TCET

DEPARTMENT OF INFORMATION TECHNOLOGY (IT)

Credit Based Grading System [CBGS - 2012(R)]/Choice Based Credit and Grading Scheme [CBCGS - 2016(R)]

University of Mumbai

Project Group Information

Group No	Roll Nos.	Student Names	Class & Div.	Project Title	Domain Name	Guide Name	Student Email ID	Student Phone Nos.
33	17	Samdharsi Kumar	BE-IT-B	Turning Design Mockups into Code using Deep Learning	Machine learning & Artificial Intelligence	Mrs.Shruti Mathur	samdharsi.kumar@gmail.com	9011124204,
33	56	Esha Vijayvargiya					eshavjy1@gmail.com	7977299823,
33	59	Harsh Vyas					harshvyas065@gmail.com	9022531484.

Index

Sr. No.	Topic	Guides Remark	Date	Sign
1	Idea Finalization	To do feasibility Study as per our scope of the project		
2	Feasibility Study	To refer related research papers and articles		
3	Software and Hardware Requirements	Prepare synopsis with literature survey		
4	Synopsis	Correction in methodology flowchart and DFD.		

Screenshot to code

Introduction

The process of implementing client-side software based on a Graphical User Interface (GUI) mock-up created by a designer is the responsibility of developers. Implementing GUI code is, however, time-consuming and prevent developers from dedicating the majority of their time implementing the actual features and logic of the software they are building. Moreover, the computer languages used to implement such GUIs are specific to each target platform; thus resulting in tedious and repetitive work when the software being built is expected to run on multiple platforms using native technologies. In this paper, we describe a system that can automatically generate platform-specific computer code given a GUI screenshot as input. We extrapolate that a scaled version of our method could potentially end the need for manually programmed GUIs.

Our first contribution is pix2code, a novel approach based on Convolutional and Recurrent Neural Networks allowing the generation of computer code from a single GUI screenshot as input. Our model is able to generate computer code from the pixel values of the input image alone. That is, no engineered feature extraction pipeline is designed to pre-process the input data. Our experiments demonstrate the effectiveness of our method for generating computer code for various platforms (i.e. iOS and Android native mobile interfaces, and multi-platform web-based HTML/CSS interfaces) without the need for any change or specific tuning to the model. In fact, pix2code can be used as such to generate code written in different target languages simply by being trained on a different dataset. A video demonstrating our system is available online1.

Our second contribution is the release of our synthesized datasets consisting of both GUI screenshots and associated source code for three different platforms. They will be made freely available2 upon publication of this paper to foster future research.

Background

The automatic generation of programs using machine learning techniques is a relatively new field and program synthesis in a human-readable format have only been addressed very recently. A recent example is DeepCoder [2], a system able to generate computer programs by leveraging statistical predictions to augment traditional search techniques. In another work by Gaunt et al. [4], the generation of source code is enabled by learning the relationships between input-output examples via differentiable interpreters. Furthermore, Ling et al. [11] recently demonstrated program synthesis from a mixed natural language and structured program specification as input. It is important to note that most of these methods rely on Domain Specific Languages (DSLs); computer languages (e.g. mark-up languages, programming languages, modelling languages) that are designed for a specialized domain but are typically more restrictive than full-featured computer languages. Using DSLs thus limit the complexity of the programming language that needs to be modelled and thus reduce the size of the search space.

Although the generation of computer programs is an active research field as suggested by these breakthroughs, code generation from visual inputs is still an unexplored research area. This paper is, to the best of our knowledge, the first work attempting to address this very problem. In order to exploit the graphical nature of our input, we can borrow methods from the computer vision literature. In fact, an important number of research [19, 18, 3, 9] have shown that deep neural networks are able to learn latent variables describing objects in an image and generate a variable-length textual description of the objects and their relationships. All these methods rely on two main components. First, a Convolutional Neural Network (CNN) transforming the raw input image into an intermediary learned representation. Second, a Recurrent Neural Network (RNN) performing language modelling on the textual description associated with the input picture. Combining both neural network architectures allows the generation of image captions with impressive results.

