An Empirical Investigation into the Use of Image Captioning for Automated Software Documentation

Kevin Moran†, Ali Yachnes*, George Purnell*, Juanyed Mahmud†,
Michele Tufano‡, Carlos Bernal Cardenas‡, Denys Poshyvanyk*, Zach H'Doubler*
†*George Mason University, VA, USA*, **William & Mary, VA, USA*, ‡*Microsoft, WA, USA*
kpmoran@gmu.edu, ayachnes@email.wm.edu, gwpurnell@email.wm.edu, jmahmud@gmu.edu
michele.tufano@microsoft.com, carlosbe@microsoft.com, denys@cs.wm.edu, pzhdoubler@email.wm.edu

***Abstract*—Existing automated techniques for software documentation typically attempt to reason between two main sources of information: code and natural language. However, this reasoning process is often complicated by the *lexical gap* between more abstract natural language and more structured programming languages. One potential bridge for this gap is the Graphical User Interface (GUI), as GUIs inherently encode salient information about underlying program functionality into rich, pixel-based data representations. This paper offers one of the first comprehensive empirical investigations into the connection between GUIs and functional, natural language descriptions of software. First, we collect, analyze, and open source a large dataset of functional GUI descriptions consisting of 45,998 descriptions for 10,204 screenshots from popular Android applications. The descriptions were obtained from human labelers and underwent several quality control mechanisms. To gain insight into the representational potential of GUIs, we investigate the ability of four Neural Image Captioning models to predict natural language descriptions of varying granularity when provided a screenshot as input. We evaluate these models quantitatively, using common machine translation metrics, and qualitatively through a large-scale user study. Finally, we offer learned lessons and a discussion of the potential shown by multimodal models to enhance future techniques for automated software documentation.**

***Index Terms*—Software Documentation, Image Captioning, Deep Learning**

I. Introduction & Motivation

Proper documentation is generally considered to be an integral component of building and distributing modern software systems. In fact, past studies have illustrated the general benefits of documentation during the development lifecycle [1], [2], [3], [4] and the importance of technical documentation to software maintenance and evolution [5]. However, despite the value of well-documented systems, modern development processes and constraints often lead to the disregard or abandonment of a range of documentation tasks [6], [5], [2], [7], [8], [1]. These difficulties have given rise to a wealth of research on automated techniques that aim to ease the burden on stakeholders by generating various types of documentation for a given task. For example, existing approaches have been developed to automatically generate natural language summaries and documentation for code [9], [10], [11], [12], [13], [14], [15], APIs [16], [17], unit tests [18], bug reports [19], [20], release notes [21], [22], and commit messages [23], [24], among other artifacts [25], [26].

Generally, existing techniques for automated software documentation have been concerned with modeling relationships that exist between two primary information modalities: code and natural language (NL). Unfortunately, reasoning between these two information sources is difficult due to the *lexical gap* resulting from the often disparate conceptual associations that connect source code lexicon and the more abstract words and phrases used in NL descriptions [27], [28]. Recently, this lexical gap was acknowledged as an *information inference* problem in a report made by Robillard *et al.* [29], wherein key research challenges exist in (i) inferring undocumented program properties, and (ii) discovering latent abstractions and rationales. These challenges suggest that overcoming the semantic disconnect between code and NL may require new knowledge sources that encode distinct program properties typically absent from traditional software or NL lexicon.

One source of information which has been left largely unexplored for the purposes of automated documentation is *visual software data* encoded into Graphical User Interfaces (GUIs). GUI-based applications predominate modern user-facing software, as can be readily seen in the popularity of desktop and mobile apps [30]. Furthermore, high quality applications with well-designed GUIs allow technically-inclined users to instinctively understand underlying program features. Thus, intuitively, certain functional properties of applications are encoded into the visual, pixel-based representation of the GUI such that cognitive human processes can determine the computing tasks provided by the interface. This suggests that there are latent *patterns* that exist within visual GUI data which indicate the presence of natural use cases capturing core functionality [31].

Given the inherent representational power of GUIs in conveying program related information, we set forth the following hypothesis that serves as the basis for work in this paper:

The representational power of graphical user interfaces to convey program-related information can be meaningfully leveraged to support automated techniques for software documentation.

While most existing work on automated documentation concerns itself with the dichotomy between code and NL, we posit that the latent information encoded within GUIs can aid in bridging the existing semantic documentation gap by providing

an additional source of knowledge that inherently reflects program functionality. In fact, GUI-based representations of software have the potential to address the two challenges set forth by Robillard *et al.* [29]. More specifically, GUIs can aid in *inferring undocumented program properties* that are inherently represented within the design of GUI controls or widgets (*e.g.,* capturing a feature which is otherwise poorly represented by low-quality code identifiers/comments). Further, GUIs could be used as source to *mine abstractions or rationales* that would otherwise remain obscure (*e.g.,* providing a use case-based explanation of a block of code connected to a GUI screen). In overcoming these challenges, we see *GUI-centric documentation* having an impact on the following types of software documentation:

Technical Documentation: Developers utilize technical documentation, such as code comments or READMEs, in order to learn about the functionality and interfaces of software to support engineering tasks. Automatically generating such documentation accurately is a challenging inference problem. However, it has been shown that GUI-related code can comprise as much as half of the code in user facing programs [32]. This means that graphical software data is connected in some way to large portions of GUI-based software projects *i.e.,* through GUI-event handlers, or code stipulating GUI layouts such as `html`. Therefore, if automated techniques are able to effectively infer salient functionality from the GUIs, they could be combined with existing techniques and leveraged to provide automation to developers, such as comment generation or code summarization with greater feature-based context awareness. As we illustrate in this paper, GUI code/metadata appears to encode *orthogonal* information compared to visual GUI data (*i.e.,* screenshots), which suggests that we may be able to infer documentation information from visual GUI data that likely can't be inferred from GUI code alone.

User Documentation: Developers typically provide users with documentation such as tutorials or walkthroughs to help clearly illustrate software features. While some experienced users can infer functionality directly from a GUI, end-users exhibit a range of technological expertise, and many *rely* upon various forms of end-user documentation [33]. Thus, building techniques capable of *automatically* generating such documentation would free up development effort for other critical tasks, such as bug fixing. Beyond typical user facing software aids, GUI-centric program documentation could also enable entirely new classes of automated accessibility features, which are sorely needed for mobile apps [34]. For example, rather than a typical text-to-speech engine, one could envision a screen-to-functionality engine that could aid a motor-impaired user with navigating the software, without extra development effort.

To investigate the potential of automated GUI-centric software documentation, we offer one of the first comprehensive empirical investigations into this new research direction's most fundamental task: *generating a natural language description given a screenshot (or screen-related information) of a software GUI*. Given that this task underlies the various potential applications discussed above, we view this as a logical first step towards investigating the feasibility of future techniques. To accomplish this, we collect and analyze a dataset for **C**omprehending visua**L** sem**A**ntics to p**R**edict applicat**I**on functional**TY** (the CLARITY dataset) consisting of 45,998 functional descriptions of 10,204 screenshots of popular Android apps available on Google Play. We provide a descriptive analysis of this dataset that investigates the "naturalness" and semantic topics of the collected descriptions by measuring cross-entropy compared to other corpora and performing a topic modeling analysis. To learn functional descriptions of the screens from this dataset, we customize, train, and test four Deep Learning (DL) models for neural image captioning—three that learn from image data and one that learns from textual GUI metadata—to predict functional descriptions of software at different granularities. We evaluate the efficacy of these models both quantitatively, by measuring the widely used BLEU metric, and qualitatively through a large-scale user study.

In summary, this paper's contributions are as follows:

- We collect the CLARITY dataset of GUIs annotated with 45,998 functional, NL descriptions from 10,204 screenshots of popular Android apps. The NL captions were obtained from human labelers, underwent several quality control mechanisms, and contain both high- and low-level descriptions of screen functionality. While other GUI datasets exist [35], [36], the CLARITY dataset differs by providing an extensively labeled set of screens, akin to Flickr8K [37] or MSCOCO [38];
- We illustrate the underlying, natural patterns that exist in the CLARITY dataset through topic modeling.
- We provide an extensive quantitative and qualitative evaluation of four tailored DL models for image captioning using standard metrics and a large scale user study;
- We offer an online appendix with examples of model-generated descriptions and experimental data [39]. Our dataset, trained models, code, and evaluation scripts are open source and accessible via the appendix.

