УДК 004.832.22

ОБЗОР СИСТЕМ ПО ГЕНЕРАЦИИ СТАТИЧНОГО КОДА ВЕБ-СТРАНИЦЫ

Никитин И.В.

Аспирант кафедры Информатика,

ФГБОУ ВО «РЭУ им. Г.В. Плеханова»,

Россия, г. Москва

Аннотация

Данное исследование является обзором особенностей реализации системы Sketch2Code по генерации статичного исходного кода веб-страницы на основе изображения макета этой страницы. Также рассмотрены и другие работы, в которых выполняется схожая задача по генерации исходного кода программы. Благодаря рассмотренным исследованиям на тему генерации исходного кода веб-страницы по изображению, сформировано представление об архитектуре систем, способных выполнять поставленную задачу, а также выявлена отличительная особенность в виде наличия предметно-ориентированного языка.

Ключевые слова: кодогенерация; изображение макета; веб-страница; предметно-ориентированный язык; архитектура;

OVERVIEW OF THE SYSTEMS FOR GENERATING STATIC WEB-PAGE CODE

Nikitin I.V.

Graduate student,

Plekhanov Russian University of Economics,

Russia, Moscow

Annotation

This study is an overview of the implementation features of the Sketch2Code system for generating static source code for a web page based on an image of the page layout. Other works that perform a similar task of generating the source code of a program are also considered. Thanks to the overviewed studies on the topic of generating the source code of a web page based on an image, an idea of the architecture of systems capable of performing the task has been formed, and a distinctive feature in the form of the presence of a domain-specific language has been identified.

Keywords: code generation; layout image; webpage; domain-specific language; architecture;

Введение

В современном мире широкое распространение получили веб-приложения. С их помощью решается огромный спектр задач: от покупки товаров, не выходя из дома (например, маркетплейс Ozon [8]), до оформления запросов в госучреждения (например, с использованием Госуслуг [1]). В связи с этим различные компании как малого, так и крупного бизнеса вкладывают большое количество средств в работу своей информационной инфраструктуры.

Разработка и создание веб-приложения – это сложный и комплексный процесс, состоящий из проработки множества различных частей, которые впоследствии объединяются в итоговый продукт. Одной из таких частей является реализация клиентской части приложения, с которой в будущем будет взаимодействовать пользователь. С точки зрения исходного кода, клиентская часть делится на три фрагментов: html-код, с помощью которого задается общая структура страницы; css-файлы, которые используются для стилевого оформления страницы, что позволяет гибко настраивать внешний вид; js-скрипты, добавляющие возможности для интерактивного взаимодействия. Для

реализации веб-приложения обычно необходима работа, как минимум, двух сотрудников: дизайнера, который предоставляет макет будущей страницы, и разработчика, который реализует страницу по макету и добавит необходимую интерактивность.

Существует несколько способов уменьшения затрат на разработку своего веб-приложения. Один из вариантов - это специальные сайты-конструкторы, в которых с помощью заданного набора готовых компонентов можно создать свое собственное веб-приложение (например, Tilda [12]). Проблема таких приложений заключается в ограниченности набора компонентов для создания своего приложения, из-за чего полученные результаты могут выглядеть однотипно. Поэтому другим вариантом решения задачи минимизации расходов может быть направление, когда на основе изображения макета веб-приложения можно было бы получить код самой веб-страницы. Макет будущего веб-приложения позволяет стилизовать его, как хочет человек, а дальше система на основе изображения данного макета может предоставить исходный код, для реализации подобной страницы.

В последние годы бурный рост переживают технологии, связанные с машинным обучении, в частности, с нейронными сетями. Одним из направлений машинного обучения является кодогенерация, цель которой – уменьшить затраты на разработку за счет использования машинного, а не человеческого, труда для создания исходного кода программы. Актуальность работы заключается в исследовании особенностей инструментов, способных с помощью технологии машинного обучения создавать исходный код. Данное исследование позволит выявить слабые места подобных систем для их устранения в будущем.

