Practical GPT Development for Business

Ernest W. Lessenger
July 27, 2023

About the Presenter

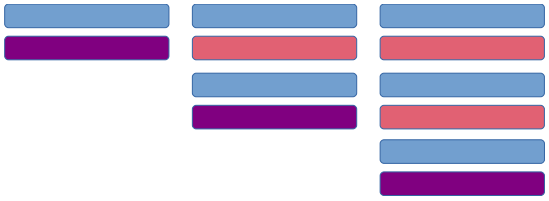
- BS in Technical Writing (CS minor) from Carnegie-Mellon University
- Master of Business Administration from Rice University
- Various Certifications from Amazon, Microsoft, SEI, and PMI

- 20 years of experience in PaaS
- Focus on integrating emerging technology into the Enterprise

www.lessenger.net

Terminology

- **Chat/Completions**



- **Training**

Input	Output
Input	Output
Input	Output
Input	Output
Input	Output
Input	Output

GPT is literally a “completion engine”: Given some input, what is the most likely response given everything that the model knows about language and the world.

So, a GPT “completion” can be viewed as a challenge and response loop: Ask a question; get a response; take action or formulate a new question; repeat.

Training is the act of teaching a GPT model what kinds of output should result from certain kinds of input. This is also known as “fine-tuning”.

Source Code

chat.openai.com

platform.openai.com

python.langchain.com

https://github.com/ernestlessenger/PracticalGPT

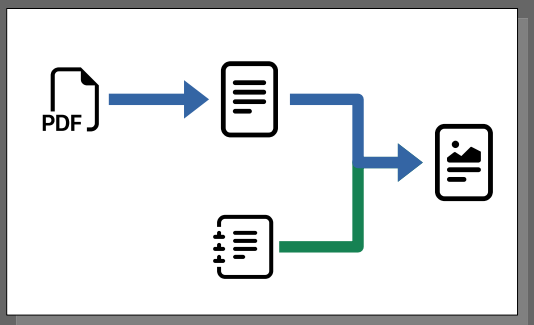
I am going to be reviewing two use-cases today.. The first use case is available as sample code along with a PDF of this presentation, here.

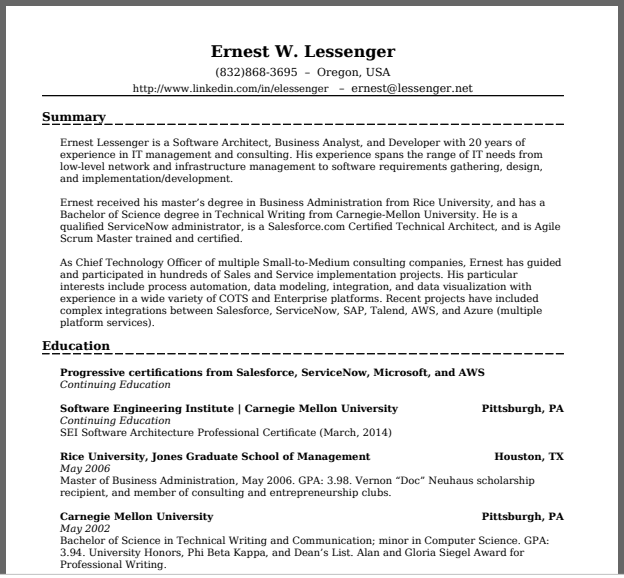
I wrote my code in Java for a few reasons:
First, it's what I'm familiar with.
Second, Java has the best and most reliable document processing libraries I know of (other than .Net), and that's most of what I do.
Third, I am a big fan of strongly-typed data structures especially when it comes to AI work.
Finally, I really like that solutions presented in Java nearly always "just work" in a platform- and version-independent way.

That said, I also provided a python file and requirements.txt based on langchain.

Use Case: Format a resume

- Given a single resume, reformat it to be easy to read.





Ernest W. Lessenger

(832)868-3695 – Oregon, USA

http://www.linkedin.com/in/elessenger – ernest@lessenger.net

Summary

Ernest Lessenger is a Software Architect, Business Analyst, and Developer with 20 years of experience in IT management and consulting. His experience spans the range of IT needs from low-level network and infrastructure management to software requirements gathering, design, and implementation/development.

