幕预测。然而，该向量 x 并不包含正确预测与数据集中每个屏幕之间的相似度。相反我们使用负采样 [31, 32]，以避免在每次训练迭代中重新计算并更新每个屏幕的嵌入，这在计算上代价高且易导致过拟合。在每次迭代中，预测与正确屏幕以及一组负样本进行比较，该负样本由: 随机抽取的 128 个其他屏幕、批次中的其他屏幕以及与正确屏幕位于同一轨迹中的屏幕组成。我们特意包含同一轨迹中的屏幕以促进屏幕特定的学习: 这样可以抑制仅基于应用 [7] 的屏幕嵌入，并强调让模型学会区分同一应用内的不同屏幕。

2.4 Baselines

2.4 基线

We compared Screen2Vec to the following three baseline models:

我们将 Screen2Vec 与以下三种基线模型进行了比较:

Text Embedding Only. The TextOnly model replicates the screen embedding method used in SOVITE [22]. It only looks at the textual content on the screen: the screen embedding vector is computed by averaging the text embedding vectors for all the text found on the screen. The pre-trained Sentence-BERT model [39] calculates the text embedding vector for each text. With the the TextOnly model, screens with semantically similar textual contexts will have similar embedding vectors.

仅文本嵌入。TextOnly 模型复现了 SOVITE [22] 中使用的屏幕嵌入方法。它只关注屏幕上的文本内容: 屏幕嵌入向量通过对屏幕上所有文本的文本嵌入向量取平均来计算。每段文本的文本嵌入向量由预训练的 Sentence-BERT 模型 [39] 计算。使用 TextOnly 模型，具有语义相似文本内容的屏幕将具有相似的嵌入向量。

Layout Embedding Only. The LayoutOnly model replicates the screen embedding method used in the original RICO paper [9]. It only looks at the visual layout of the screen: It uses the layout embedding vector computed by the layout autoencoder to represent the screen, as discussed in Section 2.2. With the LayoutOnly model, screens with similar layouts will have similar embedding vectors.

仅布局嵌入。LayoutOnly 模型复现了原始 RICO 论文 [9] 中使用的屏幕嵌入方法。它只关注屏幕的视觉布局: 使用布局自编码器计算的布局嵌入向量来表示屏幕，如第 2.2 节所述。使用 LayoutOnly 模型，具有相似布局的屏幕将具有相似的嵌入向量。

Visual Embedding Only. The VisualOnly model encodes the visual look of a screen by applying an autoencoder (described in Section 2.2) directly on its screenshot image bitmap instead of the layout bitmap. This baseline is inspired by the visual-based approach used in GUI task automation systems such as VASTA [40], Sikuli [44], and HILC [14]. With the VisualOnly model, screens that are visually similar will have similar embedding vectors.

仅视觉嵌入。VisualOnly 模型通过将自编码器 (在第 2.2 节中描述) 直接应用于屏幕截图图像位图而不是布局位图来编码屏幕的视觉外观。该基线受 GUI 任务自动化系统 (如 VASTA [40]、Sikuli [44] 和 HILC [14]) 中基于视觉的方法启发。使用 VisualOnly 模型，视觉上相似的屏幕将具有相似的嵌入向量。

2.5 Prediction Task Results

2.5 预测任务结果

We report the performance on the GUI component and GUI screen prediction tasks of the Screen2Vec model, as well as the GUI screen prediction performance for the baseline models described above.

我们报告了 Screen2Vec 模型在 GUI 组件与 GUI 屏幕预测任务上的表现，以及上述基线模型的 GUI 屏幕预测表现。

Table 2 shows the top-1 accuracy (i.e., the top predicted GUI component matches the correct one), the top-0.01% accuracy (i.e., the correct GUI component is among the top 0.01% in the prediction result), the top-0.1% accuracy, and the top-1% accuracy of the two variations of the Screen2Vec model on the GUI component prediction task, where the model tries to predict the text content for each GUI component in all the GUI screens in the RICO dataset using its context (the other GUI components around it) among the collection of all the GUI components in the RICO dataset.

表 2 展示了两种 Screen2Vec 变体在 GUI 组件预测任务上的 top-1 准确率 (即预测的首位组件与正确组件匹配)、top-0.01% 准确率 (即正确组件位于预测结果的前 0.01% 之内)、top-0.1% 准确率和 top-1% 准确率。该任务要求模型在 RICO 数据集中所有 GUI 组件的集合中，利用上下文 (该组件周围的其他 GUI 组件) 预测每个 GUI 组件的文本内容。

Similarly, Table 3 reports the accuracy of the Screen2Vec model and the baseline models (TextOnly, LayoutOnly, and VisualOnly) on the task of predicting GUI screens, where each model tries to predict each GUI screen in all the GUI interaction traces in the RICO dataset using its context (the other GUI screens around it in the trace) among the collection of all the GUI screens in the RICO dataset. For the Screen2Vec model, we compare three versions: one that encodes the locations of GUI components and the screen layouts and uses the EUCLIDEAN distance measure, one that uses such spatial information and the HIERARCHICAL

distance measure, and one that uses the EUCLIDEAN distance measure without considering spatial information. A higher accuracy indicates that that the model is better at predicting the correct screen.

同样，表 3 报告了 Screen2Vec 模型和基线模型 (TextOnly、LayoutOnly、VisualOnly) 在 GUI 屏幕预测任务上的准确率: 每个模型在 RICO 数据集中所有 GUI 交互轨迹的屏幕集合中，利用上下文 (轨迹中该屏幕周围的其他屏幕) 预测每个 GUI 屏幕。对于 Screen2Vec，我们比较了三种版本: 一种对 GUI 组件位置和屏幕布局进行编码并使用 EUCLIDEAN 距离度量，一种使用这些空间信息并采用 HIERARCHICAL 距离度量，另一种使用 EUCLIDEAN 距离但不考虑空间信息。更高的准确率表示模型在预测正确屏幕方面更优。

We also report the normalized root mean square error (RMSE) of the predicted screen embedding vector for each model, normalized by the mean length of the actual screen embedding vectors. A smaller RMSE indicates that the top prediction screen generated by the model is, on average, more similar to the correct screen.

我们还报告了各模型预测屏幕嵌入向量的归一化均方根误差 (RMSE)，以实际屏幕嵌入向量平均长度进行归一化。较小的 RMSE 表明模型生成的首选预测屏幕在平均上更接近正确屏幕。

From the results in Table 3, we can see that the Screen2Vec models perform better than the baseline models in top-1 and top-k prediction accuracy. Among the different versions of Screen2Vec, the versions that encode locations of GUI components and the screen layouts performs better than the one without spatial information, suggesting that such spatial information is useful. The model that uses the HIERARCHICAL distance performs similarly to the one that uses the EUCLIDEAN distance in GUI component prediction, but performs worse in screen prediction. In the Sample Downstream Tasks section below, we will use the Screen2Vec-EUCLIDEAN-spatial info version of the Screen2Vec model.

从表 3 的结果可以看出，Screen2Vec 模型在 top-1 和 top-k 预测准确率上优于基线模型。在不同的 Screen2Vec 版本中，编码了 GUI 组件位置和屏幕布局的版本优于不含空间信息的版本，表明此类空间信息是有用的。使用 HIERARCHICAL 距离的模型在 GUI 组件预测上与使用 EUCLIDEAN 距离的模型表现相近，但在屏幕预测上表现更差。在下文的示例下游任务中，我们将使用 Screen2Vec-EUCLIDEAN-spatial info 版本的 Screen2Vec 模型。

As we can see, adding spatial information dramatically improves the Top-1 accuracy and the Top-0.01% accuracy. However, the improvements in Top 0.1% accuracy, Top 1% accuracy, and normalized RMSE are smaller. We think the main reason is that aggregating the textual information, GUI class types, and app descriptions is useful for representing the high-level ”topic” of a screen (e.g., a screen is about hotel booking because its text and app descriptions talk about hotels, cities, dates, rooms etc.), hence the good top 0.1% and 1% accuracy and normalized RMSE for the ”no spatial info” model. But these types of information are not sufficient for reliably differentiating the different types of screens needed (e.g., search, room details, order confirmation) in the hotel booking process because all these screens in the same app and task domain would contain ”semantically similar” text. This is why the adding spatial information is helpful in identifying the top-1 and top-0.01% results.

如上所示，加入空间信息显著提升了 Top-1 和 Top-0.01% 的准确率。然而，对 Top-0.1%、Top-1% 准确率和归一化 RMSE 的提升较小。我们认为主要原因是，汇聚文本信息、GUI 类别类型和应用描述有助于表示屏幕的高层“主题”(例如: 屏幕关于酒店预订，因为其文本和应用描述都在谈论酒店、城市、日期、房间等)，因此“无空间信息”模型在 top 0.1% 和 1% 准确率以及归一化 RMSE 上表现良好。但这些信息不足以可靠区分在酒店预订流程中需要的不同类型屏幕 (例如搜索、房间详情、订单确认)，因为同一应用和任务域的这些屏幕会包含“语义上相似”的文本。这就是为何加入空间信息有助于识别 top-1 和 top-0.01% 结果的原因。

---

[7] Since the next screen is always within the same app and therefore shares an app description embedding, the prediction task favors having information about the specific app (i.e., app store description embedding) dominate the embedding.

[7] 由于下一个屏幕始终在同一应用内，因此共享应用描述嵌入，预测任务偏向于让关于特定应用的信息 (即应用商店描述嵌入) 主导嵌入。

Model	Top-1 Accuracy	Top 0.01% Accuracy	Top 0.1% Accuracy	Top 1% Accuracy	Top 5% Accuracy	Top 10% Accuracy
Screen2Vec-EUCLIDEAN-text	0.443	0.619	0.783	0.856	0.885	0.901
Screen2Vec-HIERARCHICAL-text	0.588	0.687	0.798	0.849	0.878	0.894

模型	Top-1 准确率	Top 0.01% 准确率	Top 0.1% 准确率	Top 1% 准确率	Top 5% 准确率	Top 10% 准确率
Screen2Vec-EUCLIDEAN-text	0.443	0.619	0.783	0.856	0.885	0.901
Screen2Vec-HIERARCHICAL-text	0.588	0.687	0.798	0.849	0.878	0.894

Table 2: The GUI component prediction performance of the two variations of the Screen2Vec model with two different distance measures (EUCLIDEAN and HIERARCHICAL).

表 2: 两种 Screen2Vec 模型变体在两种不同距离度量 (EUCLIDEAN 和 HIERARCHICAL) 下的 GUI 组件预测性能。

Interestingly, the baseline models beat the ”no spatial info” version of Screen2Vec in normalized RMSE: i.e., although the baseline models are less likely to predict the correct screen, their predicted screens are, on average, more similar to the correct screen. A likely explanation to this phenomenon is that both baseline models use, by nature, similarity-based measures, while the Screen2Vec model is trained on a prediction-focused loss function. Therefore Screen2Vec does not emphasize making more similar predictions when then prediction is incorrect. However, we can see that the spatial info versions of Screen2Vec perform better than the baseline models on both the prediction accuracy and the similarity measure.

有趣的是，在归一化 RMSE 上，基线模型击败了“无空间信息”版本的 Screen2Vec: 也就是说，尽管基线模型较少预测出正确的屏幕，但它们预测的屏幕平均上与正确屏幕更相似。对此现象的一个可能解释是，两种基线模型本质上使用基于相似性的度量，而 Screen2Vec 则以预测为导向的损失函数进行训练。因此当预测错误时，Screen2Vec 不会强调做出更相似的预测。然而我们可以看到，包含空间信息的 Screen2Vec 版本在预测准确率和相似性度量上都优于基线模型。

3 SAMPLE DOWNSTREAM TASKS

3 示例下游任务

Note that while the accuracy measures are indicative of how much the model has learned about GUI screens and components, the main purpose of the Screen2Vec model is not to predict GUI components or screens, but to produce distributed vector representations for them that encode useful semantic, layout, and design properties. Therefore this section presents several sample downstream tasks to illustrate important properties of the Screen2Vec representations and the usefulness of our approach.