Для того, чтобы понять, как можно улучшить существующие инструменты, связанные с кодогенерацией на основе изображения, необходимо понять, а какими свойствами такие системы уже обладают. Цель данного исследования заключается в изучении принципов и особенностей работы

инструментов кодогенерации на основе изображения на примере систем Sketch2Code и HTML Code Generation. Результатом исследования является формирование списка особенностей, на основе которых построены указанные выше системы.

Обзор системы Sketch2Code

Рассмотрим особенности работы системы Sketch2Code на основе публикации [14]. Задача инструмента - распознавание изображений ручного прототипа и генерации кода, на основе изображения.

Поднимая проблему распознавания изображения и генерации кода, авторы выделяют несколько подзадач, которые необходимо решить. Первая – научиться распознавать объекты на изображении. В этой части рассматриваются различные подходы для распознавания изображения, которые существуют, в частности архитектура RetinaNet [13]. Вторая – выбрать подходящий формат данных для вывода результатов. В частности, в этой секции оригинальной статьи приводятся примеры систем DeepCoder и HTML Code Generation, которые будут более подробно рассмотрены позже в рамках данного исследования. Кроме того, авторы упоминают программу REMAUI [9], интересная особенностью которой заключается в том, что она генерирует слишком сложный и подробный исходный код, который с трудом воспринимается человеком. Это довольно важный и интересный аспект, который необходимо учитывать в будущем: так как с полученным в результате предсказания системы исходным кодом в будущем предстоит работать человеку, необходимо, чтобы этот код был понятный и читаемый.

Стоит отметить, что кроме проблемы просто распознавания элементов на изображении, существует вероятность наложения одного элемента на другой. Здесь возможны два состояния: компоненты должны быть распознаны как два отдельных элемента и, соответственно, представлены по-разному в исходном коде, или как один элемент. В качестве примеров подобных ситуаций, авторы

статьи [14] приводят текст и интерактивный элемент, которые могут быть в одной форме, но при этом отдельными элемента; с другой стороны, у интерактивного элемента также может быть текст, без которого подобный элемент не может существовать, поэтому они должны восприниматься как единое целое. В качестве критерия выбора какая из двух ситуаций перед системой стоит, авторы остановились на определении процента пересечения: если он больше 50%, то это второй случай, иначе первый.

Полученный исходный код должен исполняться в приложении для работы с веб-страницами – браузере. Однако в настоящее время очень велико количество всевозможных архитектурных решений для построения веб-страниц (виртуальное DOM-дерево, SPA-приложения и т.д.). Поэтому в качестве выходных данных используется специальный объект, описывающий состояние изображения, который в дальнейшем может с помощью отдельного инструмента без использования систем машинного обучения преобразован в html-код. В статье [14] подобный инструмент также представлен.

С учетом описанных особенностей, была реализована система Sketch2Code. Для ее обучения использовались два отдельных набора данных: первый с описанием элементов, второй – с описанием позиций этих элементов. Кроме того, были использованы изображения с разной степенью затемненности для того, чтобы этот фактор также учитывался при обучении (в жизни фото может быть сделано при любых обстоятельствах, поэтому система должна быть готова работать с различными входными данными). В итоге система хорошо себя показала, продемонстрировав приемлемое время работы и правильность определения при различном освещении. Более подробные результаты экспериментов можно найти непосредственно в статье.

Исходного кода данной работы нет, поэтому выводы о реализации получится сделать только на основе статьи [14], которая и была рассмотрена. Важно отметить, что работает данная схема только с прототипными изображениями, что вносит ограничения в область применения. Во-первых, все

базовые изображения стандартизированы, то есть определенный набор элементов с строго заданным вариантом изображения, от которого не получится отступить в случае необходимости. Во-вторых, так как элементы прототипные, в них опущена необходимость в тексте, которая также опущена и в реализации.

Обзор системы HTML Code Generation

В статье [14] упоминаются система HTML Code Generation. Система имеет открытый исходный код [4], а поэтому был произведен его анализ для выявления особенностей работы. Данный инструмент имеет веб-интерфейс, с помощью которого можно загрузить изображение макета веб-страницы, а далее получить код этой самой страницы. В качестве примера, мной был загружен снимок экрана главной страницы доски объявлений Авито [2]. Результат работы инструмента можно увидеть на рисунке 1.

