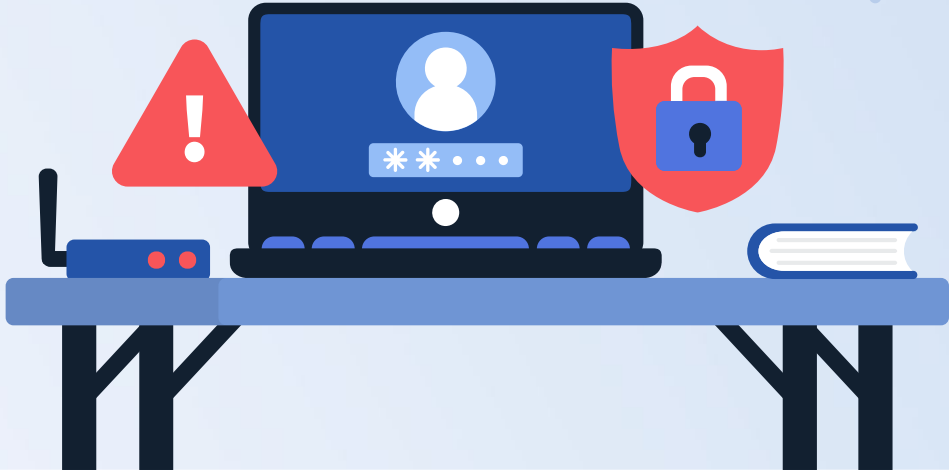
Lab 03

Visualizing Attention

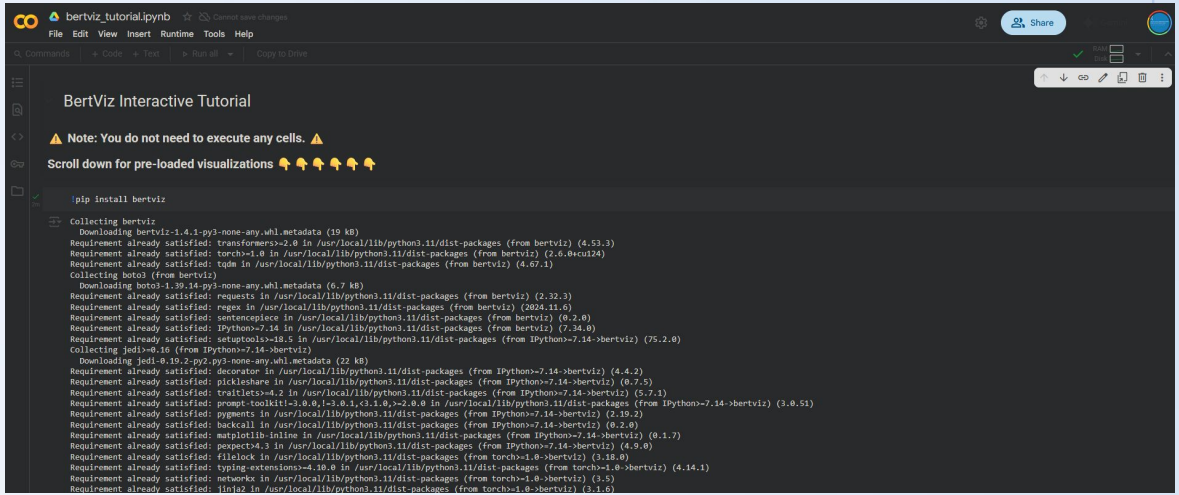
Lab Objectives

- Learn how to log into Google Colab
- Learn the basics of Colab
- Run the demo to visualize Colab



Navigate to the Demo Lab

- Navigate to the demo lab:
- https://colab.research.google.com/drive/1hXIQ77A4TYS4y3UthWF-Ci7V7vVUoxmQ?usp=sharing#scrollTo=aR07__FyOf8a



What is Google Colab?

- Google Colab is simple a web representation of Jupyter Notebooks
- “Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations, and narrative text. It is a popular tool among data scientists, researchers, and educators for interactive computing and data analysis. The name "Jupyter" is derived from the three core programming languages it originally supported: Julia, Python, and R.”
- https://www.geeksforgeeks.org/data-science/jupyter-notebook/

What is Google Colab?

