Screen2Words: Automatic Mobile UI Summarization with Multimodal Learning

Bryan Wang*
University of Toronto
Toronto, ON, Canada
bryanw@dgp.toronto.edu

Gang Li
Google Research
Mountain View, CA, USA
leebird@google.com

Xin Zhou
Google Research
Mountain View, CA, USA
zhouxin@google.com

Zhourong Chen
Google Research
Mountain View, CA, USA
zrchen@google.com

Tovi Grossman
University of Toronto
Toronto, ON, Canada
tovi@dgp.toronto.edu

Yang Li
Google Research
Mountain View, CA, USA
liyang@google.com

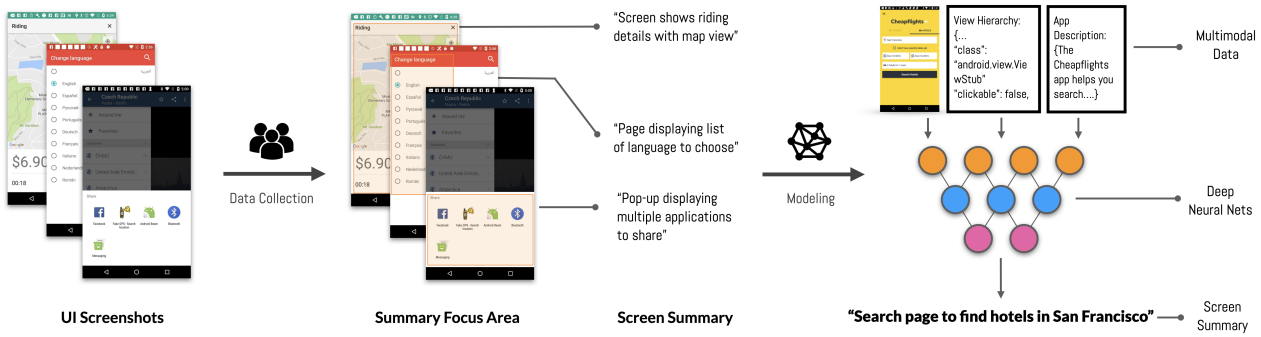


Figure 1: Screen2Words is a novel approach to automatically encapsulates essential information of a UI screen into a coherent language phrase. We collected the first large-scale screen summarization dataset, consisting of human annotations for 22,417 Android UI screens. We developed a set of deep learning models based on the dataset, leveraging the multi-modal data that a mobile screen carries. Our evaluation shows that our approach outperforms the heuristics baseline and is able to generate accurate summaries.

ABSTRACT

Mobile User Interface Summarization generates succinct language descriptions of mobile screens for conveying important contents and functionalities of the screen, which can be useful for many language-based application scenarios. We present Screen2Words, a novel screen summarization approach that automatically encapsulates essential information of a UI screen into a coherent language phrase. Summarizing mobile screens requires a holistic understanding of the multi-modal data of mobile UIs, including text, image, structures as well as UI semantics, motivating our multi-modal learning approach. We collected and analyzed a large-scale screen summarization dataset annotated by human workers. Our dataset contains more than 112k language summarization across ~22k unique UI screens. We then experimented with a set of deep models with different configurations. Our evaluation of these models with both automatic accuracy metrics and human rating shows that our approach can generate high-quality summaries for mobile screens. We demonstrate potential use cases of Screen2Words and open-source our dataset and model to lay the foundations for further bridging language and user interfaces.

CCS CONCEPTS

• **Human-centered computing** → **Human computer interaction (HCI)**.

KEYWORDS

Mobile UI summarization, screen understanding, deep learning, language-based UI, dataset.

*This work was completed while the author was an intern at Google Research.

UIST '21, October 10–14, 2021, Virtual Event, USA

ACM ISBN 978-1-4503-8635-7/21/10.
https://doi.org/10.1145/3472749.3474765

ACM Reference Format:
Bryan Wang, Gang Li, Xin Zhou, Zhourong Chen, Tovi Grossman, and Yang Li. 2021. Screen2Words: Automatic Mobile UI Summarization with Multimodal Learning. In *The 34th Annual ACM Symposium on User Interface Software and Technology (UIST '21), October 10–14, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3472749.3474765

1 INTRODUCTION

A mobile user interface screen often contains a rich set of graphical components from which a user can fulfill one or multiple functionalities. A succinct language description about a UI screen is useful for many language-based application scenarios. For example, a screen summary allows both conversational agents and end users to easily grasp the purpose and the state of a screen when accomplishing mobile tasks via language [25, 26]. A screen summary can be announced to help screen reader users quickly establish a mental model of an unknown mobile UI without waiting for the screen readers to scan through each element. [19, 37]. More broadly, representing a UI screen in language, which is highly flexible and versatile, opens many opportunities for combining the strengths of the two communication mediums. However, screen summaries are largely nonexistent in existing applications. It is difficult to compose these summaries manually because user interfaces are highly diverse and dynamic. As such, we focus on automatic screen summarization, a task that generates compact yet informative language representation from semantic understanding of UI screens.

Automatic screen summarization, while similar to image captioning [47] and text summarization [31], is unique in that it requires a holistic understanding of the multi-modal[1] data that a mobile screen carries. Existing work has investigated using deep learning to encode multi-modal UI data into vector representations [24], which has been shown useful for various downstream tasks such as UI retrieval [4, 16]. However, few efforts have been made to bridge the gap between the learned semantic representation and natural language that can be communicated with human users. On the other hand, recent work [28, 50] predicts alt-text labels for icons and widgets on mobile screens by considering multi-modal data sources, yet they focus on individual components and do not generate phrases that can describe the entire mobile UI screen, which is a more challenging task.

In this paper, we present *Screen2Words*, a novel generative approach for encapsulating complex information presented in a mobile UI into a succinct language description. To aid the development of Screen2Words, we collected the first large-scale screen summarization dataset, which consists of human annotations for 22,417 Android UI screens sampled from RICO [7], a public mobile screen corpus. Inspired by recent work on representing GUI semantics using deep learning [4, 16, 24, 28], we experimented with a set of deep models on the Screen2Words dataset to investigate the feasibility and effectiveness of our approach for screen summarization. We evaluated these models based on a set of accuracy metrics commonly used in image captioning and machine translation tasks. The result showed that all the deep models outperformed heuristic-based methods. A comparison between model variants showed that using multi-modal data sources about a UI screen leads to superior summarization accuracy. Our best results were achieved by using a combination of the Transformer encoder-decoder model [42] and ResNet [14], and by leveraging multiple data modalities including text, image, and structures of a UI. We then conducted a Mechanical Turk study to obtain human ratings on summaries generated by different model variants and heuristic baselines. The human evaluation also shows that our full model outperformed all other methods on subjective rating. Lastly, we discuss potential applications that could benefit from Screen2Words. Altogether, our paper makes the following contributions:

- We formulate mobile screen summarization, a novel task to automatically generate a descriptive language overview for mobile screens, which expands prior work for screen understanding and language generation by generating language descriptions for the entire mobile screen. The task has important implications for language-based interaction.
- We collect, analyze, and open-source[2] the first dataset dedicated for UI screen summarization. It contains 112,085 quality human annotations for 22,417 unique Android screens, collected with a carefully designed labeling process and guideline, which achieve a high inter-labeler agreement for both linguistic coherence and on-screen focus area consistency.
- We develop, train, and evaluate a set of deep models with automatic metrics and human evaluation. The results showed that our full model significantly outperforms all the heuristic baselines and model variants on both automatic metrics and subjective ratings, validating our approach with Screen2Words. The dataset, the models and the empirical results establish a solid benchmark for future research to bridge user interfaces and natural language.

2 RELATED WORK

Our work involves both dataset and model development, and builds upon several areas of existing work, including uni- and multi-modal content summarization, mobile screen understanding using deep learning, and mobile UI datasets.