注意，尽管准确率度量能反映模型对 GUI 屏幕和组件学到的程度，Screen2Vec 模型的主要目的是为它们生成编码有用语义、布局和设计属性的分布式向量表示，而不是单纯预测 GUI 组件或屏幕。因此本节展示若干示例下游任务，以说明 Screen2Vec 表示的重要特性及我们方法的实用性。

3.1 Nearest Neighbors

3.1 最近邻

The nearest neighbor task is useful for data-driven design, where the designers want to find examples for inspiration and for understanding the possible design solutions [9]. The task focuses on the similarity between GUI screen embeddings: for a given screen, what are the top-N most similar screens in the dataset? The similar technique can also be used for unsupervised clustering in the dataset to infer different types of GUI screens. In our context, this task also helps demonstrate the different characteristics between Screen2Vec and the three baseline models.

最近邻任务对数据驱动的设计非常有用，设计师可借此寻找灵感示例并理解可能的设计方案 [9]。该任务关注 GUI 屏幕嵌入之间的相似性: 给定一个屏幕，数据集中相似度最高的前 N 个屏幕是什么? 类似技术也可用于数据集的无监督聚类，以推断不同类型的 GUI 屏幕。在我们的语境中，该任务还有助于展示 Screen2Vec 与三种基线模型之间的不同特性。

We conducted a Mechanical Turk study to compare the similarity between the nearest neighbor results generated by the different models. We selected 50 screens from apps and app domains that most users are familiar with. We did not select random apps from the RICO dataset, as many apps in the dataset would be obscure to Mechanical Turk workers so they might not understand them and therefore might not be able to judge the similarity of the results. For each screen, we retrieved the top-5 most similar screens using each of the 3 models. Therefore, each of the 50 screens had up to 3 (models) $\times$5 (screen each) = 15 similar screens, but many had fewer since different models may select the same screens.

我们进行了 Mechanical Turk 调查来比较不同模型生成的最近邻结果的相似性。我们从用户最熟悉的应用及应用领域中选取了 50 个屏幕。我们没有从 RICO 数据集中随机选择应用，因为数据集中许多应用对 Mechanical Turk 工作者来说较为陌生，他们可能无法理解从而无法判断结果的相似性。对于每个屏幕，我们使用 3 个模型分别检索了前 5 个最相似的屏幕。因此，每个被选的 50 个屏幕最多会有 3 个 (模型) $\times$5 (每个屏幕) = 15 相似屏幕，但许多屏幕会更少，因为不同模型可能选到相同的屏幕。

79 Mechanical Turk workers participated in this study [8] . In total, they labeled the similarity between 5,608 pairs of screens. Each worker was paid $2 for each batch of 5 sets of source screens they labeled. A batch on average takes around 10 minutes to complete. In each batch, a worker went through a sample of 5 source screens from the 50 source screens in random order, where for each source screen, the worker saw the union of the top-5 most similar screens to the source screen generated by the 3 models in random order. For each screen, we also showed the worker the app it came from and a short description of the app from the Google Play Store, but we did not show them which model produced the screen. The worker was asked to rate the similarity of each screen to the original source screen on a scale of 1 to 5 (Figure 3). We asked the workers to consider 3 aspects in measuring similarity: (1) app similarity (how similar are the two apps); (2) screen type similarity (how similar are the types of the two screens e.g., if they are both sign up screens, search results, settings menu etc.); and (3) content similarity (how similar are the content on the two screens).

共有 79 名 Mechanical Turk 工作者参与了本研究 [8] 。他们共标注了 5,608 对屏幕的相似性。每位工人每批为标注 5 组源屏幕的工作获得 2 美元报酬。每批平均大约需要 10 分钟完成。每批中，工人按随机顺序浏览来自 50 个源屏幕中抽取的 5 个源屏幕样本；对于每个源屏幕，工人看到由 3 个模型生成的、按随机顺序合并后的前 5 个最相似屏幕。对于每个屏幕，我们还向工人展示了其所属应用及来自 Google Play 商店的简短应用描述，但未告知哪个模型生成了该屏幕。要求工人按 1 到 5 的尺度对每个屏幕与原始源屏幕的相似性进行评分 (图 3)。我们要求工人在衡量相似性时考虑 3 个方面:(1) 应用相似性 (两者应用有多相似)；(2) 屏幕类型相似性 (两者屏幕类型有多相似，例如是否都是注册页、搜索结果、设置菜单等)；及 (3) 内容相似性 (两者屏幕上的内容有多相似)。

Table 4 shows the mean screen similarity rated by the Mechanical Turk workers for the top-5 nearest neighbor results of the sample source screens generated by the 3 models. The Mechanical Turk workers rated the nearest neighbor screens generated by the Screen2Vec model to be, on average, more similar to their source screens than the nearest neighbor screens generated by the baseline TextOnly and LayoutOnly models. Tested with a nonparametric Mann-Whitney U test (because the ratings are not normally distributed), the differences between the mean ratings of the Screen2Vec model and both the TextOnly model and the LayoutOnly model are significant ($p < 0.0001$) .

表 4 显示了 Mechanical Turk 工作者对 3 个模型为示例源屏幕生成的前 5 个最近邻结果的平均屏幕相似性评分。Mechanical Turk 工作者评定由 Screen2Vec 模型生成的最近邻屏幕平均上比由基线 TextOnly 和 LayoutOnly 模型生成的最近邻屏幕更接近其源屏幕。使用非参数 Mann-Whitney U 检验 (因为评分不服从正态分布) 检验后，Screen2Vec 模型与 TextOnly 模型及 LayoutOnly 模型之间的平均评分差异均具有显著性 ($p < 0.0001$) 。

Subjectively, when looking at the nearest neighbor results, we can see the different aspects of the GUI screens that each different model captures. Screen2Vec can create more comprehensive representations that encode the textual content, visual design and layout patterns, and app contexts of the screen compared with the baseline models, which only capture one or two aspects. For example, Figure 4 shows the example nearest neighbor results for the ”request ride” screen in the Lyft app. Screen2Vec model retrives the ”get direction” screen in the Uber Driver app, ”select navigation type” screen in the Waze app, and ”request ride” screen in the Free Now (My Taxi) app. Considering the Visual and component layout aspects, the result screens all feature a menu/information card at the bottom 1/3 to 1/4 of the screen, with a MapView taking the majority of the screen space. Considering the content and app domain aspects, all of these screens are from transportation-related

主观上，在查看最近邻结果时，我们可以看到不同模型对 GUI 屏幕捕捉到的不同侧面。与只捕捉一两个方面的基线模型相比，Screen2Vec 能创建更全面的表示，编码屏幕的文本内容、视觉设计与布局模式以及应用上下文。例如，图 4 展示了 Lyft 应用中“请求用车”屏幕的最近邻示例结果。Screen2Vec 模型检索到了 Uber Driver 应用中的“获取路线”屏幕、Waze 应用中的“选择导航类型”屏幕以及 Free Now (My Taxi) 应用中的“请求用车”屏幕。就视觉和组件布局而言，这些结果屏幕都在底部三分之一到四分之一处有一个菜单/信息卡，且 MapView 占据了大部分屏幕空间。就内容和应用领域而言，所有这些屏幕都来自与交通相关的

---

[8] The protocol was approved by the IRB at our institution.

[8] 该方案已获得我们机构伦理审查委员会 (IRB) 的批准。

Model	Top-1 racy Accu-	Top 0.01% Accuracy	Top 0.1% Accuracy	Top 1% Accuracy	Top 5% Accuracy	Normalized RMSE
Screen2Vec-EUCLIDEAN-spatial info	0.061	0.258	0.969	0.998	1.00	0.853
Screen2Vec-HIERARCHICAL-spatial info	0.052	0.178	0.646	0.924	0.990	0.997
Screen2Vec-EUCLIDEAN-no spatial info	0.0065	0.116	0.896	0.986	0.999	1.723
TextOnly	0.012	0.055	0.196	0.439	0.643	1.241
LayoutOnly	0.0041	0.024	0.091	0.222	0.395	1.135
VisualOnly	0.0060	0.026	0.121	0.252	0.603	1.543

模型	Top-1 敏感度 Accu-	前 0.01% 准确率	前 0.1% 准确率	前 1% 准确率	前 5% 准确率	归一化均方根误差
Screen2Vec-欧氏-空间信息	0.061	0.258	0.969	0.998	1.00	0.853
Screen2Vec-层次-空间信息	0.052	0.178	0.646	0.924	0.990	0.997
Screen2Vec-欧氏-无空间信息	0.0065	0.116	0.896	0.986	0.999	1.723
仅文本	0.012	0.055	0.196	0.439	0.643	1.241
仅布局	0.0041	0.024	0.091	0.222	0.395	1.135
仅视觉	0.0060	0.026	0.121	0.252	0.603	1.543

Table 3: The GUI screen prediction performance of the three variations of the Screen2Vec model and the baseline models (TextOnly, LayoutOnly, and VisualOnly).

表 3: 三种 Screen2Vec 模型变体与基线模型 (TextOnly、LayoutOnly 和 VisualOnly) 的 GUI 屏幕预测性能。

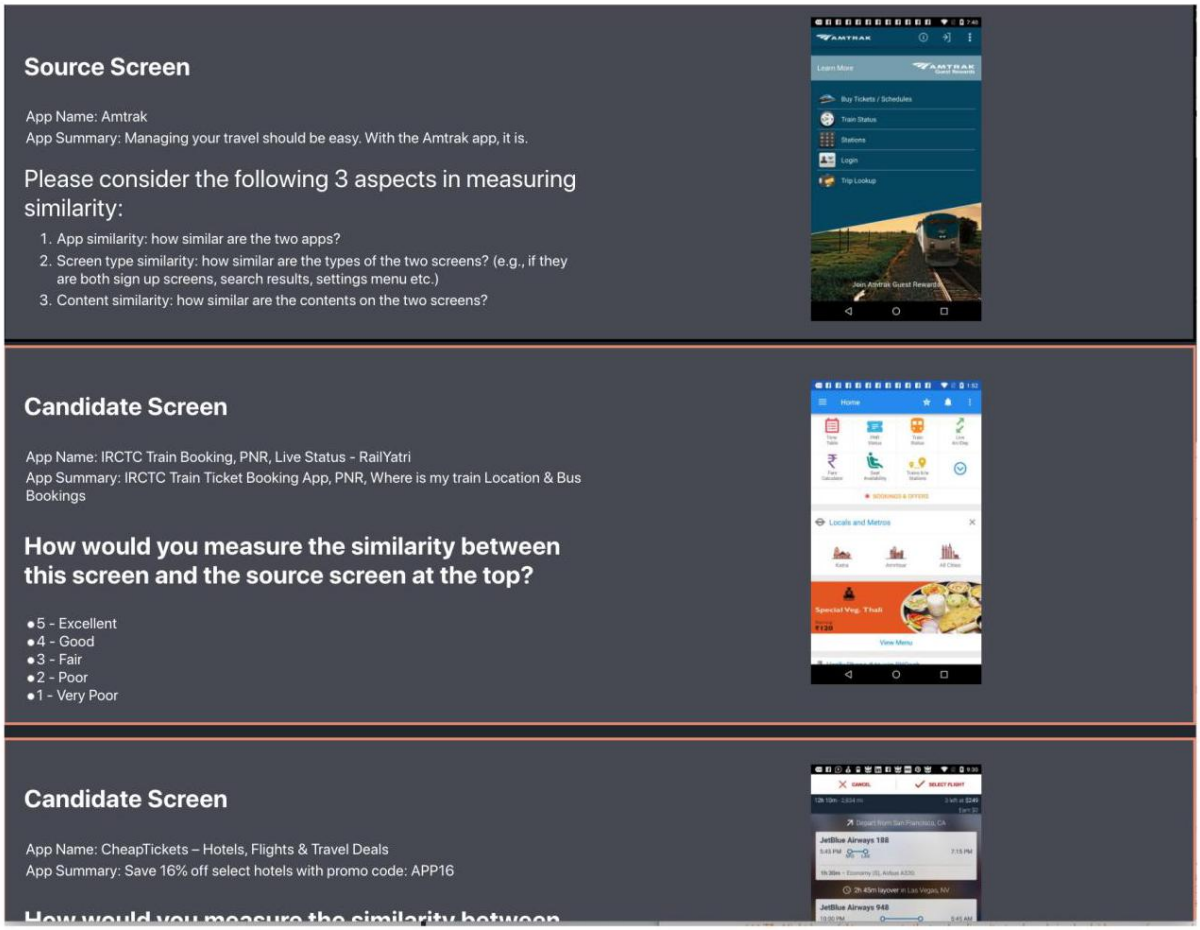

Figure 3: The interface shown to the Mechanical Turk workers for rating the similarities for the nearest neighbor results generated by different models.