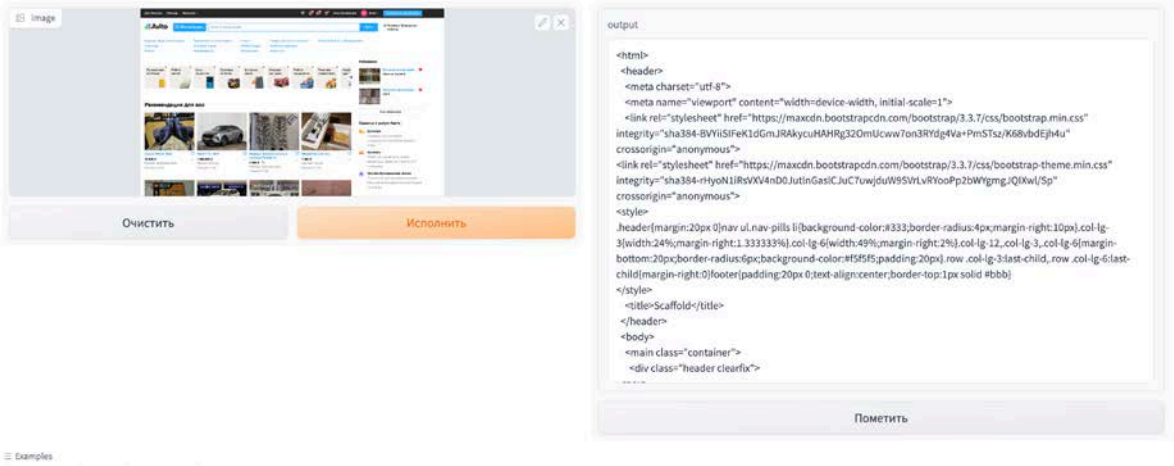

Рис. 1 – Пример работы системы HTML Code Generation (рис. автора)

Так как инструмент возвращает исходный код веб-страницы, полученный код был открыт в онлайн редакторе, в результате чего была получена веб-страница, представленная на рисунке 2.

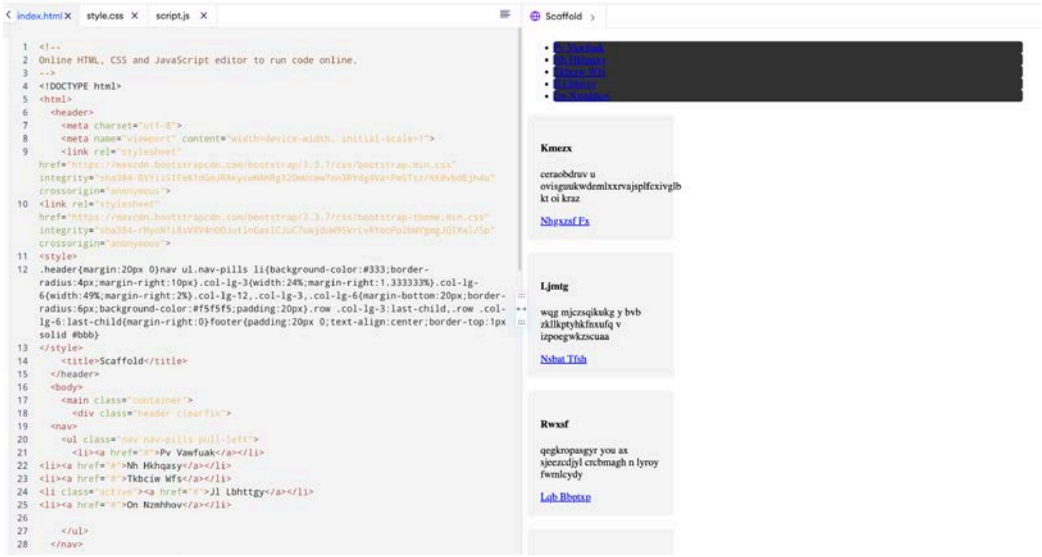

Рис. 2 – Разметка и результат ее применения по итогам работы системы HTML Code Generation (рис. автора)

Заметно, что итоговый результат не сильно похож на тот, которой был изначально передан в систему. Можно обратить внимание на то, что текст в сгенерированном варианте выглядит как случайный набор букв, а стилевое оформление является довольно скудным. При этом видно, что некоторые элементы уже имеют правильную семантику (например, являются списками с точки зрения структуры html-документа).