2.1 Uni- and Multi-Modal Content Summarization

Automatic content summarization such as text document summarization [15, 33] and video captioning [5, 10, 44] has been widely investigated over the past decades, due to its abundant applications. State-of-the-art summarization techniques typically use deep learning models to encode the underlying representation of the content for generating short, informative text summaries. For example, text summarization [35, 49] generates concise summaries for large documents, while image captioning [45, 48] generates natural language captions to describe input images. Since many real-world data is by nature multi-modal, summarization techniques that use more than one data modality have also been extensively studied [11, 52, 53]. We formulate UI screen summarization as a multi-modal summarization task because it leverages input from multiple data sources, including screenshot images, texts, and structural information of mobile UIs. This task contributes to the spectrum of problems for automatic content summarization and can potentially enhance a class of language-based human-computer interaction problems.

[1]The term multi-modal in this context refers to mechanisms that encode information from different data sources, which is different from the common HCI definition of the user interacting with devices using multiple modalities.

[2]https://github.com/google-research/google-research/tree/master/screen2words

2.2 Mobile Screen Understanding using Deep Learning

There has been an increasing interest in the field [4, 16, 24, 27, 28, 41] for using deep models to learn the latent representation of mobile UIs, which we refer to as *screen understanding*. For example, Screen2Vec [24] uses a self-supervised approach to learn the representation of a mobile UI using the textual content, visual design and layout patterns of the screen, and its app meta-data. Screen understanding has been shown crucial for many downstream tasks. For instance, TapShoe [41] predicts whether a UI element is tappable by encoding the UI screenshot with CNNs. Similarly, VINS [4] encodes UI designs and wireframes to enable content-based UI retrieval; Swire [16] allows a designer to retrieve UI designs with sketching by encoding both the sketch and UI images. Widget Captioning [28] and Screen Recognition [50] predict semantically meaningful alt-text labels for GUI components. Screen2Words extends this line of work to predict language summaries for the entire mobile GUI, which requires a model to have a holistic understanding about a screen, and the ability to summarize complex screen contents as a concise language description.

2.3 Mobile GUI and Interaction Datasets

Large-scale mobile GUI data repositories are crucial building blocks for data-driven model development. The Rico dataset [7, 30] contains visual, textual, structural, and interactive design properties of 66k unique UI screens from 9.7k Android apps spanning 27 categories in the Google Play Store. ERICA [8] provides a collection of user interaction data for mobile UIs captured while using the app. Swire [16] and VINS [4] open-sourced the datasets used for training their UI retrieval models. Another category of work in this area contributes public datasets that connect mobile UIs with natural language for both language grounding and generation [27, 28]. Based on previous work, our work contributes the first open-sourced, large-scale dataset for mobile UI summarization with high-quality human annotations, detailed analysis and benchmark models.

3 DATASET CREATION

We start our investigation of methods for automatic screen summarization by creating a dataset, which results in 112,085 human-annotated English summarization for 22,417 unique UI screens. The dataset lays the foundation for data-driven model development for screen summarization. Below we first describe our data collection process and then report an analysis over the collected data.

3.1 Mobile UI Corpus

We started by constructing a mobile UI corpus consisting of screens from an opensource dataset Rico-SCA[3] [27]. It contains a subset of screens filtered from the Rico dataset [7] to eliminate screens with missing or inaccurate view hierarchies. In our corpus, each screen comes with a screenshot image of the UI and a view hierarchy JSON file. The view hierarchy is a structural tree representation of the UI where each node, corresponding to a UI element, contains various properties such as the class, visibility-to-user, and bounds of the element. In total, we labeled 22,417 unique screens from the filtered corpus of Rico-SCA.

[3]https://github.com/google-research-datasets/seq2act

3.2 Data Annotation

We recruited 85 professional labelers to generate the summarization annotation. The labelers were a group of contractors hired for data annotation to assist ML R&D in our company. All labelers are fluent in English and had previously labeled Android UIs for other tasks. For each screen in the corpus, we presented the screenshot image, task instructions, and the app description to the labeler and collected five annotations from five different labelers. Labelers entered their answers in a text field and were able to skip a screen if they found the screen not understandable. On average, each labeler spent around 50 hours creating the annotations. During labeling, a team of quality analysts sampled and audited around 5% of the the labels (both screen summaries and SFA) to ensure quality, and incorrectly labeled screens were re-labeled. We trained the labelers with golden examples, pilot data collection, and the following guidelines containing Do's and Don'ts:

Do's

- Summarization should contain 5 to 10 words.
- Focus on the most important functionalities.
- Use the texts on the screen to help summarization.
- Summarize with the structure of "NOUN + CLAUSE".

Don'ts

- Do not summarize only the images/icons, summarize the whole screen.
- Do not describe how you feel about the screen, focus on matter of fact.
- Do not just describe color, shapes and the name of UI elements.
- Do not mention the name of the app, describe with its category.

The guidelines were adapted from those used in the data collection of Microsoft COCO Captions [6], a well-known image captioning dataset. Our guidelines are designed to encourage labelers to focus on the functionality of the UI instead of the appearance of screenshots. We iteratively refined the guidelines and the labeling interface based on the results of our pilot studies. To understand the rationale of labelers for composing a screen summary, we also asked them to select a Summary Focus Area (SFA) upon summarizing the screen and entering the phrase. An SFA is an area on the screen covering UI elements that labelers deem are most informative for them to generate the summary. The SFA annotation focuses on the *summarization importance* for a mobile UI, which is different from the visual importance that previous work [12] focuses on. We instrumented the SFA to capture potential inconsistency of labeler focus we found in pilot data collection, e.g., the ads bar versus the main content. A labeler can annotate the SFAs simply by marking a bounding box on the UI screen via drag-n-drop. Figure 2 (a) shows two sample screens with their screen summaries and SFA annotated.

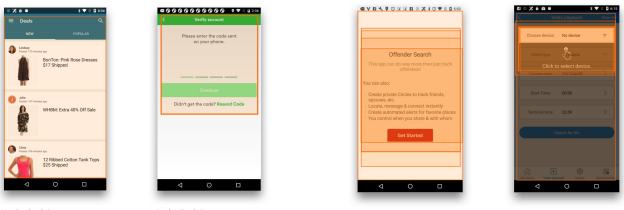


Figure 2: (a) Data annotations examples. Each screen is annotated with a language summary and a Summary Focus Area (visualized with orange rectangles) indicating where the labeler considered is most influential for their summary. (b) Overlay of 5 labelers' Summary Focus Area. Areas with the deepest colors represent the places where most labelers consider important when summarizing the screen.

3.3 Data Analysis

3.3.1 Summary Language Analysis. Our dataset contains 22,417 unique screens from 6,269 apps, resulting in 112,085 summary phrases composed by human workers. We remove a shortlist of stop phrases from collected summary phrases. These stop phrases contain eight variants of *"in the app"*, which often appear in a summary phrase but do not meaningfully contribute to summarization. The average phrase length after stop phrase removal is 6.57 words. Fig. 3 (a) shows the distribution of the summary length we collected. For each screen, to measure how consistent its summaries are across different labelers, we measure the inter-annotator agreement by computing the word-level precision and recall for all the words with two or more occurrences in the collected summaries. The same approach has been previously used in the COCO image captioning dataset [6]. The word-level precision and recall are computed using each word's true positive (TP), false positive (FP), and false negative (FN) counts accumulated across the corpus. For every screen summary, we check whether each of its words appears in the other four summary annotations for the same screen (TP) or not at all (FP). Any word presented only in the other four summaries but not in the one being checked is counted as an FN case. We iterated this calculation through each of the five summary labels for every screen to avoid sampling variance. This process yields corpus-wide accumulative TP/FP/FN counts for each word, which we use for precision and recall calculation. We focus our analysis on the top 4.5K words in the dataset, which appear more than once and amount to 99.7% of all the word occurrences in the summaries. Fig. 3 (b) shows that the collected summaries have a reasonable word-level agreement for each screen across different human labelers. Specifically, for the 4.5K words, we report the mean precision and recall of every 10 consecutive words in the vocabulary (sorted by word frequency). As a result, the figure contains 450 data points, each representing precision/recall of 10 words. The ranks of the words in the vocabulary are used to color the data points. Lower rank indicates higher word frequencies in the corpus.

