Co-writing with AI language models

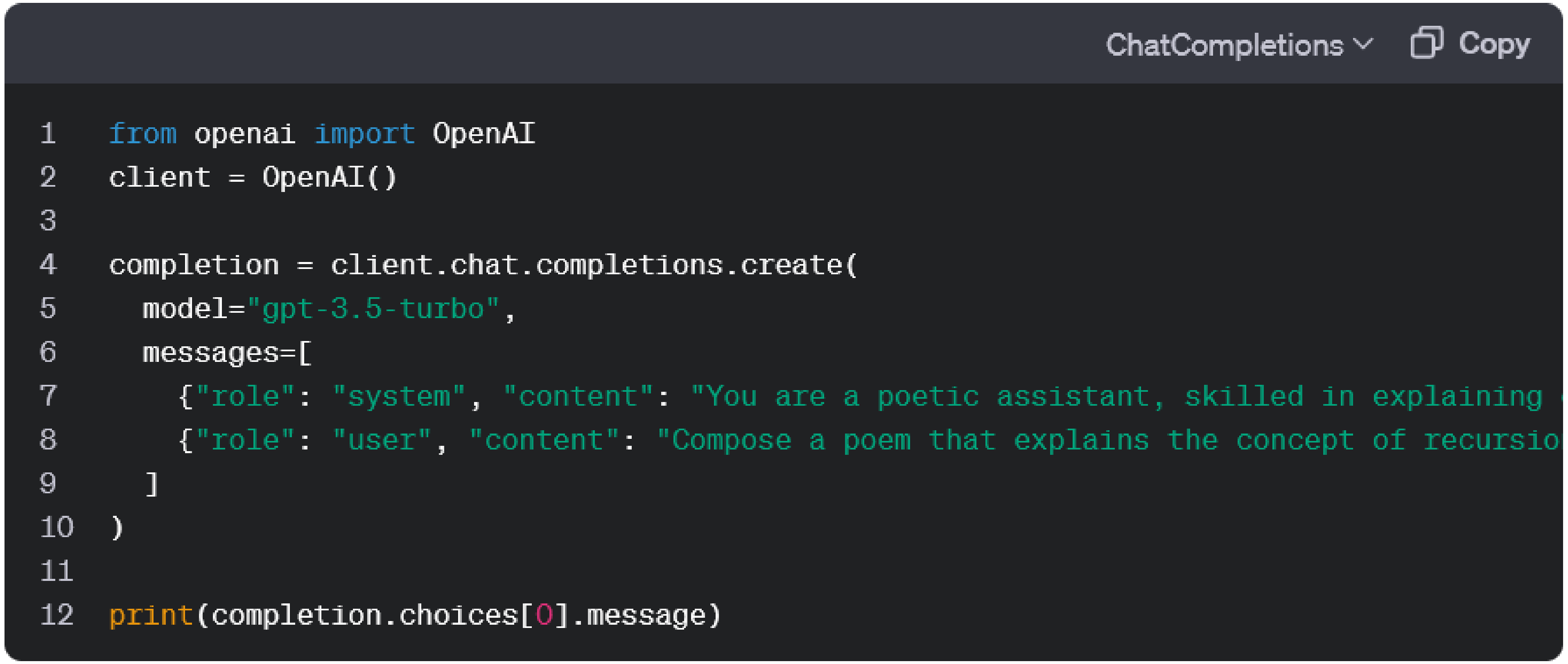
Exercises

Prof. Perttu Hämäläinen, Jan 2024

Tools

- For everyone:
  - OpenAI's GPT series models (Bing chat, Microsoft CoPilot mobile app, ChatGPT, OpenAI Playground, https://platform.openai.com)
  - Anthropic Claude 2, https://www.anthropic.com/index/claude-2
  - Write with Laika (GPT-2 models finetuned with specific public domain literary works, e.g., Kafka) https://www.writewithlaika.com/
- For programmers
  - OpenAI API
  - Amazon Bedrock
  - Huggingface Transformers library: Easy access to many models. Recommended: Meta's Llama 2 series, Mistral (Llama finetunes), Microsoft Phi 2 (very good and small model), Mamba (SSM instead of Transformer, soon available via Huggingface too)
  - OpenAI cookbook: https://cookbook.openai.com/