Ernest received his master's degree in Business Administration from Rice University, and has a Bachelor of Science degree in Technical Writing from Carnegie-Mellon University. He is a qualified ServiceNow administrator, is a Salesforce.com Certified Technical Architect, and is Agile Scrum Master trained and certified.

As Chief Technology Officer of multiple Small-to-Medium consulting companies, Ernest has guided and participated in hundreds of Sales and Service implementation projects. His particular interests include process automation, data modeling, integration, and data visualization with experience in a wide variety of COTS and Enterprise platforms. Recent projects have included complex integrations between Salesforce, ServiceNow, SAP, Talend, AWS, and Azure (multiple platform services).

Education

Progressive certifications from Salesforce, ServiceNow, Microsoft, and AWS
Continuing Education

Software Engineering Institute | Carnegie Mellon University **Pittsburgh, PA**
Continuing Education
SEI Software Architecture Professional Certificate (March, 2014)

Rice University, Jones Graduate School of Management **Houston, TX**
May 2006
Master of Business Administration, May 2006. GPA: 3.98. Vernon "Doc" Neuhaus scholarship recipient, and member of consulting and entrepreneurship clubs.

Carnegie Mellon University **Pittsburgh, PA**
May 2002
Bachelor of Science in Technical Writing and Communication; minor in Computer Science. GPA: 3.94. University Honors, Phi Beta Kappa, and Dean's List. Alan and Gloria Siegel Award for Professional Writing.

The use case is formatting a resume.

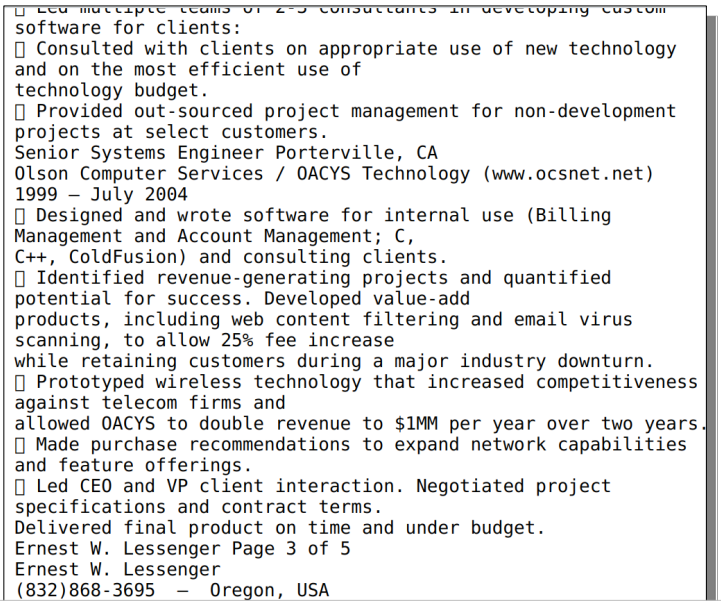
This is a great use case because everything important should already be available in a single document. It can be useful to reorganize the content to make a point, and it might even be useful to take advantage of general knowledge. But there's generally no room or opportunity for hallucinations or misinterpretation.

The general workflow is:

1. Extract the current version of the resume, either your old Word document or LinkedIn, or whatever.
2. Write a prompt that will result in a new, better resume.
3. Mash the two together and send it off to a potential employer.

Step 1: Convert the resume to text

- For each page...
- Extract the text
- There will be a lot of extra junk, bad formatting, etc… ChatGPT handles this well!



In the source code that I provided, I'm using the iText Core library. This is one of several that are available, including a few open-source versions from Apache.

Personally, I like iText for simple PDF work because it is a very clean and targeted API.

I like Aspose for general document manipulation.

If I don't need the reliability of a native solution, Google Chrome does a good job of rendering html pages to PDF.

If you have patience and control over the data, Apache POI is fine for OpenDocument and PDF creation.