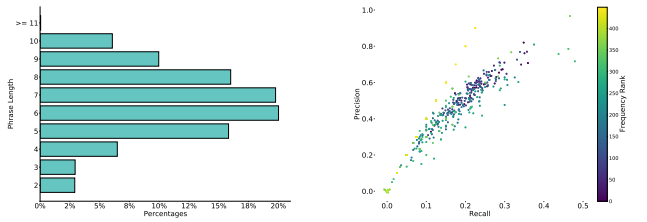


Figure 3: (a) The distribution of summary length of the collected annotations. (b) The distribution of precision and recall for the top 4.5K in the summaries. See section 3.3.1 for more details.

3.3.2 Summary Focus Area Analysis. The SFAs are approximate indicators of which parts of the UI the labelers focused on when performing summarization, and we use it to understand the perceived importance of UI elements on the screen across labelers for summarization purposes. On average, the SFAs cover 66.1% of the screen, which is within our expectation as the goal is to summarize the entire screen. We use the IoU score (Intersection over Union) to measure the agreement on the areas labeled by different labelers for the same screen. The average pair-wise IoU score is 74.1%, showing a decent consensus among labelers on which area on the screen was focused when producing summaries for a screen. The SFAs were initially introduced to account for potential variance in annotations. As we revised our data collection guidelines and labeling interface through iterations, we found the discrepancy between labelers' focus became less—their SFAs often contain most of the relevant UI elements on the screen. The consistency is desirable for an ML model, and SFA provides a safeguard and accountability for screen summarization consistency. Fig. 2 (b) shows two sample screens overlaid with the five SFAs annotated by different labelers. Areas with the deepest colors represent the parts where the most labelers consider important when summarizing the screen.

4 MODEL DESIGN

We utilize deep learning models to understand the challenges and feasibility of the proposed screen summarization task. We designed our models based on the encoder-decoder architecture, a commonly used architecture in image captioning and machine translation. In an encoder-decoder architecture, the model first encodes input data into hidden representations—referred to as encoding, and then decodes outputs based on the encoded information.

Figure 4 shows the architecture of our screen summarization model. Similar to models of Widget Captioning [28], we use a dual encoder for encoding multi-modal information of a screen. Our encoder consists of 1) a Transformer encoder to encode the UI structure of a screen along with the text description of the app that the screen belongs to, and 2) a ResNet to encode the image pixels of each element on the screen. The outputs of the two encoders are then combined via a late fusion, i.e., the concatenation of the outputs of the two encoders forms the encoding of each element. The resulted encodings of all the UI elements are the multi-modal semantic understanding of the screen, which is then fed to the

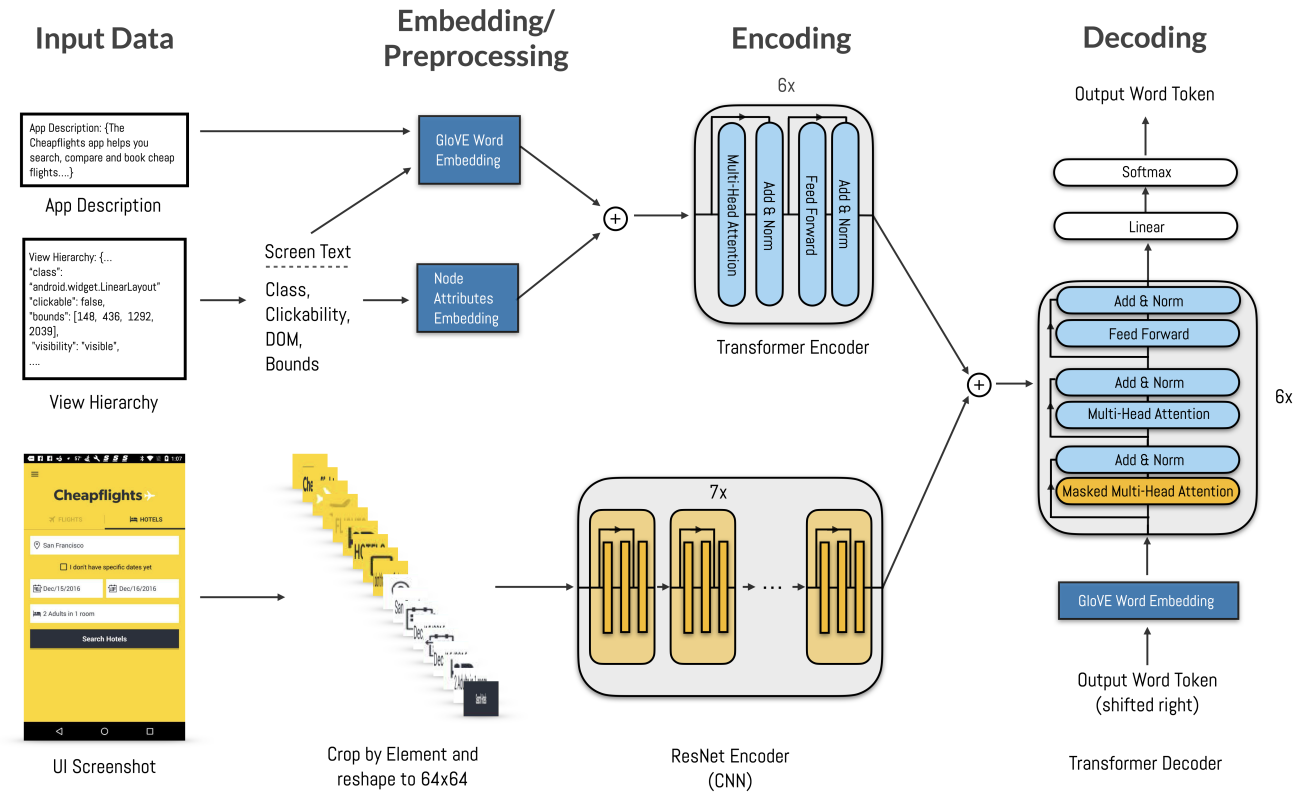


Figure 4: The Screen2Words deep model is designed based on the Transformer Encoder-Decoder architecture [42]. It leverages three input modalities, including the UI screenshot, the view hierarchy structure, and the screen text and app description. Each modality is encoded and then fused to participate in the decoding for summary generation.

Transformer decoder for generating the phrase for screen summarization. We next elaborate on each individual component of our model. Although many aspects of our model designs follow best practice, we would like to describe them here for completeness and reproducibility. Meanwhile, we highlight unique aspects in our design, such as incorporating app description as an additional "element" to participate in Transformer self-attention during encoding.

4.1 Encoding a Mobile UI Screen

To form a holistic understanding of a mobile UI screen, we encode both the structural-textual information in its view hierarchy and app description, and the raw pixels of its screenshot, using a dual encoder. The outputs of each encoder are then concatenated to form the multimodal encoding of each UI element on the screen.

4.1.1 Encoding Structural and Textual Information. We use a Transformer model [42] to encode both the structural and textual information of a mobile UI. The core idea behind the Transformer model is self-attention, which the model learns to represent each screen element by leveraging the information from all the elements coexisting on the screen via neural attention mechanisms. As a result, the approach allows us to acquire a contextual representation of each UI element on the screen. A Transformer encoder requires both content embedding and positional encoding as input, and we follow the approach that has been shown effective for embedding the mobile [27, 28]. We perform breadth-first traversal to flatten the view hierarchy tree and obtain the linear order of input elements required by the Transformer model. We compute the positional embedding using the elements' spatial positions on the screen and the tree (DOM) positions in the view hierarchy. Each element of the flattened tree carries both spatial and structural positional information.