图 3: 展示给 Mechanical Turk 工作者用于评估不同模型生成最近邻结果相似性的界面。

Screen2Vec		TextOnly		LayoutOnly	
Mean Rating	Std. Dev.	Mean Rating	Std. Dev.	Mean Rating	Std. Dev.
3.295*	1.238	3.014*	1.321	2.410*	1.360

Screen2Vec		仅文本		仅布局	
平均评分	标准差	平均评分	标准差	平均评分	标准差
3.295*	1.238	3.014*	1.321	2.410*	1.360

Table 4: The mean screen similarity rated by the Mechanical Turk workers for the top-5 nearest neighbor results of the sample source screens generated by the 3 models: Screen2Vec, Text0nly, and Layout0nly (*p<0.0001).

表 4:Mechanical Turk 工作者对由三个模型生成的样本源屏幕的前 5 个最近邻结果的平均屏幕相似度评分:Screen2Vec、TextOnly 和 LayoutOnly(*p<0.0001)。

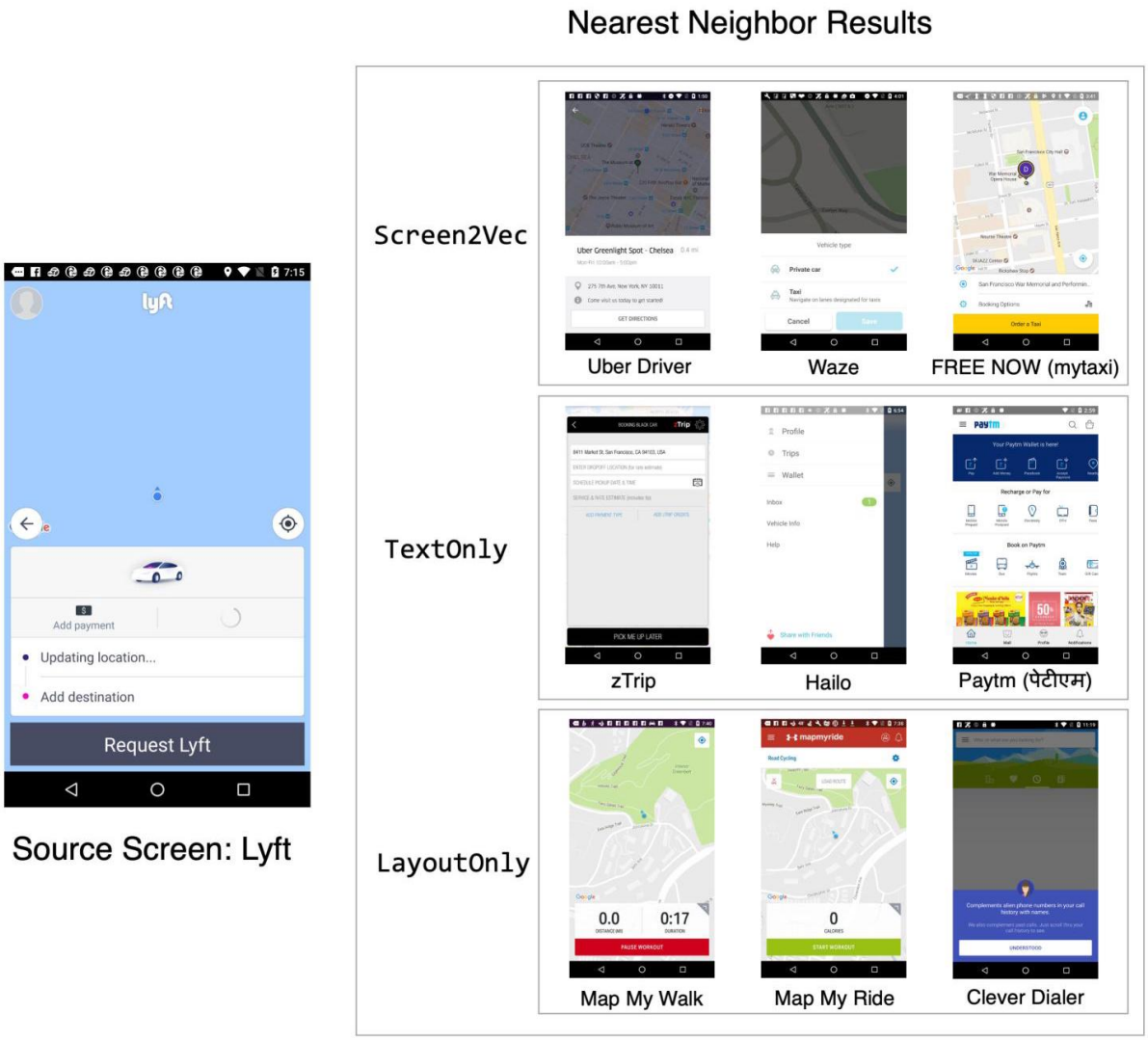

Figure 4: The example nearest neighbor results for the Lyft "request ride" screen generated by the Screen2Vec, Text0nly, and LayoutOnly models. apps that allow the user to configure a trip. In comparison, the TextOnly model retrieves the "request ride" screen from the zTrip app, the "main menu" screen from the Hailo app (both zTrip and Hailo are taxi hailing apps), and the home screen of the Paytm app (a mobile payment app in India). The commonality of these screens is that they all include text strings that are semantically similar to "payment" (e.g., add payment type, wallet, pay, add money), and strings that are semantically similar to "destination" and "trips" (e.g., drop off location, trips, bus, flights). But the model did not consider the visual layout and design patterns of the screens nor the app context. Therefore the result contains the "main menu" (a quite different type of screen) in the Hailo app and the "home screen" in the Paytm app (a quite different type of screen in a different type of app). The LayoutOnly model, on the other hand, retrieves the "exercise logging" screens from the Map My Walk app and the Map My Ride app, and the tutorial screen from the Clever Dialer app. We can see that the content and app-context similarity of the result of the LayoutOnly model is quite lower than those of the Screen2Vec and TextOnly models. However, the result screens all share similar layout features as the source screen, such as the menu/information card at the bottom of the screen and the screen-wide button at the bottom of the menu.

图 4: 针对 Lyft “请求乘车”屏幕，Screen2Vec、TextOnly 和 LayoutOnly 模型示例性最近邻结果。Screen2Vec 检索到的是允许用户配置行程的应用界面。相比之下，TextOnly 模型检索到 zTrip 应用的“请求乘车”屏幕、Hailo 应用的“主菜单”屏幕 (zTrip 和 Hailo 都是叫车应用)，以及印度移动支付应用 Paytm 的主屏幕。这些屏幕的共性是都包含与“支付”语义相近的文本字符串 (例如添加支付类型、钱包、支付、充值) 以及与“目的地”和“行程”语义相近的字符串 (例如下车地点、行程、巴士、航班)。但该模型没有考虑屏幕的视觉布局与设计模式，也未考虑应用上下文，因此结果中出现了 Hailo 应用的“主菜单”(一种截然不同的屏幕类型) 和 Paytm 应用的“主屏幕”(在不同类型应用中截然不同的屏幕)。另一方面，LayoutOnly 模型检索到的是 Map My Walk 和 Map My Ride 应用的“运动记录”屏幕，以及 Clever Dialer 应用的教程屏幕。可见 LayoutOnly 的结果在内容和应用上下文相似性上明显低于 Screen2Vec 和 TextOnly 的结果，然而这些结果屏幕在布局特征上与源屏幕相似，例如屏幕底部的菜单/信息卡片以及菜单底部跨屏宽度的按钮。

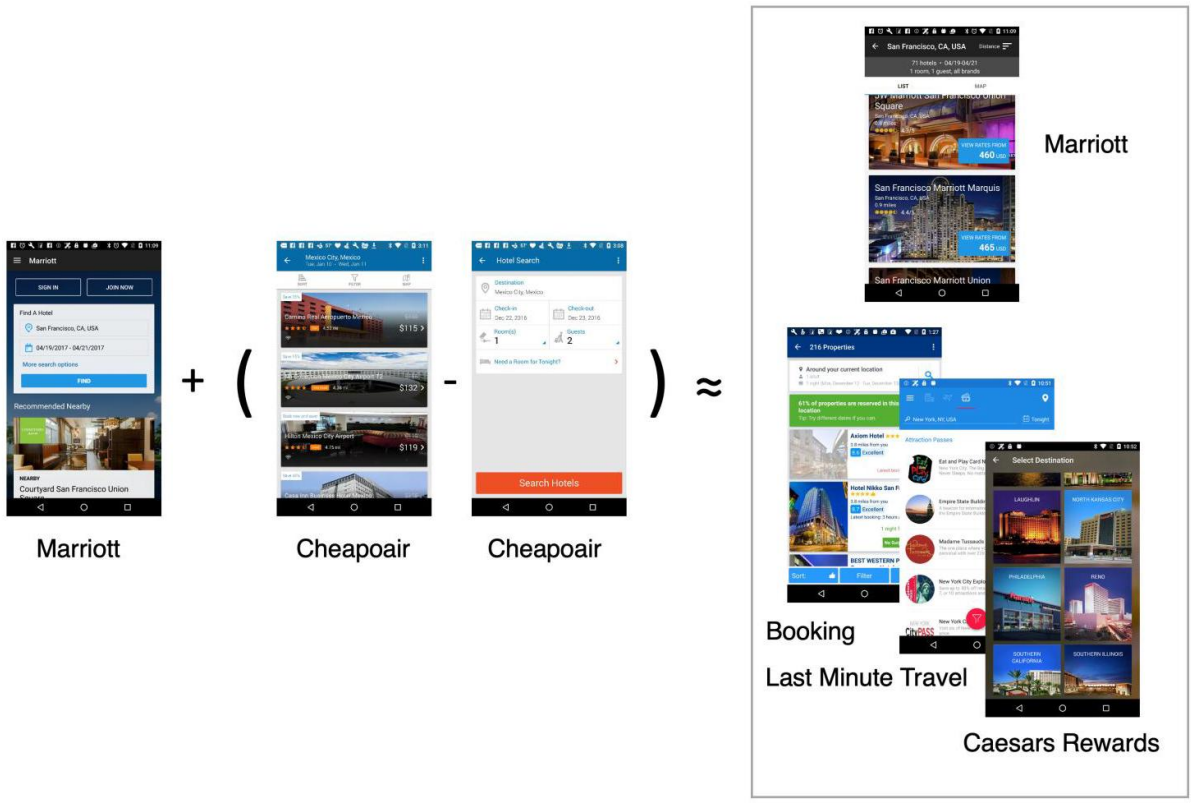

Figure 5: An example showing the composability of Screen2Vec embeddings: running the nearest neighbor query on the composite embedding of Marriott app ’s hotel booking page + Cheapoair app’s hotel booking page - Cheapoair app’s search result page can match the Marriott app’s search result page and the similar pages of a few other travel apps.