Step 2: Create a prompt

```
You are a recruiter for a Fortune-500 IT department.

Reformat this resume to be easy to read.

Use only information provided in this resume. Do not add
information or use information from other sources. Be concise
and use business formal language.

Include four sections: A two-paragraph summary; Education;
Experience; Familiar Technologies

Format your response as Markdown with the first header at level
1. Enclose your response in <response></response> tags.

<resume>…</resume>
```

The second step is to write a prompt and inject both the prompt and the resume into the “context” of the request. Note that OpenAI has an 8k or 32k context limit (depending on how you call it).

There are a few important points to the prompt, and a LOT of YouTube videos teaching you how to write them. But in this case, you need to tell it:

* What persona to use (how to act)
* What you want it to do
* Any restrictions or special instructions
* What specifically you want it to output
* How you want the response formatted
* The actual context that you want it to act on

Step 3: Call ChatGPT

- ChatGPT takes an array of chat messages, and simply adds a new message to the end.
- Anthropic Claude just completes a string with “\n\nAssistant:” as the delimiter.

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

completion = openai.ChatCompletion.create(
  model="gpt-4",
  messages=[
    {"role": "system", "content": "You are a
helpful assistant."},
    {"role": "user", "content": "Hello!"}
  ]
)

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

https://platform.openai.com/docs/api-reference

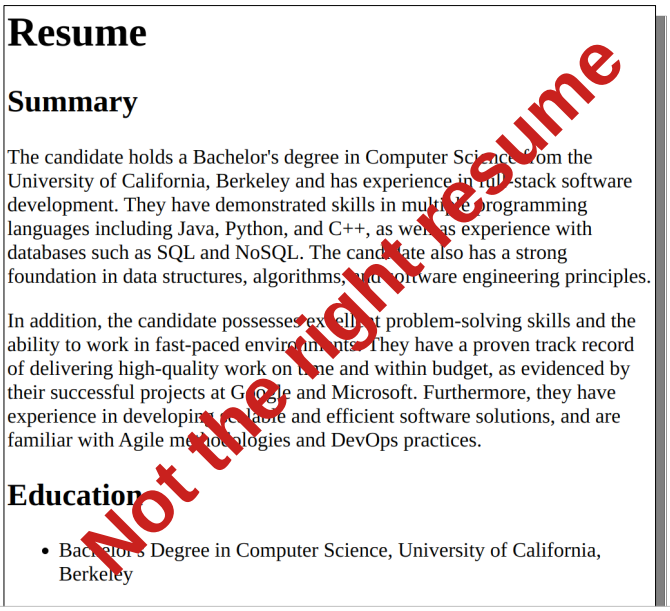
Make the API call. The APIs are actually VERY simple, consisting of the ENTIRE context of the conversation in each request. Both the context and the response are included in the 8k token limit, and in your bill.

OpenAI uses a JSON structure to record the system messages, user messages, assistant responses, and function requests in an array.
Claude uses a single String delimited by “\n\nHuman:” or “\n\nAssistant:”.

One trick that you can use is to build the context dynamically each time instead of actually keeping a chat history. For example: Delete any context that wasn’t used in a response. Because each API call is independent, you don’t actually have to embrace the “chat” paradigm.

Step 4: Profit!

- Use Commonmark to convert Md to HTML
- Use a tool to convert from HTML to PDF
- If you make a stupid mistake, ChatGPT generally won't warn you.



In the earlier example, I told it to output Markdown. This is a good format because, unlike XML/HTML, there is no need to terminate XML tags. Making life easier for the system saves money, and more importantly saves processing time for useful "thought".

BUT WAIT!!!

I forgot to include my resume in the context. ChatGPT hallucinated a resume that would get me hired.

Interlude: Problems with ChatGPT

- Hallucinations
  - Writing FROM the context vs
    Writing ABOUT the context vs
    Writing BASED ON the context
- Consistency of results
- Reliability of formatting
- Accuracy of answers
- Precision of answers
- Small context
- Access to context data
- Lack of training material
- Speed of APIs

So how do we avoid hallucinations?