Element Embeddings: We first need to represent each individual element on the UI as an embedding vector. An element comes with a rich set of properties. From view hierarchy, the *class*, the *clickability* and the *bounds* of an element respectively represent the type of the element, whether it is clickable, and its 2D positional information on the screen. The tree positions of the element, captured as the pre-order and post-order traversal position and its depth in the view hierarchy, reflect the structural position of the element on the UI. Both the spatial and tree positions of the element constitute the positional information of the element. Last but not the least, an element might come with text content, which is a good source of semantic information about the element. We embed each of these sources of information separately for an element. For the text content, we use pre-trained GloVe word embedding [36] to represent

each word and then perform sum pooling over all the words of the element to acquire a fixed-size vector for the element. For all other types of attributes, we treat them as categorical variables and embed them separately. The final embedding of an element is acquired via concatenation.

App Description Embedding: As described in our data collection section, we allow the human labelers to read the app description while performing screen summarization, because the app description provides the background information about the app that a screen belongs to. To train a computational model to achieve human intelligence in screen summarization, it is important to give the model the same access to this information. As such, we include the app description from Google Play Store as another source of input. Similar to representing the text content in an element, we first embed each word in an app description using pre-trained GloVe word embedding, and then simply treat the app description as a "bag of words" and acquire the final embedding of the app description using sum pooling over all the words. For screens with missing app descriptions, we assign an all-zero embedding embedding for an empty description. Note that more complex methods, such as LSTM, can be easily plugged into our model here for embedding app descriptions, although it is not the focus of this paper. The outcome of this process is to obtain a fixed-size embedding vector for the app description.

Transformer Encoding: With each element and the app description represented as an embedding vector, we linearly project (P) each embedding vector to a target dimension that is required by the Transformer encoder, as shown in Figure 4. Note that we treat the app description as a special "element" to participate in the Transformer encoding process. This design naturally allows each UI element to reference the app description embedding when forming the element's contextual representation, due to the self-attention mechanism of the Transformer encoder model. Specifically, we append the app description embedding to element embeddings, over the element dimension, which yields an embedding tensor, E, in the shape of `[num_of_elements + 1, embedding_size]`. To enable the Transformer encoder to differentiate the embedding vectors of a real UI element from that of the app description, we tag each embedding to indicate whether it is an element embedding or app description embedding, simply by appending a one-hot vector to the embedding before the linear projection P. The embedding tensor, E, was then fed to the Transformer encoder to produce the structural-textual encoding of the UI.

4.1.2 Encoding the UI Screenshot. To represent the visual aspect of a screen, we encode the image of each element on the screen. To do so, we crop the image of each element from the UI screenshot, and then re-scale the cropped image to a fixed-size tensor in the shape of `64x64x1`, where `64x64` are the spatial dimensions and `1` is the single channel for grayscale. We resize the images to ensure each element's image encoding has a fixed shape expected by the downstream computation. While resizing loses the aspect ratio, it makes our model more efficient in computation compared to padding element images to a larger target size. We use grayscale images because the semantics of most UI elements are color-invariant. Because the view hierarchy that specifies each element on the screen is given, it is sensible to directly use the cropped element image instead of asking the model to encode the entire screen. This design frees the image encoder from having to learn to extract UI elements from the screenshot pixels—an object detection problem—which by itself is a nontrivial task. We used ResNet, a multi-layer Convolutional Neural Net (CNN), to encode the pixel information of each UI element. The building block of our ResNet is a residual block consisting of three convolutional 2D layers with a residual connection−−the input of the 1st layer is added to the input of the 3rd layer. The last layer of each block uses a stride of 2 that halves both the vertical and horizontal spatial dimensions. The output of the multi-layer CNN represents the visual encoding of the elements on the screen. We then concatenate the structural-textual encoding, from the previous section, and the visual encoding to form the *final encoding* of each UI element. Note that because the structural-textual encoding has an extra element from the app description, we add a padding image encoding as its corresponding visual encoding. We used the padding encoding since the image encoding of each element is already made available to the model. An alternative approach is to use the image encoding of the entire screen.

4.2 Decoding Screen Summaries

The final encodings acquired from the previous section encompass both the structural-textual and visual information about each element on the screen. Such a representation is contextual as each encoding is computed by attending to other elements on the screen. Based on these encodings, we use a Transformer decoder [42] model to generate a varying-length natural language summary for the screen. Similar to the encoder, the Transformer decoder uses self-attention to attend to the context of the generated word tokens during summarization. To avoid information leaks from future tokens that have not been generated during supervised training, it uses masked self-attention to allow its multi-head attention to only attend to previous token representations. In addition, the Transformer decoder accesses the encoder's output, i.e., the encodings, for each step of the decoding process. This is achieved with the encoder-decoder attention layer. Internally, it adds the weighted sum of screen encodings to the attention output of each decoding step, before feeding into a position-wise, multi-layer perceptron (FFN). Unlike prior work focusing on captioning each individual element on the screen [28], our Transformer decoder attends to every element on the screen to decode a holistic screen summary. The probability distribution of each token of the summarization is finally computed using the softmax over the Transformer decoder output. The entire model, including both the encoders and decoder, is trained end-to-end, by minimizing $\mathcal{L}$, the average cross-entropy loss for decoding each token of each screen's summary:

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{M}\sum_{j=1}^{M}\text{Cross\_Entropy}(y'_{i,j}, y_{i,j}),$$

where M is the number of word tokens to decode and N is the number of screens in the mini batch, $y'_{i,j}$ is the jth token in the groundtruth summary and $y_{i,j}$ is the corresponding prediction. Training is conducted in a teacher-forcing manner where the groundtruth summary words are fed into the decoder. During prediction time, the model decodes autoregressively and beam search is used to generate the top summarization candidates.

Table 1: Experimental Dataset statistics

Dataset	#Apps	#Screens	#Summaries
Training	4,390	15,743	78,715
Validation	625	2,364	11,820
Test	1,254	4310	21,550
Total	6,269	22,417	112,085

5 EXPERIMENTS

We conducted screen summarization experiments to investigate the effectiveness of our proposed Screen2Words approach based on multi-modal deep learning. The goal of the experiments is to validate whether 1) deep models perform better than heuristics methods and 2) whether incorporating multiple data modalities would lead to better summarization results. We first discuss the experimental setup and training configurations. We then report the performance of our models against several commonly-used metrics and an analysis of the model behavior.

5.1 Experimental Datasets

We split our dataset into training, validation and test set for model development and evaluation, as shown in Table 1. To avoid information leaks because screens in the same app might share similar styles and semantics, we split the data app-wise so that all the screens from the same app will not be shared across different splits. Consequently, all the apps and screens in the test dataset are unseen during training, which allows us to examine how each model configuration generalizes to unseen conditions at the test. Our vocabulary includes 10,000 most frequent words and the rest of the words encountered in the training dataset is assigned a unique unknown token <UNK>. During validation and testing, any <UNK> in the decoded phrase is removed before evaluation. Since in our dataset, each screen has five summarization label phrases (from five different labelers), one of its labels is randomly sampled as the training target each time during stochastic training. During the testing phase, all the five screen summaries are used to form the reference set for automatic evaluation metrics such as BLEU and CIDEr.

5.2 Model Configurations and Training Details

We tuned our model architectures based on the training and validation datasets. We initialize the word embeddings with pre-trained 400K-vocab 300-dimensional GLOVE embeddings [36], which are then projected onto a 128-dimensional vector space. The embedding weights were shared by both the screen encoder and the decoder. Both the Transformer encoder and decoder use 6 Transformer layers with a hidden size of 128 and 8-head attention. A 7-layer ResNet was used for encoding the pixels of elements on the screen. The ResNet in total involves 21 convolutional layers, and the output of the final layer is flattened into a 256-sized vector. Batch normalization was used for each convolutional layer. The final encoding of each element's image is a 128-dimensional vector that is concatenated with the Transformer encoder's output for decoding. Each word during summarization is decoded sequentially, and we use beam search with a beam size of 5 to generate top-5 summarization predictions during testing. We implemented the models with TensorFlow, and trained the models with 8 Tesla V100 GPU cores using a batch size of 128 screens, and the Adam optimizer [18]. All the models we experiment with were converged in less than two days.

