Templatized Email body Generation using Large Language Models

Akshathaholla · Follow
4 min read · Sep 27, 2023

Imagine having a conversation with a computer that not only understands your words but can generate entire essays, poetry, or code at your command. Welcome to the era of Large Language Models, or LLMs for short, where machines have evolved to understand and create text in a way that seems almost uncanny.

A large language model (LLM) is a type of artificial intelligence (AI) algorithm that uses deep learning techniques and massively large data sets to understand, summarize, generate and predict new content.

In this blog we will explore how we can leverage the power of LLMs to automate

list of their existing customers with appropriate information. Let us take a look at the steps involved.

Reading customers list and Product list

Here we will use a sample customer list and product list available in the links below for customizing the content of the email body. We first save the data from the links into local csv files.

```
import requests

customers_url = "https://media.githubusercontent.com/media/datablist/sample-csv
customers_file_name = "./customers.csv"

result=requests.get(customers_url)

with open(customers_file_name, "w") as file:

    file.write(result.text)

products_url = "https://diviengine.com/downloads/Divi-Engine-WooCommerce-Sample
products_file_name = "./products.csv"
result=requests.get(products_url)
with open(products_file_name, "w") as file:
    file.write(result.text)
```

Next we read the csv files and load it into memory

```
import csv

def read_csv(filename):
    with open(filename, 'r') as file:
        return list(csv.DictReader(file))

products = read_csv(products_file_name)
customers = read_csv(customers_file_name)
```

Generating prompt with required variables for customers

Here we create a sample prompt template which plays an important role in ensuring that the email generated has the right context and content. The customer and product information is utilized to customize the email body so that the content generated is highly personalized. The prompt also needs to be descriptive about what exactly is expected as part of the body.

The generated content is only as good as the prompt template that is designed.

```
def generate_email_prompt(info: dict):
    prompt = """ Your goal is to write a personalized outbound email from {sale
    A good email is personalized and informs the customer on how your product c
    Be sure to use value selling: A sales methodology that focuses on how your
    % INFORMATION ABOUT {company}:
    a fashion and apparel company
    products list:
    {products_info}

    % INFORMATION ABOUT {prospect}:
    {text}

    % INCLUDE THE FOLLOWING PIECES IN YOUR RESPONSE:
    - Start the email with the sentence: "We have amazing discounts on our prem
    - The sentence: "We observed that XYZ product suits your taste " Replace XY
    - A 1-2 sentence description about the products, be brief
    - End your email with a call-to-action such as asking them to set up time t

    % YOUR RESPONSE:"""
    for key in info.keys():
        prompt=prompt.replace("{"+key+"}",info[key])
    return prompt

customer_infos = []
for customer in customers[:5]:
    info = {}
    info['company'] = "XYZ"
    info['sales_rep'] = "ABC"
    info['prospect'] = customer["First Name"] + " " + customer["Last Name"]
    info['text'] = customer["Job Title"]
    products_info= ""
    i=1
    for product in products[:5]:
        products_info += f"\n {i}. Product Name: {product['Name']} \n  Descript
        i+=1
    info['products_info']=products_info
    customer_infos.append(info)
llm_chain=LLMChain(prompt=prompt_template,llm=llm, verbose = True)


prompts = [generate_email_prompt(info) for info in customer_infos]
```

Once the prompt is generated for various customers we use OpenAI Completion for generating the final email content from the prompt that is provided.

