REVOLUTIONIZING PERSONALIZED WEB PRESENCE: AI-POWERED AUTOMATED WEBSITE GENERATION FOR STREAMLINED INDIVIDUAL EXPRESSION AND ACCESSIBILITY

Zhen Yang[1] and Tiancheng Xu[2]

[1]Shanghai United International School Qingpu Campus, No. 32, Yejin Road, Qingpu District, Shanghai
[2]Computer Science Department, California State Polytechnic University, Pomona, CA 91768

ABSTRACT

In the present information technology era, the desire for personalized individual websites is notably significant [1]. The creation of such personal websites necessitates proficient knowledge of HTML-based programming, a skill predominantly possessed by professional programmers [2]. As an alternative to this limited option, individuals often resort to purchasing websites from service providers, incurring substantial costs and time investments. In order to address this widespread demand for personalized websites, we propose the implementation of an AI program capable of automatically generating websites based on user instructions [3]. Such a program would considerably reduce the financial and temporal expenditures associated with purchasing pre-made websites.

KEYWORDS

Chat GPT, Large Language Model, Prompt Engineering

1. INTRODUCTION

In the present information technology era, the desire for personalized individual websites is notably significant. The creation of such personal websites necessitates proficient knowledge of HTML-based programming, a skill predominantly possessed by professional programmers [4]. As an alternative to this limited option, individuals often resort to purchasing websites from service providers, incurring substantial costs and time investments.

We are the first batch to generate HTML using trained large language models [5]. Compared to finding someone to customize HTML, telling them our requirements and understanding them. Or go to the official learning document and create the webpage you want yourself. The time we save and the accuracy of the generated graphics and text are very high.

Nowadays, everyone wants personalized websites and a lot of money. If someone is hired to create HTML, this price is incomparable to ours. They usually need $500 to $600 in expenses. By comparison, our price is as light as a feather in a feather.

David C. Wyld et al. (Eds): SEAS, NLPI, SP, BDBS, CMCA, DMML, CSITEC -2023
pp. 11-19, 2023. CS & IT - CSCP 2023 DOI: 10.5121/csit.2023.131702

In order to address this widespread demand for personalized websites, we propose the implementation of an AI program capable of automatically generating websites based on user instructions. Such a program would considerably reduce the financial and temporal expenditures associated with purchasing pre-made websites.

Every time I test, I test the accuracy. Of course, the accuracy of the generated HTML is quite high. I will set the topic for each experiment, topic_ Demand, user demand denmark, number, quality, image_ Style and website_ Demand. These are the basic requirements for running code. I am very satisfied with the content generated every time, including images, text, and layout. I requested the color of the background at that time, and he also followed my request. I believe that being precise and filling out my requirements fully is the reason for generating these perfect web pages.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. Images or Videos

Chat GPT is famous for generating text-based context [6]. However, modern websites usually include images or videos, which cannot be generated by GPT directly. The challenge is how to generate high-quality images and plug them into the GPT generated html file [7]. One way to solve it is to use a different AI image generator, and tell GPT the path to the images when giving prompts to GPT.

2.2. The Output

Although GPT is a powerful AI tool, it is essentially a black box process, meaning that the output doesn't have to be following the user's instructions [8]. This becomes a problem when we want it to automatically generate a html file, since html files have strict syntax requirements. One way to address this issue is to clearly state that we want a formatted html file, and it should follow all html rules.

2.3. 3D Effects

When I was writing code for a web-page, the original effect was 3D effects [9]. But I am not proficient in art and learning the front-end design of 3D interaction. It requires lots of professional technology such as 3DS Max, Cinema 4D [10]. They are not friendly for a novice who has no artistic skills and has never been exposed before.

3. SOLUTION

The program starts with asking the users for information of what they want to generate. To be more specific, we ask the user for a topic and a user demand, which describe what the website should be in detail. In addition, we also have an optional text field needing the user to fill. That is a user demand remark entry, where the user could give explanations or examples of their demands. Then we ask for a number which means how many blocks they want on the website. Along with these, we also ask for generating quality and image style, so that the user has more freedom on choosing. At the end, we ask for a website demand, where the user can specify what kind of website they want.