II. BACKGROUND

A. The Connection between Images and NL

The task of image captioning is much more difficult than that of classification or labeling, as an effective model must be able to both learn salient features from images automatically and semantically equate these features with the proper NL words and grammar that describe them. This task of semantically aligning two completely different modalities of information has led to the development of multimodal DL architectures that jointly embed NL and pixel-based information in order to predict an appropriate description of a given input image. These techniques are typically trained on large-scale datasets that contain images annotated with multiple captions, such as MSCOCO [38], and have largely drawn inspiration from encoder-decoder neural language models traditionally applied to machine translation tasks. In this

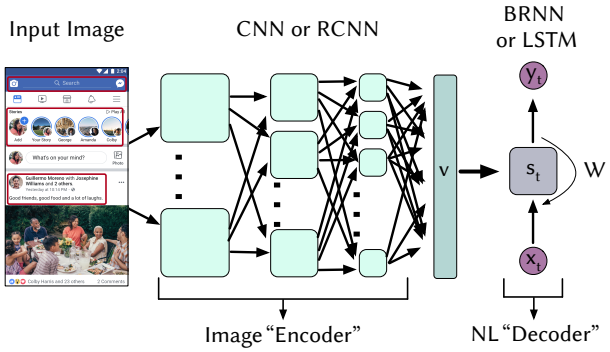


Fig. 1: Generalized overview of multimodal DL architectures for image captioning (with RCNN)

paper, we adapt three recent architectures for image captioning, `neuraltalk2` [40], the `im2txt` [41], and the `show, attend and tell (SAT)` [42] frameworks to predict functional descriptions of software screenshots through the use of custom pre-training and fine-tuning procedures. Additionally, we explore the `seq2seq` neural language model.

DL models for image captioning build upon the success of encoder-decoder neural language models. The `im2txt` framework treats image captioning as a machine translation problem, wherein the source "sentence" is an image, and the target "translation" is a NL description. The generalized architecture of such models is shown in Fig. 1. As illustrated, these architectures replace the encoder RNN with a Convolutional Neural Network (CNN), which have been shown to be highly capable of learning rich image features [43], [44], [45]. Google's implementation of `im2txt` uses a Long-Short Term Memory (LSTM) RNN [46] for the "decoder" module, which has also proven extremely effective when applied to machine translation tasks. The decoder module of the `neuraltalk2` architecture is composed of a Bidirectional RNN (BRNN) [47] as opposed to an LSTM. Finally, the `show, attend, & tell (SAT)` model [42] uses an LSTM decoder but with the addition of an attention mechanism that can "attend" to salient parts of the image representation by combining "hard" and "soft" attention mechanisms.

III. Overview

In this section, we provide an "at-a-glance" overview of the data-collection procedures and various analyses performed in this paper. Figure 2 illustrates the four major components of the paper. The first major task of our study is to derive a suitable dataset of screenshot-caption pairs. We describe this process in two parts: (i) the collection of screenshots (Sec. IV-A), and (ii) the collection of captions from human workers (Sec. IV-B). The result of this data-collection effort is the CLARITY dataset, which contains 45,998 captions of 10,204 Android screenshots. Next, we aim to understand the *lexical* properties of our captions through an empirical analysis in order to better understand how easily they might be modeled (Sec. V). Thus, we perform both a comparison of the the cross-entropy of language models trained our caption corpus to other popular SE corpora, and perform an LDA-based topic analysis. Next, we discuss the process of configuring and training three neural image captioning models, and one

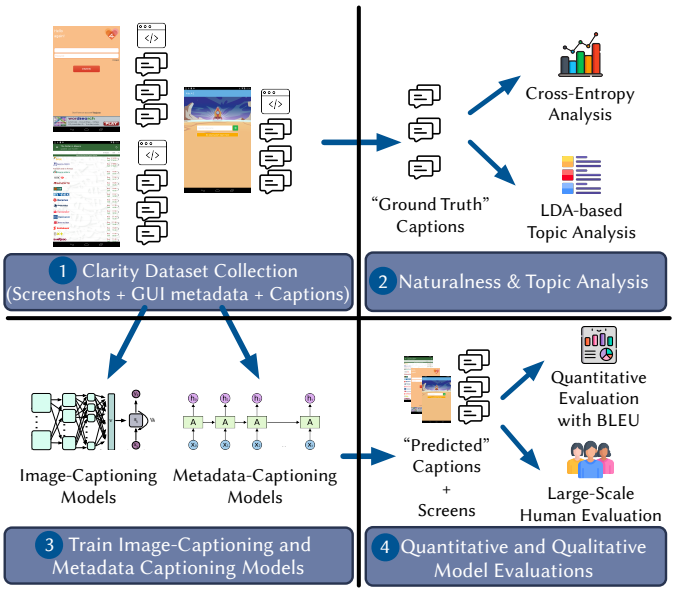


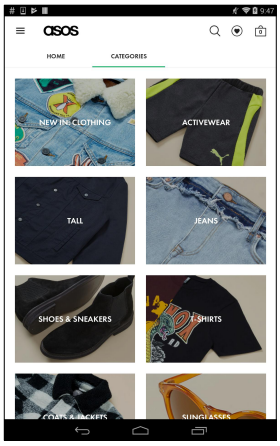
Fig. 2: Overview of Dataset Collection and Analysis

sequence-based model to predict functional descriptions of software GUIs (Sec. VI). Finally, we conclude our analysis by measuring the accuracy of our trained models according to both automated reference-based metrics (*i.e.,* BLEU@n) and via a large-scale human evaluation. (Sec. VII)

IV. Dataset Collection

A. Screen & GUI Metadata Collection

The first step in deriving the CLARITY dataset is the collection of a sizable and diverse dataset of screenshots and GUI-metadata. We chose to focus this dataset derivation on the Android platform for three main reasons: (i) Android is currently the most popular OS in the world [30], (ii) Android apps are highly GUI-and gesture driven, making them a suitable target for our investigation, and (iii) the Android `screencap` and `uiautomator` tools facilitate the extraction of screenshots and GUI-metadata from running apps. Fortunately, large-scale datasets of Android screenshots and metadata are publicly available in related literature [48], [35]. For this work, we took advantage of the REDRAW [48], [36] dataset which contains nearly 17k unique screenshots from 8,655 of the top-rated apps from the Google Play Store. It should be noted that another large-scale Android GUI dataset that contains a larger number of screenshots, RICO, is also available [35]. However, we chose to utilize the REDRAW dataset as it aligned with one of our primary study objectives. That is, we aim to learn latent feature information from both screenshots and GUI-metadata. However, for the GUI-metadata to properly align with the displayed content on a screen, the app must make use of *native* Android components. Therefore, apps that primarily display their information using web technologies, so-called *hybrid apps*, would obscure the GUI-metadata and impact our study findings. The REDRAW dataset contains a set of screenshots that underwent several stages of filtering to remove instances of hybrid apps along with other noise. Furthermore, the REDRAW dataset contains a set of GUI-component images



High Level Caption

The screen allows the user to look at clothing categories

Low Level Captions

The top left icon allows the user to access the menu

The top right icon allows the user to access the shopping cart

The center list of categories allows the user to make a selection

The heart icon to the left of the shopping cart allows the user to view favorites

Fig. 3: Example of captions from the CLARITY dataset.

labeled with their corresponding types (*e.g.,* `Button`) which we utilize later in our study (Sec. VI). The end result of this filtering process was a total set of 17,203 candidate screens for labeling. We refer readers to the REDRAW paper for complete details of the filtering process [48].

B. Collection of Functional Descriptions

Once we derived a suitable set of screens, we needed to manually label these screens with functional captions. This process occurred in two steps: (i) first, we conducted a pilot labeling study in order to develop and prove out a tagging methodology suitable for large scale caption collection; (ii) second, we performed a full scale data collection study using Amazon's Mechanical Turk Crowd-worker platform to collect over 10k screens with functional descriptions.

*1) **Caption Granularity**:* Intuitively, GUIs encode functional information at multiple levels of granularity. For example, if you were to ask a user or developer what the high-level purpose of a given screen is, they might say *"This screen allows users to browse clothing categories"*, as shown in Fig. 3. These types of descriptions constitute the "high-level" functionality of a given screen. However, a single screen rarely implements only one functionality, and there may be multiple functional properties that enable the screen's high-level functional purpose. User descriptions of these types of functional properties are typically centered around the *interactive* components of a screen, since these represent the instances of actions (*e.g.,* users "doing something") that are easily attributed to implemented functions. For example, in the screen in Fig. 3, underlying functions include viewing favorites, accessing a shopping cart, or selecting an item from a list. These types of "low-level" screen properties centered around GUI-components describe key constituent functionality. Hence, in order to capture a holistic functional view of each screen, we tasked participants with labeling each screen with one "high-level" functional caption, and up to four "low-level" functional captions. Fig. 3 shows these two categories using actual captions collected as part of the CLARITY dataset.