Interface with LLMs and generate Email content

The last step is to interface with the hosted LLMs of your choice using Hugging face or OpenAI or Cohere (Here we chose OpenAI)

```
import os
import openai

openai.api_key = os.getenv("OPENAI_API_KEY")

responses = [openai.Completion.create(
  model="text-davinci-003",
  prompt=prompt,
  temperature=0,
  max_tokens=3000,
  top_p=1.0,
  frequency_penalty=0.0,
  presence_penalty=0.0
) for prompt in prompts]

email_body = responses[0]["choices"][0].text
```

Here the temperature parameter is a hyperparameter that can be used to control the randomness and creativity of the generated text in a generative language model. When the temperature is set to a low value it results in more conservative and predictable text. The temperature value can be varied based on the creativity you wish to see in the generated text. We have chosen text-davinci-003 which is a text generation model available as part of OpenAI. We have set a max token limit of 3000. Depending on the model used, requests can use up to 3000 tokens shared between prompt and completion. If your prompt is 1000 tokens, your completion can be 2000 tokens at most. top_p limits the pool of available tokens that can be chosen from, for completion. The value 1 represents a big pool so lot of tokens are available for the completion. Frequency_penalty is used to discourage the model from repeating frequently used/present words while presence_penalty encourages the model to use a more diverse range of words in the generated text.

Sample Response

```
Dear Shelby Terrell,

We have amazing discounts on our premium products such as the Divi Engine Strir
We would love to discuss how our products can help you reach your goals. Please

Thank you for your time and consideration.

Sincerely,
ABC
XYZ
```

In conclusion, the capability of LLMs as an automated email/content generation tool is undeniable. From crafting persuasive marketing emails to streamlining customer support responses, these AI-driven solutions have proven their worth in today's fast-paced digital landscape.

However, as we embrace this technology, it's crucial to remember that while AI can automate the process of generating emails, it cannot replace the human touch entirely. Personalization, empathy, and context-awareness are qualities that only human writers can truly provide. Thus, the synergy between AI-generated content and human creativity is the key to unlocking the full potential of email communication.

Llm Ai Email Generator OpenAI

Follow

Written by Akshathaholla

2 Followers

More from Akshathaholla

Akshathaholla

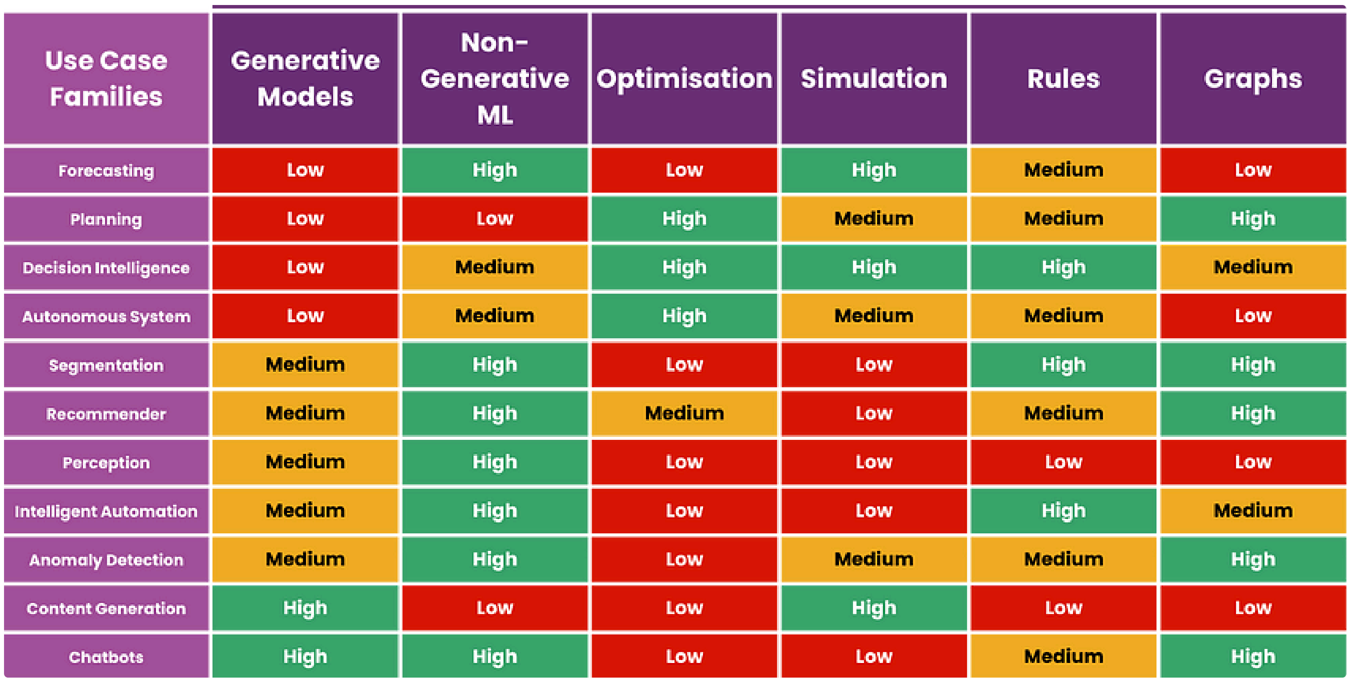
Scheduling Poll based Tasks using Timer in Java