When we collect the topic and user demand, we make a prompt locally and send it to GPT. We ask the GPT to generate a number of sentences and corresponding prompts for generating images. We also generate a slogan. Then we use the prompts to generate images. In addition, we use the slogan to generate a background image. Once we have the text and images ready, we send everything to GPT, asking it to generate a html file, representing the website.

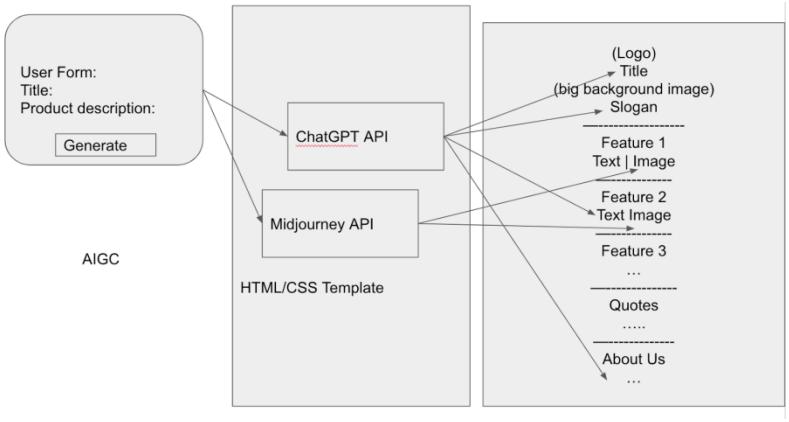

Figure 1. Overview of the solution

The first component is a function that asks for topic and user demand and returns a GPT response. It will take all the information needed for generating sentences, and merge them into a well-designed local prompt. Then it will send this local prompt via OpenAI API to the right GPT model, and receive the response at the end [12].

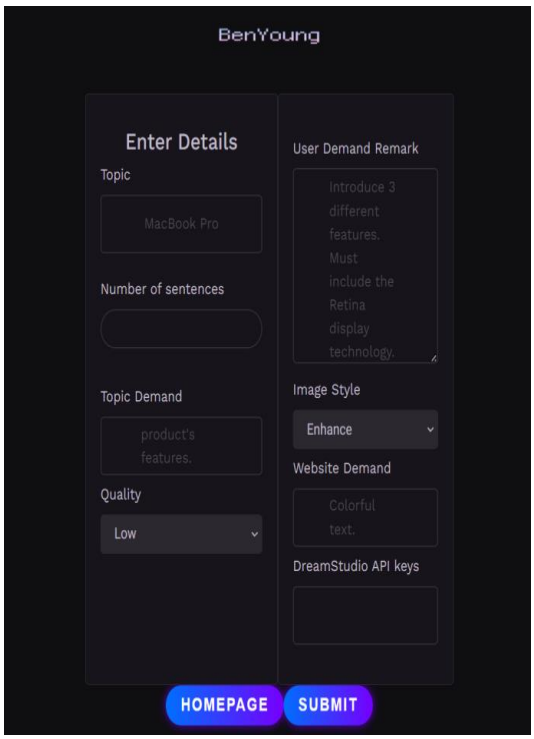

Figure 2. Screenshot of the AI model 1

```
def gen_sentences_and_prompts(gpt_model, topic, tasks, tasks_remark, number):
  image_guideline = """Here are some guidelines for generating high-quality images. You
should write the prompts following these guidelines. \
1 - Be specific and detailed about the image. \
2 - Use sensory language to describe textures, sounds, smells, and other sensory details. \
3 - Consider composition and perspective by providing guidelines on camera angles, framing,
and object arrangement. \
4 - Set the mood and atmosphere by describing the desired emotional context of the image. \
5 - Incorporate storytelling elements to give the AI image a sense of narrative or drama. \
6 - Reference existing art or images as inspiration or guidance for the desired aesthetic."""
  system_message = """You are a knowledgable assistant. You need to write sentences using
the best of your knowledge."""
  user_message = f"""Your job is to generate {number} sentences in the topic and tasks
delimited by triple backticks. \
```Topic: {topic}. Tasks: {tasks}``` \
Make sure that you must satisfy the requirements delimited by <>. \
<Requirements: {tasks_remark}> \
For each sentence, write a prompt for generating its corresponding images. \
{image_guideline} \
The prompts must include information about the topic. \
The Output must be {number} sentences and {number} prompts. \
Output it as a Json object, where the key is the sentence and the value is the prompt."""

