Code from Captures: Building HTML Markup from Screenshots

Michael Zhang
University of Waterloo
m468zhan@uwaterloo.ca

Jonathan Polina
University of Waterloo
jonathan.polina@uwaterloo.ca

Richard Sha
University of Waterloo
r4sha@uwaterloo.ca

Reeto Ghosh
University of Waterloo
reeto.ghosh@uwaterloo.ca

***Abstract*—This paper investigates the efficacy of Transformer models in the realm of web development, focusing on the generation of HTML and CSS code from website screenshots. Leveraging the groundwork laid by the Pix2Code model, which employs LSTM networks for a similar task, we conduct a comparative analysis between Transformers and LSTMs. By delving into their performances, we aim to provide insights into their respective strengths and weaknesses, thereby advancing automated web development tools. Our findings not only contribute to the ongoing discourse in machine learning for web development but also pave the way for future research directions and practical applications.**

I. Introduction

A. Motivation

In this paper, we aim to explore the effectiveness of transformer models in the domain of web development, specifically in generating HTML and CSS code from website screenshots. Building upon the foundation laid by the Pix2Code model, which utilizes Long short-term memory (LSTM) networks for the same task, we seek to compare the performance of transformers against LSTMs [Beltramelli, 2017]. By conducting this comparative analysis, we hope to gain insights into the strengths and weaknesses of each approach, thereby contributing to the advancement of automated web development tools. Our findings have the potential to inform future research directions and practical applications in the field of machine learning for web development.

B. Problem Definition

We aim to compare the performances of the Transformer model to the Long Short-Term Memory model for generating HTML code from website screenshots.

II. Related Work

Our approach mirrors the objective outlined in the Pix2Code paper authored by Tony Beltramelli, where the aim is to derive code representations from screenshots of websites. Leveraging a Convolutional Neural Network (CNN) for image encoding alongside Long Short-Term Memory (LSTM) models for token processing and decoding, the Pix2Code architecture has demonstrated an impressive 77% accuracy rate. Notably, the Pix2Code model adopts a streamlined domain-specific language (DSL) for delineating GUI layouts, which contributes to a reduction in overall vocabulary complexity. This DSL methodology simplifies the representation by disregarding textual values of labels, eliminating the need for computationally intensive word embeddings.

Research into the comparative effectiveness of LSTM networks and Transformer networks has been a subject of considerable investigation, particularly within the realm of image captioning tasks. Notably, a study conducted by Zouitni and colleagues demonstrated that Transformer networks outperform LSTMs in image captioning, showcasing advantages in both accuracy and computational efficiency [Zouitni et al., 2023]. This holds significance for our current project due to the analogous nature of image captioning and code generation tasks. In both scenarios, the objective revolves around generating a coherent sequence of words or tokens based on input data - in one case, visual information, and in the other, code structures and syntax. Thus, insights seen in advancements in image captioning techniques are highly pertinent and potentially applicable to our project.

III. Methodology

A. Model Architecture

The architectural framework, illustrated in Fig. 1, is elaborated upon below.

To encode the website screenshot images, we opted for the convolutional layers of the ResNet-50 neural network, diverging from the conventional encoder architecture of a typical transformer. This selection was based on the effectiveness of convolutional neural networks (CNNs) like ResNet-50 in extracting intricate features from images [He et al., 2015].

For decoding the image features, we employed the transformer decoder architecture, comprising an embedding layer, a positional encoding layer, a multi-head self-attention layer (comprising 16 heads), and a fully connected layer [Vaswani et al., 2023].

The vector obtained from the decoder undergoes an argmax operation, where the index with the highest score in the vector is selected. This index corresponds to a token within the vocabulary. The chosen token is appended to the output and cycled back into the decoder until the End Of Sequence token is chosen by the decoder.

B. Dataset

The LLM was trained on the Pix2Code dataset which had .gui files and their corresponding images. This was beneficial

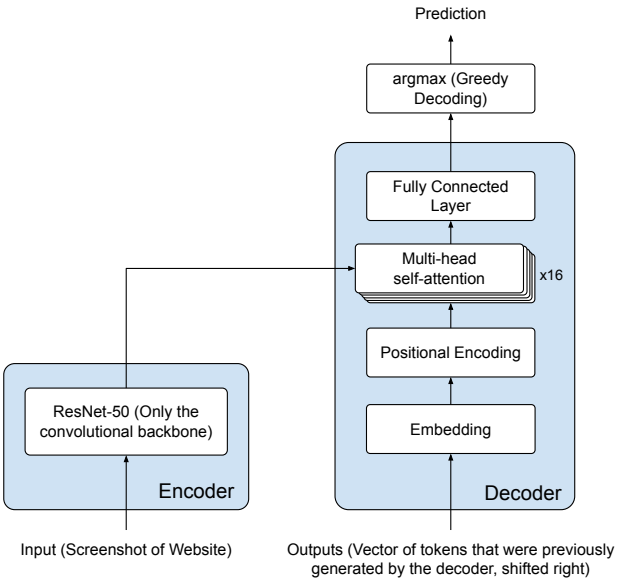

Fig. 1. The transformer model architecture.

for several reasons. Namely, the dataset is abundant with instances of GUI designs paired with their corresponding images, offering a wide variety of data for our model to learn from. The .gui files were also very simple to tokenize, which led to a smaller vocabulary. This simplified the learning process for our model, leading to more accurate and efficient predictions. Additionally, it reduced the dimensionality of the problem, making the model less prone to overfitting and more adaptable to unseen data. Using an established dataset like Pix2Code ensures that the data we're training on is of high quality, which is crucial for the performance of our model. Thus, this approach enhances the model's ability to accurately and efficiently convert screenshots to functional websites.