Many projects require automation in accepting/reading data from an external source on a timely basis and require certain tasks to be…

Sep 29, 2022

See all from Akshathaholla

Recommended from Medium



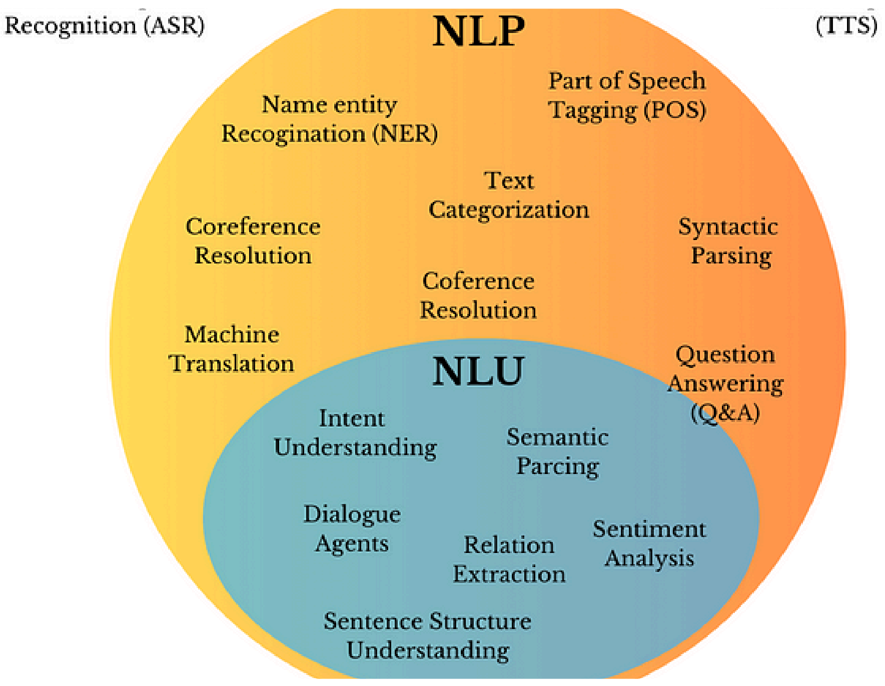
Use Case Families	Generative Models	Non-Generative ML	Optimisation	Simulation	Rules	Graphs
Forecasting	Low	High	Low	High	Medium	Low
Planning	Low	Low	High	Medium	Medium	High
Decision Intelligence	Low	Medium	High	High	High	Medium
Autonomous System	Low	Medium	High	Medium	Medium	Low
Segmentation	Medium	High	Low	Low	High	High
Recommender	Medium	High	Medium	Low	Medium	High
Perception	Medium	High	Low	Low	Low	Low
Intelligent Automation	Medium	High	Low	Low	High	Medium
Anomaly Detection	Medium	High	Low	Medium	Medium	High
Content Generation	High	Low	Low	High	Low	Low
Chatbots	High	High	Low	Low	Medium	High

Christopher Tao in Towards AI

Do Not Use LLM or Generative AI For These Use Cases

Choose correct AI techniques for the right use case families

Aug 11 1.8K 22



Vipra Singh

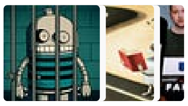
LLM Architectures Explained: NLP Fundamentals (Part 1)

Deep Dive into the architecture & building of real-world applications leveraging NLP Models starting from RNN to the Transformers.