  contents_response = openai.ChatCompletion.create(
    model=gpt_model,
    messages=[
      {"role": "system", "content": system_message},
      {"role": "user", "content": user_message},
    ],
    temperature=0,
  )
contents_response_text = contents_response["choices"][0]["message"]["content"]
contents_response_text = eval(contents_response_text)

sentences = []
prompts = []
for k,v in contents_response_text.items():
  sentences.append(k)
  prompts.append(v)

return sentences, prompts
```

Figure 3. Screenshot of code 1

In the code, we make our messages clear by prompt engineering. First we set up the role of GPT agent via system messages, clearly claiming that it should use its best knowledge to help create content. In the user messages, we first state that we are requesting for a certain number of sentences that are related to the topic and user demand. For each sentence, we need a high-quality prompt as well. To better help GPT generate a good image generation prompt, we write a detailed AI image generation guideline. This guideline includes some general skills on generating high-quality images. Once we have all the messages needed, we form them as a complete message and send it to GPT via the Open AI API [11]. We extract the content from the GPT response at the end.

The second component of this program is to randomly generate images given a specific image generating prompt. It will take the image generating prompts that generated from the previous component and send it to stability AI API. Stability AI is an image generating AI model that takes prompts and generates the corresponding images.
```

```
def gen_images(prompts, quality, image_style):
  if quality == "low":
    engine_type = "stable-diffusion-v1"
    n_steps = 15
  elif quality == "medium":
    engine_type = "stable-diffusion-v1"
    n_steps = 30
  elif quality == "high":
    engine_type = "stable-diffusion-512-v2-0"
    n_steps = 20

  stability_api = client.StabilityInference(
      key=os.environ['STABILITY_KEY'],  # API Key reference.
      verbose=True,  # Print debug messages.
      engine=engine_type #"stable-diffusion-v1",  # Set the engine to use for generation.
      # Available engines: stable-diffusion-v1 stable-diffusion-v1-5 stable-diffusion-512-v2-0
  stable-diffusion-768-v2-0
      # stable-diffusion-512-v2-1 stable-diffusion-768-v2-1 stable-diffusion-xl-beta-v2-2-2
  stable-inpainting-v1-0 stable-inpainting-512-v2-0
  )

  images = []
 for i, prompt in enumerate(prompts):
  answers = stability_api.generate(
    prompt=prompt,
    style_preset = image_style,
    seed=992446758,  # If a seed is provided, the resulting generated image will be
deterministic.
    # What this means is that as long as all generation parameters remain the same, you can
always recall the same image simply by generating it again.
    # Note: This isn't quite the case for CLIP Guided generations, which we tackle in the CLIP
Guidance documentation.
    steps=n_steps,  # Amount of inference steps performed on image generation. Defaults to
30.
    cfg_scale=8.0,  # Influences how strongly your generation is guided to match your prompt.
    # Setting this value higher increases the strength in which it tries to match your prompt.
    # Defaults to 7.0 if not specified.
    width=512,  # Generation width, defaults to 512 if not included.
    height=512,  # Generation height, defaults to 512 if not included.
    samples=1,  # Number of images to generate, defaults to 1 if not included.
    sampler=generation.SAMPLER_K_DPMPP_2M  # Choose which sampler we want to
denoise our generation with.
    # Defaults to k_dpmpp_2m if not specified. Clip Guidance only supports ancestral
samplers.
    # (Available Samplers: ddim, plms, k_euler, k_euler_ancestral, k_heun, k_dpm_2,
k_dpm_2_ancestral, k_dpmpp_2s_ancestral, k_lms, k_dpmpp_2m, k_dpmpp_sde)
  )
 for resp in answers:
  for artifact in resp.artifacts:
   if artifact.finish_reason == generation.FILTER:
    print("Generation failed.")
   if artifact.type == generation.ARTIFACT_IMAGE:
    img = Image.open(io.BytesIO(artifact.binary))
    img.save(str(i)+".png")
    images.append(str(i)+".png")

return images
```