5.3 Model Variants and Baseline

To investigate the effectiveness of fusing multi-modal data representation for screen summarization, we compared different variants of our model:

- *PixelOnly*. Building upon the existing image captioning approach [32, 45, 48], this variant leverages only the visual information encoded by the multi-layer CNN from the UI element images to generate summaries.
- *LayoutOnly*. This variant uses the structural representation encoded with the Transformer encoder based on the UI properties of the elements extracted from the view hierarchy.
- *Pixel+Layout*. This variant uses both the image encoding and structural information.
- *Pixel+Layout+Text*. Based on Pixel+Layout, this model additionally uses the textual information encoded from the screen texts.
- *Pixel+Layout+Text+AppDesc*. This is our full model that uses all the above information plus the app descriptions.

Since there are no existing baseline techniques dedicated to automatic screen summarization, we create several template-based baselines for comparison in our experiment. The baselines predict the summary of an unseen screen—the query screen—by retrieving the summary of the screen, from the training dataset, that is most similar to the query screen—the nearest neighbor approach. In these baseline methods, we featurize each screen as either a TF-IDF score vector, which is well-known in the literature of text analysis, or a vector of pixel values, or a combination of the two.

The length of the TF-IDF vector is the vocabulary size. Each value in the vector corresponds to the TF-IDF score of a word in the vocabulary. Based on the original TF-IDF terminology, we treat each screen as a *document* and each word in the screen as a *term*. The Term-Frequency (TF) represents how often a word token appears in the screen, whereas Inverse Document Frequency (IDF) tells how rare the word is across screens in the training corpus. Together, a TF-IDF score measures how unique a word is to the screen. We implement the TF-IDF baseline using sklearn [3]. For the pixel-value vector, we convert the screen to grayscale and resize the screen to a dimension of 100×100 to match the TF-IDF vector size. Additionally, we include *Pixel-DL*, another pixel-based baseline that uses image encoding learned by a CNN-based autoencoder[4] to represent the pixel-value vector. With each screen represented as a TF-IDF vector and/or a pixel-value vector, we can then compute the cosine similarity scores $\in [0, 1]$ between a pair of screens, and find the most similar screens using either scores or the sum of the two scores. We investigate baseline variants using either or both vectors to match a query screen to screens in the training set, and report

[4]The Pixel-DL baseline uses a CNN-based autoencoder trained to recover the screenshot image with mean square loss. The model consists of 3 convolutional layers (kernel sizes: 128, 64, 32) and 3 transposed convolutional layers (kernel sizes: 32, 64, 128), all with a filter size of 3x3 and strides of 2. The latent state of a screenshot is a vector of size 100, which is used to retrieve the most similar screenshot and its human-labeled summary.

Table 2: Model Performance on Automatic Metrics

Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4	CIDEr	ROUGE-L	METOER
Template (TF-IDF)	47.9	29.3	20.7	16.5	31.8	36.5	21.1
Template (Pixel)	42.8	25.3	17.9	14.6	14.1	33.9	18.6
Template (Pixel-DL)	43.6	26.0	18.3	14.8	16.1	34.2	19.1
Template (TF-IDF+Pixel+AppDesc)	45.7	27.3	19.2	15.5	23.7	35.5	20.1
Template (TF-IDF+Pixel)	49.0	30.0	21.1	16.8	33.2	37.6	21.5
Pixel Only	56.8	37.0	25.3	19.9	31.4	42.6	25.6
Layout Only	58.7	39.6	27.4	21.7	35.2	44.3	26.0
Pixel+Layout	62.1	40.6	28.4	22.1	35.4	45.5	26.1
Pixel+Layout+ScreenText	63.6	43.5	30.9	23.9	55.5	47.4	29.0
Pixel+Layout+ScreenText+AppDesc	**65.5**	**45.8**	**32.4**	**25.1**	**61.3**	**48.6**	**29.5**

their performance in Table 2. The baselines allow us to understand the challenge of using a traditional nearest-neighbor or template-based retrieval method and the benefit of using a deep generative model for automatic screen summarization. In the following reports, we prefix each baseline method with *Template* because they are all based on template-based approaches.

5.4 Experimental Results

In this section, we report our model performance based on metrics commonly used in machine translation and image captioning tasks: BLEU [34], CIDEr [43], ROUGE-L [29], and METOER [9] (see Table 2). A higher number means better model performance for these metrics—the closer distances between the predicted and the ground truth phrases. For all model variants, the top-1 prediction by beam search is used for calculating metrics; for the template-based baselines, we sample one summarization from the most similar screen for its prediction. All the scores were calculated on the test dataset, which was not seen by the model during the training phase.

The results show that all the generative models outperformed all the template-based retrieval baselines across these metrics by a large margin, which justifies our proposed modeling approach. Among the baselines, using both TF-IDF and pixel-value vectors achieved the best performance, demonstrating the usefulness of multimodal information. Adding app description for computing the TF-IDF vector (Template (TF-IDF+Pixel+AppDesc) in Table 2) did not help the performance, possibly due to that all the screens from the same app would have the same app description, making them indistinguishable from each other if the screen text is sparse. The Template (Pixel-DL) baseline, which uses DL-based image encoding, offers only modest improvements over using raw pixels.

On the other hand, the Pixel-Only model, which only uses the image encoding of an element, performs significantly better than all the baselines. The Layout-Only model, which incorporates the mobile UI structural representation, achieves slightly better performances than the Pixel-Only model. Combining the two inputs, Pixel+Layout offers further improvements across the metrics. Similarly, Pixel+Layout+ScreenText, which additionally leverages the textual information from screen, achieves even better results. Our full model, which leverages app description, achieves the best accuracy among all the models. These results show that the multi-modal data of mobile UI complement each other, and their combination leads to better summarization results.

Figure 5 shows example summarization results by different model variants on the same test set screens. All of our models are able to compose coherent, understandable summaries, with our full model having the most substantial capability to capture the underlying semantics of a screen and generate the most accurate summaries. While the Pixel-Only model can sometimes generate relevant descriptions (row 1), it often fails to capture the meaning of a more complex screen. Examples in row 2 demonstrate that additional layout information does help the Pixel+Layout model boost the summarization performance. Our full model results reveal that textual information is helpful to provide more contexts and details when summarizing a screen, such as app categories (row 3). To showcase when the model may fail, we also include examples where all models could only generate very generic descriptions (row 4, left), and where all of them were unable to generate a relevant summary (row 4, right).

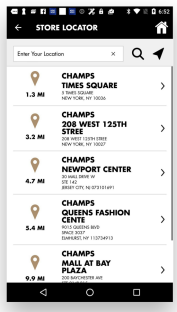
6 HUMAN EVALUATION

To further understand the quality of screen summarization, we conducted a Mechanical Turk study to ask human to assess the quality of the generated screen summary, and validate how automatic metrics correlate with human judgment.

6.1 Study Setup

We compare the model variants *Pixel Only*, *Pixel+Layout*, and the full model *Pixel+Layout+ScreenText+AppDesc*, as well as two template baselines *Template (Pixel)* and *Template (TF-IDF+Pixel)* on the same set of 1000 screens that we randomly sampled from the test set. For each screen, we recruited three raters to assess the quality of summarization. In total, there were 1041 unique human workers participated in the rating, and none of them was involved in the dataset labeling. During rating, human raters were presented with a screen summary generated by one of these models along the corresponding UI screenshot. Human raters are not aware of which model generates the screen summary being evaluated. We asked the raters to consider three aspects:

- *Screen Type*: If the summary mentions the type of screen, e.g., sign-in or sign-up screen, settings menu, search results, how accurate or relevant is it?

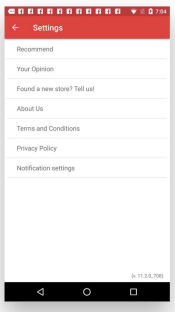


Human Label:
Page displaying with different location options for the stores.

Full Model:
Page displaying list of nearby locations

Pixel+Layout:
Page displaying list of countries.

