Pixel to Code Introduction

Geping Chen
Tianbo Lu

pix2code: Generating Code from a Graphical User Interface Screenshot

- Tony Beltramelli

Team member

Geping Chen

MEng in CS

gepingc@vt.edu

Tianbo Lu

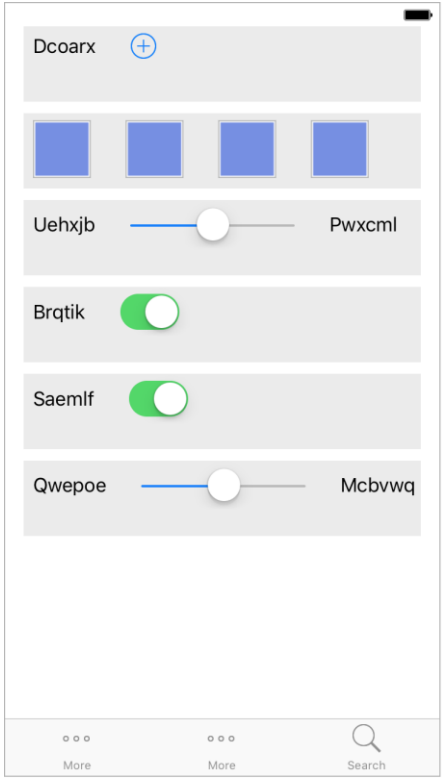
MEng in CS

ltianbo21@vt.edu

Problem to solve

- Implementing Graphical User Interface (GUI) code
  - time-consuming
  - specific to each target runtime system

- Transforming user interface screenshots provided by designers into computer code
  - Preventing developers from doing repeated work



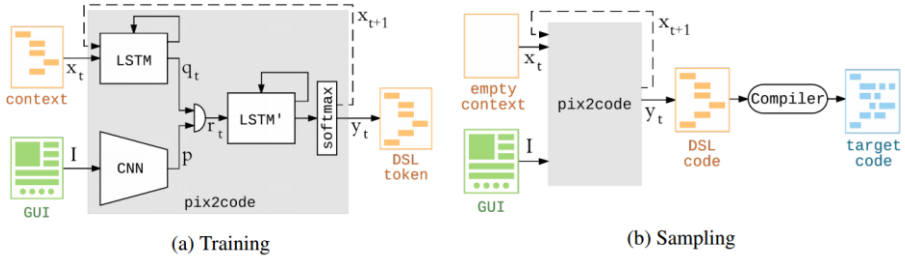
Approaches

Training

- Convolutional Neural Network (CNN)
  - unsupervised feature learning
- Recurrent Neural Network (RNN)
  - language modeling
- Decoder (LSTM)

Sampling

- Updating the input context each prediction (gradient descent)
- Compiling the output sequence of DSL token



(a) Training (b) Sampling

CNN

Encoder - Mapping the input GUI image to a fixed-length vector

- simple preprocessing
  - resize the image to 256 x 256 pixels
  - normalize the pixel values
- convolution
  - 3 x 3 receptive fields
  - stride 1
- pooling layer
  - max pooling
- activation
  - Rectified Linear Units (ReLU)

RNN

Encoder - context to feature vector

- DSL code
  - token-level modeling
  - reducing the size of the vocabulary
- Long Short-Term Memory (LSTM)
  - tokens spread out in a sequence
  - vanishing and exploding gradients

$$
\begin{aligned}
i_t &= \phi(W_{ix}x_t + W_{iy}h_{t-1} + b_i) \\
f_t &= \phi(W_{fx}x_t + W_{fy}h_{t-1} + b_f) \\
o_t &= \phi(W_{ox}x_t + W_{oy}h_{t-1} + b_o) \\
c_t &= f_t \bullet c_{t-1} + i_t \bullet \sigma(W_{cx}x_t + W_{cy}h_{t-1} + b_c) \\
h_t &= o_t \bullet \sigma(c_t)
\end{aligned}
$$

```
stack {
  row {
    label, switch
  }
  row {
    label, btn-add
  }
  row {
    label, slider, label
  }
  row {
    img, label
  }
}
footer {
  btn-more, btn-contact, btn-search, btn-download
}
```

Decoder and Classification

- the second LSTM layers
  - single feature vector (vision-encoded + language-encoded)
  - learns to model the relationship