Figure 4. Screenshot of code 2

This function takes prompts, quality and image style. The prompts have a certain number of prompts, which describe the image requirements in great detail. The quality is either low, medium or high, indicating the generated images' quality. The low quality uses a v1 engine and a step size of 15; the medium quality uses a v1 engine and a step size of 30; the high quality uses a v2 engine and a step size of 20. The newer the engine, the stronger the performance. Same rule applies to step size. Furthermore, we allow the user to choose an image style. The image styles are categorical, meaning that the user has to choose one from a large range of image style options. Some examples are "enhance", "anime", "photographic" and so on. Once we collected all that

information, we combined all that information into a huge chunk of prompts. At the end, we send the prompt to the server via the stability AI API [15]. The API returns a response which contains the images generated. We save the images to the local machine.

The last component is a function generating html file. It takes a topic, a list of sentences and their corresponding images, a slogan, a background image and a website demand. It will reorganize all the collected information, and send it to the GPT model, asking it to write an html file. At the end, the GPT returns a html as the output. We save the outputting html to the local machine.

```
def gen_html(gpt_model, topic, sentences, slogan, images, background, website_demand):
  system_message = "I will give you a topic, some related sentences, the corresponding
images and a slogan. Your job is to generate a html file. "
  system_message += "The header contains the topic only. "
  system_message += "A container contains the slogan and a background image. The source
of the background image is" + str(background) + ". "
  system_message += "The background image is not the background for the whole html page.
It is only local for the slogan container. It should somehow be transparent, so that the user
can read the slogan. "
  system_message += "The text color of the slogan should be the contrast color of the
background image. "
  system_message += "In addition, another container shows a list of images and their
corresponding sentences. The source of the images are " + str(images)
  system_message += website_demand
  user_message = "The topic is: " + topic + ". The sentences are " + " ".join(sentences) + " The
slogan is " + slogan


  html_response = openai.ChatCompletion.create(
    model=gpt_model,
    messages=[
      {"role": "system", "content": system_message},
      {"role": "user", "content": user_message},
    ],
    temperature=0,
  )

  html_response_text = html_response["choices"][0]["message"]["content"]

  return html_response_text
```

Figure 5. Screenshot of code 3

The function takes a topic, a certain number of sentences, a certain number of images, a slogan, a background image, a website demand and a specific GPT model [14]. It does some prompt engineering, combining all the information into a prompt, sending it to the GPT model. The GPT model which receives the message, will return a html file as its response. We write the prompt carefully so that it only returns the html raw code if everything goes well. At the end, we save the returning html file to the local machine.

4. EXPERIMENT

4.1. Experiment 1

We plan to introduce the Mac Book Pro product. Introduce 3 different features, must include the Retina display. I set up some variables called topic, tasks, tasks_remark, number, quality, image_style. I input the topic “Macbook Pro”, task is “Introduce the Macbook Pro product’s features”. Tasks_remark is Introducing 3 different features, must include the Retina display. Number is 3, which means that AI will give me 3 different pictures. Quality is high, I will get 3 different high quality images. Also, the image_style is “3d-model”.

Figure 6. Figure of experiment 1

I think the quality of the pictures is very shocking to me, as if it were real and also including an introduction to the display - Retina display. The only regret is that typesetting and font art are too common and lack characteristics. Nowadays, a webpage with such a simple layout is not attractive, but I already think it's great to be able to generate such a website in just a few minutes.

4.2. Experiment 2

We plan to introduce motorcycles.
I set up some variables called topic, topic_demand, user demand remark, number, quality, image_style and website_demand. I input the topic “Motorcycle”, the task is “Introduce motorcycle”. Tasks_remark is to “introduce 3 motorcycles, the motorcycle brands should compare performance, price, and usage. Examples of uses include whether it is for running a race track or riding a motorcycle.” Number is 3, which means that AI will give me 3 different pictures. Quality is high, I will get 3 different high quality images. Also, the image_style is “3d-model”.