Реализация системы HTML Code Generation

Так как решение находится в открытом доступе, было проведено исследование его реализации, чтобы понять какие подходы использовал автор и какие есть особенности работы данной системы. В исходном коде присутствует изображение архитектуры приложения, которое представлено на рисунке 3. HTML Code Generation состоит из двух частей: части по распознаванию изображения и части по генерации кода.

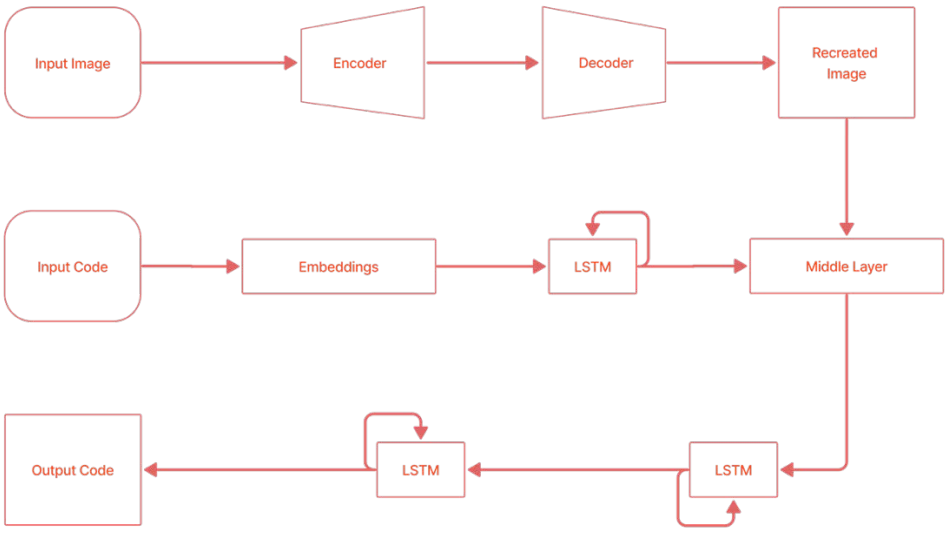

Рис. 3 – Архитектура системы HTML Code Generation (источник - https://github.com/taneemishere/html-code-generation-from-images-with-deep-neural-networks/blob/main/resources-for-md/model-architecture.png)

В данной работе нет распознавания текста, т.е. при составлении итогового кода, который выводится как результат работы программы, на месте необходимого текста выводится случайно сгенерированный текст.

Для решения задачи генерации валидного кода в работе [4] также, как и в работе [14], используется свой специальный формат для описания представления страницы. Этот формат описан с помощью предметно-ориентированного языка или DSL языка. Использование такого языка предполагает наличие специального словаря, который сопоставляет определенный символ словаря с кодом, соответствующим этому символу.

Работа модели состоит из нескольких этапов. Во-первых, набор данных разбивается на тренировочный набор данных и тестовый. Изначальный набор данных заранее подготовлен, состоит из файлов двух расширений: png (изображения) и gui (представление изображений через предметно-ориентированный язык, DSL). Разбиение на два набора происходит по принципу случайной выборки.

Затем на тренировочном наборе данных выполняется функция, подготавливающая изображения для дальнейшей работы. Каждая картинка уменьшается в размере до квадрата размером 255 на 255, который затем сохраняется и используется уже непосредственно в обучении. Выбор такого размера картинок связан с тем, что при самой обработке изображения используются маленькие квадраты размером 3 на 3. Эффективность работы подобной системы описана в статье [5]. Выбор квадратов размером 3 на 3 обусловлен тем, что это минимальный размер, при котором есть понимание ориентации в пространстве. Обучение на такой выборке показало себя наиболее эффективным.