图 5: 展示 Screen2Vec 嵌入可组合性的示例: 在 Marriott 应用的酒店预订页嵌入 + Cheapoair 应用的酒店预订页嵌入 - Cheapoair 应用的搜索结果页嵌入的复合嵌入上运行最近邻查询，可以匹配到 Marriott 应用的搜索结果页以及其他若干旅行应用的类似页面。

3.2 Embedding Composability

3.2 嵌入的可组合性

A useful property of embeddings is that they are composable-meaning that we can add, subtract, and average embeddings to form a meaningful new one. This property is commonly used in word embeddings. For example, in Word2Vec, analogies such as ”man is to woman as brother is to sister” is reflected in that the vector (man - woman) is similar to the vector (brother - sister). Besides representing analogies, this embedding composability can also be utilized for generative purposes-for example, (brother - man + woman) results in an embedding vector that represents ”sister”.

嵌入的一个有用属性是可组合性——我们可以对嵌入进行加、减、平均以构成有意义的新嵌入。这个属性在词嵌入中常被使用。例如在 Word2Vec 中，“man 对应 woman，等同于 brother 对应 sister”这种类比体现为向量 (man - woman) 与 (brother - sister) 的相似性。除了表示类比，这种嵌入可组合性也可用于生成目的——例如 (brother - man + woman) 会产生表示“sister”的嵌入向量。

This property is also useful in screen embeddings. For example, we can run a nearest neighbor query on the composite embedding of (Marriott app ’s ”hotel booking” screen + (Cheapoair app’s ”search result” screen - Cheapoair app’s ”hotel booking” screen)). The top result is the ”search result” screen in the Marriott app (see Figure 5). When we filter the result to focus on screens from apps other than Marriott, we get screens that show list results of items from other travel-related mobile apps such as Booking, Last Minute Travel, and Caesars Rewards.

这种属性在屏幕嵌入中同样有用。例如，我们可以对复合嵌入 (Marriott 应用的“酒店预订”屏幕 + (Cheapoair 应用的“搜索结果”屏幕 - Cheapoair 应用的“酒店预订”屏幕)) 运行最近邻查询。首个结果是 Marriott 应用的“搜索结果”屏幕 (见图 5)。当我们过滤结果以关注非 Marriott 的应用屏幕时，会得到显示其他旅游相关移动应用 (如 Booking、Last Minute Travel 和 Caesars Rewards) 中条目列表结果的屏幕。

The composability can make Screen2Vec particularly useful for GUI design purposes-the designer can leverage the composability to find inspiring examples of GUI designs and layouts. We will discuss more about its potential applications in Section 4.

这种可组合性使 Screen2Vec 对 GUI 设计特别有用——设计师可以利用可组合性寻找富有启发性的 GUI 设计与布局示例。我们将在第 4 节讨论其更多潜在应用。

3.3 Screen Embedding Sequences for Representing Mobile Tasks

3.3 用屏幕嵌入序列表示移动任务

GUI screens are not only useful data sources individually on their own, but also as building blocks to represent a user’s task. A task in an app, or across multiple apps, can be represented as a sequence of GUI screens that makes up the user interaction trace for performing this task using app GUIs. In this section, we conduct a preliminary evaluation on the effectiveness of embedding mobile tasks as sequences of Screen2Vec

screen embedding vectors.

GUI 屏幕不仅单独作为有用的数据源，也可作为表示用户任务的构建模块。一个应用内或跨应用的任务可以表示为一系列 GUI 屏幕，构成执行该任务的用户交互轨迹。在本节中，我们对将移动任务嵌入为 Screen2Vec 屏幕嵌入向量序列的有效性进行初步评估。

Similar to GUI screens and components, the goal of embedding mobile tasks is to represent them in a vector space where more similar tasks are closer to each other. To test this, we recorded the scripts of completing 10 common smartphone tasks, each with two variations that use different apps, using our open-sourced SUGILITE [20] system on a Pixel 2 XL phone running Android 8.0. Each script consists of a sequence of "perform action X (e.g., click, long click) on the GUI component Y in the GUI screen Z". In this preliminary evaluation, we only used the screen context: we represented each task as the average of the Screen2Vec screen embedding vectors for all the screens in the task sequence.

类似于 GUI 屏幕与组件，嵌入移动任务的目标是将它们表示在向量空间中，使得更相似的任务彼此更接近。为验证这一点，我们在运行 Android 8.0 的 Pixel 2 XL 手机上使用我们开源的 SUGILITE [20] 系统记录了完成 10 个常见智能手机任务的脚本，每个任务有两种使用不同应用的变体。每个脚本由一系列“在 GUI 屏幕 Z 的 GUI 组件 Y 上执行操作 X(例如点击、长按)”组成。在本次初步评估中，我们只使用了屏幕上下文: 将每个任务表示为该任务序列中所有屏幕的 Screen2Vec 屏幕嵌入向量的平均值。

Table 5 shows the 10 tasks we tested on, the two apps used for each task, and the number of unique GUI screens in each trace used for task embedding. We queried for the nearest neighbor within the 20 task variations for each task variation, and checked if the model could correctly identify the similar task that used a different app. The Screen2Vec model achieved a 18/20 (90%) accuracy in this test. In comparison, when we used the TextOnly model for task embedding, the accuracy was 14/20 (70%).

表 5 显示了我们测试的 10 个任务、每个任务使用的两个应用以及用于任务嵌入的每条轨迹中唯一 GUI 屏幕的数量。对于每个任务变体，我们在 20 个任务变体中查询最近邻，并检查模型是否能正确识别使用不同应用的相似任务。Screen2Vec 模型在该测试中取得了 18/20(90%) 的准确率。相比之下，当我们使用 TextOnly 模型进行任务嵌入时，准确率为 14/20(70%)。

While the task embedding method we explored in this section is quite primitive, it illustrates that the Screen2Vec technique can be used to effectively encode mobile tasks into the vector space where semantically similar tasks are close to each other. For the next steps, we plan to further explore this direction. For example, the current method of averaging all the screen embedding vectors does not consider the order of the screens in the sequence. In the future, we may collect a dataset of human annotations of task similarity, and use techniques that can encode the sequences of items, such as recurrent neural networks (RNN) and long short-term memory (LSTM) networks, to create the task embeddings from sequences of screen embeddings. We may also incorporate the Screen2Vec embeddings of the GUI components that were interacted with (e.g., the button that was clicked on) to initiate the screen change into the pipeline for embedding tasks.

尽管本节探讨的任务嵌入方法相当原始，但它说明了 Screen2Vec 技术可以有效地将移动任务编码到向量空间中，使语义相近的任务靠得更近。下一步我们计划进一步在该方向上探索。例如，当前将所有屏幕嵌入向量取平均的方法并未考虑屏幕序列的顺序。未来我们可能收集人工标注的任务相似性数据集，并使用能够编码序列项的技术，如循环神经网络 (RNN) 和长短期记忆网络 (LSTM)，从屏幕嵌入序列中创建任务嵌入。我们也可能将触发屏幕变化的被交互 GUI 组件 (例如被点击的按钮) 的 Screen2Vec 嵌入并入任务嵌入流程。

4 POTENTIAL APPLICATIONS

4 潜在应用

This section describes several potential applications where the new Screen2Vec technique can be useful based on the downstream tasks described in Section 3.

本节描述了基于第 3 节所述下游任务，Screen2Vec 新技术可发挥作用的若干潜在应用。

Screen2Vec can enable new GUI design aids that take advantage of the nearest neighbor similarity and composability of Screen2Vec embeddings. Prior work [9, 13, 16] has shown that data-driven tools that enable designers to curate design examples are useful for interface designers. Unlike [9], which uses a content-agnostic approach that focuses on the visual and layout similarities, Screen2Vec considers the textual content and app meta-data in addition to the visual and layout patterns, often leading to different nearest neighbor results as discussed in Section 3.1. This new type of similarity results will also be useful when focusing on interface design beyond just visual and layout issues, as the results enable designers to query for example designs that display similar content or screens that are used in apps in a similar domain. The composability in Screen2Vec embeddings enables querying for design examples at a finer granularity. For example, suppose a designer wishes to find examples for inspiring the design of a new checkout page for app A. They may query for the nearest neighbors of the synthesized embedding App A's order page + (App B's checkout page - App B's order page). Compared with only querying for the nearest neighbors of App B's checkout page, this synthesized query encodes the interaction context (i.e., the desired page should be the checkout page for App A's order page) in addition to the "checkout" semantics.

Screen2Vec 可启用利用最近邻相似性与可组合性的新型 GUI 设计辅助工具。先前工作 [9, 13, 16] 表明，支持设计师策划设计示例的数据驱动工具对界面设计师很有用。与 [9] 使用侧重视觉与布局相似性的内容不可知方法不同，Screen2Vec 在考虑视觉与布局模式的同时还纳入了文本内容与应用元数据，常常产生如第 3.1 节所述的不同最近邻结果。这种新型相似性结果在关注界面设计而不仅仅是视觉与布局问题时也很有用，因为结果使设计师能够查询显示相似内容或在相似领域中使用的应用屏幕示例。Screen2Vec 嵌入的可组合性使得以更细粒度查询设计示例成为可能。例如，假设设计师希望为应用 A 的新结账页面寻找灵感示例，他们可以查询合成嵌入 App A 的订单页 +(App B 的结账页 - App B 的订单页) 的最近邻。与仅查询 App B 的结账页的最近邻相比，这一合成查询在“结账”语义之外还编码了交互上下文 (即期望的页面应为针对 App A 订单页的结账页)。

The Screen2Vec embeddings can also be useful in generative GUI models. Recent models such as the neural design network (NDN) [18] and LayoutGAN [19] can generate realistic GUI layouts based on user-specified constraints (e.g., alignments, relative positions between GUI components). Screen2Vec can be used

in these generative approaches to incorporate the semantics of GUIs and the contexts of how each GUI screen and component gets used in user interactions. For example, the GUI component prediction model can estimate the likelihood of each GUI component given the context of the other components in a generated screen, providing a heuristic of how likely the GUI components would fit well with each other. Similarly, the GUI screen prediction model may be used as a heuristic to synthesize GUI screens that would better fit with the other screens in the planned user interaction flows. Since Screen2Vec has been shown effective in representing mobile tasks in Section 3.3, where similar tasks will yield similar embeddings, one may also use the task embeddings of performing the same task on an existing app to inform the generation of new screen designs. The embedding vector form of Screen2Vec representations would make them particularly suitable for use in the recent neural-network based generative models.

Screen2Vec 嵌入在生成式 GUI 模型中也很有用。近期模型如神经设计网络 (NDN)[18] 与 LayoutGAN [19] 能在用户指定约束 (例如对齐、组件之间的相对位置) 下生成逼真的 GUI 布局。Screen2Vec 可在这些生成方法中用于纳入 GUI 的语义以及每个 GUI 屏幕与组件在用户交互中如何被使用的上下文。例如，GUI 组件预测模型可在给定生成屏幕中其他组件的上下文下估计各组件的可能性，为组件之间是否契合提供启发。同样，GUI 屏幕预测模型可作为启发式用于合成更适配预期用户交互流程中其他屏幕的 GUI 屏幕。由于 Screen2Vec 在第 3.3 节中已被证明能有效表示移动任务，使相似任务产生相似嵌入，因此也可利用在现有应用上执行相同任务的任务嵌入来指导新屏幕设计的生成。Screen2Vec 表示的向量形式使其特别适合用于近期基于神经网络的生成模型。

Screen2Vec's capability of embedding tasks can also enhance interactive task learning systems. Specifically, Screen2Vec may be used to enable more powerful procedure generalizations of the learned tasks. We have shown that the Screen2Vec model can effectively predict screens in an interaction trace. Results in Section 3.3 also indicated that Screen2Vec can embed mobile tasks so that the interaction traces of completing the same task in different apps will be similar to each other in the embedding vector space. Therefore, it is quite promising that Screen2Vec may be used to generalize a task learned from the user by demonstration in one app to another app in the same domain (e.g., generalizing the procedure of ordering coffee in the Starbucks app to the Dunkin' Donut app). In the future, we plan to further explore this direction by incorporating Screen2Vec into open-sourced mobile interactive task learning agents such as our SUGILITE system [20].