Literature Survey

Reference No.	Paper Title	Author	Year of Publication	Key Findings	Research Gaps
1	Pix2code: Generating	Tony Beltramelli		Transforming a graphical	The code, which is

	Code from a Graphical User Interface Screenshot		2016	user interface screenshot created by a designer into computer code is a typical task conducted by a developer in order to build customized Software.	generated from the screenshot, is restricted to only few coding platforms.
2	What You Get Is What You See: A Visual Markup Decompiler	Yuntian Deng, Anssi Kanervisto, Alexander M. Rush	2015	Building on recent advances in image caption generation and optical character recognition (OCR).	Standard mode of language is required for the generation of code.
3	Neural machine translation by jointly learning to align and translate	Dzmitry Bahdanau	2017	Neural machine translation is a recently proposed approach to machine translation.	The time required to generate code from the screenshot depends upon the

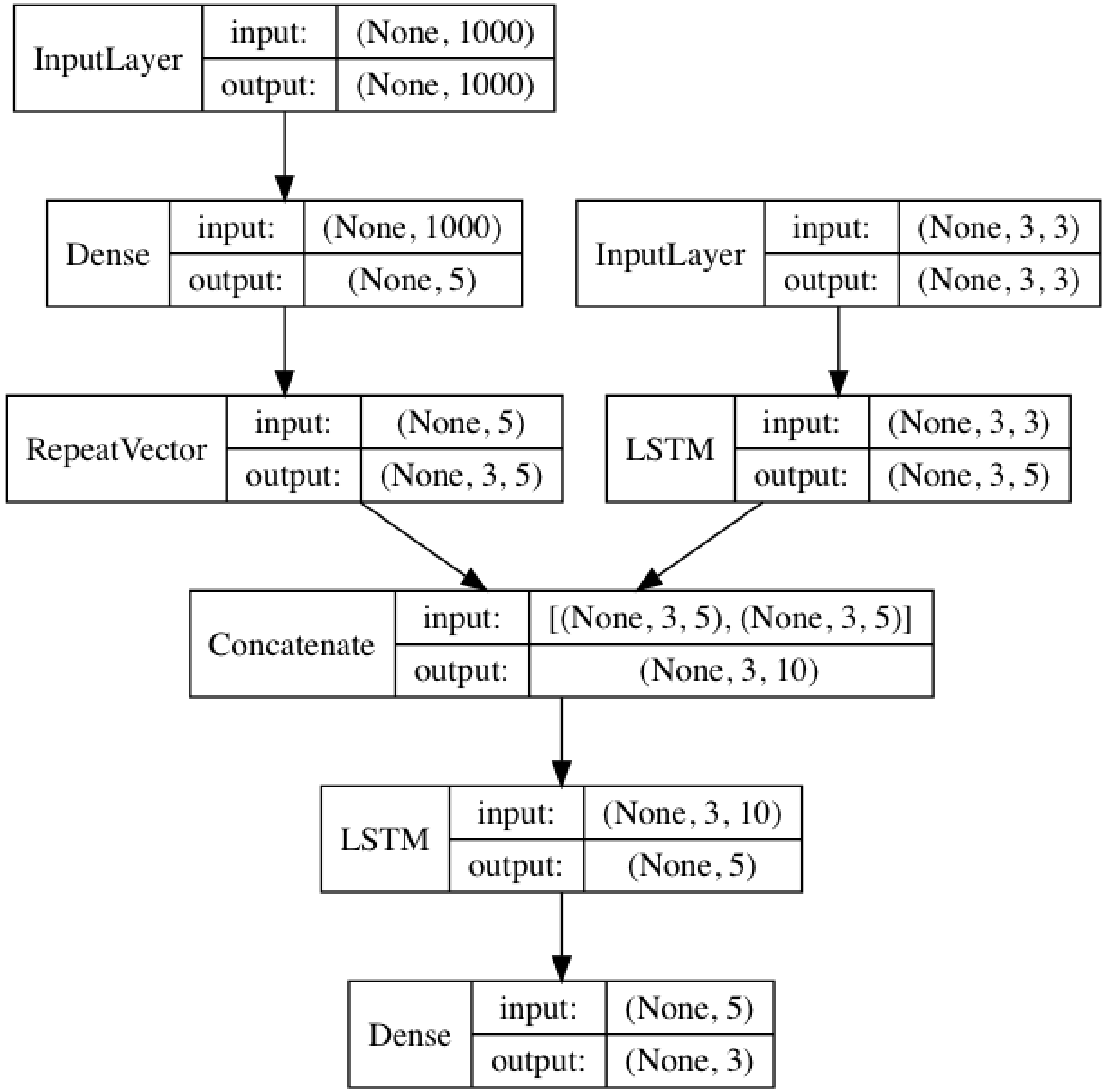
				Unlike the traditional statistical machine translation.	complexity of the webpage.