Figure 7. Figure of experiment 2

I think the quality of the pictures is very shocking to me. The picture is very realistic and perfectly displays the effect I want. And it includes different types of motorcycles. The only regret is that typesetting and font art are too common and lack characteristics. Nowadays, a webpage with such a simple layout is not attractive, but I already think it's great to be able to generate such a website in just a few minutes.

5. RELATED WORK

We are the first batch to generate HTML using trained large language models. Compared to finding someone to customize HTML, telling them our requirements and understanding them. Or go to the official learning document and create the webpage you want yourself. The time we save and the accuracy of the generated graphics and text are very high.

Nowadays, everyone wants personalized websites and a lot of money. If someone is hired to create HTML, this price is incomparable to ours. They usually need $500 to $600 in expenses. By comparison, our price is as light as a feather in a feather.

6. CONCLUSIONS

The largest limitation of our program is that our website must be following a certain form of pattern. Our working flow is that we need to generate a couple of sentences following the topic at first. Then we generate an image for each sentence. It’s impossible for our program to generate 2 images in total, if there are 3 sentences. That limitation decreases the user’s freedom to create a website. To fix this issue, we need to develop a more general pattern, so that the user has all freedom to design their websites, and at the same time with the ability to generate the corresponding image along with it.

Overall, we developed an AI-powered HTML generator using the large language model technology [13]. We designed some experiments to test out its generating ability, and we were shocked at the quality of generating websites, in terms of both the text and the images quality. We think it could be a good alternative to purchasing expensive customized personal websites.

REFERENCES

[1] Thurman, Neil, and Steve Schifferes. "The future of personalization at news websites: Lessons from a longitudinal study." Journalism studies 13.5-6 (2012): 775-790.

[2] Megantoro, Prisma, et al. "Real-time monitoring system for weather and air pollutant measurement with HTML-based UI application." Bulletin of Electrical Engineering and Informatics 10.3 (2021): 1669-1677.

[3] Shortliffe, Edward H., et al. "An artificial intelligence program to advise physicians regarding antimicrobial therapy." Computers and Biomedical Research 6.6 (1973): 544-560.

[4] [Gonvalves, F. A. S., and Carlos A. Canesin. "Java applets for a WWW-HTML-based course in power electronics." 2001 IEEE 32nd Annual Power Electronics Specialists Conference (IEEE Cat. No. 01CH37230). Vol. 1. IEEE, 2001.

[5] Melis, Gábor, Chris Dyer, and Phil Blunsom. "On the state of the art of evaluation in neural language models." arXiv preprint arXiv:1707.05589 (2017).

[6] Hill-Yardin, Elisa L., et al. "A Chat (GPT) about the future of scientific publishing." Brain Behav Immun 110 (2023): 152-154.

[7] Chen, Zhutian, et al. "Beyond Generating Code: Evaluating GPT on a Data Visualization Course." arXiv preprint arXiv:2306.02914 (2023).

[8] Venkatesh, Viswanath. "Adoption and use of AI tools: a research agenda grounded in UTAUT." Annals of Operations Research (2022): 1-12.

[9] To, Akiko, et al. "3D effects of sharp boundaries at the borders of the African and Pacific Superplumes: Observation and modeling." Earth and Planetary Science Letters 233.1-2 (2005): 137-153.
[10] Kadirbergenovna, Bagbekova Laylo. "Create 3d graphics with the hand of 3d max software." Conference Zone. 2022.
[11] Gozalo-Brizuela, Roberto, and Eduardo C. Garrido-Merchan. "ChatGPT is not all you need. A State of the Art Review of large Generative AI models." arXiv preprint arXiv:2301.04655 (2023).
[12] Aydın, Ömer, and Enis Karaarslan. "OpenAI ChatGPT generated literature review: Digital twin in healthcare." Available at SSRN 4308687 (2022).
[13] Subramonyam, Hariharan, Colleen Seifert, and Eytan Adar. "Protoai: Model-informed prototyping for ai-powered interfaces." 26th International Conference on Intelligent User Interfaces. 2021.
[14] Floridi, Luciano, and Massimo Chiriatti. "GPT-3: Its nature, scope, limits, and consequences." Minds and Machines 30 (2020): 681-694.
[15] Mackenzie, Adrian. "From API to AI: platforms and their opacities." Information, Communication & Society 22.13 (2019): 1989-2006.