Screen2Vec 嵌入任务的能力也能增强交互式任务学习系统。具体来说，Screen2Vec 可用于实现对已学任务更强的过程泛化。我们已展示 Screen2Vec 模型能够有效地预测交互轨迹中的屏幕。第 3.3 节的结果也表明 Screen2Vec 能将移动任务嵌入，使得在不同应用中完成相同任务的交互轨迹在嵌入向量空间中彼此相似。因此，Screen2Vec 很有希望被用来将用户在一个应用中通过示范学到的任务泛化到同一领域的另一个应用 (例如，将在星巴克应用中点咖啡的流程泛化到唐恩都乐应用)。未来，我们计划通过将 Screen2Vec 纳入诸如我们的 SUGILITE 系统 [20] 之类的开源移动交互式任务学习代理，进一步探索这一方向。

5 LIMITATIONS AND FUTURE WORK

5 限制与未来工作

There are several limitations of our work in Screen2Vec. First, Screen2Vec has only been trained and tested on Android app GUIs. However, the approach used in Screen2Vec should apply to any GUI-based apps

with hierarchical-based structures (e.g., view hierarchies in iOS apps and hierarchical DOM structures in web apps). We expect embedding desktop GUIs to be more difficult than mobile ones, because individual screens in desktop GUIs are usually more complex with more heterogeneous design and layout patterns.

我们的 Screen2Vec 工作存在若干限制。首先，Screen2Vec 仅在 Android 应用 GUI 上训练和测试。然而，Screen2Vec 使用的方法应适用于任何基于 GUI 且具有层级结构的应用 (例如 iOS 应用中的视图层级和网页应用中的层级 DOM 结构)。我们预计桌面 GUI 的嵌入会比移动端更困难，因为桌面 GUI 中的单个屏幕通常更复杂，具有更多异质的设计和布局模式。

Second, the RICO dataset we use only contains interaction traces within single apps. The approach used in Screen2Vec should generalize to interaction traces across multiple apps. We plan to evaluate its prediction performance on cross-app traces in the future with an expanded dataset of GUI interaction traces. The RICO dataset also does not contain screens from paid apps, screens that require special accounts/privileges to access to (screens that require free accounts to access are included when the account registration is readily available in the app), or screens that require special hardware (e.g., in the companion apps for smart home devices) or specific context (e.g., pages that are only shown during events) to access. This limitation of the RICO dataset might affect the performance of the pre-trained Screen2Vec model on these underrepresented types of app screens.

其次，我们使用的 RICO 数据集仅包含单一应用内的交互轨迹。Screen2Vec 使用的方法应可推广到跨多应用的交互轨迹。我们计划在未来使用扩展的 GUI 交互轨迹数据集评估其对跨应用轨迹的预测性能。RICO 数据集也不包含付费应用的屏幕、需要特殊账户/权限才能访问的屏幕 (当应用中注册免费账户即可访问时则包含)、或需要特殊硬件 (例如智能家居设备配套应用) 或特定上下文 (例如仅在活动期间显示的页面) 才能访问的屏幕。RICO 数据集的这些限制可能影响预训练 Screen2Vec 模型在这些欠代表类型的应用屏幕上的表现。

Task Description	App 1	Screen Count	App 2	Screen Count
Request a cab	Lyft	3	Uber	2
Book a flight	Fly Delta	4	United Airlines	4
Make a hotel reservation	Booking.com	7	Expedia	7
Buy a movie ticket	AMC Theaters	3	Cinemark	4
Check the account balance	Chase	4	American Express	3
Check sports scores	ESPN	4	Yahoo! Sports	4
Look up the hourly weather	AccuWeather	3	Yahoo! Weather	3
Find a restaurant	Yelp	3	Zagat	4
Order an iced coffee	Starbucks	7	Dunkin' Donuts	8
Order takeout food	GrubHub	4	Uber Eats	3

任务描述	应用 1	屏幕数量	应用 2	屏幕数量
叫一辆车	Lyft	3	Uber	2
预订机票	乘坐达美航空	4	联合航空	4
预订酒店	Booking.com	7	Expedia	7
购买电影票	AMC 影院	3	Cinemark	4
查询账户余额	Chase	4	American Express	3
查看体育比分	ESPN	4	Yahoo! Sports	4
查询每小时天气	AccuWeather	3	Yahoo! Weather	3
找餐厅	Yelp	3	Zagat	4
点一杯冰咖啡	Starbucks	7	Dunkin' Donuts	8
点外卖	GrubHub	4	Uber Eats	3

Table 5: A list of 10 tasks we used for the preliminary evaluation of using Screen2Vec for task embedding, along with the apps used and the count of screens used in the task embedding for each variation.

表 5: 用于初步评估用 Screen2Vec 做任务嵌入的 10 个任务列表，列出所用应用及每种变体在任务嵌入中使用的屏幕数量。

A third limitation is that the current version of Screen2Vec does not encode the semantics of graphic icons that have no textual information. Accessibility-compliant apps all have alternative texts for their graphic icons, which Screen2Vec already encodes in its GUI screen and component embeddings as a part of the text embedding. However, for non-accessible apps, computer vision-based (e.g., [8, 30]) or crowd-based (e.g., [45]) techniques can be helpful for generating textual annotations for graphic icons so that their semantics can be represented in Screen2Vec. Another potentially useful kind of information is the rules and examples in GUI design systems (e.g., Android Material Design, iOS Design Patterns). While Screen2Vec can, in some ways, "learn" these patterns from the training data, it will be interesting to explore a hybrid approach that can leverage their explicit notions. We will explore incorporating these techniques into the Screen2Vec pipeline in the future.

第三个局限是当前版本的 Screen2Vec 无法编码没有文本信息的图标语义。符合无障碍要求的应用都会为图标提供替代文本，Screen2Vec 已将这些替代文本作为文本嵌入的一部分编码到其 GUI 屏幕和组件嵌入中。然而，对于非无障碍应用，基于计算机视觉 (例如 [8, 30]) 或基于众包 (例如 [45]) 的技术可用于为图标生成文本注释，从而使其语义能在 Screen2Vec 中表示。另一类可能有用的信息是 GUI 设计系统中的规则和示例 (例如 Android Material Design、iOS 设计范式)。虽然 Screen2Vec 在某种程度上可以从训练数据"学习"这些模式，但探索能利用其显式概念的混合方法将很有意义。我们将来会探索将这些技术并入 Screen2Vec 流水线。

6 RELATED WORK

6 相关工作

6.1 Distributed Representations of Natural Language

6.1 自然语言的分布式表示

The study of representing words, phrases, and documents as mathematical objects, often vectors, is central to natural language processing (NLP) research [32, 42]. Conventional non-distributed word embedding methods represent a word using a one-hot representation where the vector length equals the size of the vocabulary, and only one dimension (that corresponds to the word) is on [42]. This representation does not encode the semantics of the words, as the vector for each word is perpendicular to the others. Documents represented using a one-hot word representation also suffer from the curse of dimensionality [3] as a result of the extreme sparsity in the representation.

将词、短语和文档表示为数学对象 (通常是向量) 的研究是自然语言处理 (NLP) 研究的核心 [32, 42]。传统的非分布式词表示方法使用独热表示，其中向量长度等于词汇表大小，且只有对应单词的维度为 1 [42]。此表示不编码词的语义，因为各词的向量彼此正交。使用独热词表示的文档也因表示极度稀疏而遭受维度灾难 [3]。

By contrast, a distributed representation of a word represents the word across multiple dimensions in a continuous-valued vector (word embedding) [4]. Such distributed representations can capture useful syntactic and semantic properties of the words, where syntactically and semantically related words are similar in this vector space [42]. Modern word embedding approaches usually use the language modeling task. For example, Word2Vec [32] learns the embedding of a word by predicting it based on its context (i.e., surrounding words), or predicting the context of a word given the word itself. GloVe [37] is similar to Word2Vec on a high level, but focuses on the likelihood that each word appears in the context of other words with in the whole corpus of texts, as opposed to Word2Vec which uses local contexts. More recent work such as ELMo [38] and BERT [11] allowed contextualized embedding. That is, the representation of a phrase can vary depending on a word's context to handle polysemy (i.e., the capacity for a word or phrase to have multiple meanings). For example, the word "bank" can have different meanings in "he withdrew money from the bank" versus "the river bank"

相比之下，分布式表示将单词跨多个维度表示为连续值向量 (词嵌入)[4]。此类分布式表示能捕捉有用的句法和语义属性，使句法或语义相关的词在该向量空间中相似 [42]。现代词嵌入方法通常使用语言建模任务。例如，Word2Vec [32] 通过根据上下文 (即周围词) 预测目标词，或根据目标词预测其上下文来学习词的嵌入。GloVe [37] 在高层次上与 Word2Vec 相似，但侧重于在整个语料库中某词出现在其他词上下文中的概率，而 Word2Vec 使用的是局部上下文。更新的工作如 ELMo [38] 和 BERT [11] 引入了上下文化嵌入，即短语的表示可随词的上下文变化以处理多义性 (即词或短语具有多重含义的能力)。例如，“bank”在“he withdrew money from the bank”与“the river bank”中具有不同含义。

While distributed representations are commonly used in natural language processing, to our best knowledge, the Screen2Vec approach presented in this paper is the first to seek to encode the semantics, the contexts, and the design patterns of GUI screens and components using distributed representations. The Screen2Vec approach is conceptually similar to Word2Vec on a high level-like Word2Vec, Screen2Vec is trained using a predictive modeling task where the context of a target entity (words in Word2Vec, GUI components and screens in Screen2Vec) is used to predict the entity (known as the continuous bag of words (CBOW) model in

Word2Vec). There are also other relevant Word2Vec-like approaches for embedding APIs based their usage in source code and software documentations (e.g., API2Vec [35]), and modeling the relationships between user tasks, system commands, and natural language descriptions in the same vector space (e.g., CommandSpace [1]).

尽管分布式表示在自然语言处理中被广泛使用，据我们所知，本论文提出的 Screen2Vec 方法是首个尝试使用分布式表示来编码 GUI 屏幕和组件的语义、上下文和设计模式的方法。从概念上看，Screen2Vec 在高层次上类似于 Word2Vec——像 Word2Vec 一样，Screen2Vec 通过预测建模任务训练，其中目标实体的上下文 (Word2Vec 中为词，Screen2Vec 中为 GUI 组件和屏幕) 被用来预测该实体 (在 Word2Vec 中称为连续词袋 (CBOW) 模型)。还有其他类似 Word2Vec 的相关方法，用于根据 API 在源代码和软件文档中的使用来嵌入 API(例如 API2Vec [35])，以及在同一向量空间中建模用户任务、系统命令与自然语言描述之间关系的方法 (例如 CommandSpace [1])。

Besides the domain difference between our Screen2Vec model and Word2Vec and its follow-up work, Screen2Vec uses both a (pre-trained) text embedding vector and a class type vector, and combines them with a linear layer. It also incorporates external app-specific meta-data such as the app store description. The hierarchical approach allows Screen2Vec to compute a screen embedding with the embeddings of the screen's GUI components, as described in Section 2. In comparison, Word2Vec only computes word embed-dings using word contexts without using any other meta-data [32].

除了 Screen2Vec 与 Word2Vec 及其后续工作之间的领域差异外，Screen2Vec 使用了 (预训练的) 文本嵌入向量和类类型向量，并通过线性层将它们结合。它还加入了应用商店描述等外部应用特定元数据。层次化方法使 Screen2Vec 能够如第 2 节所述，使用屏幕 GUI 组件的嵌入来计算屏幕嵌入。相比之下，Word2Vec 仅使用词上下文来计算词嵌入，未使用任何其他元数据 [32]。

6.2 Modeling GUI Interactions

6.2 建模 GUI 交互

Screen2Vec is related to prior research on computationally modeling app GUIs and the GUI interactions of users. The interaction mining approach [10] captures the static (UI layout, visual features) and dynamic (user flows) parts of an app's design from a large corpus of user interaction traces with mobile apps, identifies 23 common flow types (e.g., adding, searching, composing), and can classify the user's GUI interactions into these flow types. A similar approach was also used to learn the design semantics of mobile apps, classifying GUI elements into 25 types of GUI components, 197 types of text buttons, and 135 types of icon classes [30]. App-stract [12] focused on the semantic entities (e.g., music, movie, places) instead, extracting entities, their properties, and relevant actions from mobile app GUIs. These approaches use a smaller number of discrete types of flows, GUI elements, and entities to represent GUI screens and their components, while our Screen2Vec uses continuous embedding in a vector space for screen representation.