После всех подготовительных работ модель начинает обучатся. В начале каждый файл с gui расширением обрабатывается, и составляется словарь соответствия между тегом языка DSL и изображением, соответствующим этому тегу. При этом для задания соответствия используется подход с быстрым кодированием (one-hot encoding). Сам словарь состоит из 15 элементов, поэтому кодирование рассчитано на массив размером 15 элементов. Каждый файл обрабатывается поэлементно для того, чтобы модель в будущем могла сравнить тот вариант, который предложила она, с тем, который реально используется. После этого с использование библиотек Tenserflow [11] и Keras [6] система начинает поиск зависимостей между входными и выходными данными.

Сама система HTML Code Generation основана на другой работе – pix2code, в которой также создавалась система, способная генерировать исходный код на основе изображения. Исследования и разработка системы pix2code представлены в статье [10]. В ней автор выделяет три основные проблемы, которые необходимо решить для получения рабочей системы: распознать изображение; сгенерировать работоспособный и правильный код; совместить промежуточные результаты в выходные данные (итоговую разметку).

Если сравнить работы pix2code и HTML Code Generation, то можно заметить, что архитектурно они почти повторяют друг друга: в обоих случаях процесс обучения идет способом, описанным выше. Основное отличие между ними заключается в том, что HTML Code Generation использует более современные технологии (те же Keras и Tenserflow), а также в ней присутствуют методы для оценки по системе BLEU.

Работы [14], [4], [10] используют свой отдельный формат для того, чтобы система во время обучения работала с ним, а не с исходным кодом напрямую. Как упоминалось ранее, для этого используется предметно-ориентированный язык. Использование именно такого подхода основано на работе системы DeepCoder [7], являющейся примером системы, построенной на дифференцируемых интерпретаторах и предметно-ориентированном языке, и которая представила способ упрощения генерации исходного кода.

Система DeepCoder [7] была разработана совместно инженерами из Майкрософт и университета Кембридж. В этой работе авторы хотели научиться генерировать и запускать программу с использованием решений с нейронными сетями. В процессе исследования, авторы DeepCoder столкнулись со следующей особенностью: для успешной генерации исходного кода, необходимо иметь словарь потенциальной программы, который можно использовать - предметно-ориентированный язык. Одна из проблем – это наполнение такого языка он не должен быть достаточно низкоуровневым, иначе в нем сложно будет ориентироваться, но при этом и не должен быть достаточно высокоуровневым, для сохранения гибкости при построении программ. Другая проблема – это выбор определенного знака этого языка. Для решения одной и той же задачи может подходить несколько разных знаков DSL-языка. Поэтому для более успешного поиска составить отдельную функцию, которая добавит ранжирование этих знаков, чтобы выбирать по нескольким критериям наиболее подходящий. В этом месте уже помогают нейронные сети, которые и использовали авторы статьи [7]. В работах [4] и [10]

используются нейронные сети для поиска подходящего значения из языка DSL. Для этого использовалась сеть, состоящая из энкодера и декодера для получения связи между набор знаков и вектора значений и наоборот. Энкодер реализован через архитектуру нейронной сети с прямой связью. Плюсом такого подхода авторы назвали простоту обучения, но при этом к минусам относят наличие верхней границы в входных и выходных данных.

Рассмотренный подход с DSL-языком может показаться несколько устаревшим, так как были представлены более 5 лет назад, однако они является актуальным и сегодня, так как лежит в основе других работ, которые в основном улучшают систему pix2code для получения более точных результатов. Так, например, в работе [15] улучшают подход для определения метрики BLEU для более точного определения метрик систему в купе с улучшением самой системы. Другой пример – работа [3]. В ней автор сосредоточился на улучшении генерации именно Android экранов, воспользовавшись более актуальными архитектурными решениями: двунаправленными LSTM, а также используя архитектуру ResNet для обработки изображения.

Заключение

Генерация статичного кода на основе изображения макета может звучать задачей сложной, однако уже сейчас есть примеры реализаций и исследований в данном направлении, которые показывают, что успешно решать подобные задачи вполне реально.

В ходе исследования, была рассмотрена система Sketch2Code, схожие системы HTML Code Generation и pix2code, а также подсистема DeepCodeer, которая идеи которой лежат в основе рассмотренных систем.