OpenAI API is super easy to use

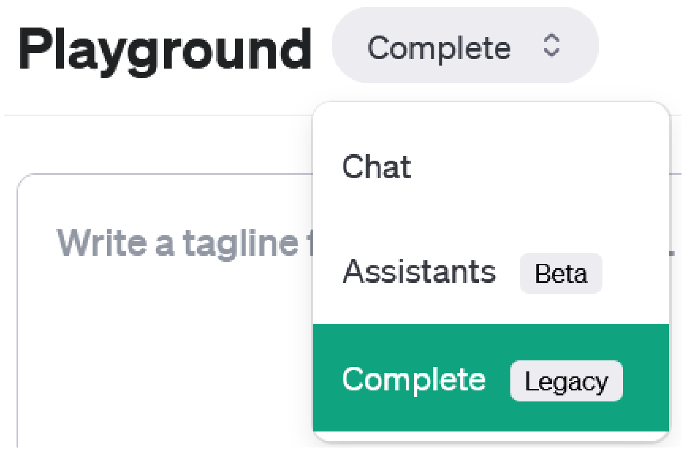


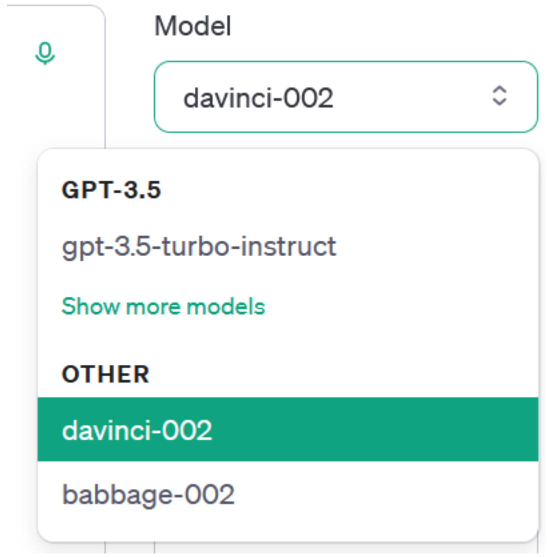
```python
from openai import OpenAI
client = OpenAI()

completion = client.chat.completions.create(
  model="gpt-3.5-turbo",
  messages=[
    {"role": "system", "content": "You are a poetic assistant, skilled in explaining
    {"role": "user", "content": "Compose a poem that explains the concept of recursio
  ]
)

print(completion.choices[0].message)
```

https://platform.openai.com/docs/quickstart?context=python

General guidance

- Try and compare different models to get the feel of their strengths and weaknesses
  - ChatGPT is very knowledgeable but has a specific default "voice" and low variablility in responses, davinci-002 requires more examples to work well but has higher variability & creativity. You can access davinci-002 in OpenAI Playground, by selecting "Complete" as the model type and "davinci-002" as the model. https://platform.openai.com/playground
- Remember the 2 principles: be specific and give concrete examples
- Try and practice well established prompting techniques: https://platform.openai.com/docs/guides/prompt-engineering





Exercise: Generate game ideas (everyone)

- Learning goal: See the effect of examples and practice finding & writing good examples
- Try the following with ChatGPT and compare the results:
    1. A simple prompt: “Please give me an innovative game idea”
    2. More specific prompt: “Please add more innovative game ideas to the list below. All ideas must have a clear design hook that makes players want to wishlist and try the game.” Then add one or more examples at the end of the prompt, e.g., as bullet points.
    3. Modify how the examples are written, e.g., make them more and less detailed. How does the generated output change?
    4. Try using davinci-002 instead of ChatGPT (see the previous slide). In this case, you have to change the prompt to something like “A list of experimental indie game ideas:” followed by ideas as bullet points.
    5. Repeat the above a few times, pick the best generations, and add them to the examples.
    6. Remember that when generating a list of things, **each generated item becomes an example for the next generated items**. Thus, the quality will eventually degrade, and it’s better to add a bit of code to prompt the model multiple times and collate the results. Here’s Colab notebook tutorial: http://colab.research.google.com/github/PerttuHamalainen/MediaAI/blob/master/Code/Jupyter/few_shot_prompting.ipynb

Variation: **If games are not your thing, generate book opening sentences or quotes.** For example material, you can Google “best opening sentences” or check Amazon’s most highlighted Kindle text segments.

Exercise: Stories with archetypes

Learning goal: practice finding example material for your prompts.