Screen2Vec 与以往对应用 GUI 及用户 GUI 交互进行计算建模的研究相关。交互挖掘方法 [10] 从大量移动应用的用户交互轨迹中捕捉应用设计的静态 (UI 布局、视觉特征) 和动态 (用户流程) 部分，识别出 23 种常见流程类型 (例如，添加、搜索、撰写)，并能将用户的 GUI 交互归类到这些流程类型中。类似的方法也被用于学习移动应用的设计语义，将 GUI 元素分类为 25 种 GUI 组件、197 种文本按钮和 135 种图标类别 [30]。App-stract [12] 则关注语义实体 (例如，音乐、电影、地点)，从移动应用 GUI 中抽取实体、其属性及相关操作。这些方法使用较少的离散流程类型、GUI 元素和实体来表示 GUI 屏幕及其组件，而我们的 Screen2Vec 则使用向量空间中的连续嵌入来表示屏幕。

Some prior techniques specifically focus on the visual aspect of GUIs. The RICO dataset [9] shows that it is feasible to train a GUI layout embedding with a large screen corpus, and retrieve screens with similar layouts using such embeddings. Chen et al.'s work [8] and Li et al.'s work [29] show that a model can predict semantically meaningful alt-text labels for GUI components based on their visual icon. Screen2Vec provides a more holistic representation of GUI screens by encoding textual content, GUI component class types, and app-specific meta-data in addition to the visual layout.

有些先前技术专注于 GUI 的视觉方面。RICO 数据集 [9] 显示在大规模屏幕语料上训练 GUI 布局嵌入并使用该嵌入检索相似布局屏幕是可行的。Chen 等 [8] 与 Li 等 [29] 的工作表明，模型可以基于视觉图标为 GUI 组件预测语义上有意义的替代文本标签。Screen2Vec 则通过对视觉布局之外的文本内容、GUI 组件类别类型和应用特定元数据进行编码，提供更全面的 GUI 屏幕表示。

Another category of work in this area focuses on predicting GUI actions for completing a task objective. Pasupat et al.'s work [36] maps the user's natural language commands to target elements on web GUIs. Li et al.'s work [28] goes a step further by generating sequences of actions based on natural language commands. These works use a supervised approach that requires a large amount of manually-annotated training data, which limits its utilization. In comparison, Screen2Vec uses a self-supervised approach that does not require any manual data annotation of user intents and tasks. Screen2Vec also does not require any annotations of the GUI screens themselves, unlike [46] which requires additional developer annotations as meta-data for GUI components.

另一类工作侧重于预测为完成任务目标而需执行的 GUI 操作。Pasupat 等 [36] 将用户的自然语言命令映射到网页 GUI 上的目标元素。Li 等 [28] 更进一步，根据自然语言命令生成动作序列。这些工作采用监督方法，需要大量人工标注的训练数据，限制了其应用。相比之下，Screen2Vec 使用自监督方法，不需要任何关于用户意图和任务的人工数据标注。Screen2Vec 也不需要对 GUI 屏幕本身进行任何注释，不像 [46] 需要开发者为 GUI 组件额外注释元数据。

6.3 Interactive Task Learning

6.3 交互式任务学习

Understanding and representing GUIs is a central challenge in GUI-based interactive task learning (ITL). When the user demonstrates a task in an app, the system needs to understand the user's action in the context of the underlying app GUIs so that it can generalize what it has learned to future task contexts [23]. For example, SUGILITE represents each app screen as a graph where each GUI component is an entity [24]. Properties of GUI components, their hierarchical relations, and the spatial layouts are represented as edges in the

graph. This graph representation allows grounding natural language instructions to GUIs [23, 24] with graph queries, allowing a more natural end user development experience [33]. It also supports personal information anonymization on GUIs [21]. However, this graph representation is difficult to aggregate or compare across different screens or apps. Its structure also does not easily fit into common machine learning techniques for computationally modeling the GUI tasks. As a result, the procedure generalization capability of systems like SUGILITE is limited to parameters within the same app and the same set of screens.

理解和表示 GUI 是基于 GUI 的交互式任务学习 (ITL) 中的核心挑战。当用户在应用中示范一个任务时，系统需要在底层应用 GUI 的语境中理解用户的操作，以便将所学推广到未来的任务情境 [23]。例如，SUGILITE 将每个应用屏幕表示为图，其中每个 GUI 组件是一个实体 [24]。GUI 组件的属性、层级关系和空间布局被表示为图中的边。该图表示允许通过图查询将自然语言指令落地到 GUI [23, 24]，从而提供更自然的终端用户开发体验 [33]，并支持 GUI 上的个人信息匿名化 [21]。然而，该图表示难以在不同屏幕或应用间聚合或比较，其结构也不易适配常见的用于计算建模 GUI 任务的机器学习技术。因此，像 SUGILITE 这样的系统在过程泛化能力上仅限于同一应用和同一组屏幕内的参数。

Some other interactive task learning systems such as VASTA [40], Sikuli [44], and HILC [14] represent GUI screens visually. This approach performs segmentation and classification on the video of the user performing GUI actions to extract visual representations (e.g., screenshot segments/icons) of GUI components, allowing replay of actions by identifying target GUI components using computer vision object recognition techniques. This approach supports generalization based on visual similarity (e.g., perform an action on all PDF files in a file explorer because they all have visually similar icons). However, this visual approach is limited by its lack of semantic understanding of the GUI components. For example, the icon of a full trash bin is quite different from an that of an empty one pixel count wise, but they should have the same meaning when the user intent is ”open the trash bin”. The icon for a video file can be similar to that of an audio file (with the only difference being the tiny ”mp3” and ”mp4” at a corner), but the system should differentiate them in intents like ”select all the video files”.

其他一些交互式任务学习系统如 VASTA [40]、Sikuli [44] 和 HILC [14] 则以视觉方式表示 GUI 屏幕。这种方法对用户执行 GUI 操作的视频进行分割和分类，从而提取 GUI 组件的视觉表示 (例如截图片段/图标)，并通过计算机视觉对象识别技术识别目标 GUI 组件以重放操作。该方法支持基于视觉相似性的泛化 (例如在文件管理器中对所有 PDF 文件执行同一操作，因为它们的图标在视觉上相似)。但这种视觉方法受限于对 GUI 组件语义理解的缺乏。例如，从像素计数看，满垃圾桶图标与空垃圾桶图标差别较大，但在用户意图为“打开回收站”时它们应具有相同含义。又如视频文件的图标可能与音频文件相似 (仅在角落有很小的“mp3”或“mp4”区别)，但在“选择所有视频文件”之类的意图中系统应能区分它们。

The Screen2Vec representation presented in this paper encodes the textual content, visual layout and design patterns, and app-specific context of GUI screens in a distributed vector form that can be used across different apps and task domains. We think this representation can be quite useful in supplementing the existing graph and visual GUI representations in ITL systems. For example, as shown in Section 3.3, sequences of Screen2Vec screen embedding can represent tasks in a way that allows the comparison and retrieval of similar tasks among different apps. The results in Section 3.3 also suggest that the embedding can help an ITL agent transfer procedures learned from one app to another.

本文提出的 Screen2Vec 表示将 GUI 屏幕的文本内容、视觉布局与设计模式以及应用特有的上下文编码为可跨应用与任务域使用的分布式向量形式。我们认为该表示可有效补充现有 ITL 系统中的图与视觉 GUI 表示。例如，如第 3.3 节所示，Screen2Vec 屏幕嵌入序列可以以一种允许在不同应用间比较和检索相似任务的方式来表示任务。第 3.3 节的结果还表明，该嵌入可帮助 ITL 代理将从一个应用学到的操作流程迁移到另一个应用。

7 CONCLUSION

7 结论

We have presented Screen2Vec, a new self-supervised technique for generating distributed semantic representations of GUI screens and components using their textual content, visual design and layout patterns, and app meta-data. This new technique has been shown to be effective in downstream tasks such as nearest neighbor retrieval, composability-based retrieval, and representing mobile tasks. Screen2Vec addresses an important gap in computational HCI research, and could be utilized for enabling and enhancing interactive systems in task learning (e.g., [25, 40]), intelligent suggestive interfaces (e.g., [7]), assistive tools (e.g., [5]), and GUI design aids (e.g., [17, 41]).

我们提出了 Screen2Vec，一种利用文本内容、视觉设计与布局模式以及应用元数据自监督生成 GUI 屏幕与组件分布式语义表示的新方法。该方法在最近邻检索、基于可组合性的检索和移动任务表示等下游任务中表现有效。Screen2Vec 弥补了计算人机交互研究中的重要空白，可用于支持与增强任务学习中的交互式系统 (例如 [25, 40])、智能建议界面 (例如 [7])、辅助工具 (例如 [5]) 以及 GUI 设计辅助 (例如 [17, 41])。

ACKNOWLEDGMENTS

致谢

This research was supported in part by Verizon through the Yahoo! InMind project, a J.P. Morgan Faculty Research Award, Google Cloud Research Credits, NSF grant IIS-1814472, and AFOSR grant FA95501710218. Any opinions, findings or recommendations expressed here are those of the authors and do not necessarily reflect views of the sponsors. We would like to thank our anonymous reviewers for their feedback and Ting-Hao (Kenneth) Huang, Monica Lam, Vanessa Hu, Michael Xieyang Liu, Haojian Jin, and Franklin Mingzhe Li for useful discussions.

本研究部分由 Verizon(通过 Yahoo! InMind 项目)、J.P. Morgan 教授研究奖、Google Cloud 研究额度、NSF 资助 IIS-1814472 以及 AFOSR 资助 FA95501710218 支持。此处所表达的任何观点、发现或建议均为作者个人观点，并不必然代表资助方意见。我们感谢匿名审稿人的反馈以及 Ting-Hao (Kenneth) Huang、Monica Lam、Vanessa Hu、Michael Xieyang Liu、Haojian Jin 和 Franklin Mingzhe Li 的有益讨论。

REFERENCES

参考文献

[1] Eytan Adar, Mira Dontcheva, and Gierad Laput. 2014. CommandSpace: Modeling the Relationships Between Tasks, Descriptions and Features. In Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology (UIST '14). ACM, New York, NY, USA, 167-176. https://doi.org/10.1145/2642918.2647395

[1] Eytan Adar, Mira Dontcheva, and Gierad Laput. 2014. CommandSpace: Modeling the Relationships Between Tasks, Descriptions and Features. In Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology (UIST '14). ACM, New York, NY, USA, 167-176. https://doi.org/10.1145/2642918.2647395

[2] Tanzirul Azim, Oriana Riva, and Suman Nath. 2016. uLink: Enabling User-Defined Deep Linking to App Content. In Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '16). ACM, New York, NY, USA, 305-318. https://doi.org/10.1145/2906388.2906416

[2] Tanzirul Azim, Oriana Riva, and Suman Nath. 2016. uLink: Enabling User-Defined Deep Linking to App Content. In Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '16). ACM, New York, NY, USA, 305-318. https://doi.org/10.1145/2906388.2906416

[3] Richard Bellman. 1966. Dynamic Programming. Science 153, 3731 (1966), 34-37. https://doi.org/10.1126/science.153.3

[3] Richard Bellman. 1966. Dynamic Programming. Science 153, 3731 (1966), 34-37. https://doi.org/10.1126/science.153.3731.34

[4] Yoshua Bengio. 2009. Learning deep architectures for AI. Now Publishers Inc.

[4] Yoshua Bengio. 2009. Learning deep architectures for AI. Now Publishers Inc.