В результате проведенного исследования были выявлены следующие особенности:

- все рассмотренные системы в своей основе содержат предметно-ориентированный или DSL язык. Благодаря наличию такого языка, задача кодогенерации становится реальной, так как сводится к формированию правильной последовательности слов из словаря;

- основная проблема работы с DSL языком – поиск правильного символа из словаря;

- ни в одной из этих работ не решается проблема выделения текста из макета и его вставка в исходный код;

- полученный в результате работы код должен быть понятный и читаемый, так дальше с ним предстоит работать человеку;

Благодаря описанным выше особенностям, возможно построить систему, способную генерировать исходный код на основе изображения макета веб-страницы.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Портал государственных услуг Российской Федерации // [Электронный ресурс] – Режим доступа - URL: https://www.gosuslugi.ru/ (Дата обращения: 24.10.2024);
2. Avito // [Электронный ресурс] – Режим доступа - URL: https://www.avito.ru/ (Дата обращения: 22.10.2024);
3. D. Zou, G. Wu. Automatic Code Generation for Android Applications Based on Improved pix2code / D. Zou, G. Wu // Journal of Artificial Intelligence and Technology – 2024 - № 4 – pp. 325-331
4. HTML Code Generation from Images With Deep Neural Networks // Github – [Электронный ресурс] – Режим доступа - URL: https://github.com/taneemishere/html-code-generation-from-images-with-deep-neural-networks (Дата обращения: 24.10.2024);

5. K. Simonyan, A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition / K. Simonyan, A. Zisserman // arXiv preprint – 2014 - [Электронный ресурс] – Режим доступа - URL: https://arxiv.org/abs/1409.1556 (Дата обращения: 22.10.2024)
6. Keras // [Электронный ресурс] – Режим доступа - URL: https://keras.io/ (Дата обращения: 22.10.2024);
7. M. Balog, A. L Gaunt, M. Brockschmidt, S. Nowozin. DeepCoder: Learning to Write Programs / M. Balog, A. L Gaunt, M. Brockschmidt, S. Nowozin // arXiv preprint – 2016 - [Электронный ресурс] – Режим доступа - URL: https://arxiv.org/abs/1611.01989 (Дата обращения: 24.10.2024);
8. Ozon // [Электронный ресурс] – Режим доступа - URL: https://www.ozon.ru/ (Дата обращения: 24.10.2024);
9. T. A. Nguyen, C. Csallner. Reverse Engineering Mobile Application User Interfaces with REMAUI / T. A. Nguyen, C. Csallner // IEEE/ACM International Conference on Automated Software Engeneering (ASE) – 2015 – pp. 248-259
10. T. Beltramelli. pix2code: Generating Code from a Graphical User Interface Screenshot / T. Beltramelli // Proceeding of ACM SIGCHI Symposium on Engeneering Interactive Computing Systems (ACM) – 2018 – p. 3
11. Tensorflow // [Электронный ресурс] – Режим доступа - URL: https://www.tensorflow.org/?hl=ru (Дата обращения: 22.10.2024);
12. Tilda // [Электронный ресурс] – Режим доступа - URL: https://tilda.cc/ru/ (Дата обращения: 24.10.2024);
13. Tsung-Yi Lin, P. Goyal, R. Girshick, K. He, P. Dollár. Focal Loss for Dense Object Detection / Tsung-Yi Lin, P. Goyal, R. Girshick, K. He, P. Dollár // IEEE Transactions on Pattern Analysis and Machine Intelligence PP (99): 1-1 - 2018;
14. V. Jain, P. Agrawal, S. Bange, R. Kapoor. Sketch2Code: Transformation of Sketches to UI in Real-time Using Deep Neural Network / V. Jain, P. Agrawal, S. Bange, R. Kapoor // arXiv preprint – 2019 - [Электронный ресурс] – Режим доступа - URL: https://arxiv.org/abs/1910.08930 (Дата обращения: 24.10.2024)

15. X. Yao, M. H. Yap, Y. Zhang. Towards a Deep Learning Approach for Automatic GUI Layout Generation / X. Yao, M. H. Yap, Y. Zhang // Proceeding of International Conference on Computing and Communications Networks – 2022 – pp. 19-27

Оригинальность 81%