- Google Colab-ratory is a free service that allows you to run Juypter notebooks in the google ecosystem
- It is free
  - Depends on available resources
  - Typically a T4 instance in GCP
- https://research.google.com/colaboratory/faq.html

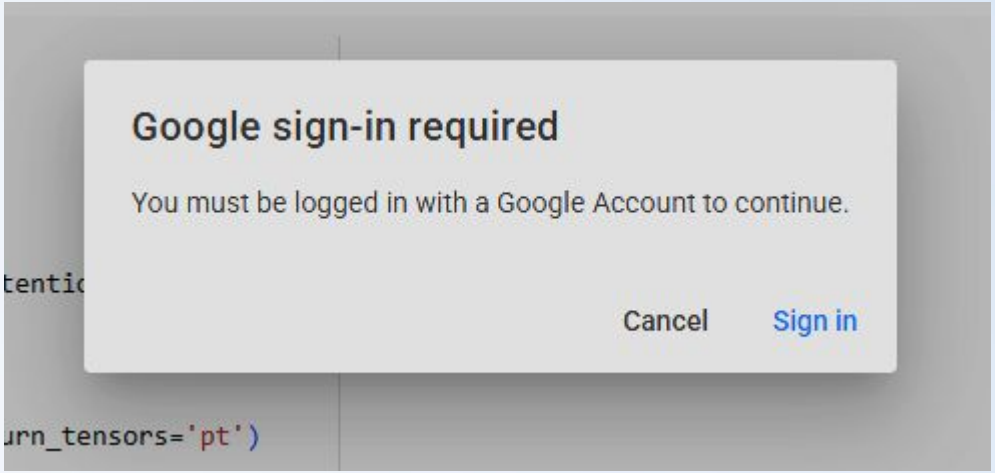
Google Colab
Absolute beginners

What are Jupyter notebooks

- A way to execute code via a web browser
  - But so much more
- Notebook kernels are computational engines that execute blocks of code in a notebook document
- Client-Server model
- “Notebook documents (or “notebooks”, all lower case) are documents produced by the Jupyter Notebook App, which contain both computer code (e.g. python) and rich text elements (paragraph, equations, figures, links, etc…). Notebook documents are both human-readable documents containing the analysis description and the results (figures, tables, etc..) as well as executable documents which can be run to perform data analysis.”

Getting started

- Log into a google account to get started
  - If you need assistance with creating a real account or a “sock puppet” account please let us know



Running through the demo lab

- Once you are signed in you will run through the playbook “chunk” by “chunk” by clicking the play buttons next to each chunk

Running through the demo lab

- As you finish each section, you will see a little green checkmark next to a successful run

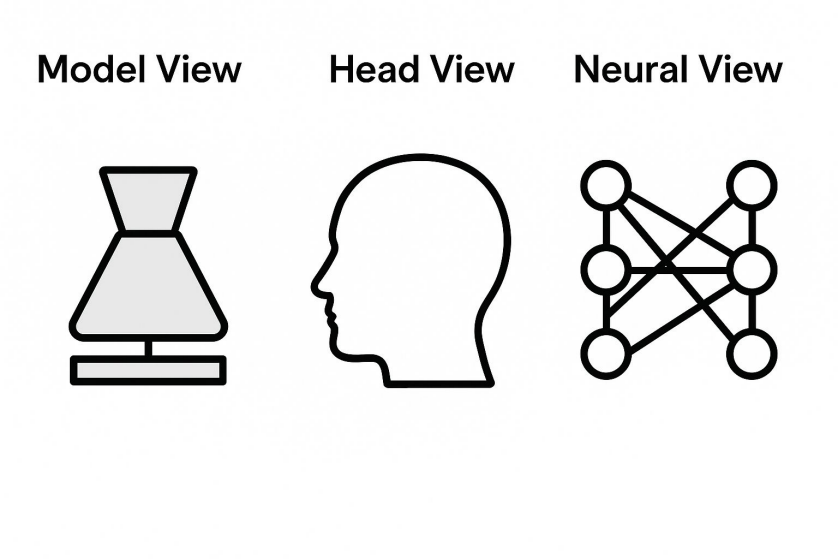
```
# Load model and retrieve attention weights

from bertviz import head_view, model_view
from transformers import BertTokenizer, BertModel

model_version = 'bert-base-uncased'
model = BertModel.from_pretrained(model_version, outp
tokenizer = BertTokenizer.from_pretrained(model_versi
sentence_a = "The cat sat on the mat"
sentence_b = "The cat lay on the rug"
inputs = tokenizer.encode_plus(sentence_a, sentence_b
input_ids = inputs['input_ids']
```

Model Views

- Once completed you will see several views demonstrated in the following slides



Visualizing Attention

- Head View:

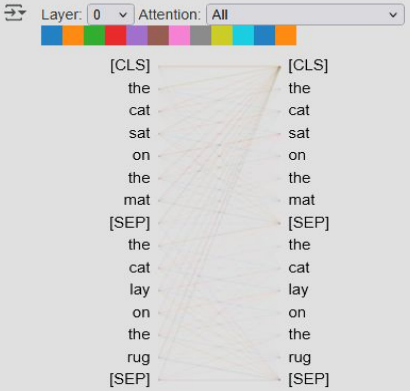
Head View

The head view visualizes attention in one or more heads from a single Transformer layer. Each line shows the attention from one token (left) to another (right). Line weight reflects the attention value (ranges from 0 to 1), while line color identifies the attention head. When multiple heads are selected (indicated by the colored tiles at the top), the corresponding visualizations are overlaid onto one another. For a more detailed explanation of attention in Transformer models, please refer to the blog.