[5] Jeffrey P. Bigham, Tessa Lau, and Jeffrey Nichols. 2009. Trailblazer: Enabling Blind Users to Blaze Trails through the Web. In Proceedings of the 14th International Conference on Intelligent User Interfaces (Sanibel Island, Florida, USA) (IUI '09). ACM, New York, NY, USA, 177-186. https://doi.org/10.1145/1502650.1502677

[5] Jeffrey P. Bigham, Tessa Lau, and Jeffrey Nichols. 2009. Trailblazer: Enabling Blind Users to Blaze Trails through the Web. In Proceedings of the 14th International Conference on Intelligent User Interfaces (Sanibel Island, Florida, USA) (IUI '09). ACM, New York, NY, USA, 177-186. https://doi.org/10.1145/1502650.1502677

[6] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. ACL, Lisbon, Portugal, 632-642. https://doi.org/10.18653/v1/D15-1075

[6] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. ACL, Lisbon, Portugal, 632-642. https://doi.org/10.18653/v1/D15-1075

[7] Fanglin Chen, Kewei Xia, Karan Dhabalia, and Jason I. Hong. 2019. MessageOn-Tap: A Suggestive Interface to Facilitate Messaging-Related Tasks. In Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (Glasgow, Scotland Uk) (CHI '19). ACM, New York, NY, USA, Article 575, 14 pages. https://doi.org/10.1145/3290605.3300805

[7] Fanglin Chen, Kewei Xia, Karan Dhabalia, and Jason I. Hong. 2019. MessageOn-Tap: A Suggestive Interface to Facilitate Messaging-Related Tasks. In Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (Glasgow, Scotland Uk) (CHI '19). ACM, New York, NY, USA, Article 575, 14 pages. https://doi.org/10.1145/3290605.3300805

[8] Jieshan Chen, Chunyang Chen, Zhenchang Xing, Xiwei Xu, Liming Zhu, Guo-qiang Li, and Jinshui Wang. 2020. Unblind Your Apps: Predicting Natural-Language Labels for Mobile GUI Components by Deep Learning. In Proceedings of the 42nd International Conference on Software Engineering (ICSE '20).

[8] Jieshan Chen, Chunyang Chen, Zhenchang Xing, Xiwei Xu, Liming Zhu, Guo-qiang Li, and Jinshui Wang. 2020. Unblind Your Apps: Predicting Natural-Language Labels for Mobile GUI Components by Deep Learning. In Proceedings of the 42nd International Conference on Software Engineering (ICSE '20).

[9] Biplab Deka, Zifeng Huang, Chad Franzen, Joshua Hibschman, Daniel Afergan, Yang Li, Jeffrey Nichols, and Ranjitha Kumar. 2017. Rico: A Mobile App Dataset for Building Data-Driven Design Applications. In Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology (UIST '17). ACM, New York, NY, USA, 845-854. https://doi.org/10.1145/3126594.3126651

[9] Biplab Deka, Zifeng Huang, Chad Franzen, Joshua Hibschman, Daniel Afergan, Yang Li, Jeffrey Nichols, and Ranjitha Kumar. 2017. Rico: 一个用于构建数据驱动设计应用的移动应用数据集. In Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology (UIST '17). ACM, New York, NY, USA, 845-854. https://doi.org/10.1145/3126594.3126651

[10] Biplab Deka, Zifeng Huang, and Ranjitha Kumar. 2016. ERICA: Interaction Mining Mobile Apps. In Proceedings of the 29th Annual Symposium on User Interface Software and Technology (UIST '16). ACM, New York, NY, USA, 767-776. https: //doi.org/10.1145/2984511.2984581

[10] Biplab Deka, Zifeng Huang, and Ranjitha Kumar. 2016. ERICA: 交互挖掘移动应用. In Proceedings of the 29th Annual Symposium on User Interface Software and Technology (UIST '16). ACM, New York, NY, USA, 767-776. https: //doi.org/10.1145/2984511.2984581

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,

Volume 1 (Long and Short Papers). ACL, Minneapolis, Minnesota, 4171-4186. https://doi.org/10.18653/v1/N19-1423

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: 用于语言理解的深度双向变换器预训练. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). ACL, Minneapolis, Minnesota, 4171-4186. https://doi.org/10.18653/v1/N19-1423

[12] Earlence Fernandes, Oriana Riva, and Suman Nath. 2016. Appstract: On-the-fly App Content Semantics with Better Privacy. In Proceedings of the 22Nd Annual International Conference on Mobile Computing and Networking (MobiCom '16). ACM, New York, NY, USA, 361-374. https://doi.org/10.1145/2973750.2973770

[12] Earlence Fernandes, Oriana Riva, and Suman Nath. 2016. Appstract: 实时应用内容语义与更佳隐私保护. In Proceedings of the 22Nd Annual International Conference on Mobile Computing and Networking (MobiCom '16). ACM, New York, NY, USA, 361-374. https://doi.org/10.1145/2973750.2973770

[13] Forrest Huang, John F. Canny, and Jeffrey Nichols. 2019. Swire: Sketch-Based User Interface Retrieval. In Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (Glasgow, Scotland Uk) (CHI '19). ACM, New York, NY, USA, 1-10. https://doi.org/10.1145/3290605.3300334

[13] Forrest Huang, John F. Canny, and Jeffrey Nichols. 2019. Swire: 基于草图的用户界面检索. In Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (Glasgow, Scotland Uk) (CHI '19). ACM, New York, NY, USA, 1-10. https://doi.org/10.1145/3290605.3300334

[14] Thanapong Intharah, Daniyar Turmukhambetov, and Gabriel J. Brostow. 2019. HILC: Domain-Independent PbD System Via Computer Vision and Follow-Up Questions. ACM Trans. Interact. Intell. Syst. 9, 2-3, Article 16 (March 2019), 27 pages. https://doi.org/10.1145/3234508

[14] Thanapong Intharah, Daniyar Turmukhambetov, and Gabriel J. Brostow. 2019. HILC: 通过计算机视觉与后续提问实现领域无关的基于示例编程系统. ACM Trans. Interact. Intell. Syst. 9, 2-3, Article 16 (March 2019), 27 pages. https://doi.org/10.1145/3234508

[15] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1412.6980

[15] Diederik P. Kingma and Jimmy Ba. 2015. Adam: 一种用于随机优化的方法. In 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1412.6980

[16] Ranjitha Kumar, Arvind Satyanarayan, Cesar Torres, Maxine Lim, Salman Ahmad, Scott R. Klemmer, and Jerry O. Talton. 2013. Webzeitgeist: Design Mining the Web. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (Paris, France) (CHI '13). ACM, New York, NY, USA, 3083-3092. https: //doi.org/10.1145/2470654.2466420

[16] Ranjitha Kumar, Arvind Satyanarayan, Cesar Torres, Maxine Lim, Salman Ahmad, Scott R. Klemmer, and Jerry O. Talton. 2013. Webzeitgeist: 网络的设计挖掘. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (Paris, France) (CHI '13). ACM, New York, NY, USA, 3083-3092. https: //doi.org/10.1145/2470654.2466420

[17] Chunggi Lee, Sanghoon Kim, Dongyun Han, Hongjun Yang, Young-Woo Park, Bum Chul Kwon, and Sungahn Ko. 2020. GUIComp: A GUI Design Assistant with Real-Time, Multi-Faceted Feedback. In Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems (Honolulu, HI, USA) (CHI '20). ACM, New York, NY, USA, 1-13. https://doi.org/10.1145/3313831.3376327

[17] Chunggi Lee, Sanghoon Kim, Dongyun Han, Hongjun Yang, Young-Woo Park, Bum Chul Kwon, and Sungahn Ko. 2020. GUIComp: 一款提供实时、多方面反馈的 GUI 设计助手. In Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems (Honolulu, HI, USA) (CHI '20). ACM, New York, NY, USA, 1-13. https://doi.org/10.1145/3313831.3376327

[18] Hsin-Ying Lee, Weilong Yang, Lu Jiang, Madison Le, Irfan Essa, Haifeng Gong, and Ming-Hsuan Yang. 2020. Neural Design Network: Graphic Layout Generation with Constraints. European Conference on Computer Vision (ECCV) (2020).

[18] Hsin-Ying Lee, Weilong Yang, Lu Jiang, Madison Le, Irfan Essa, Haifeng Gong, and Ming-Hsuan Yang. 2020. Neural Design Network: 在约束下的图形布局生成. European Conference on Computer Vision (ECCV) (2020).

[19] Jianan Li, Jimei Yang, Aaron Hertzmann, Jianming Zhang, and Tingfa Xu. 2019. LayoutGAN: Synthesizing Graphic Layouts with Vector-Wireframe Adversarial Networks. IEEE Transactions on Pattern Analysis and Machine Intelligence (2019).

[19] 李佳楠, 杨洁美, Aaron Hertzmann, 张建明, 许廷发. 2019. LayoutGAN: 用向量线框对抗网络合成图形布局. IEEE Transactions on Pattern Analysis and Machine Intelligence (2019).

[20] Toby Jia-Jun Li, Amos Azaria, and Brad A. Myers. 2017. SUGILITE: Creating Multimodal Smartphone Automation by Demonstration. In Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17). ACM, New York, NY, USA, 6038-6049. https://doi.org/10.1145/3025453.3025483

[20] Toby Jia-Jun Li, Amos Azaria, Brad A. Myers. 2017. SUGILITE: 通过示范创建多模态手机自动化. 载于 Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17). ACM, New York, NY, USA, 6038-6049. https://doi.org/10.1145/3025453.3025483

[21] Toby Jia-Jun Li, Jingya Chen, Brandon Canfield, and Brad A. Myers. 2020. Privacy-Preserving Script Sharing in GUI-Based Programming-by-Demonstration Systems. Proc. ACM Hum.-Comput. Interact. 4, CSCW1, Article 060 (May 2020), 23 pages. https://doi.org/10.1145/3392869

[21] Toby Jia-Jun Li, Jingya Chen, Brandon Canfield, Brad A. Myers. 2020. 在基于 GUI 的编程即示范系统中隐私保护的脚本共享. Proc. ACM Hum.-Comput. Interact. 4, CSCW1, Article 060 (May 2020), 23 页. https://doi.org/10.1145/3392869

[22] Toby Jia-Jun Li, Jingya Chen, Haijun Xia, Tom M. Mitchell, and Brad A. Myers. 2020. Multi-Modal Repairs of Conversational Breakdowns in Task-Oriented Dialogs. In Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology (UIST 2020). ACM. https://doi.org/10.1145/3379337.3415820

[22] Toby Jia-Jun Li, Jingya Chen, Haijun Xia, Tom M. Mitchell, Brad A. Myers. 2020. 面向任务对话中会话中断的多模态修复. 载于 Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology (UIST 2020). ACM. https://doi.org/10.1145/3379337.3415820

[23] Toby Jia-Jun Li, Igor Labutov, Xiaohan Nancy Li, Xiaoyi Zhang, Wenze Shi, Tom M. Mitchell, and Brad A. Myers. 2018. APPINITE: A Multi-Modal Interface for Specifying Data Descriptions in Programming by Demonstration Using Verbal Instructions. In Proceedings of the 2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC 2018). https://doi.org/10.1109/VLHCC.2018.8506506

[23] Toby Jia-Jun Li, Igor Labutov, Xiaohan Nancy Li, Xiaoyi Zhang, Wenze Shi, Tom M. Mitchell, Brad A. Myers. 2018. APPINITE: 在编程即示范中使用口头指令指定数据描述的多模态界面. 载于 Proceedings of the 2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC 2018). https://doi.org/10.1109/VLHCC.2018.8506506

[24] Toby Jia-Jun Li, Tom Mitchell, and Brad Myers. 2020. Interactive Task Learning from GUI-Grounded Natural Language Instructions and Demonstrations. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations. ACL, Online, 215-223. https://doi.org/10.18653/v1/2020.acl-demos.25

[24] Toby Jia-Jun Li, Tom Mitchell, Brad Myers. 2020. 从基于 GUI 的自然语言指令和示范中交互式学习任务. 载于 Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations. ACL, Online, 215-223. https://doi.org/10.18653/v1/2020.acl-demos.25