*2) **Pilot Data Collection Study**:* We developed an initial image caption collection platform using a Java-based web application. Using this system, the authors manually labeled 743 screens with the caption granularities described earlier. During this study, we discovered some instances of screens with relatively little information displayed on them, making it difficult to label them with functional attributes, even after the filtering techniques discussed previously. Therefore, before moving onto the large-scale caption collection with Mechanical Turk (MTurk), at least one author manually inspected each of the 17,203 candidate screens, and discarded those with a severe lack of functionality. This resulted in a set of 16,311 candidate screens for the next phase of the study.

*3) **Mechanical Turk Data Collection Study**:* To set up our large-scale data-collection process, we adapted our web application caption collection mechanism to work with MTurk's crowd worker platform. This involved configuring a Human Intelligence Task (HIT) that provided workers with a set of detailed instructions, displaying a screenshot from our dataset alongside text entry boxes for one high-level functional caption and up to four low-level functional captions (a limit was imposed to normalize the amount of time workers would spend on the HIT). This study was approved by the Institutional Review Board of the authors' affiliated institution.

Given that we aimed to collect high-quality functional descriptions of screens in natural English, we targeted MTurk users from primarily English speaking countries that had completed at least 1,000 HITs and had a HIT approval rate of at least 90%. We provided a detailed set of instructions for labeling images with captions that clearly explained the concept of high-level and low-level captions with examples, and provided users with explicit instructions as well as DOs and DONTs for the labeling task. The full set of instructions is available in our online appendix [39]. With regard to caption quality, we specifically had three major requirements: (i) that the caption describes the perceived *functionality* of a screen and not simply its appearance, (ii) that *spatial references* are given for low-level captions (*e.g., "the button in the top-left corner of the screen"*), and (iii) that captions be written in complete English sentences with reasonably proper grammar.

We published batches of HIT tasks by sampling unique screens from our set of 16,311 candidate screens, ensuring that no user was assigned the same screen twice. The quality of work from crowd-sourced tasks is not always optimal, so as captions were submitted, they needed to be vetted for quality. Thus, the captions for each screen were examined by at least one author for the three quality attributes mentioned above. If an author was unsure about whether a screen met these quality attributes, it was reviewed by at least one other author to reach a consensus. In total, 2,419 screens were rejected and republished as new HITs due to quality issues. In summary, 2,150 MTurk workers collected 45,998 captions (across granularities) for 10,204 screens (≈5 screens per participant), and over $2,400 was paid out.

V. EMPIRICAL DATASET ANALYSIS

The CLARITY dataset provides a rich source of data for exploring the relationship between GUI-based and lexical

TABLE I: LDA topics learned over high-level captions $k = 15$

Assigned Label	Top 7 Words
"color options"	screen show app option color book differ
"login or create acccount"	user screen allow account log creat app
"select image from a list"	user screen allow select view list imag
"map search by location"	screen locat search map user show find

TABLE II: LDA Topics learned on low-level captions $k = 25$

Assigned Label	Top 7 Words
"page button"	page button top center bottom side left
"select date"	avail date select one option theme present
"camera button"	video imag photo pictur bottom camera
"privacy policy banner"	titl just term blue banner privaci polici

software data. However, it is important to investigate the semantic makeup of the collected captions in order to better understand: (i) the *latent topics* they capture as well as (ii) their *naturalness* and, hence, predictability. In this section we carry out an empirical analysis of this phenomena guided by the following two Research Questions (RQs):

- **RQ$_1$:** *What are the latent topics captured within the high- and low-level captions in the* CLARITY *dataset?*
- **RQ$_2$:** *How natural (i.e., predictable) are the high- and low-level captions in the* CLARITY *dataset?*

A. Analysis Methodology

*1) **RQ$_1$: Investigating Dataset Topics**:* To investigate the latent topics in the CLARITY dataset, we learned topic models over caption corpora representing different granularities of functional descriptions. More specifically, we applied Latent Dirichlet Allocation (LDA) [49] to both segmented high- and low- level captions from the CLARITY dataset. In our analysis, the set of captions for a specific screenshot in the CLARITY dataset represents a document, and the entire set of captions across screenshots for a given granularity (*i.e.,* high or low level) constitutes a corpus. LDA has several configurable hyper-parameters that impact the smoothing of generated topics. These include k, the number of topics, n which denotes the number of iterations of the sampling algorithm (Gibbs sampling [50], in our case), as well as α and β which impact topic distributions. We set α and β to standard values for NL corpora, set n to 500, which proved to be a sufficient for model convergence, and varied k between 15, 25, 50, and 75 topics.

*2) **RQ$_2$: Analyzing the Naturalness of GUI Descriptions**:* Past work has pioneered the notion of the *naturalness* of software [51], which illustrated the fact that software, even more so than NL, exhibits repetitive patterns that make it predictable. This finding was recently further investigated and the existence of certain natural patterns was confirmed [52]. To illustrate naturalness, these past studies have learned statistical n-gram language models over software corpora, and measured the "perplexity" (or a log-transformed version, *cross-entropy*) of these models, which represents the degree to which a model is "surprised" by the patterns on a test corpus when trained on a corpus from the same domain. A model with lower measured

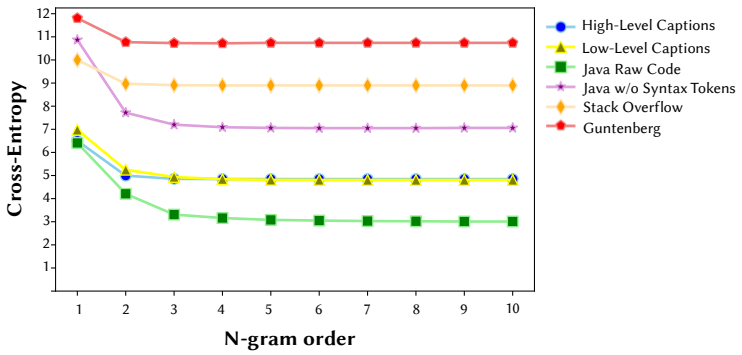


Fig. 4: Cross-entropy of the CLARITY dataset's high and low-Level captions compared to other corpora.

cross-entropy represents a higher predictive power, and thus, a more *natural* underlying corpus.

We follow the methodology of these past studies to explore the naturalness of the CLARITY dataset captions. Thus, similar to the methodology for the previous RQ, we split the collected captions into two corpora, one for the high-level descriptions, and one for the low-level descriptions. We then learned interpolated n-gram models, using the `mitlm` [53] implementation of Kneser-Ney smoothing [54], which has been shown to be the most effective n-gram smoothing method [51], following a ten-fold cross-validation procedure. We report the average cross-entropy values across these experiments for both the high and low-level corpora, compared to prior results [51], [52] for other NL and software corpora.

B. Analysis Results

*1) **RQ$_1$: Results of Dataset Topic Modeling**:* We present selected results of some of the most representative topics in Tables I & II, complete with descriptive labels that we provide for readability, and include all the results in our appendix [39]. These topics help to provide a descriptive illustration of some of the latent patterns that exist in both the high and low level CLARITY captions. The high-level captions illustrate several screen-level topics, including searching on a map and adjusting app settings. The low-level captions conversely capture topics that describe component-level functionality, such as date selectors, camera buttons, and back buttons. These results indicate the existence of logical topics specific to the domain of GUIs in our collected captions.

*2) **RQ$_2$: The Naturalness of Clarity Descriptions**:* The results of our naturalness analysis are illustrated in Figure 4. This figure shows the average cross entropy of the high- and low- level captions from the CLARITY dataset compared to several other corpora as calculated by Rahman *et al.* [52]. More specifically, the graph depicts the average ten-fold cross entropy for: (i) The Gutenberg corpus containing over 3k English books written by over a hundred different authors, (ii) Java code from over 134 open source projects on GitHub, (iii) Java without *Syntax Tokens* (*i.e.,* separators, keywords, and operators), and (iv) a Stack Overflow corpus consisting of only the English descriptions from over 200k posts.