Problem Definition

The task of generating computer code written in a given programming language from a GUI screenshot can be compared to the task of generating English textual descriptions from a scene photography. In both scenarios, we want to produce variable-length strings of tokens from pixel values. We can thus divide our problem into three sub-problems. First, a computer vision problem of understanding the given scene (i.e. in this case, the GUI image) and inferring the objects present, their identities, 2 positions, and poses (i.e. buttons, labels, element containers). Second, a language-modelling problem of understanding text (i.e. in this case, computer code) and generating syntactically and semantically correct samples. Finally, the last challenge is to use the solutions to both previous sub-problems by exploiting the latent variables inferred from scene understanding to generate corresponding textual descriptions (i.e. computer code rather than English) of the objects represented by these variables.

Methodology

Dataflow Diagram:


InputLayer
input:
(None, 1000)
output:
(None, 1000)
Dense
input:
(None, 1000)
output:
(None, 5)
InputLayer
input:
(None, 3, 3)
output:
(None, 3, 3)
RepeatVector
input:
(None, 5)
output:
(None, 3, 5)
LSTM
input:
(None, 3, 3)
output:
(None, 3, 5)
Concatenate
input:
[(None, 3, 5), (None, 3, 5)]
output:
(None, 3, 10)
LSTM
input:
(None, 3, 10)
output:
(None, 5)
Dense
input:
(None, 5)
output:
(None, 3)

Objective & Scope

Neural machine translation is a recently proposed approach to machine translation. Unlike the traditional statistical machine translation, the neural machine translation aims at building a single neural network that can be jointly tuned to maximize the translation performance. The models proposed recently for neural machine translation often belong to a family of encoder–decoders and encode a source sentence into a fixed-length vector from which a decoder generates a translation. In this paper, we conjecture that the use of a fixed-length vector is a bottleneck in improving the performance of this basic encoder–decoder architecture, and propose to extend this by allowing a model to automatically (soft-) search for parts of a source sentence that are relevant to predicting a target word, without having to form these parts as a hard segment explicitly. With this new approach, we achieve a translation performance comparable to the existing state-of-the-art phrase-based system on the task of English-to-French translation. Furthermore, qualitative analysis reveals that the (soft-) alignments found by the model agree well with our intuition.

References

- D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473, 2014.
- M. Balog, A. L. Gaunt, M. Brockschmidt, S. Nowozin, and D. Tarlow. Deepcoder: Learning to write programs. arXiv preprint arXiv:1611.01989, 2016.
- B. Dai, D. Lin, R. Urtasun, and S. Fidler. Towards diverse and natural image descriptions via a conditional gan. arXiv preprint arXiv:1703.06029, 2017.

Conclusion

In this project, we presented pix2code, a novel method to generate computer code given a single GUI image as input. While our work demonstrates the potential of such a system to automate the process of implementing GUIs, we only scratched the surface of what is possible. Our model consists of relatively few parameters and was trained on a relatively small dataset. The quality of the generated code could be drastically improved by training a bigger model on significantly more data for an extended number of epochs. Implementing a now-standard attention mechanism [1, 22] could further improve the quality of the generated code. Using one-hot encoding does not provide any useful information about the relationships between the tokens since the method simply assigns an arbitrary vectorial representation to each token. Therefore, pre-training the language model to learn vectorial representations would allow the relationships between tokens in the DSL to be inferred (i.e. learning word embedding's such as word2vec [13]) and as a result alleviate semantical error in the generated code. Furthermore, one-hot encoding does not scale to very big vocabulary and thus restrict the number of symbols that the DSL can support.

Gantt chart

	January		February		March		April		May	
Objective 1										
Activity 1.1	🟦									
Activity 1.2		🟦								
Objective 2										
Activity 2.1		🟦	🟦	🟦						
Activity 2.2				🟦						
Activity 2.3				🟦	🟦					
Objective 3										
Activity 3.1					🟦					
Activity 3.2						🟦				
Objective 4										
Activity 4.1						🟦				
Activity 4.2							🟦	🟦		
Activity 4.3								🟦	🟦	
Activity 4.4									🟦	🟦

(Project Guide)	(Project Coordinator)	(Signature of HOD)