[25] Toby Jia-Jun Li, Marissa Radensky, Justin Jia, Kirielle Singarajah, Tom M. Mitchell, and Brad A. Myers. 2019. PUMICE: A Multi-Modal Agent that Learns Concepts and Conditionals from Natural Language and Demonstrations. In Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology (UIST 2019). ACM. https://doi.org/10.1145/3332165.3347899

[25] Toby Jia-Jun Li, Marissa Radensky, Justin Jia, Kirielle Singarajah, Tom M. Mitchell, Brad A. Myers. 2019. PUMICE: 一个从自然语言和示范中学习概念与条件的多模态代理. 载于 Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology (UIST 2019). ACM. https://doi.org/10.1145/3332165.3347899

[26] Toby Jia-Jun Li and Oriana Riva. 2018. KITE: Building conversational bots from mobile apps. In Proceedings of the 16th ACM International Conference on Mobile Systems, Applications, and Services (MobiSys 2018). ACM. https://doi.org/10.1145/ 3210240.3210339

[26] Toby Jia-Jun Li, Oriana Riva. 2018. KITE: 从移动应用构建会话机器人. 载于 Proceedings of the 16th ACM International Conference on Mobile Systems, Applications, and Services (MobiSys 2018). ACM. https://doi.org/10.1145/ 3210240.3210339

[27] Toby Jia-Jun Li, Shilad Sen, and Brent Hecht. 2014. Leveraging Advances in Natural Language Pro-

cessing to Better Understand Tobler's First Law of Geography. In Proceedings of the 22Nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL '14). ACM, New York, NY, USA, 513-516. https://doi.org/10.1145/2666310.2666493

[27] Toby Jia-Jun Li, Shilad Sen, Brent Hecht. 2014. 利用自然语言处理进展更好地理解托布勒的地理第一定律. 载于 Proceedings of the 22Nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL '14). ACM, New York, NY, USA, 513-516. https://doi.org/10.1145/2666310.2666493

[28] Yang Li, Jiacong He, Xin Zhou, Yuan Zhang, and Jason Baldridge. 2020. Mapping Natural Language Instructions to Mobile UI Action Sequences. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. ACL, Online, 8198-8210. https://doi.org/10.18653/v1/2020.acl-main.729

[28] 杨立, 何家聪, 周昕, 张远, Jason Baldridge. 2020. 将自然语言指令映射到移动界面操作序列. 载于 Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. ACL, Online, 8198-8210. https://doi.org/10.18653/v1/2020.acl-main.729

[29] Yang Li, Gang Li, Luheng He, Jingjie Zheng, Hong Li, and Zhiwei Guan. 2020. Widget Captioning: Generating Natural Language Description for Mobile User Interface Elements. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). ACL, Online, 5495-5510. https://doi.org/ 10.18653/v1/2020.emnlp-main.443

[29] 杨立, 李刚, 何璐衡, 郑靖杰, 李宏, 管志伟. 2020. 控件说明: 为移动用户界面元素生成自然语言描述. 载于 Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). ACL, Online, 5495-5510. https://doi.org/ 10.18653/v1/2020.emnlp-main.443

[30] Thomas F. Liu, Mark Craft, Jason Situ, Ersin Yumer, Radomir Mech, and Ranjitha Kumar. 2018. Learning Design Semantics for Mobile Apps. In Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology (Berlin, Germany) (UIST '18). ACM, New York, NY, USA, 569-579. https://doi.org/10.1145/324

[30] Thomas F. Liu, Mark Craft, Jason Situ, Ersin Yumer, Radomir Mech, and Ranjitha Kumar. 2018. 为移动应用学习设计语义。载于第 31 届年度 ACM 用户界面软件与技术研讨会论文集 (德国柏林)(UIST '18)。ACM，纽约，NY，USA，569-579。https://doi.org/10.1145/3242587.3242650

[31] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. arXiv:1301.3781 [cs] (Jan. 2013). http://arxiv.org/abs/1301.3781 arXiv: 1301.3781.

[31] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. 向量空间中词表示的高效估计。arXiv:1301.3781 [cs](2013 年 1 月)。http://arxiv.org/abs/1301.3781 arXiv: 1301.3781。

[32] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their com-positionality. In Advances in neural information processing systems. 3111- 3119. http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality

[32] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. 词和短语的分布式表示及其可组合性。载于《神经信息处理系统进展》。3111-3119。http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality

[33] Brad A. Myers, Amy J. Ko, Chris Scaffidi, Stephen Oney, YoungSeok Yoon, Kerry Chang, Mary Beth Kery, and Toby Jia-Jun Li. 2017. Making End User Development More Natural. In New Perspectives in End-User Development. Springer, Cham, 1-22. https://doi.org/10.1007/978-3-319-60291-2_1

[33] Brad A. Myers, Amy J. Ko, Chris Scaffidi, Stephen Oney, YoungSeok Yoon, Kerry Chang, Mary Beth Kery, and Toby Jia-Jun Li. 2017. 使终端用户开发更自然。载于《终端用户开发新视角》。Springer, Cham，1-22。https://doi.org/10.1007/978-3-319-60291-2_1

[34] Vinod Nair and Geoffrey E. Hinton. 2010. Rectified Linear Units Improve Restricted Boltzmann Machines. In Proceedings of the 27th International Conference on International Conference on Machine Learning (Haifa, Israel) (ICML'10). Omni-press, Madison, WI, USA, 807-814.

[34] Vinod Nair and Geoffrey E. Hinton. 2010. 修正线性单元改进受限玻尔兹曼机。载于第 27 届国际机器学习大会论文集 (以色列海法)(ICML'10)。Omni-press，Madison，WI，USA，807-814。

[35] Trong Duc Nguyen, Anh Tuan Nguyen, Hung Dang Phan, and Tien N. Nguyen. 2017. Exploring API Embedding for API Usages and Applications. In Proceedings of the 39th International Conference on Software Engineering (Buenos Aires, Argentina) (ICSE '17). IEEE, 438-449. https://doi.org/10.1109/ICSE.2017.47

[35] Trong Duc Nguyen, Anh Tuan Nguyen, Hung Dang Phan, and Tien N. Nguyen. 2017. 探索 API 嵌入用于 API 使用和应用。载于第 39 届国际软件工程大会论文集 (阿根廷布宜诺斯艾利斯)(ICSE '17)。IEEE，438-449。https://doi.org/10.1109/ICSE.2017.47

[36] Panupong Pasupat, Tian-Shun Jiang, Evan Liu, Kelvin Guu, and Percy Liang. 2018. Mapping natural language commands to web elements. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP '18). ACL, Brussels, Belgium, 4970-4976. https://doi.org/10.18653/v1/D18-1540

[36] Panupong Pasupat, Tian-Shun Jiang, Evan Liu, Kelvin Guu, and Percy Liang. 2018. 将自然语言命令映射到网页元素。载于 2018 年实证方法自然语言处理会议论文集 (EMNLP '18)。ACL，比利时布鲁塞尔，4970-4976。https://doi.org/10.18653/v1/D18-1540

[37] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP '14). ACL, Doha, Qatar, 1532-1543. https://doi.org/10.3115/v1/D14-1162

[37] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: 用于词表示的全局向量。载于 2014 年实证方法自然语言处理会议论文集 (EMNLP '14)。ACL，卡塔尔多哈，1532-1543。https://doi.org/10.3115/v1/D14-1162

[38] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In Proceedings of the 2018 Conference

of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers) (NAACL '18). ACL, New Orleans, Louisiana, 2227-2237. https://doi.org/10.18653/v1/N18-1202

[38] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. 深度上下文化词表示。载于 2018 年北美计算语言学学会分会会议论文集: 人类语言技术，卷 1(长论文)(NAACL '18)。ACL，新奥尔良，路易斯安那，2227-2237。https://doi.org/10.18653/v1/N18-1202

[39] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics. http://arxiv.org/abs/1908.10084

[39] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: 使用双胞胎 BERT 网络的句子嵌入。载于 2019 年实证方法自然语言处理会议论文集。计算语言学协会。http://arxiv.org/abs/1908.10084

[40] Alborz Rezazadeh Sereshkeh, Gary Leung, Krish Perumal, Caleb Phillips, Min-fan Zhang, Afsaneh Fazly, and Iqbal Mohomed. 2020. VASTA: a vision and language-assisted smartphone task automation system. In Proceedings of the 25th International Conference on Intelligent User Interfaces (IUI '20). 22-32.

[40] Alborz Rezazadeh Sereshkeh, Gary Leung, Krish Perumal, Caleb Phillips, Min-fan Zhang, Afsaneh Fazly, and Iqbal Mohomed. 2020. VASTA: 一种视觉与语言辅助的智能手机任务自动化系统。载于第 25 届智能用户界面国际会议论文集 (IUI '20)。22-32。

[41] Amanda Swearngin, Mira Dontcheva, Wilmot Li, Joel Brandt, Morgan Dixon, and Amy J. Ko. 2018. Rewire: Interface Design Assistance from Examples. In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (Montreal QC, Canada) (CHI '18). ACM, New York, NY, USA, 1-12. https://doi.org/10.1145/3173574.3174078

[41] Amanda Swearngin, Mira Dontcheva, Wilmot Li, Joel Brandt, Morgan Dixon, and Amy J. Ko. 2018. Rewire: 基于示例的界面设计辅助。收录于 2018 年人因与计算系统会议论文集 (蒙特利尔 QC，加拿大)(CHI '18)。ACM，纽约，NY，USA，1-12。https://doi.org/10.1145/3173574.3174078

[42] Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word Representations: A Simple and General Method for Semi-Supervised Learning. In Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (Uppsala, Sweden) (ACL '10). ACL, USA, 384-394.

[42] Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. 词表示: 一种用于半监督学习的简单而通用的方法。收录于第 48 届计算语言学协会年会论文集 (乌普萨拉，瑞典)(ACL '10)。ACL，USA，384-394。

[43] Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers). ACL, New Orleans, Louisiana, 1112-1122. https://doi.org/10.18653/v1/N18-1101

[43] Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. 面向通过推理进行句子理解的广覆盖挑战语料库。收录于 2018 年北美计算语言学协会人类语言技术分会会议论文集，第 1 卷 (长篇论文)。ACL，新奥尔良，路易斯安那，1112-1122。https://doi.org/10.18653/v1/N18-1101

[44] Tom Yeh, Tsung-Hsiang Chang, and Robert C. Miller. 2009. Sikuli: Using GUI Screenshots for Search and Automation. In Proceedings of the 22Nd Annual ACM Symposium on User Interface Software and Technology (UIST '09). ACM, New York, NY, USA, 183-192. https://doi.org/10.1145/1622176.1622213

[44] Tom Yeh, Tsung-Hsiang Chang, and Robert C. Miller. 2009. Sikuli: 使用 GUI 屏幕截图进行搜索与自动化。收录于第 22 届年度 ACM 用户界面软件与技术研讨会论文集 (UIST '09)。ACM，纽约，NY，USA，183-192。https://doi.org/10.1145/1622176.1622213

[45] Xiaoyi Zhang, Anne Spencer Ross, Anat Caspi, James Fogarty, and Jacob O. Wobbrock. 2017. Interaction Proxies for Runtime Repair and Enhancement of Mobile Application Accessibility. In Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17). ACM, New York, NY, USA, 6024-6037. https://doi.org/10.1145/3025453.3025846

[45] Xiaoyi Zhang, Anne Spencer Ross, Anat Caspi, James Fogarty, and Jacob O. Wobbrock. 2017. 交互代理用于移动应用可访问性的运行时修复与增强。收录于 2017 年人因与计算系统会议论文集 (CHI '17)。ACM，纽约，NY，USA，6024-6037。https://doi.org/10.1145/3025453.3025846

[46] Xiaoyi Zhang, Anne Spencer Ross, and James Fogarty. 2018. Robust Annotation of Mobile Application Interfaces in Methods for Accessibility Repair and Enhancement. In Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology.

[46] Xiaoyi Zhang, Anne Spencer Ross, and James Fogarty. 2018. 移动应用界面在可访问性修复与增强方法中的稳健注释。收录于第 31 届年度 ACM 用户界面软件与技术研讨会论文集。