As described earlier, the lower the cross-entropy is for a particular dataset, the more *natural* it is. That is, the corpora that exhibit lower cross entropy tend to exhibit stronger latent patterns that can be effectively modeled and predicted. As we see from Fig. 4, the CLARITY high and low level captions are *more natural* than every dataset excluding raw Java code. It should be noted that, comparatively, there are several factors that could account for the observed lower cross entropy of the CLARITY captions. For instance, such factors could include other corpora having a larger size or having a more diverse set of human authors and writing styles. However, we mainly provide entropy measures of other datasets to provide context for the predictability of the CLARITY dataset compared to other popular corpora. Regardless of dataset differences, the average ≈ 5 bits of entropy measured for the two datasets of CLARITY captions signals that our collected descriptions exhibit strong semantic patterns that can be effectively modeled for prediction. Additionally, we observe that the cross-entropy for the high and low-level captions are surprisingly similar. Intuitively, one might expect that the low-level CLARITY captions would exhibit more prevalent patterns due to the repetitive use cases of certain GUI-components such as menu buttons. This indicates the tendency of both datasets to exhibit patterns that can be appropriately modeled. However, as we illustrate in Sec. VII the ability for GUI-related information to predict captions differs according to granularity.

VI. Deep Learning Functional Descriptions from Software GUIs

The results of the analysis from the previous section demonstrate the presence of the latent patterns in the CLARITY dataset of screenshots and captions. In this section, we detail our methodology for investigating the capability of different customized DL models to learn these patterns to predict functional descriptions from two GUI representations.

A. Clarity Dataset Segmentation

We collected two different granularities of captions from users to derive the CLARITY dataset (Sec. IV-B). For the experiments in this section, we want to explore the model's ability to learn both high- and low-level functional descriptions. Thus, we split the CLARITY dataset into two groups, one containing only high level captions, and one containing only low level ones. We also created a third dataset combining the high and low level captions, in order to explore whether the predictive capabilities of the models improved by aggregating multiple granularities. It should be noted that each low-level caption was treated as a single caption (i.e. each low-level caption was treated as a separate data point) as is convention with datasets containing multiple captions [38]. Each screen in the dataset has both an associated screenshot and a GUI-metadata file. In order to make for a fair comparison of performance across various model configurations, we created consistent training, validation and test partitions (80%, 10%, 10% according to the number of images/GUI metadata files) to be used across models. The NL text used as input to the models was preprocessed according to the specific requirements for each model implementation [55], [56], [57].

TABLE III: Image Captioning Model Configs. used in Study

Model	Identifier	Caption Config.	Model Config.
im2txt	im2txt-h-imgnet	High	inception v3 trained on imagenet
	im2txt-l-imgnet	Low	
	im2txt-c-imgnet	Combined	
	im2txt-h-comp	High	Inception v3 fine-tuned on Component Dataset
	im2txt-l-comp	Low	
	im2txt-c-comp	Combined	
	im2txt-h-fs	High	Inception v3 fine-tuned on Full Screen Dataset
	im2txt-l-fs	Low	
	im2txt-c-fs	Combined	
NeuralTalk 2	ntk2-h-imgnet	High	VGGNet pre-trained on ImageNet
	ntk2-l-imgnet	Low	
	ntk2-c-imgnet	Combined	
	ntk2-h-ft	High	VGGNet pre-trained on ImageNet with Fine Tuning
	ntk2-l-ft	Low	
	ntk2-c-ft	Combined	
SAT	sat-h	High	VGGNet pre-trained on ImageNet
	sat-l	Low	
	sat-c	Combined	

TABLE IV: Subset of Model Hyper-paramters

Hyperparameter	im2txt	NeuralTalk2	SAT	Seq2Seq
Batch Size	64	16	17	64
Embedding Size	512	512	512	128
Decoder RNN Size/Units	512	512	1024	128
Optimizer	SGD	SGD	Adam	Adam
Initial Learning Rate	2	2	0.001	0.0001
Dropout Probability	0.7	0.7	0.3	0.8

B. Image Captioning Model Configurations

We customize, train, and test the three neural image captioning models, `im2txt`, `neuraltalk2`, and `show, attend, & tell (SAT)` (Sec. II-A), on the screenshots and captions of the CLARITY dataset. We choose to explore these three models due to their different underlying design decisions related to the type of utilized CNNs and RNNs (Sec. II-A), as these differences may affect their performance in our domain. It should be noted that in the course of our experiments, we make several customizations to these models through adaptions to pre-training and fine-tuning procedures. However, given the typical number of parameters that constitute these models, the training time can be quite prohibitive, even on modern hardware. Thus, to control our experimental complexity and investigate a number of model configurations that can be trained in a reasonable amount of time, we fix the values of the hyper-parameters for each model in our experiments. We derived our utilized hyper-parameter values by conducting random searches for optimal values of certain parameters, and chose optimal parameters reported in prior work for others. While we fix the hyper-parameters for these models, we instead customize the configurations of our image captioning models at the architectural level. Specifically, we investigate how training the "encoder" CNN using different datasets and training procedures effects the efficacy of the model predictions. This type of analysis allows us to more effectively flush out broader patterns related to the benefits and drawbacks of model design decisions. In the end, we trained more than 15 different configurations of the models (see Table III) over several machine months of computation.

*1) **im2txt Model Configurations & Training**:* For `im2txt`, we adapted Google's open source implementa-

tion of the model in TensorFlow [55]. Given the incredibly large number of parameters that need to be trained for the `im2txt` model, performing even relatively simple hyper-paramter searches proved to be computationally prohibitive for our experiments. Therefore, for this model we utilized the optimal set of parameters reported by Vinyals *et al.* [41] on similarly sized datasets. A subset of these hyper-parameter values are given in Table IV, whereas full configuration details can be found in our appendix. The publicly available implementation of Google's `im2txt` model utilizes the Inception v3 [58] image captioning architecture as its encoder CNN.

In past work, the inception model weights were initialized by training on the large-scale image classification dataset ImageNet [59], which contains "commonplace" image categoires. However, given that we are applying these models to very particular domain (predicting descriptions of software) it is unclear if an Inception v3 model trained on the broader ImageNet dataset would capture subtle semantic patterns in the CLARITY dataset. Therefore, we explored three different model configurations to explore this phenomena: one with Inception v3 pre-trained on ImageNet, and two with Inception v3 fine-tuned on domain specific-datasets. The first domain specific image dataset we utilize is the ReDraw cropped image dataset outlined in Sec. IV-A, which contains over 190k images of native Android GUI-components labeled with their type (*e.g.,* `Button`, `TextView`). The second domain specific image dataset we use consists of the full screenshots from the CLARITY dataset, labeled with their Google Play categories.

*2) **NeuralTalk2 Model Configurations & Training**:* For `neuraltalk2`, we adapted Karpathy *et al.'s* implementation written in Torch and lua [56]. We performed a brief randomized hyper-parameter search for this model, given its more efficient training time, using the optimal `im2txt` parameters as a starting point. The optimal values resulting from this search are provided in Table IV. For its CNN decoder, `neuraltalk2` makes use of a VGGNet [44] architecture pre-trained on the ImageNet [59] dataset. Unlike our im2txt configurations, we explore the effect of jointly fine-tuning `neuraltalk2`'s CNN and RNN. Thus, we explore two configurations of `neuraltalk2`, one that jointly fine tunes the pre-trained VGGNet on the CLARITY dataset, and one that does not perform fine-tuning. We followed a training procedure similar to that of our `im2txt` models, in that we trained our models on the high, low, and combined CLARITY caption training data for 500K iterations, saving model checkpoints every 2K iterations.

*3) **Show, Attend and Tell Model Configurations & Training**:* For the `SAT` model, we adapted the open-source implementation of the model in Tensorflow [60]. The hyperparameters that we used to train our model are shown in Table IV. The implementation used VGG16 [44] as its encoder CNN. We trained the `SAT` model on the CLARITY dataset for the low, high and combined captions for 500K iterations and kept the checkpoints after every 1K iterations. Note that due to the prohibitive training cost of this model, we did not explore using a fine-tuned VGGNet as we did with `neuraltalk2`.