Prompt a story using one or more archetypes: https://thescriptlab.com/blogs/38406-250-character-archetypes-to-use-in-your-screenplay/

Include the archetype descriptions in your prompt and describe the desired interaction between the characters on a high level, e.g., what kind of a conflict the characters should have, or how the characters feel about each other.

250 Character Archetypes to Use in Your Screenplay

By Ken Miyamoto from ScreenCraft · August 30, 2023

All movie, television, and literary characters come from a general mold of character types called archetypes, which contain familiar character DNA that readers and audiences can instantly identify with. They are familiar and common within stories.

Exercise: Stories with archetypes (programmers)

Learning goal: Practice constructing prompts in code and scraping reference material.

Scrape the archetypes to a Python string list: https://thescriptlab.com/blogs/38406-250-character-archetypes-to-use-in-your-screenplay/

Using Python, randomize an archetype combination and construct a prompt to generate a story with the archetypes.

Optional: Also randomize an interaction between the characters.

250 Character Archetypes to Use in Your Screenplay

By Ken Miyamoto from ScreenCraft · August 30, 2023

All movie, television, and literary characters come from a general mold of character types called archetypes, which contain familiar character DNA that readers and audiences can instantly identify with. They are familiar and common within stories.

Exercise: Generate game ideas (programmers)

- Learning goal: Practice constructing prompts in code and automating tedious manual prompting. Choose one:
  1. Use OpenAI API or equivalent (in Colab or locally on your computer)
  2. Use an open source LLM locally or in Colab: Llama 2 7B, Mistral, or Microsoft Phi. Google "mistral colab" or "phi llm colab" to get started
- Scrape reference game descriptions, e.g., from Steam. Google "python web scraping" or ask ChatGPT how
- In your code, select a random sample of the descriptions and construct the prompt, starting with "Please add more innovative game ideas to the list below. All ideas must have a clear design hook that makes players want to wishlist and try the game."
- Send the prompt to the LLM
- Construct another prompt asking the model to select the best idea, starting e.g., "Which of the game ideas below is the most innovative and likely to compel players to wishlist and buy?"

Exercise: Visualize text (programmers)

- Learning goal: Practice using embeddings to explore and visualize text data.
- Starting point: 100 game ideas or opening sentences, either generated in the previous exercise or scraped from the Internet
- Compute embedding vectors for the texts using either OpenAI API or Sentence Transformers (google "sentence transformers embeddings colab"
- Reduce the dimensionality of the embeddings to 2 or 3 for visualization using PCA, UMAP, or MDS (google e.g. "python dimensionality reduction UMAP")
- Create an interactive scatterplot visualization using Plotly's Scatter function (google "plotly scatter colab"). Example: https://github.com/PerttuHamalainen/LLMCode/blob/master/test_results/bopp_test_visualization.gif

Exercise: Retrieval-augmented generation (Build your own ChatPDF, programmers)

- Retrieval-augmented generation means that the user can ask questions about some corpus of text, e.g., a pdf of a book
  1. Preprocess the corpus by computing embeddings of each paragraph
  2. Preprocess the question by computing its embedding
  3. Construct a prompt that includes paragraphs with the lowest cosine distance between the question and the paragraph embeddings
  4. Make a query to a LLM using the prompt

  There are multiple details to get right here, but a basic version is doable in a short time. This can also become your final project.

  Challenge: Also plug in a speech-to-text model such as OpenAI Whisper. Using it, create your data by transcribing a podcast such as https://www.idlethumbs.net/designernotes/

Exercise: Google Sheets automation (programmers)

- Learning goal: Learn to read a Google Sheets spreadsheet in Colab, use an LLM to process some column, and write the results back.
- Starting point: You are designing a spell casting system where the spells combine the following elements: wind, fire, water, earth
- Task: You want to generate good names for each combination. Make a spreadsheet that lists combinations in one column and some example names in another column. E.g., wind + fire = Scorching Hurricane, fire + water = Suffocating Steam. In Colab, prompt an LLM by providing the existing names as few-shot examples. Populate the empty cells of the name column with generated names.
- If your sheet is public, you can simply use:

```
data = pd.read_csv("<sheet .csv export URL here>")
```

- If you need to authenticate, see this example:

https://colab.research.google.com/notebooks/snippets/sheets.ipynb