PixelOnly:
List of options available in the app.

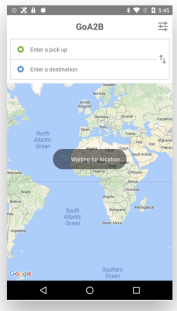


Human Label:
Page showing different options in settings.

Full Model:
List of options available in the app settings.

Pixel+Layout:
List of options available in the app settings.

PixelOnly:
Settings page.



Human Label:
Page displaying search option to find location.

Full Model:
Page showing search bar to find location.

Pixel+Layout:
Page displaying the location finder app.

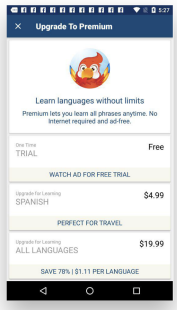
PixelOnly:
Screen displaying the page of a medical app.

Human Label:
Page showing statistics and points in a sports app.

Full Model:
Page displaying the team statistics.

Pixel+Layout:
Display of teams in a sport app.

PixelOnly:
Display of a page showing of flight booking app.

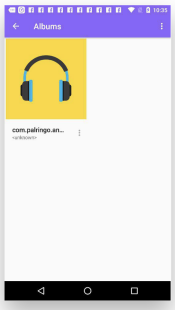


Human Label:
Page displaying upgrade options.

Full Model:
Display of upgrade plans in a learning app.

PixelOnly:
Screen displaying the page of a medical app.

Pixel+Layout:
Screen displaying the page of a fitness app.



Human Label:
Page displaying an album in the app.

Full Model:
Display of albums in a music player app.

Pixel+Layout:
List of options available in the app.

PixelOnly:
Page showing different options on an app.

Human Label:
Welcome page of a scholarship finder app.

Full Model:
Sign up page.

Pixel+Layout:
Welcome page.

PixelOnly:
Welcome page.

Human Label:
Display of tips in a baby care app.

Full Model:
Display of a page showing of dating app.

PixelOnly:
Screen displaying the page of a food app.

Pixel+Layout:
Screen displaying the page of a medical app.

Figure 5: Summarization examples generated by our models. All the examples are sampled from the test set.

- *UI Elements*: If the summary mentions UI elements, e.g., pop-up, list, options, search bar, video, how accurate or relevant is it?
- *App Type*: If the summary mentions the type of the app, e.g., social app, news app, learning app, how accurate or relevant is it?

For the rating scales, we adopted the rating system previously used in evaluating image captioning quality with human judgments [45], but added the fifth point, which indicates the summary not only describes the screen without errors but also provides useful details of the screen, instead of a generic description. The exact rating criteria provided to the labelers are as follows:

Table 3: Mean Summarization Scores Rated by Mechanical Turk Workers.

Model	Mean	STD
Template (Pixel)	2.583	1.075
Template (TF-IDF+Pixel)	3.240	1.081
Pixel-Only	2.886	1.180
Pixel+Layout	3.166	1.060
Pixel+Layout+ScreenText+AppDesc	**3.436***	1.119

- describes the screen without any errors and provides sufficient details.
- describes the screen without any errors.
- describes the screen with minor errors.
- describes the screen with a somewhat related description.
- describes the screen with an unrelated description.

6.2 Study Results

For each screen, we averaged the ratings from three different raters to obtain a score for the summarization quality. As shown in Table 3, the mean score of each deep model variant from human evaluation correlates with the results of automatic evaluation metrics well. The full model is rated the best, which is followed by the *Pixel+Layout*, and then the *Pixel-Only*. The *Template (Pixel)* baseline has the lowest average rating. Interestingly, the strongest baseline on automatic metrics, *Template (TF-IDF+Pixel)*, achieves a higher human rating than the two deep model variants that do not use textual information, even though these deep models outperformed the baseline method on automatic metrics as discussed previously. We speculate this discrepancy between subjective ratings and automatic metric scores might be due to the following. The ScreenText feature may contain keywords that play a decisive role in determining human-perceived relevance of a screen and its summary, which is utilized by TF-IDF+Pixel but deprived from these two deep model variants. The disadvantage does not show in automatic metrics as the metrics treats each word equally instead of weighting them according to their salience to human perception about summarization relevance.

Nonetheless, the results of human evaluation, combined with the automatic metric evaluation, clearly show that our full model outperforms all the template baselines and deep model variants. A non-parametric Mann-Whitney U test shows that the differences between the mean ratings of our full model and all the other settings are significant ($p < 0.0001$). Moreover, 38.6% of the ratings on the full model received the highest score, i.e., 5 points, and the percentage drops to 32.2% and 29.5% for the *Template (TF-IDF+Pixel)* and the *Pixel+Layout*, respectively. These indicate that compared to baselines, our full model is more capable of generating summaries that are not only without any errors but also contain sufficient details.

7 POTENTIAL APPLICATIONS

We have introduced Screen2Words, an approach for automatic screen summarization based on multimodal UI information, and demonstrated the capability of our proposed model for generating quality screen summaries. We now demonstrate the potential use cases of Screen2Words by proposing three mock-up applications: 1) Language-Based User Interface Retrieval, 2) Enhancing Screen Readers, 3) Screen Indexing for Conversational Mobile Applications.

7.1 Language-Based User Interface Retrieval

Screen2Words can empower design search by enabling language-based mobile UI retrieval with which a designer can retrieve design examples using natural language queries. Unlike keyword-based retrieval, which retrieves UIs by directly matching the words in the query with those on the screen texts, Screen2Words models can capture UI semantics beyond text content that appears on a screen, evident by our summary results and accuracy achieved. Figure 6 shows a mock-up interface demonstrating a language-based UI retrieval system. With the UI semantics captured by Screen2Words, designers can not only search with UI keywords like "sign up page" (Figure 6, left), but also use queries that contain more specific semantic details, such as "sign up page of a social network application" (Figure 6, right). We use the mock-up to map out the envisioned user experiences of UI retrieval based on Screen2Words. A fully functioning system could be built by training the Screen2Words models with a triplet loss based on our dataset.

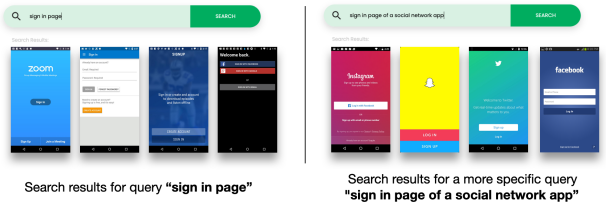


Figure 6: Mock-up interface of language-based UI retrieval system based on Scree2Words. Left: the designer searches for UI design example using the query "sign in page". Right: the designer refines the query to search for sign-in pages of a social network app.

7.2 Enhancing Screen Readers

Screen readers, e.g., VoiceOver and TalkBack, render text and image content on a mobile screen into speech based on the metadata describing the UI. They are essential accessibility features for a visually impaired person yet often suffer from the missing metadata [39, 40, 51], which have motivated recent work in predicting missing metadata using machine learning [28, 50]. Screen2Words can contribute to this effort by predicting screen summaries to provide an overview description for screen reader users. Unlike sighted users who can quickly understand the purpose of an unknown screen with the rich visual interface, visually impaired users have to scan through the elements on a screen, with a screen reader, before they form a mental model for a new UI. Therefore, studies [2, 19, 37] have called out for the need of screen overviews for visually-impaired users so that they can quickly decide whether they should spend time on the current page.

For a more concrete scenario, when a user switches apps with the app switcher (e.g., Android Overview or iOS App Switcher),

the OS typically caches the last viewed screen for each app so that the user would not need to always start from the home screen. However, since visually impaired users cannot see the visuals and the screen readers only read the name of each app, there is no way they could know which specific screen is cached for each app. As shown in Figure 7, Screen2Words can provide a screen summary to help them locate where they are in an app immediately after switching to the app.