TABLE V: Metadata Captioning Model Congfigurations

Model	Identifier	Caption Config.	Model Config.
Seq2Seq	seq2seq-h-type	High	Trained on GUI Component Types
	seq2seq-l-type	Low	
	seq2seq-c-type	Combined	
	seq2seq-h-text	High	Trained on GUI-Component Text
	seq2seq-l-text	Low	
	seq2seq-c-text	Combined	
	seq2seq-h-tt	High	Trained on GUI-Component Type + Text
	seq2seq-l-tt	Low	
	seq2seq-c-tt	Combined	
	seq2seq-h-ttl	High	Trained on GUI-component Type + Text + Location
	seq2seq-l-ttl	Low	
	seq2seq-c-ttl	Combined	

C. Metadata Captioning Model Configurations

To explore the ability to translate between the lexical representations of GUI-metadata and NL functional descriptions, we train and test an encoder-decoder neural language model using Google's `seq2seq` [57] framework. Note that recent work has proposed new models that take advantage of structural text properties [61], however, implementations of such models are generally not available, hence we leave the study of more advanced models for future work. We chose to utilize the default general-purpose architecture and hyper-parameters for this model, as they have been shown to be effective across a wide-range of machine translation tasks [62]. More specifically, our encoder network consists of a BRNN with Gated Recurrent Units (GRUs) and our decoder network consists of an RNN with LSTM units; hyperparameters are listed in Table IV.

To investigate the representative power of different attributes included in Android GUI-metadata, we create four configurations of GUI-metadata consisting of different attribute combinations (Table V). We chose to utilize these attribute combinations as they represent (i) the attributes that are most likely to have values, and (ii) represent a wide range of information types (*e.g.,* displayed text, component types, and spatial information). Note that `seq2seq` did not consistently converge for the high level caption dataset, thus we do not report these results. Consistent with the training of the other models, our implementation of the `seq2seq` model was trained to 500k iterations, with checkpoints every 2k iterations.

VII. Deep Learning Model Evaluation

To explore our core hypothesis set forth at the beginning of this paper, and evaluate our DL models described in Sec. VI, we perform a comprehensive empirical evaluation with two main *goals*: (i) intrinsically evaluate the predictive power of the models according to a well accepted machine translation effectiveness metric, and (ii) extrinsically evaluate the models by examining and rating the quality of the predicated functional NL descriptions. The *quality focus* of this evaluation is our studied models' ability to effectively predict accurate, concise, and complete functional descriptions. To aid in achieving our study goals, we define the following RQs:

- **RQ$_3$:** *How accurate are our model's predicted NL descriptions?*
- **RQ$_4$:** *How accurate, complete, & understandable are our model's predicted NL descriptions from the viewpoint of evaluators?*

TABLE VI: BLEU Score Evaluation Results for Models

Model	Capt.	Model Type	B_c	B_1	B_2	B_3	B_4
im2txt	High	im2txt-h-fs	12.4	24.8	12.6	6.7	5.3
	Low	im2txt-l-comp	27.0	45.6	31.8	20.0	10.1
	Comb.	im2txt-c-comp	30.3	51.7	35.9	22.1	11.6
NeuralTalk2	High	ntk2-h-imgnet	13.3	27.4	13.5	7.3	5.3
	Low	ntk2-l-ft	27.4	47.5	32.8	19.5	9.6
	Comb.	ntk2-c-ft	30.1	52.1	36.0	21.8	10.8
seq2seq	Low	seq2seq-l-type	18.1	44.6	17.0	7.9	0.24
	Comb.	seq2seq-c-type	16.9	38.9	14.7	6.0	0.08
SAT	High	sat-h	17.7	30.1	18.3	12.9	9.8
	Low	sat-l	35.0	52.5	38.7	28.1	20.7
	Comb.	sat-c	37.7	56.8	42.0	30.5	22.0
NeuralTalk2	Trained on Flickr8K		34.0	57.9	38.3	24.5	16.0
NeuralTalk2	Trained on MSCOCO		40.7	62.5	45.0	32.1	23.0
im2txt			42.6	66.6	46.1	32.9	24.6
SAT			45.7	71.8	50.4	35.7	25.0

A. Evaluation Methodology

*1) **RQ$_3$: Empirically Evaluating Model Accuracy**:* To evaluate the accuracy of our trained model's generated captions, we follow past work [40], [41] and report BLEU scores [63] of the predicted captions on the shared CLARITY test set of images and GUI-metadata. The BLEU score is a standard metric used in machine translation research that measures the textual similarity between a predicted caption (the output from a model) and a reference caption (the collected descriptions from humans in the CLARITY test set). The BLEU score can be measured according to the similarity of different subsequence lengths (*i.e.,* BLEU$_n$), and we report BLEU$_1$ through BLEU$_4$, as well as a composite score calculated as the average of these, as is convention [40], [41]. For the image captioning models, we use the *coco-caption* implementation of the BLEU score adapted for the CLARITY test set. For each test image across all image captioning models, three captions were generated using a beam width of 3 for the beam search across candidate predictions. The seq2seq models were evaluated in the same manner. We chose to utilize a beam width of 3 as an initial qualitative examination of our models' predictions showed this size to achieve a reasonable balance between prediction accuracy and model confidence. For the high-level captions, the three candidate captions were compared to the reference, and the overall average BLEU$_n$ scores were calculated for each model. For the low-level and combined captions, the predicted captions and reference captions were compared in a pairwise manner and overall average BLEU$_n$ scores were calculated for each model configuration.

*2) **RQ$_4$: Human Perceptions of Predicted Captions**:* To qualitatively evaluate our studied model's generated captions, we performed a large-scale study involving an additional 220 participants recruited from MTurk. We randomly sampled 220 screens from the CLARITY test set, and then predicted high, low, and combined captions for them using the optimal configurations of `im2txt`, `NeuralTalk2`, and `seq2seq` according to the composite BLEU score for each model and caption level combination. The `SAT` captions were not included in this study due to time constraints related to the model's training. We created a HIT wherein each participant viewed 11 screenshots paired with captions. Two of the 11 captions were reference high and low to serve as a control, while the other 9 captions came from the model predictions. Screens and caption pairs were arranged into HITs such that 1) no single HIT had two of the same screenshot, 2) each of the 11 types of captions (2 reference, 9 model) were included only once per HIT. The order of these captions was randomized per HIT to prevent bias introduced by identical caption ordering between HITs. By this arrangement, each screen-caption pair was evaluated by 11 participants. After viewing these screenshot-caption pairs, participants were asked to answer six evaluation questions. Three of these questions (EQ$_1$-EQ$_3$) were adapted from prior work that assessed the quality of automatically generated code summaries [21], and inquired about *accuracy*, *completeness*, and *understandability*, respectively. The three remaining questions (EQ$_4$-EQ$_6$), were free response and asked participants to explain *accuracies*, *inaccuracies*, and *improvements*. The full set of questions and HIT are in our online appendix [39]. Similar to the CLARITY dataset collection, each participant's response was thoroughly vetted by at least one author, and discarded if the answers were incomplete. Responses were collected until 220 HITs were completed by unique respondents.

B. Evaluation Results

*1) **RQ$_3$ Results: Evaluating BLEU Scores**:* We illustrate the BLEU score results for the most effective model configuration and checkpoint across all of our trained models in Table VI, whereas the results for other model configurations can be found in our online appendix [39] in addition to caption examples. The cells highlighted in blue illustrate the highest performing model configuration for each caption type. In general we observe that `SAT` exhibits the highest overall BLEU scores across all caption granularities. We speculate that this is attributable to the addition of the advanced attention mechanism in this model that is able to "focus" on varying image regions or features to effectively handle multiple caption granularities. In general, the `seq2seq` model performed quite poorly across the varying caption types, indicating a lower tendency for rich representation. Perhaps most interestingly, we see that the optimal model configurations for the `im2txt` framework were those where the CNN was conditioned on domain specific datasets. More specifically, the best high-level caption model was conditioned on full screenshots and the best low-level caption was conditioned on the cropped GUI-component screenshots.

Another general trend that emerges is the low-level and combined caption models tend to exhibit higher overall BELU scores compared to the high-level captions. This is somewhat intuitive, as it indicates that there are more natural connections between visual GUI and lexical patterns in the low-level captions, compared to the high-level captions that reflect more abstract functional descriptions. When examining the captions generated by the optimal configurations of each model, it is clear that `im2txt` and `SAT` produces a more diverse set of output captions than `neuraltalk2`, which could be considered as more useful in many software documentation tasks.