C. Experiment Setup

In terms of training hyperparameters, it was decided to train with 30 epochs, 2811 warm-up steps, an l2 penalty of 0.5, a learning rate of 0.0001, a gradient clipping of 2, and an evaluation period of 3.

The decoder has 6 layers, 16 attention heads, and a dropout of 0.5.

The batch size for training is 128 and the batch size for evaluation is 64. A learning rate scheduler is used to decrease the learning rate when a plateau is reached during training. Furthermore, the learning rate scheduler factor is 0.2, the patience is 1, and the threshold is 0.01.

D. Evaluation Methods

We used the BLEU (Bilingual Evaluation Understudy) score as the evaluation metric for our model. Commonly used in text generation and machine-translating tasks, the BLEU score is useful for us because it gives us a quantitative measure (between 0 and 1) of the similarity between the generated code from the model and the ground truth from the dataset.

IV. Results and Discussion

The changes in the loss function values of the two models over each epoch are shown in the Figure 2.

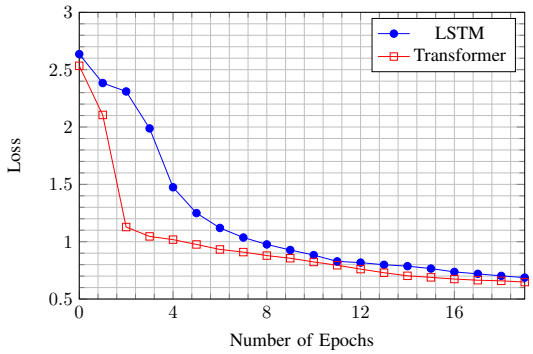

Fig. 2. Training Loss of LSTM vs. Transformer

The loss function of the Transformer model converges faster than the loss function of LSTM. This may be due to several factors. Transformers utilize self-attention mechanisms that enable each token in the input sequence to attend to all other tokens simultaneously, facilitating more effective capturing of long-range dependencies and contextual information. This parallel processing ability can allow the Transformer to update its parameters more efficiently during training compared to the LSTM model, which process sequences sequentially. Moreover, Transformers employ techniques like layer normalization and residual connections, which contribute to smoother gradient flow and faster convergence during optimization.

Figure 3. shows the BLEU scores of the transformer model during training and testing respectively. The score increases over the first 10 epochs and then oscillates at around 60-70. The oscillation could be due to the capacity of the Transformer model being too complex relative to the size of the dataset, making it struggle to generalize effectively, leading to oscillations in performance as it tries to fit the training data too closely.

Figure 4. shows the changes of the BLEU scores for LSTM on the training and testing dataset respectively. Both show a faster increase than the scores of the Transformer model, and converges to around 80 - 90 after 5 epochs. The better performance of the LSTM model compared to the Transformer model can be caused by the limitation of the dataset size.

The LSTM model has fewer parameters compared to the Transformer model. With limited data, a smaller model like the LSTM model might generalize better due to its lower risk of overfitting. The Transformer, with a larger number of parameters, might struggle to effectively learn from the limited data, leading to poorer performance. This may also be the reason why the BLEU score fluctuates more for the Transformer despite its loss function is consistently decreasing.

Furthermore, the length of each file within the dataset is also smaller than usual web pages. A reason for Transformers