- softmax layer
  - multi-class classification
  - sample one token at a time

$$
\begin{aligned}
p &= CNN(I) \\
q_t &= LSTM(x_t) \\
r_t &= (q, p_t) \\
y_t &= softmax(LSTM'(r_t)) \\
x_{t+1} &= y_t
\end{aligned}
$$

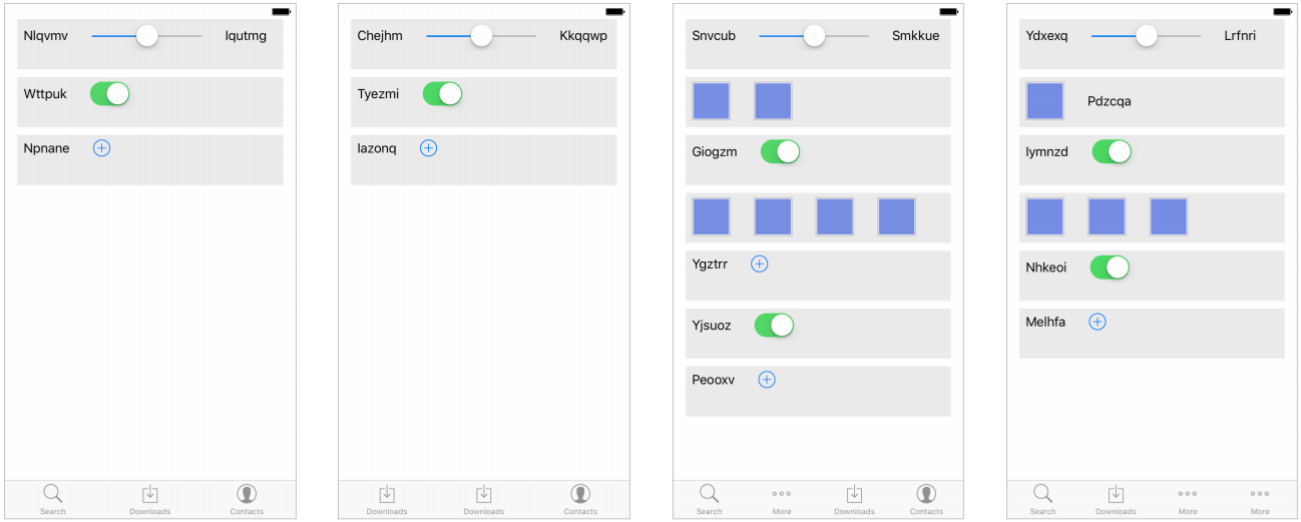
Training and Sampling

- sliding window
  - segment the DSL input files
  - trade-off (long-term dependencies & cost)
- loss (backpropagation)

$$L(I, X) = -\sum_{t=1}^{T} x_{t+1} \log(y_t)$$

- RMSProp algorithm
  - learning rate set to 1e^(-4)
  - clipping the output gradient to the range [-1.0, 1.0]
- dropout regularization
- update token sequence
  - $\mathbf{X}t$, ..., $\mathbf{X}T$-1 are set to $\mathbf{X}t$+1, ..., $\mathbf{X}T$ (discard $\mathbf{X}t$)

Results



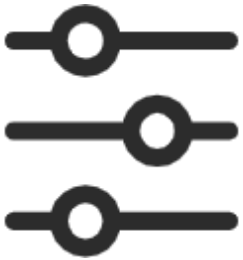
(a) Groundtruth GUI 1 (b) Generated GUI 1 (c) Groundtruth GUI 2 (d) Generated GUI 2

Dataset type	Error (%)		
	greedy search	beam search 3	beam search 5
iOS UI (Storyboard)	**22.73**	25.22	23.94
Android UI (XML)	**22.34**	23.58	40.93
web-based UI (HTML/CSS)	12.14	**11.01**	22.35

Problem

- select the right color/ style
- GUI with long lists of graphical components

Lesson learned



environment

dependency conflict

preprocessing

incomplete dataset

modeling

long training time

Future work

- Generative Adversarial Networks GANs
  - fine-tune results
- More training data
  - crawling the internet to collect a dataset
- Focusing on web-based GUIs
  - no need to do data synthesis

Thank you