Finally, it is worth discussing how the BLEU scores of our models compare to those of the same models trained on the

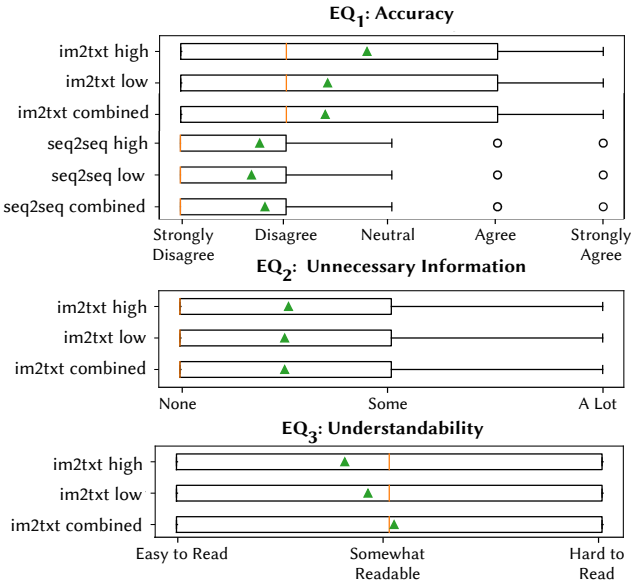


Fig. 5: Responses across models for EQ_1-EQ_3

more traditional Flickr8k [37] and MSCOCO [38] datasets given at the bottom of Table VI. Given the data-intensive nature of our DL models, and the much larger size of the MSCOCO dataset (≈123k images, each with 5 captions), we did not expect our models trained on the CLARITY dataset to outperform those trained on MSCOCO. Thus, unsurprisingly, we observe that on average, `im2txt`, `neuraltalk2`, and `SAT` models trained on the MSCOCO dataset outperform the same models trained on the CLARITY datasets by ≈ 10 BLEU score points for the combined and low level captions, and ≈ 27 points on high-level captions. However, when we examine the performance of Neuraltalk2 on the more similarly sized Flickr8K dataset (≈ 8K images, each with 5 captions) we observe comparable performance to the CLARITY low-level and combined datasets, with the `SAT` model narrowly outperforming the Flickr8K `neuraltalk2` model, with a slightly bigger discrepancy for the high-level captions. Overall, these results indicate that when compared with datasets of similar size, DL models trained on the CLARITY dataset exhibit similar performance.

*2) **RQ_4 Results: Human Evaluations**:* The results of EQ_1-EQ_3 for the model configurations with the best performance during the human study, in addition to the `seq2seq` accuracy scores, are summarized in Fig. 5. Complete results across all model configurations can be found in our online appendix. The responses to EQ_4-EQ_6 varied by the type of caption, and are provided in our appendix in full. Generally, `im2txt` fared the best in terms of accuracy, and was followed by `neuraltalk2` and `seq2seq` respectively. For `im2txt`, despite mixed reactions from participants, in many cases respondents verified that the caption was accurate (*e.g., "The description accurately describes the screen, it is in fact a terms and conditions screen."*) and suggested minor improvements similarly to the reference captions (*e.g., "It could add specifics about what the settings pertain to (i.e. security)"*). As illustrated in Fig. 5 the `im2txt` predictions were consistently rated as being readable and containing relevant information. It is also interesting to note that there appears to a mismatch between the performance as indicated by BLEU scores, and human perceptions, with the participants consistently rating the `im2txt` captions better than other models across EQ_1-EQ_3, despite `neuraltalk2` achieving a higher BLEU score for two model configurations.

VIII. Discussion & Learned Lessons

Lesson 1: Functional Descriptions of GUIs exhibit a high degree of naturalness and can be modeled using DL techniques. We observed that DL models trained on the low-level and combined datasets exhibit similar performance to models trained on general image captioning datasets of similar size (*e.g.,* Flickr8K). This indicates that GUI screenshots could be used to augment approaches for automated documentation.

Lesson 2: GUI-centric software documentation models benefit from being pre-trained on domain specific GUI data, as opposed to general image datasets (e.g., MSCOCO) The qualitative results of our model analysis illustrate that for `im2txt`, the most effective configurations were those trained on domain specific CNN datasets. This suggests a perceptible difference between the utility of image features learned from general datasets, compared to those learned on datasets more specific to software. This suggests that future work aiming to leverage DL models for GUI-centric program documentation should look to collecting and extracting features from large-scale GUI-related datasets.

Lesson 3: Future automated approaches for GUI-centric program documentation would likely benefit from combining the orthogonal semantics of screenshots and GUI-metadata. Our evaluation in this paper illustrates that the representational power of screenshots appears to be superior when applied to a software documentation task. However, given stark differences between these two modalities of information, we also observed that they encode orthogonal semantic patterns that could be combined for more effective documentation generation. One property we observed of certain captions generated by the image-based models was the effect of their limited vocabulary. For example, certain predicted captions similar to the following: *"The screen allows the user to select a <UNK>"*, wherein the `UNK` token represents missing token, which should be mapped to some unobserved app property, such as a "album cover" or "store location". However, such predictions could be combined with the vocabulary present in GUI metadata to help predict more complete, and accurate descriptions. Thus, a promising direction for future work is to *jointly encode* both screenshots and lexical GUI-metadata.

Lesson 4: Training image captioning models to predict specific or diverse pieces of functionality is difficult. Practical models for GUI-centric documentation should able to predict both specific pieces of information (e.g. the functionality of a particular button for a given method handler), and diverse functionality (being able to generate descriptions of functionality anywhere on a given screen). However, one aspect we observed across our models is that the most common observed types of functionality (*e.g.,* back buttons, menu buttons) corresponded to the functionalities that our

models predicted most often and most confidently on unseen screenshots. This is somewhat expected, as the models saw the most examples of such functionalities during training. Thus, the diversity of predictions is an open problem for future research. This problem can be partially mitigated by larger, more diverse datasets with specifically curated descriptions (such as extensions to the CLARITY dataset). However, it is likely that domain-specific models, or ensembles of models, may be required to more effectively predict diverse features.

Lesson 5: Future studies that evaluate automated GUI-centric documentation approaches should include human studies, as human perceptions of models may differ from automated reference-based metrics. One of the more surprising results of our study is that there seems to be a mismatch between humans perceptions of the captions generated by our DL models and the BLEU score metrics typically used to asses the accuracy of model predictions. This signifies that there are aspects of human perception that are not effectively captured in the BLEU metric, and possibly other translation metrics.

IX. Limitations & Threats to Validity

Internal Validity. Threats to *internal validity* correspond to unexpected factors in the experiments that may contribute to observed results. To derive our dataset we rely on MTurk and its workers to extract the high- and low-level descriptions per each screenshot. It should be noted that we did not ask MTurk workers to provide technical software documentation descriptions, but rather general descriptions of screen functionality at differing granularities. To minimize low quality captions we published the jobs for workers with more than 1k HITS, from English speaking countries, and HIT approval rate of more than 90%. Also, each successfully completed HIT was vetted by at least one of the authors to assure quality. If there was any question related to caption quality, at least one of the other authors stepped in to resolve the ambiguity. As a result 2,429 HITs were rejected due to low quality descriptions.

External Validity. Threats to *external validity* concern the generalization of the results. As with any collected dataset, there is a threat to external validity about the generalizability of the CLARITY dataset. However, we used a diverse set of popular apps from the Android domain, extracted popular screenshots from these apps, and the apps were captioned by a large and diverse set of MTurk workers. During our data collection process, we only collected 4 low-level captions per each screen in order to make the task feasible for MTurk workers as workers tend to abandon or perform poorly on long tasks. This means that, for certain screens with many GUI-components, some components may lack natural language descriptions. However, given the size of our dataset and the diversity of our screenshots and captions, we assert that our low-level captions are reasonably representative.

X. Related Work

DL for Image Captioning and GUIs. Hossain *et al.* [64] recently performed a wide-ranging study on DL models for image captioning, surveying the many different architectures and datasets used to evaluate them. However, this survey did not examine the ability of any image captioning model to predict functional descriptions of software. There have been a limited number of papers in the SE community that have applied DL techniques to GUI related data. Chen *et al.* [65] designed an approach that uses an NMT to translate an Android screenshot into a GUI-skeleton. However, their technique is able to predict GUI structure given an image, not functional natural language descriptions. Recently, Zhang et. al. [66] created a dataset of iOS image captions to train a model for captioning accessibility data. However, the authors do not make their dataset publicly available and target a different goal of accessibility data compared our goal of generating functional captions. Chen *et al.* investigated the use of DL image captioning models for applying labels to GUI-components in mobile apps [67], however, this approach only aims to predict short labels for a limited subset of GUI-components, whereas our study focuses upon predicting functional descriptions consisting of complete sentences for both individual GUI-components and entire screenshots.

