Speedreading with Large Language Models

Text Summarization in Python

A Reproducible Research Workshop

Simon Stone
Research Data Services
Dartmouth College

Reproducible Research @ DARTMOUTH

About the Reproducible Research Group

- Joint venture of **Research Computing @ ITC** and **Research Data Services @ Library**
- Consult with **experts** on
  - research data management,
  - data visualization,
  - biomedical research support,
  - spatial data and GIS,
  - high performance and research computing,
  - statistical analysis,
  - economics and social sciences data
- **Meet** the people on campus that support your reproducible research lifecycle
- **Engage** in community discussions to learn from other researchers on campus
- Attend a workshop to **learn** practical tools and tips

About Research Data Services

Research Data Management

Data Management Plans (DMPs) for sponsored projects

Finding and using 3rd party data

Collection and cleaning of data

Organization and documentation

Publishing and Repositories

Data Analysis/Visualization

Textual, numeric, spatial data

Reproducible research workflows

Scripting in R: tidyverse core package (i.e. ggplot, dplyr, tydr, tibble, etc.)

Scripting in Python: NumPy, SciPy, Pandas, Scikit-learn, Matplotlib, Seaborn, (OpenCV, PyTorch, TensorFlow, Tesseract, NLTK, etc.)

Computational Scholarship

Computational project planning

Collections as Data

Storytelling with data and visualizations

Text and data mining

Digital Humanities support

Computational Pedagogy

Work with us

ResearchDataHelp@groups.dartmouth.edu

Jeremy Mikecz
Research Data Science Specialist
jeremy.m.mikecz@dartmouth.edu
dartgo.org/jeremyappts

Simon Stone
Research Data Science Specialist
simon.stone@dartmouth.edu
dartgo.org/meetwithsimon

Lora Leligdon
Head of Research Data Services
lora.c.leligdon@dartmouth.edu
dartgo.org/lora

Why use Large Language Models?

- 📚 Large Language Models are artificial intelligence systems trained on massive amounts of text to “understand” and generate human language
- 🤖 Large Language Models are not just chat bots
- 💪 They can be powerful text processing and analysis tools
- 💬 They allow natural language interaction instead of code
- 🏇 They can do many tasks a human reader could do, but at scale

Why *not* use Large Language Models?

- 😱 Large Language Models are Large and require staggering resources
- 😕 The best Large Language Models are accessible through commercial APIs (OpenAI, Google VertexAI):
  - 💸 Cost may become significant for large amounts of text
  - 👀 Privacy and confidentiality is at risk

But...

- 🛠️ We can pick and choose the right model for the right task
  - 💵 Helps to manage cost (e.g., GPT-3.5 is 1/10th of the cost of GPT-4)
- 💻 We can run smaller, less general-purpose models on our own machine
  - 🤝 Trade-offs have to be made between performance and speed

What you will learn in this workshop

- **How** to interact with a Large Language Model in Python
- **How** to efficiently use prompts through code
- **How** to summarize documents using an LLM
- **How** to perform other text analysis tasks with LLMs

What we will work with in this workshop

- Platform: https://jhub.Dartmouth.edu
- **Python**
- LangChain
- OpenAI's GPT 3.5
- Materials: **www.dartgo.org/computational-text-analysis-week**

Let’s get started…

Getting an OpenAI API key

- To interact with OpenAI's models through code, you need an API key
- This key is used to identify you as a user and bill you for the cost
- To get your own API key:
  1. Sign up for an OpenAI account and log in
  2. Go to the API section in your account
  3. Generate a new API key and save it (you will only see it here once!)
  4. Set up billing and usage limits to avoid surprise charges

Using your API key

🔑 An API key is as sensitive as a password

😱 Anyone with your API key could use the paid (!) service in your name

🔒 Keep it secret, keep it safe!

Good practice:

☝️ Never use the key explicitly in your code

☝️ Refer to the key from an environment variable

☝️ Never check your key into version control (e.g., git)!

Setting up the key in your environment on JHub

- Create a new file in the folder `RR-workshops/text-analysis/computational-text-analysis-week`:
  - Navigate to the folder using the file explorer pane
  - Right-click in the empty area and select “New File”
  - Rename the file to “secrets.env” (make sure to also change the extension txt!)
- Add the key to `secrets.env`
  - Open the file by double-clicking
  - Add the following line (replacing everything between and including ”<>” with your key):

    `OPENAI_API_KEY=“<your_key_here>”`

  - For today’s session, you can use the provided key (courtesy of Dartmouth College Library)
- We will read the file `secrets.env` from the Python code and only refer to the key using the variable name

Takeaways

🚀 LLMs are a foundational technology

📚 They are a great tool for text processing

🤯 We have mature libraries available to quickly create complex applications involving LLMs

🛠️ LLMs add a completely new, very powerful tool to our belt

📦 While they are fast, how LLMs produce results is very opaque

Next steps

- 💬 Browse prompts for all kinds of applications: https://smith.langchain.com/hub
- 👨‍💻 Create a UI for your application using Streamlit
- 🤖 Try out local models by GPT4All
- 🔎 Explore Retrieval Augmented Generation (RAG)

Thank you.