Usage

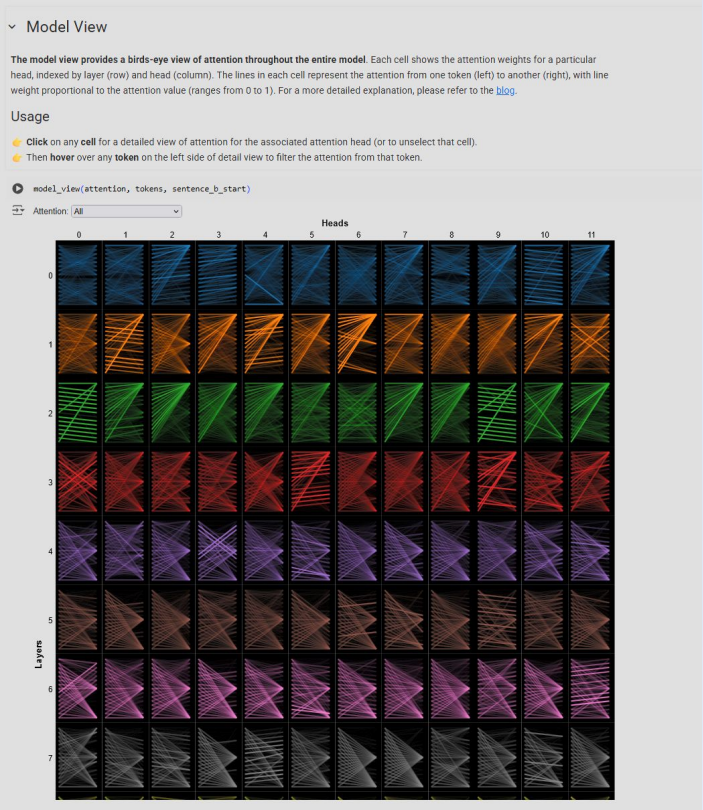
- 👉 **Hover** over any **token** on the left/right side of the visualization to filter attention from/to that token.
- 👉 **Double-click** on any of the **colored tiles** at the top to filter to the corresponding attention head.
- 👉 **Single-click** on any of the **colored tiles** to toggle selection of the corresponding attention head.
- 👉 **Click** on the **Layer** drop-down to change the model layer (zero-indexed).

```
head_view(attention, tokens, sentence_b_start)
```



Visualizing Attention

- Model View:



Visualizing Attention

- Neural View

Neuron View

The neuron view visualizes the intermediate representations (e.g. query and key vectors) that are used to compute attention. In the collapsed view (initial state), the lines show the attention from each token (left) to every other token (right). In the expanded view, the tool traces the chain of computations that produce these attention weights. For a detailed explanation of the attention mechanism, please refer to the blog.

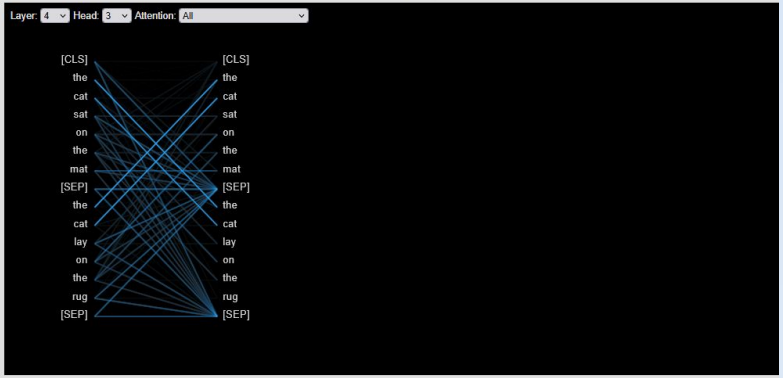
Usage

👉 **Hover** over any of the tokens on the left side of the visualization to filter attention from that token.
👉 Then **click** on the **plus** icon that is revealed when hovering. This exposes the query vectors, key vectors, and other intermediate representations used to compute the attention weights. Each color band represents a single neuron value, where color intensity indicates the magnitude and hue the sign (blue=positive, orange=negative).
👉 Once in the expanded view, **hover** over any other **token** on the left to see the associated attention computations.
👉 **Click** on the **Layer** or **Head** drop-downs to change the model layer or head (zero-indexed).

+ Code + Text

```
from bertviz.transformers_neuron_view import BertModel, BertTokenizer
from bertviz.neuron_view import show

model_type = 'bert'
model_version = 'bert-base-uncased'
model = BertModel.from_pretrained(model_version, output_attentions=True)
tokenizer = BertTokenizer.from_pretrained(model_version, do_lower_case=True)
show(model, model_type, tokenizer, sentence_a, sentence_b, layer=4, head=3)
```



Visualizing Attention

- Github repository: https://github.com/jessevig/bertviz?tab=readme-ov-file

Lab End