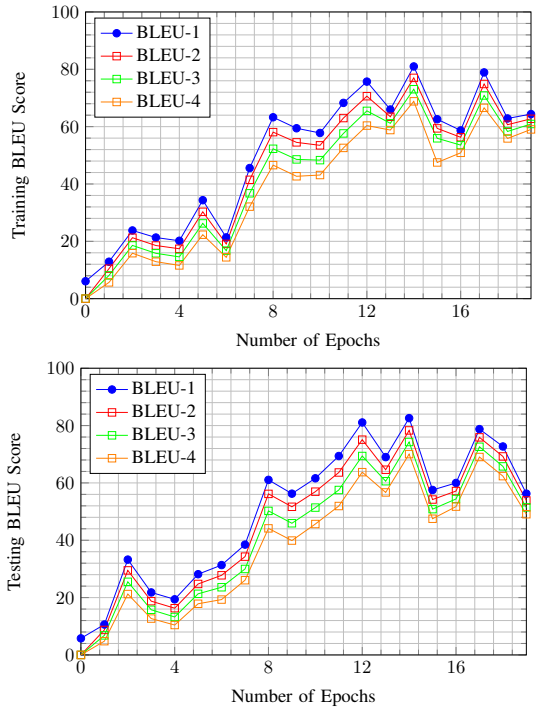

Fig. 3. BLEU Scores of Transformer in Training and Testing

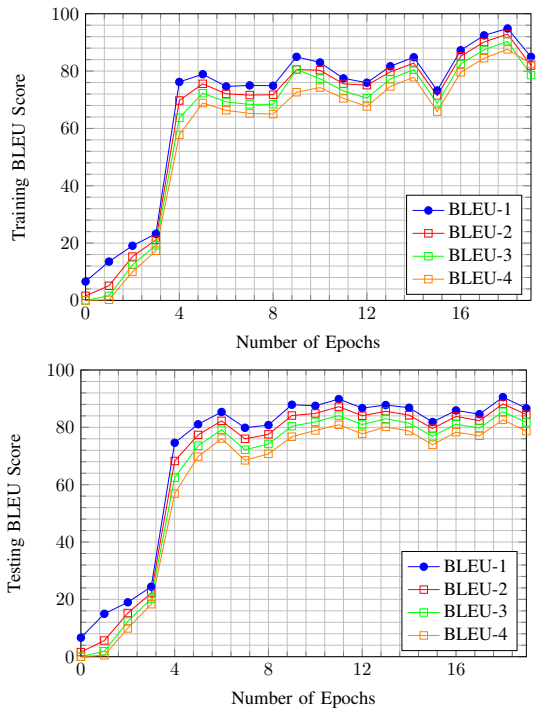

Fig. 4. BLEU Scores of LSTM in Training and Testing

outperforming LSTMs is that they are not as affected by the vanishing gradient or exploding gradient problem as LSTMS. With shorter sequences, this issue may not be an apparent factor in the performance of either models.

The limitation in dataset size can also be shown in both models having similar scores on their training and testing sets, despite the testing sets containing unseen data. The data provided by the pix2code dataset may be too simple in its structure and both models can generalize well enough from their training set to produce similar results on the testing set.

A. Ethical Considerations

We strive to make our research transparent and accessible to both technical and non-technical audiences. This includes clearly documenting our methodology, disclosing any limitations or potential biases in our approach, and providing explanations for model predictions to ensure accountability and trustworthiness.

We also have a responsibility to consider the potential societal impacts of our work and to use technology in ways that promote positive outcomes and minimize harm. This includes considering the potential implications of automated web development tools on job displacement, user experience, and accessibility.

Although the data used to train the model in this project was created for training purposes and was not scraped from the internet, we must ensure that any data used in similar studies or extensions, including website screenshots and associated code, is obtained and used in accordance with relevant privacy regulations. This includes obtaining consent from website owners and users, anonymizing sensitive information, and securely storing data to prevent unauthorized access.

V. Conclusion

To examine the more suitable architecture for website generation, we experimented with models using LSTM and Transformer as decoders and used the BLEU score for measuring and comparing their performances. The results of the experiments were different from the initial expectations of Transformers outperforming LSTMs in such tasks. In contrast, the model using LSTM as the decoder reached higher BLEU scores within fewer epochs compared to the model using Transformer as the decoder. We suggest that the reason for the result may be due to the limitation in the size and variability of the dataset. The small data volume makes LSTMs less likely to overfit the data compared to Transformers, while also making the advantages of Transformers over LSTMs in long-range dependencies obsolete.

VI. Future Work

While the team's model can generate basic websites, more work could be done to expand the capabilities of website generation. For instance, investigations into generating actual HTML and CSS code or frontend code utilizing popular frontend frameworks such as React or Angular will allow users to create more complicated and sophisticated websites to be generated. Additionally, the use of static images as the input to the model limits the use of animations or videos on the generated websites. Investigations into using video as input will allow developers to integrate stunning transitions and animations into the generated website. Another avenue of

investigation could be the support of multiple images in the model to allow the generation of multiple pages and support the navigation routing of websites. As was with the original website generation model, all these future endeavors will need in-depth research and careful execution.

VII. Limitations

The team faced many limitations throughout the project. One of the biggest limitations was the lack of clean training data. Initially, the team sought to scrape HTML and CSS documents online using automation tools such as Selenium but found that the data was too messy. This limitation affected the tokenization process, as websites used different HTML tags or had different frontend frameworks present. As a result, it's difficult for the model to make consistent outputs with the existing data that the team scraped from online. Realizing this, the team pivoted to existing data provided by the Pix2Code GUI format for training.

VIII. Acknowledgements

We acknowledge that our dataset is from the pix2code paper by Tony Beltramelli.

IX. Appendix

Use this section to add additional figures and tables.

References

[Beltramelli, 2017] Beltramelli, T. (2017). pix2code: Generating code from a graphical user interface screenshot.

[He et al., 2015] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition.

[Vaswani et al., 2023] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2023). Attention is all you need.

[Zouitni et al., 2023] Zouitni, C., Sabri, M. A., and Aarab, A. (2023). A comparison between lstm and transformers for image captioning. In Motahhir, S. and Bossoufi, B., editors, *Digital Technologies and Applications*, pages 492–500, Cham. Springer Nature Switzerland.