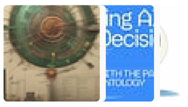
Aug 15 1K 9

Lists

Natural Language Processing
1656 stories · 1233 saves

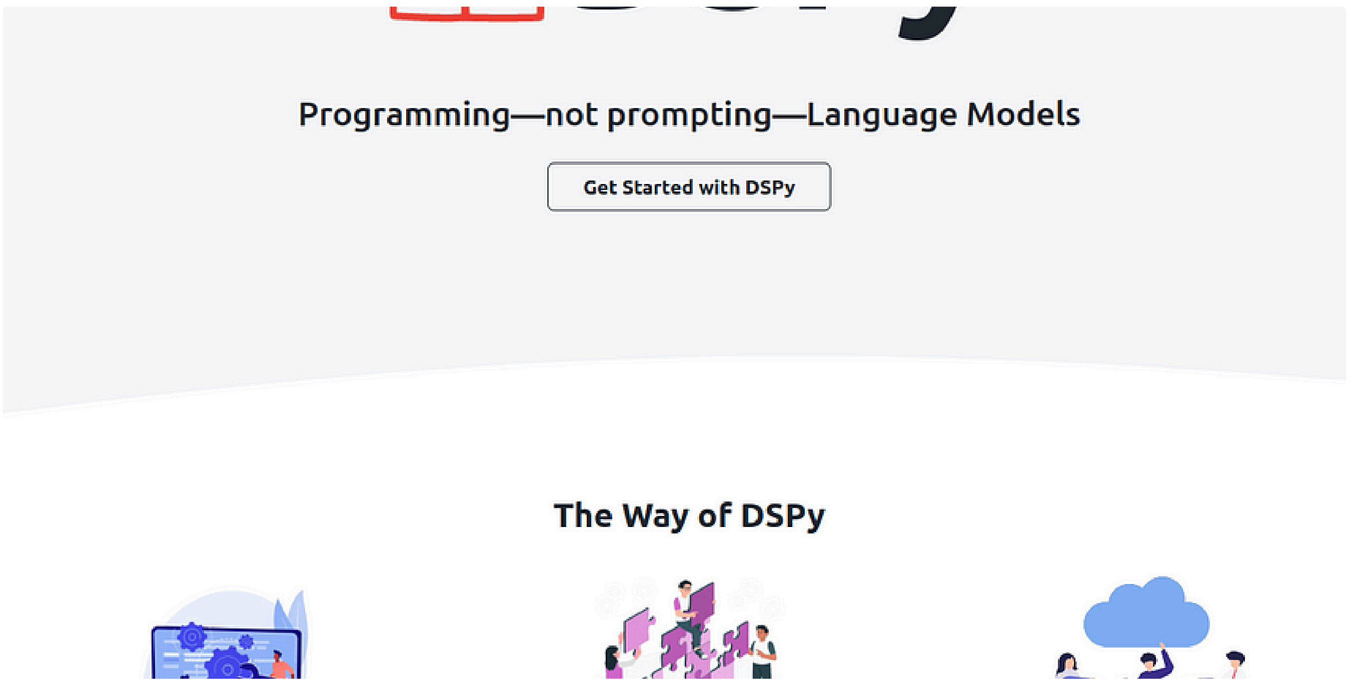


AI Regulation
6 stories · 544 saves



data science and AI
40 stories · 223 saves

Coding & Development
11 stories · 754 saves



Vishal Rajput in AIGuys

Prompt Engineering Is Dead: DSPy Is New Paradigm For Prompting

DSPy Paradigm: Let's program—not prompt—LLMs

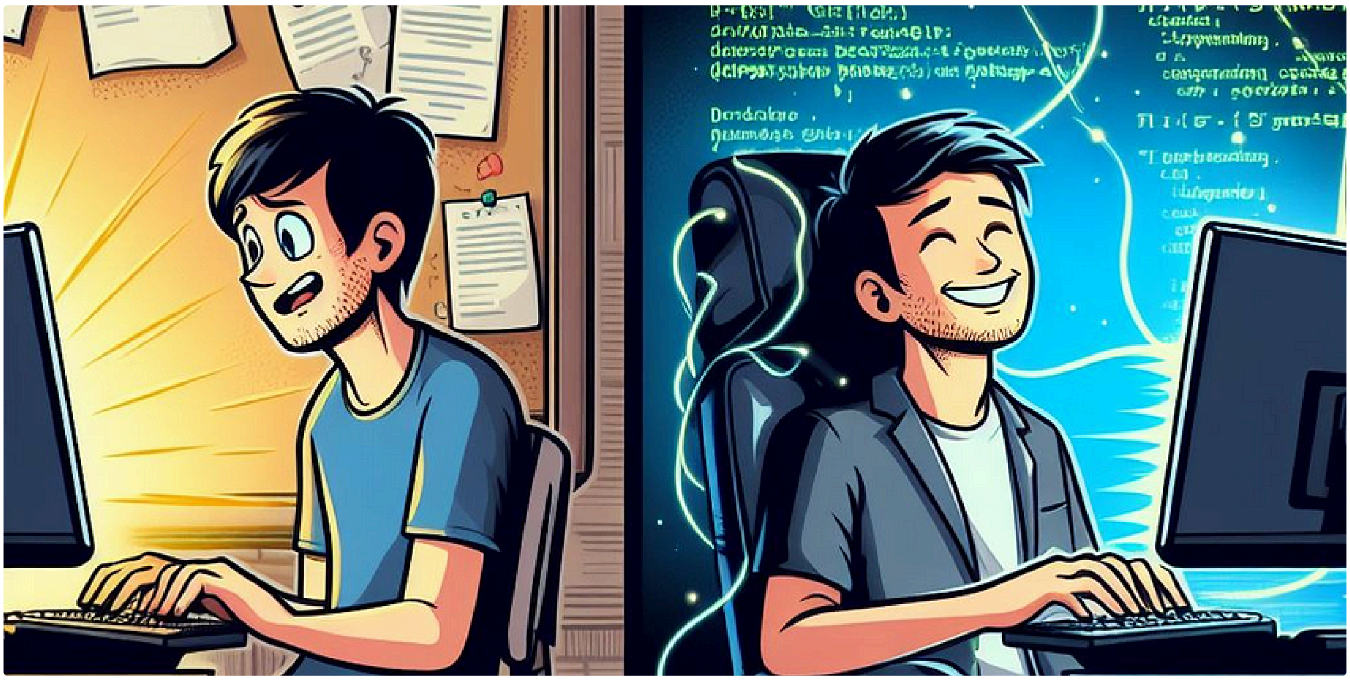
May 29 4.5K 46

Thuwarakesh Murallie in Towards Data Science

How to Build Helpful RAGs with Query Routing.

An LLM can handle general routing. Semantic search can handle private data better. Which one should you pick?

Aug 16 272 1



Abhay Parashar in The Pythoneers

17 Mindblowing Python Automation Scripts I Use Everyday

Scripts That Increased My Productivity and Performance

1d ago 6.8K 56

Datadrifters

Say Hello to 'Her': Real-Time AI Voice Agents with 500ms Latency, Now Open Source

Voice Mode is hands down one of the coolest features in ChatGPT, right?

6d ago 527 7

See more recommendations