GUI-based Analysis of Mobile Apps. GVT and GCat analyze the visual properties of GUIs to detect design violations and evolutionary changes [68], [69]. In contrast, we focus solely on image captioning techniques to provide functional program descriptions of screenshots. Approaches such as REMAUI [70], REDRAW [48], and `pix2code` [71] aim to automatically generate mobile app code given an app screenshot. Conversely, we leverage DL techniques to generate functional descriptions rather than source code using a pixel-based image as input. Chen *et al.* [72] introduced StoryDroid, for automatically generating visual storyboards of Android apps to help aid in the app design process. However, their approach is not capable of generating a functional description of an application from GUI data. Furthermore, Deka *et al.* showed how the Rico dataset could be navigated via semantic search using autoencoders [35]. UiRef [73] is an approach for resolving security and privacy concerns by considering semantics of GUI-components that request user's inputs. Moreover, Liu *et al.* [74] presented an approach for automatically classifying mobile app icons according to semantic GUI patterns. Xiao *et al.* proposed IconIntent that combines program analysis and icon classification to detect privacy sensitive GUI-components [75]. Different from this body of work, we aim to predict functional descriptions of GUIs for software documentation.

XI. Conclusion

In this paper, we have conducted one of the first comprehensive empirical investigations into the connection between GUI-related information, and functional descriptions of programs. We have derived the CLARITY dataset of GUI screenshots/metadata and NL captions, trained DL models on this dataset, and demonstrated their ability to bridge the semantic gap between visual and lexical program information.

Acknowledgment

This work was supported by the NSF CCF-2007246 & CCF-1955853 grants. Any opinions, findings, and conclusions expressed herein are the authors' and do not necessarily reflect those of the sponsors.

REFERENCES

[1] J.-C. Chen and S.-J. Huang, "An empirical analysis of the impact of software development problem factors on software maintainability," *J. Syst. Softw.*, vol. 82, pp. 981–992, June 2009.

[2] M. Kajko-Mattsson, "A survey of documentation practice within corrective maintenance," *Empirical Software Engineering*, vol. 10, pp. 31–55, Jan 2005.

[3] B. Dagenais and M. P. Robillard, "Creating and evolving developer documentation: Understanding the decisions of open source contributors," in *Proceedings of the Eighteenth ACM SIGSOFT International Symposium on Foundations of Software Engineering*, FSE '10, (New York, NY, USA), pp. 127–136, ACM, 2010.

[4] G. Garousi, V. Garousi-Yusifoglu, G. Ruhe, J. Zhi, M. Moussavi, and B. Smith, "Usage and usefulness of technical software documentation: An industrial case study," *Information and Software Technology*, vol. 57, pp. 664 – 682, 2015.

[5] A. Forward and T. C. Lethbridge, "The relevance of software documentation, tools and technologies: A survey," in *Proceedings of the 2002 ACM Symposium on Document Engineering*, DocEng '02, (New York, NY, USA), pp. 26–33, ACM, 2002.

[6] "The agile manfiesto http://agilemanifesto.org."

[7] J. Zhi, V. Garousi-Yusiflu, B. Sun, G. Garousi, S. Shahnewaz, and G. Ruhe, "Cost, benefits and quality of software development documentation," *J. Sys. Sof.*, Jan. 2015.

[8] B. Fluri, M. Wursch, and H. C. Gall, "Do code and comments co-evolve? on the relation between source code and comment changes," in *Proceedings of the 14th Working Conference on Reverse Engineering*, WCRE '07, (Washington, DC, USA), pp. 70–79, 2007.

[9] L. Moreno, A. Marcus, L. Pollock, and K. Vijay-Shanker, "JSummarizer: An automatic generator of natural language summaries for Java classes," in *2013 21st International Conference on Program Comprehension*, ICPC'13, pp. 230–232, May 2013.

[10] L. Moreno, J. Aponte, G. Sridhara, A. Marcus, L. Pollock, and K. Vijay-Shanker, "Automatic generation of natural language summaries for Java classes," in *2013 21st International Conference on Program Comprehension*, ICPC'13, pp. 23–32, May 2013.

[11] P. C. Rigby and M. P. Robillard, "Discovering Essential Code Elements in Informal Documentation," in *Proceedings of the 2013 International Conference on Software Engineering*, ICSE'13, (San Francisco, CA, USA), pp. 832–841, 2013.

[12] A. T. T. Ying and M. P. Robillard, "Selection and Presentation Practices for Code Example Summarization," in *Proceedings of the 22Nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, FSE'14, (Hong Kong, China), pp. 460–471, 2014.

[13] A. T. T. Ying and M. P. Robillard, "Code Fragment Summarization," in *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*, FSE'13, pp. 655–658, 2013.

[14] I. K. Ratol and M. P. Robillard, "Detecting fragile comments," in *2017 32nd IEEE/ACM International Conference on Automated Software Engineering*, ASE'17, pp. 112–122, Oct. 2017.

[15] M. J. Howard, S. Gupta, L. Pollock, and K. Vijay-Shanker, "Automatically mining software-based, semantically-similar words from comment-code mappings," in *2013 10th Working Conference on Mining Software Repositories*, MSR'13, pp. 377–386, may 2013.

[16] C. Treude and M. P. Robillard, "Augmenting API Documentation with Insights from Stack Overflow," in *Proceedings of the 38th International Conference on Software Engineering*, ICSE '16, (New York, NY, USA), pp. 392–403, 2016.

[17] G. Petrosyan, M. P. Robillard, and R. De Mori, "Discovering Information Explaining API Types Using Text Classification," in *Proceedings of the 37th International Conference on Software Engineering - Volume 1*, ICSE '15, (Florence, Italy), pp. 869–879, 2015.

[18] B. Li, C. Vendome, M. Linares-Vásquez, D. Poshyvanyk, and N. A. Kraft, "Automatically Documenting Unit Test Cases," in *2016 IEEE International Conference on Software Testing, Verification and Validation*, ICST'16, pp. 341–352, Apr. 2016.

[19] L. Moreno, W. Bandara, S. Haiduc, and A. Marcus, "On the Relationship between the Vocabulary of Bug Reports and Source Code," in *2013 IEEE International Conference on Software Maintenance*, ICSM'13, pp. 452–455, Sept. 2013.

[20] K. Moran, M. Linares-Vásquez, C. Bernal-Cárdenas, and D. Poshyvanyk, "Auto-completing Bug Reports for Android Applications," in *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, FSE'15, (Bergamo, Italy), pp. 673–686, 2015.

[21] L. Moreno, G. Bavota, M. D. Penta, R. Oliveto, A. Marcus, and G. Canfora, "Arena: An approach for the automated generation of release notes," *IEEE Transactions on Software Engineering*, vol. 43, pp. 106–127, Feb 2017.

[22] L. Moreno, G. Bavota, M. Di Penta, R. Oliveto, A. Marcus, and G. Canfora, "Automatic Generation of Release Notes," in *Proceedings of the 22Nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, FSE'14, pp. 484–495, 2014.

[23] S. Jiang, A. Armaly, and C. McMillan, "Automatically generating commit messages from diffs using neural machine translation," in *2017 32nd IEEE/ACM International Conference on Automated Software Engineering*, ASE'17, pp. 135–146, oct 2017.

[24] L. F. Cortés-Coy, M. Linares-Vásquez, J. Aponte, and D. Poshyvanyk, "On automatically generating commit messages via summarization of source code changes," in *2014 IEEE 14th International Working Conference on Source Code Analysis and Manipulation*, SCAM'14, Sep. 2014.

[25] P. Chatterjee, B. Gause, H. Hedinger, and L. Pollock, "Extracting Code Segments and Their Descriptions from Research Articles," in *Proceedings of the 14th International Conference on Mining Software Repositories*, MSR '17, (Buenos Aires, Argentina), pp. 91–101, 2017.

[26] M. Linares-Vásquez, B. Li, C. Vendome, and D. Poshyvanyk, "Documenting database usages and schema constraints in database-centric applications," in *Proceedings of the 25th International Symposium on Software Testing and Analysis*, ISSTA 2016, (New York, NY, USA), pp. 270–281, 2016.

[27] T. J. Biggerstaff, B. G. Mitbander, and D. E. Webster, "Program understanding and the concept assignment problem," *Commun. ACM*, vol. 37, pp. 72–82, May 1994.

[28] J. Guo, J. Cheng, and J. Cleland-Huang, "Semantically enhanced software traceability using deep learning techniques," in *Proceedings of the 39th International Conference on Software Engineering*, ICSE '17, (Piscataway, NJ, USA), pp. 3–14, 2017.

[29] M. P. Robillard, A. Marcus, C. Treude, G. Bavota, O. Chaparro, N. Ernst, M. A. Gerosa, M. Godfrey, M. Lanza, M. Linares-Vásquez, G. C. Murphy, L. Moreno, D. Shepherd, and E. Wong, "On-demand Developer Documentation," in *2017 IEEE International Conference on Software Maintenance and Evolution*, ICSME'17, pp. 479–483, Sept. 2017.