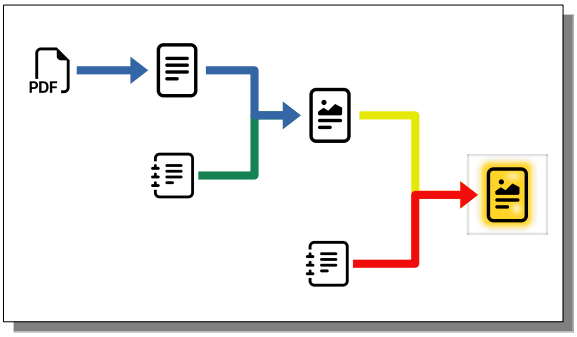
1. Always provide context.
2. Tell the AI to warn you if it doesn't have enough information, or to ask for more information.

3. The AI is probabilistic. This means that Exegesis (inference from the context) will give more reliable results than Eisegesis (interpretation of the prompt using general world knowledge).
4. But, asking it to be creative can lead to interesting results.

   In this case, it's a good idea to give it direction like: Come up with five ways that I could improve my resume and make me more likely to get hired.

Step 5: Profit!

- ChatGPT is good at extending, correcting, and augmenting
- But it gets really cool when you mashup contexts...



Ernest W. Lessenger

Contact: (832)868-3695
Location: Oregon, USA
LinkedIn: http://www.linkedin.com/in/elessenger
Email: ernest@lessenger.net

Summary

Ernest Lessenger is a seasoned Software Architect, Business Analyst, Developer, and IT leader with two decades of experience in IT management and consulting. His expertise spans low-level network and infrastructure management to software requirements gathering, design, and implementation/development. Ernest has served as Chief Technology Officer for several small to medium consulting companies, guiding numerous Sales and Service implementation projects. His interests encompass process automation, data modeling, integration, and data visualization, with experience in a wide variety of COTS and Enterprise platforms.

He possesses a master's degree in Business Administration from Rice University and a Bachelor of Science degree in Technical Writing from Carnegie-Mellon University. He is Agile Scrum Master trained and certified, a qualified ServiceNow administrator, and a Salesforce.com Certified Technical Architect. His recent projects have involved complex integrations between Salesforce, ServiceNow, SAP, Talend, AWS, and Azure (multiple platform services).

Education

- Master of Business Administration, Rice University, Jones Graduate School of Management, Houston, TX, May 2006
- Bachelor of Science in Technical Writing and Communication (Minor in Computer Science), Carnegie Mellon University, Pittsburgh, PA, May 2002
- SEI Software Architecture Professional Certificate, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, March 2014
- Progressive certifications from Salesforce, ServiceNow, Microsoft, and AWS

Experience

When I provide my resume as context. Things look a lot better.

But wait, there's more!

What if, given my new and improved resume, I give it a job description and ask it to write a cover letter, using my resume, for that job description?

There is an example of this in the source code.

Steps 6+: Moar Profit!

```
You are a recruiter for a Fortune-500 IT department.
Write a 1000 word cover letter explaining why this candidate is
the perfect person for the job described below.
This job does not yet exist, so you will need to also explain
why this job position would benefit Peace Health.
Format your response as Markdown with the first header at level
1. Enclose your response in <response></response> tags.

<job>Peace Health in Eugene, Oregon is looking for the right
person to manage their Artificial Intelligence practice. The
ideal candidate will have recent experience working with
Generative AI, Machine Learning, and Data Science.</job>

<resume>
# Ernest W. Lessenger

## Summary

Ernest Lessenger is a seasoned Software Architect, Business
Analyst, and Developer with two decades of experience in IT
management and consulting. His vast knowledge spans from low-
level network and infrastructure management to software
requirements gathering, design, and implementation/development.
Ernest has served as the Chief Technology Officer for multiple
Small-to-Medium consulting companies, guiding and participating
in numerous Sales and Service implementation projects. He has a
keen interest in process automation, data modeling, integration,
and data visualization with experience in a broad spectrum of
COTS and Enterprise platforms. Recent projects have included
sophisticated integrations between Salesforce, ServiceNow, SAP,
Talend, AWS, and Azure (multiple platform services)
```

Ernest W. Lessenger: The Ideal Candidate for AI Practice Manager at Peace Health

Peace Health, an esteemed healthcare organization in Eugene, Oregon, is advancing its technological capabilities to leverage the power of Artificial Intelligence (AI). To effectively manage this transformation, it is crucial to onboard an individual with proven expertise in AI, Machine Learning (ML), and Data Science. Ernest W. Lessenger, with his robust experience in IT management, consulting, and technical knowledge, is the perfect fit for this position.