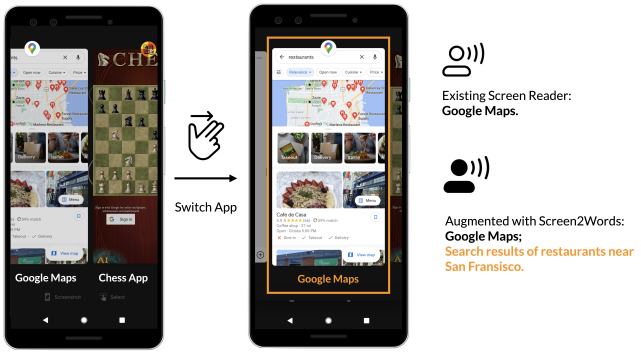


Figure 7: Illustration of how Screen2Words can enhance existing screen readers when using apps switcher.

7.3 Screen Indexing for Conversational Mobile Applications

Screen summarization can be used to equip each screen with additional language metadata so that they can be indexed on the phone. By combining with other app metadata, the user could quickly launch a desired screen by saying *"Setting page of Gmail"* or *"Ordering page of the Starbucks app"* without manual navigation. Moreover, Screen2Words can be integrated into a multi-modal conversational agent that cooperates with users to accomplish mobile tasks [23, 25, 26]. For example, SUGILITE [22] learns to carry out novel tasks by user demonstration. With the screen indexing powered by Screen2Words, speech interaction can replace part of the manual demonstration. For instance, if the user wants to demonstrate how to order an Americano in the Starbucks app, they can say *"Open the ordering page of the Starbucks app."* Once the ordering page is opened, the user can continue to demonstrate the remaining steps manually. The hybrid approach is valuable especially when manually navigating into the desired screen, for completing the task, takes a longer time.

8 DISCUSSIONS AND FUTURE WORK

Screen2Words represents a step closer to bridge mobile UIs with natural language. The proposed approach is applicable to other tasks that require a holistic understanding of the mobile UI and may generalize to different types of user interfaces such as the web UI. However, there are still several limitations which future work could address. Firstly, our screen summarization model is not always accurate. As shown in Figure 5, it sometimes produces generic or wrong summaries. This is akin to the long-standing problem for image captioning where the model is prone to "hallucinating"

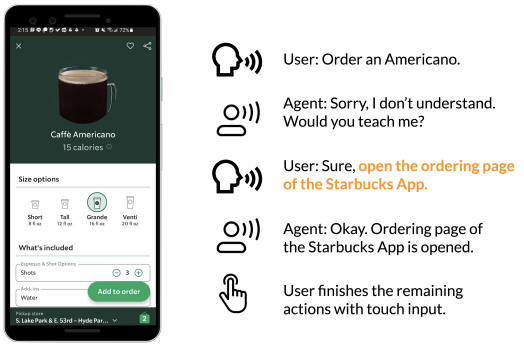


Figure 8: Illustration of how screen indexing enabled by Screen2Words can facilitate multi-modal interactions to accomplish mobile tasks with conversational agents.

objects that are not actually in a scene [38]. Moreover, similar to prior work on unconventional vision-to-language tasks [17, 46], we found that human rating and automatic scores may not always be well-correlated, signaling the need for better automatic evaluation metrics. Future work could investigate new model architectures and evaluation metrics to obtain better summarization results.

In some scenarios, app metadata may be missing and the performance of Screen2Words may decrease. However, our approach is still useful as our configurations such as PixelOnly rely on pixel input only—that is always available. Particularly, we found pixel input to be crucial when ScreenText is sparse or entirely absent. Such scenarios reinforce the motivation of our multimodal approach, which performs even if a modality is missing.

Another limitation is that screen summarization only generates descriptions for the entire screen. It cannot be steered by the user to describe the information of a specified section on the screen. Therefore, a natural extension to Screen2Words is towards Visual Question Answering (VQA) [1, 13] for UI screens, which takes a mobile screen and a free-form, open-ended, natural language question as input, and produces a natural language answer as the output. Screen2Words could be viewed as a model specialized for answering questions such as "what's on the screen?". A full-range Screen VQA should be able to answer further questions such as "what actions can I take with the screen?" or "what's the title of the first news article on the screen?" Such technologies could greatly facilitate eye-free, speech interaction with mobile devices.

Lastly, one of the use cases of Screen2Words is to facilitate UI retrieval with language queries. An immediate next step is to explore language-based UI generation based on our dataset, which generates mobile UI designs based on language descriptions. Future work could leverage graphical layout generation models [20, 21] conditioning on a screen summary to generate user interface designs. To this end, our open-source dataset would be a valuable source to fuel all these research directions.

9 CONCLUSION

We have presented Screen2Words, an approach for automatically summarizing the multi-modal information of a mobile UI screen as

a concise language summary using deep learning methods. We collected and analyzed the first large-scale human-annotated dataset to investigate the task. Based on the dataset, we trained and evaluated a set of deep models to examine the feasibility of automatic UI screen summarization. Our evaluation with various automatic metrics shows that deep learning models outperform the heuristic baselines with a significant margin. Our full model, which leverages image, text, and UI structural information, achieves the best results among all the model variants. We have also conducted a human evaluation using Mechanical Turk and show that our full model significantly outperforms other models and baselines on subjective rating. Lastly, we outline three application scenarios that would benefit from the Screen2Words approach. Our dataset, benchmark models and experimental results lay the groundwork for future work on automatic UI screen summarization, which contributes to the effort for bridging natural language and mobile user interface.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their constructive feedback, which has helped improve this paper. We also thank the participants in our human evaluation and Google data team for their tremendous help on data collection.

REFERENCES

[1] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. 2015. VQA: Visual Question Answering. In *International Conference on Computer Vision (ICCV)*.

[2] Y. Borodin, Jeffrey P. Bigham, Glenn Dausch, and I. Ramakrishnan. 2010. More than meets the eye: a survey of screen-reader browsing strategies. In *W4A*.

[3] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. 2013. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*. 108–122.

[4] Sara Bunian, Kai Li, Chaima Jemmali, Casper Harteveld, Yun Fu, and Magy Seif El-Nasr. 2021. VINS: Visual Search for Mobile User Interface Design. *arXiv preprint arXiv:2102.05216* (2021).

[5] Bor-Chun Chen, Yan-Ying Chen, and Francine Chen. 2017. Video to Text Summary: Joint Video Summarization and Captioning with Recurrent Neural Networks.. In *BMVC*.

[6] Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollar, and C. Lawrence Zitnick. 2015. Microsoft COCO Captions: Data Collection and Evaluation Server. arXiv:1504.00325 [cs.CV]

[7] 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* (Québec City, QC, Canada) *(UIST '17)*. Association for Computing Machinery, New York, NY, USA, 845–854. https://doi.org/10.1145/3126594.3126651

[8] 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* (Tokyo, Japan) *(UIST '16)*. ACM, New York, NY, USA, 767–776. https://doi.org/10.1145/2984511.2984581

[9] Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the ninth workshop on statistical machine translation*. 376–380.

[10] Aniqa Dilawari and Muhammad Usman Ghani Khan. 2019. ASoVS: Abstractive summarization of video sequences. *IEEE Access* 7 (2019), 29253–29263.

[11] B. Erol, D. . Lee, and J. Hull. 2003. Multimodal summarization of meeting recordings. In *2003 International Conference on Multimedia and Expo. ICME '03. Proceedings (Cat. No.03TH8698)*, Vol. 3. III–25. https://doi.org/10.1109/ICME.2003.1221239

[12] Camilo Fosco, Vincent Casser, Amish Kumar Bedi, Peter O'Donovan, Aaron Hertzmann, and Zoya Bylinskii. 2020. Predicting Visual Importance Across Graphic Design Types. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) *(UIST '20)*. Association for Computing Machinery, New York, NY, USA, 249–260. https://doi.org/10.1145/3379337.3415825

[13] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2017. Making the V in VQA Matter: Elevating the Role of Image Understanding in Visual Question Answering. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. arXiv:1512.03385 [cs.CV]

[15] Eduard Hovy, Chin-Yew Lin, et al. 1999. Automated text summarization in SUMMARIST. *Advances in automatic text summarization* 14 (1999), 81–94.