[30] "Android inches ahead of windows as most popular os - https://www.cnet.com/news/android-most-popular-os-beats-windows-statcounter/."

[31] K. Moran, C. Bernal-Crdenas, M. Linares-Vsquez, and D. Poshyvanyk, "Overcoming lanaguge dichotomies: Toward effective program comprehension for mobile app development," in *26th IEEE International Conference on Program Compreehnsion*, May 2018. To Appear.

[32] A. M. Memon, *A comprehensive framework for testing graphical user interfaces*. Ph.D., 2001. Advisors: Mary Lou Soffa and Martha Pollack; Committee members: Prof. Rajiv Gupta (University of Arizona), Prof. Adele E. Howe (Colorado State University), Prof. Lori Pollock (University of Delaware).

[33] A. Solem, "Designing computer documentation that will be used: Understanding computer user attitudes," in *Proceedings of the 4th Annual International Conference on Systems Documentation*, SIGDOC 85, (New York, NY, USA), p. 5556, Association for Computing Machinery, 1986.

[34] A. Alshayban, I. Ahmed, and S. Malek, "Accessibility issues in android apps: State of affairs, sentiments, and ways forward," in *Proceedings of the 42nd International Conference on Software Engineering*, ICSE 2020, (New York, NY, USA), p. to appear, ACM, 2020.

[35] B. Deka, Z. Huang, C. Franzen, J. Hibschman, D. Afergan, Y. Li, J. Nichols, and R. Kumar, "Rico: A mobile app dataset for building data-driven design applications," in *Proceedings of the 30th Annual Symposium on User Interface Software and Technology*, UIST '17, 2017.

[36] K. Moran, C. Bernal-Cardenas, M. Curcio, R. Bonett, and D. Poshyvanyk, "The ReDraw Dataset: A Set of Android Screenshots, GUI Metadata, and Labeled Images of GUI Components," Jan. 2019. Supported by NSF Grant CCF-1815186.

[37] M. Hodosh, P. Young, and J. Hockenmaier, "Framing image description as a ranking task: Data, models and evaluation metrics," *J. Artif. Int. Res.*, vol. 47, no. 1, pp. 853–899, 2013.

[38] T. Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common Objects in Context," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8693 LNCS, pp. 740–755, 2014.

[39] "Online appendix http://sagelab.io/Clarity/."
[40] A. Karpathy and L. Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, pp. 664–676, Apr. 2017.
[41] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: Lessons learned from the 2015 mscoco image captioning challenge," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, pp. 652–663, Apr. 2017.
[42] K. Xu, J. Ba, R. Kiros, K. Cho, A. C. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *ICML*, 2015.
[43] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1097–1105, 2012.
[44] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *International Conference on Learning Represen.*, vol. abs/1409.1556, 2014.
[45] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going Deeper with Convolutions," in *Computer Vision and Pattern Recognition*, 2015.
[46] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[47] M. Schuster and K. Paliwal, "Bidirectional recurrent neural networks," *Trans. Sig. Proc.*, vol. 45, pp. 2673–2681, Nov. 1997.
[48] K. P. Moran, C. Bernal-Cárdenas, M. Curcio, R. Bonett, and D. Poshyvanyk, "Machine learning-based prototyping of graphical user interfaces for mobile apps," *IEEE Transactions on Software Engineering*, pp. 1–1, 2018.
[49] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, Mar. 2003.
[50] I. Porteous, D. Newman, A. Ihler, A. Asuncion, P. Smyth, and M. Welling, "Fast collapsed gibbs sampling for latent dirichlet allocation," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, (New York, NY, USA), pp. 569–577, 2008.
[51] A. Hindle, E. T. Barr, Z. Su, M. Gabel, and P. Devanbu, "On the Naturalness of Software," in *International Conference on Software Engineering*, ICSE'12, pp. 837–847, 2012.
[52] M. Rahman, D. Palani, and P. Rigby, "Natural software revisted," in *Proceedings of the 41st International Conference on Software Engineering Companion*, ICSE '19, (Montreal, QC Canada), p. to appear, 2019.
[53] "Mit language modeling toolkit https://github.com/mitlm/mitlm"
[54] P. Koehn, *Statistical Machine Translation*. 1st ed., 2010.
[55] "Google's im2txt implementation https://github.com/tensorflow/models/tree/master/research/im2txt."
[56] "Neuraltalk2 implementation https://github.com/karpathy/neuraltalk2."
[57] "Google seq2seq framework https://github.com/google/seq2seq."
[58] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *2016 IEEE Conference on Computer Vision and Pattern Recognition*, June 2016.
[59] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.
[60] "Show, attend and tell implementation https://github.com/coldmanck/show-attend-and-tell."
[61] L. Sha, L. Mou, T. Liu, P. Poupart, S. Li, B. Chang, and Z. Sui, "Order-planning neural text generation from structured data," *The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI'17)*, vol. abs/1709.00155, 2017.
[62] D. Britz, A. Goldie, T. Luong, and Q. Le, "Massive Exploration of Neural Machine Translation Architectures," *ArXiv e-prints*, Mar. 2017.
[63] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: A method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, (Stroudsburg, PA, USA), pp. 311–318, 2002.
[64] M. Z. Hossain, F. Sohel, M. F. Shiratuddin, and H. Laga, "A comprehensive survey of deep learning for image captioning," *ACM Comput. Surv.*, vol. 51, pp. 118:1–118:36, Feb. 2019.
[65] S. Chen, L. Fan, C. Chen, T. Su, W. Li, Y. Liu, and L. Xu, "Storydroid: Automated generation of storyboard for android apps," in *Proceedings of the 41st International Conference on Software Engineering*, ICSE '19, (Piscataway, NJ, USA), pp. 596–607, IEEE Press, 2019.
[66] X. Zhang, L. de Greef, A. Swearngin, S. White, K. Murray, L. Yu, Q. Shan, J. Nichols, J. Wu, C. Fleizach, A. Everitt, and J. P. Bigham, *Screen Recognition: Creating Accessibility Metadata for Mobile Applications from Pixels*. New York, NY, USA: Association for Computing Machinery, 2021.
[67] J. Chen, C. Chen, Z. Xing, X. Xu, L. Zhu, G. Li, and J. Wang, "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 2020, (New York, NY, USA), p. to appear, ACM, 2020.
[68] K. Moran, C. Watson, J. Hoskins, G. Purnell, and D. Poshyvanyk, "Detecting and summarizing gui changes in evolving mobile apps," in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, ASE 2018, (New York, NY, USA), pp. 543–553, 2018.
[69] K. Moran, B. Li, C. Bernal-Cárdenas, D. Jelf, and D. Poshyvanyk, "Automated Reporting of GUI Design Violations in Mobile Apps," in *Proceedings of the 40th International Conference on Software Engineering Companion*, ICSE '18, (Gothenburg, Sweden), p. to appear, 2018.
[70] T. A. Nguyen and C. Csallner, "Reverse Engineering Mobile Application User Interfaces with REMAUI," in *Proceedings of the 2015 30th IEEE/ACM International Conference on Automated Software Engineering*, ASE'15, (Washington, DC, USA), pp. 248–259, 2015.
[71] T. Beltramelli, "Pix2code: Generating Code from a Graphical User Interface Screenshot," *CoRR*, vol. abs/1705.07962, 2017.
[72] Q. Chen and M. Zhou, "A neural framework for retrieval and summarization of source code," in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, ASE 2018, (New York, NY, USA), pp. 826–831, ACM, 2018.
[73] B. Andow, A. Acharya, D. Li, W. Enck, K. Singh, and T. Xie, "UiRef: analysis of sensitive user inputs in Android applications," in *Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, WiSec'17, (New York, New York, USA), pp. 23–34, 2017.
[74] T. F. Liu, M. Craft, J. Situ, E. Yumer, R. Mech, and R. Kumar, "Learning design semantics for mobile apps," in *The 31st Annual ACM Symposium on User Interface Software and Technology*, UIST '18, (New York, NY, USA), pp. 569–579, 2018.
[75] X. Xiao, X. Wang, Z. Cao, H. Wang, and P. Gao, "Iconintent: Automatic identification of sensitive ui widgets based on icon classification for android apps," in *Proceedings of the 41st International Conference on Software Engineering Companion*, ICSE '19, (Montreal, QC Canada), p. to appear, 2019.