Why Peace Health Needs an AI Practice Manager

Healthcare organizations worldwide are realizing the transformative potential of AI. It is revolutionizing patient care, predicting diseases, improving diagnostics, and streamlining administrative processes. To stay competitive, Peace Health needs to harness the power of AI, which calls for the creation of a dedicated position - the AI Practice Manager.

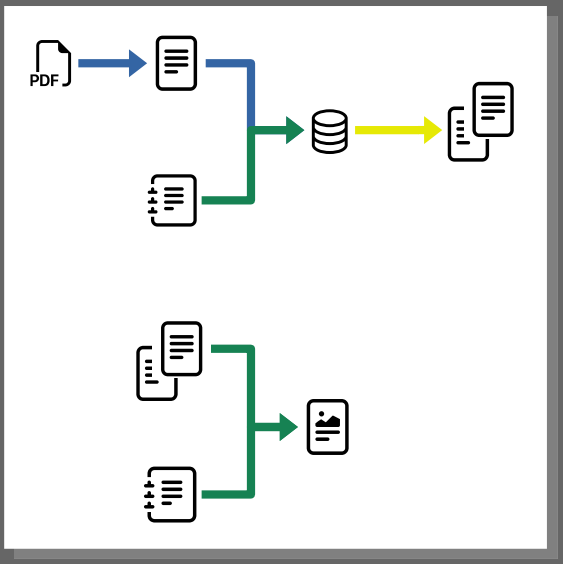
This role will be responsible for developing and implementing AI strategies, aligning AI initiatives with business objectives, and ensuring smooth AI

Use Case: Chatting with PDFs

1. I have a bunch of documents
2. I want to be able to ask questions about the documents and engage in a chat about it.

   Examples:

   1. Does my health plan cover hearing aids?
   2. What types of hearing aid are covered?
   3. What brands and models of hearing aid are covered?



Another popular use case is “Chatting with PDFs”. Essentially this means creating a database of text documents and using the entire database as context.

But you can’t do that… Most databases are more than 32k bytes of data.

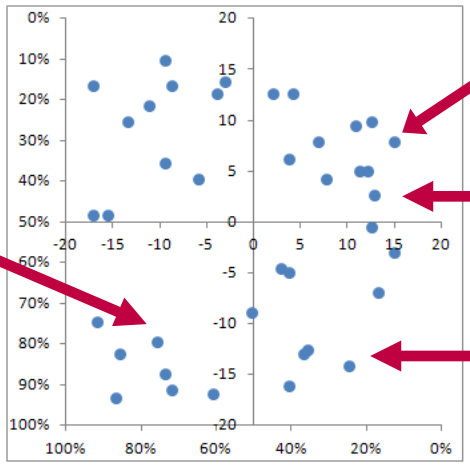
So you use a multi-step process:

1. Add all your documents to the database.
2. Use the question (the prompt) to obtain documents that are likely to have the answer.
3. Inject those documents *and* the original prompt into the context.
4. RECOMMENDED: Train the AI with answers to a lot of questions, possibly using real-world examples.

SORRY, NO CODE SAMPLES

More Terminology

- **Embeddings**
- **Vector Database**



"What hearing aids are covered by my plan?"

"The 'XYZ' plan covers most unfitted hearing aids..."

"The 'ABC' plan pays for annual doctors visits..."

"The XYZ plan covers hip replacements with a prior approval..."

This use-case requires a special database called a vector database. This is REALLY old technology, btw.

An Embedding is a vector of 1,500(ish) numbers between negative one and one. You can think of this as a point in 1500-dimensional concept space.

They "compress" a long, messy document into a concise numeric point in space. Different documents might be close to each other on the map, which implies that they talk about similar things.

The numbers don't have any actual meaning to humans, but it might help to think of them as "the degree to which this document relates to a concept."

For example: In the chart above, the x-axis might refer to plan coverage, while the y-axis refers to hearing aids. So a document that refers to plan coverage for hearing aids would appear in the top right.