[16] 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)*. Association for Computing Machinery, New York, NY, USA, 1–10. https://doi.org/10.1145/3290605.3300334

[17] Ting Hao Huang, Francis Ferraro, Nasrin Mostafazadeh, Ishan Misra, Aishwarya Agrawal, Jacob Devlin, Ross Girshick, Xiaodong He, Pushmeet Kohli, Dhruv Batra, C. Lawrence Zitnick, Devi Parikh, Lucy Vanderwende, Michel Galley, and Margaret Mitchell. 2016. Visual storytelling. In *2016 Conference of the North American Chapter of the Association for Computational Linguistics (2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL HLT 2016 - Proceedings of the Conference)*. Association for Computational Linguistics (ACL), 1233–1239. https://doi.org/10.18653/v1/n16-1147 Publisher Copyright: ©2016 Association for Computational Linguistics. Copyright: Copyright 2020 Elsevier B.V., All rights reserved.; 15th Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL HLT 2016 ; Conference date: 12-06-2016 Through 17-06-2016.

[18] Diederik P. Kingma and Jimmy Ba. 2017. Adam: A Method for Stochastic Optimization. arXiv:1412.6980 [cs.LG]

[19] R. Kuber, Amanda Hastings, Matthew Tretter, and D. Fitzpatrick. 2012. DETERMINING THE ACCESSIBILITY OF MOBILE SCREEN READERS FOR BLIND USERS.

[20] Hsin-Ying Lee, Lu Jiang, Irfan Essa, Phuong B Le, Haifeng Gong, Ming-Hsuan Yang, and Weilong Yang. 2020. Neural Design Network: Graphic Layout Generation with Constraints. arXiv:1912.09421 [cs.CV]

[21] Jianan Li, Jimei Yang, Aaron Hertzmann, Jianming Zhang, and Tingfa Xu. 2019. LayoutGAN: Generating Graphic Layouts with Wireframe Discriminators. arXiv:1901.06767 [cs.CV]

[22] 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* (Denver, Colorado, USA) *(CHI '17)*. Association for Computing Machinery, New York, NY, USA, 6038–6049. https://doi.org/10.1145/3025453.3025483

[23] 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 *(UIST '20)*. Association for Computing Machinery, New York, NY, USA, 1094–1107. https://doi.org/10.1145/3379337.3415820

[24] Toby Jia-Jun Li, Lindsay Popowski, Tom M Mitchell, and Brad A Myers. 2021. Screen2Vec: Semantic Embedding of GUI Screens and GUI Components. *arXiv preprint arXiv:2101.11103* (2021).

[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* (New Orleans, LA, USA) *(UIST '19)*. Association for Computing Machinery, New York, NY, USA, 577–589. 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 Annual International Conference on Mobile Systems, Applications, and Services* (Munich, Germany) *(MobiSys '18)*. Association for Computing Machinery, New York, NY, USA, 96–109. https://doi.org/10.1145/3210240.3210339

[27] 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*. Association for Computational Linguistics, Online, 8198–8210. https://doi.org/10.18653/v1/2020.acl-main.729

[28] 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. arXiv:2010.04295 [cs.LG]

[29] Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*. 74–81.

[30] Thomas F. Liu, Mark Craft, Jason Situ, Ersin Yumer, Radomir Mech, and Ranjitha Kumar. 2018. Learning Design Semantics for Mobile Apps. In *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/10.1145/3242587.3242650

[31] P. Magdum and Sheetal Rathi. 2021. *A Survey on Deep Learning-Based Automatic Text Summarization Models*. 377–392. https://doi.org/10.1007/978-981-15-3514-7_30

[32] Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, Zhiheng Huang, and Alan Yuille. 2014. Deep captioning with multimodal recurrent neural networks (m-rnn). *arXiv preprint arXiv:1412.6632* (2014).

[33] Mani Maybury. 1999. *Advances in automatic text summarization.* MIT press.

[34] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*. 311–318.

[35] Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A Deep Reinforced Model for Abstractive Summarization. arXiv:1705.04304 [cs.CL]

[36] 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)*. Association for Computational Linguistics, Doha, Qatar, 1532–1543. https://doi.org/10.3115/v1/D14-1162

[37] André Rodrigues, Hugo Nicolau, Kyle Montague, João Guerreiro, and Tiago Guerreiro. 2019. Open Challenges of Blind People using Smartphones.

[38] Anna Rohrbach, Lisa Anne Hendricks, Kaylee Burns, Trevor Darrell, and Kate Saenko. 2019. Object Hallucination in Image Captioning. arXiv:1809.02156 [cs.CL]

[39] Anne Spencer Ross, Xiaoyi Zhang, James Fogarty, and Jacob O. Wobbrock. 2017. Epidemiology as a Framework for Large-Scale Mobile Application Accessibility Assessment. In *Proceedings of the 19th International ACM SIGACCESS Conference on Computers and Accessibility* (Baltimore, Maryland, USA) *(ASSETS '17)*. Association for Computing Machinery, New York, NY, USA, 2–11. https://doi.org/10.1145/3132525.3132547

[40] Anne Spencer Ross, Xiaoyi Zhang, James Fogarty, and Jacob O. Wobbrock. 2018. Examining Image-Based Button Labeling for Accessibility in Android Apps through Large-Scale Analysis. In *Proceedings of the 20th International ACM SIGACCESS Conference on Computers and Accessibility* (Galway, Ireland) *(ASSETS '18)*. Association for Computing Machinery, New York, NY, USA, 119–130. https://doi.org/10.1145/3234695.3236364

[41] Amanda Swearngin and Yang Li. 2019. Modeling Mobile Interface Tappability Using Crowdsourcing and Deep Learning. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–11. https://doi.org/10.1145/3290605.3300305

[42] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762* (2017).

[43] Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4566–4575.

[44] Subhashini Venugopalan, Marcus Rohrbach, Jeffrey Donahue, Raymond Mooney, Trevor Darrell, and Kate Saenko. 2015. Sequence to sequence-video to text. In *Proceedings of the IEEE international conference on computer vision*. 4534–4542.

[45] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Computer Vision and Pattern Recognition*. http://arxiv.org/abs/1411.4555

[46] Xin Wang, Wenhu Chen, Yuan-Fang Wang, and William Yang Wang. 2018. No Metrics Are Perfect: Adversarial Reward Learning for Visual Storytelling. arXiv:1804.09160 [cs.CL]

[47] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In *Proceedings of the 32nd International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 37)*, Francis Bach and David Blei (Eds.). PMLR, Lille, France, 2048–2057. http://proceedings.mlr.press/v37/xuc15.html

[48] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*. PMLR, 2048–2057.

[49] Mahmood Yousefi-Azar and Len Hamey. 2017. Text summarization using unsupervised deep learning. *Expert Systems with Applications* 68 (2017), 93–105.

[50] Xiaoyi Zhang, Lilian de Greef, Amanda Swearngin, Samuel White, Kyle Murray, Lisa Yu, Qi Shan, Jeffrey Nichols, Jason Wu, Chris Fleizach, Aaron Everitt, and Jeffrey P. Bigham. 2021. Screen Recognition: Creating Accessibility Metadata for Mobile Applications from Pixels. arXiv:2101.04893 [cs.HC]

[51] 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* (Berlin, Germany) *(UIST '18)*. Association for Computing Machinery, New York, NY, USA, 609–621. https://doi.org/10.1145/3242587.3242616

[52] Junnan Zhu, Haoran Li, Tianshang Liu, Yu Zhou, Jiajun Zhang, and Chengqing Zong. 2018. MSMO: Multimodal summarization with multimodal output. In *Proceedings of the 2018 conference on empirical methods in natural language processing*. 4154–4164.

[53] Junnan Zhu, Yu Zhou, Jiajun Zhang, Haoran Li, Chengqing Zong, and Changliang Li. 2020. Multimodal summarization with guidance of multimodal reference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 9749–9756.