Vector Databases like Pinecone are simply really good at finding the p-nearest neighbors in n-space.

Step 1: Build the database

1. Convert all the documents to text
2. Split the documents into reasonable-sized chunks
    1. Because there is an 8k token (2k word) limit per completion
    2. Fewer topics per document results in more precise search results
3. Obtain embeddings for each document
4. Upload the embeddings to Pinecone

The first step is the split the document into smaller chunks of data.

This is necessary because, remember, all GPT models have a context limit.

Also, the fewer concepts a single document talks about, the more precise and accurate the embeddings will be, and a smaller returned chunk will refer specifically to the topic of interest.

And, finally, it is helpful if the database returns a page number and document name that you can show to the end user.

Does this look familiar? It’s just a search engine!

Step 2: Search the database

1. Get a question from the end-user
2. Obtain an embedding from OpenAI
3. Using the new embedding, search Pinecone for the nearest 5 “chunks” of document.
4. You now have:
    1. The text of the chunk of document
    2. A link to the original document

The next step is the get the question from the user and convert it to an embedding. It’s not the same type of document as the source material from the previous slide…

But because it’s talking about the same kind of things (hearing aids, plan coverage) we expect that they will be nearby to each other in 1500-dimensional concept-space. Just maybe on the other side of the street, because one is a legal document while the other is a question.

Ask the database for the five chunks of plan document that are most similar to the question being asked.

Step 3: Build the Prompt

1. Inject the document chunks into a prompt, along with the question:

```
Use the five document fragments below to answer the
question “Does my insurance plan cover hearing aids?”.
If you need additional information, use the provided
APIs to obtain it. Format your response as an XML
document enclosed in <response> tags, and list the
source documents that you used in <source> tags.
<fragments>…</fragments>
```

Then, put the five plan documents (which you HOPE contain the answer) into the context along with the question.

Step 4: Answer the customer

1. Parse the response and extract any structured content.
2. Display the answer to the customer:

```
According to your insurance plan document
XYZ12b, hearing aids are covered if your
request is submitted on a form AB34z that
has been signed by your primary care
physician, your case worker, and three
specialists.
```

You may want to consider:

1. Using the OpenAI functions api to give ChatGPT a way to ask for more information.
2. Tell ChatGPT to use only information provided in the plan documents.
3. Instruct ChatGPT to refuse to answer if it doesn’t have an exact answer.

You may also want to train the API with real-world questions and answers.

And you might want to make a separate API call to check the response for bad answers.

Conclusion

Where I see things going

- APIs will mature, and models will become less important
  - OpenAI ChatGPT has a better API, and is supported by Microsoft.
  - Anthropic Claude has a more reliable and trustworthy responses.
  - AWS is focused on providing tools to call any number of models with a single API.
  - Google PaLM2 focuses more on citations, and has an “okay” API
- Structured access to data will become more important
  - OpenAI steered conversations
  - Functions
  - Structured input

Lessons Learned

- Writing good prompts is critical
  - Writing good pipelines is even more critical
- Create escape hatches for unexpected business situations, or use functions
- Context size is a limiting factor… The public, paid APIs have enough context to do cool things but not REALLY cool things.
- Structured interaction is a challenge
  - Obtaining context, getting clarification, responding, and acting on the response
- I don’t know which APIs and which companies will be around in 12 months

Thank you!

https://github.com/ernestlessenger/PracticalGPT

Tools I Use

- Salesforce
- ServiceNow
- Talend
- Laptop, Operating System
- Office365
- IntelliJ IDEA
- GitHub Copilot
- GitHub (or Bitbucket)
- Snowflake, Databricks
- Fivetran and DBT
- Postman
- GitKraken

Tools I Use for AI

- OpenAI ChatGPT
  - Chat API
  - Embeddings
- Anthropic Claude
  - Completion API
- Pinecone
  - Vector Database
- Amazon Lambda/S3/ElasticSearch
- Microsoft Teams (transcription)
- Vaadin (Java UI)
- Python and langchain
- DBT and Snowflake

Thank you!

https://github.com/ernestlessenger/PracticalGPT