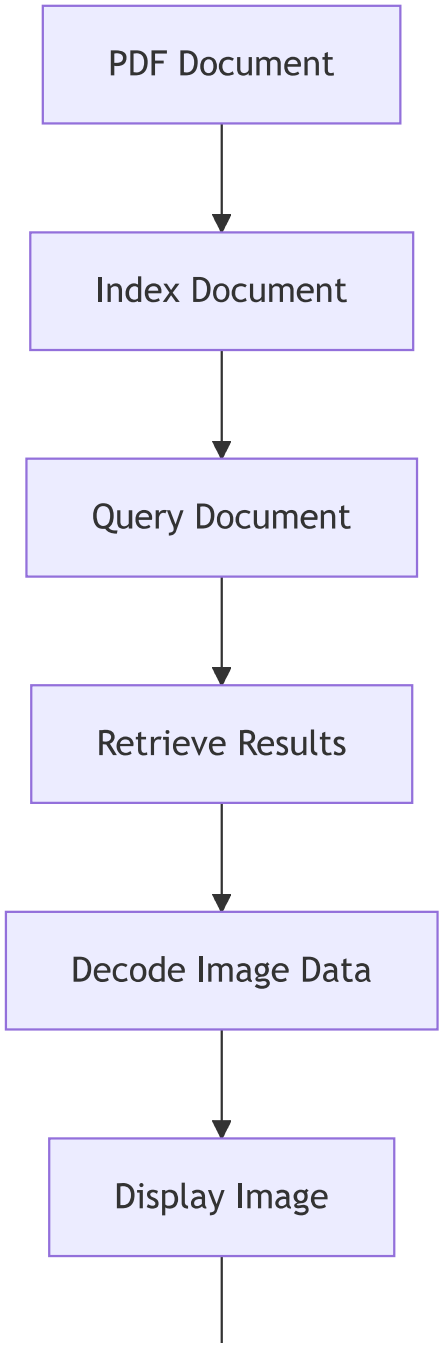
Overview:

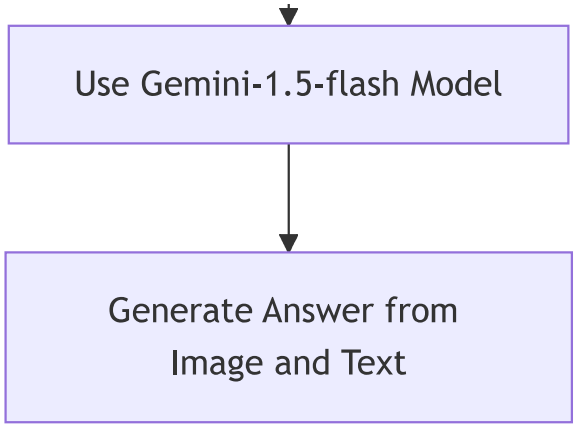
This code implements one of the multiple ways of multi-model RAG. This project processes a PDF file, retrieves relevant content using Colpali, and generates answers using a multi-modal RAG system. The process includes document indexing, querying, and summarizing with the Gemini model.

Key Components:

- **RAGMultiModalModel**: Used for document indexing and retrieval.
- **PDF Processing**: Downloads and processes "Attention is All You Need" paper.
- **Gemini Model**: Used for content generation from retrieved images and queries.
- **Base64 Encoding/Decoding**: Manages image data retrieved during search.

Diagram:





Motivation:

To enable efficient querying and content generation from multi-modal documents (PDFs with text and images) in response to natural language queries.

Method Details:

- Indexing: The PDF is indexed using the `RAGMultiModalModel`, storing both text and image data.
- Querying: Natural language queries retrieve relevant document segments.
- Image Processing: Images from the document are decoded, displayed, and used in conjunction with the Gemini model to generate content.

Benefits:

- Multi-modal support for both text and images.
- Streamlined retrieval and summarization pipeline.
- Flexible content generation using advanced LLMs (Gemini model).

Implementation:

- PDF is indexed, and the content is split into text and image segments.
- A query is run against the indexed document to fetch the relevant results.
- Retrieved image data is decoded and passed through the Gemini model for answer generation.

Summary:

This project integrates document indexing, retrieval, and content generation in a multi-modal setting, enabling efficient queries on complex documents like research papers.

Setup

In [ ]:

```
!pip install -q byaldi
!sudo apt-get install -y poppler-utils # for windows - https://stackoverflow.
```

In [ ]:
```
!pip install -q git+https://github.com/huggingface/transformers.git qwen-vl-u
```



```
os.environ["HF_token"] = 'your-huggingface-api-key' # to download the ColPali
from byaldi import RAGMultiModalModel
```

In [ ]:
```
RAG = RAGMultiModalModel.from_pretrained("vidore/colpali-v1.2", verbose=1)
```

Download the "Attention is all you need" paper

In [ ]:
```
!wget https://arxiv.org/pdf/1706.03762
!mkdir docs
!mv 1706.03762 docs/attention_is_all_you_need.pdf
```

```
--2024-09-20 12:36:38--  https://arxiv.org/pdf/1706.03762
Resolving arxiv.org (arxiv.org)... 151.101.3.42, 151.101.131.42, 151.101.67.42,
...
Connecting to arxiv.org (arxiv.org)|151.101.3.42|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2215244 (2.1M) [application/pdf]
Saving to: ‘1706.03762’


1706.03762            0%[                    ]       0  --.-KB/s
1706.03762          100%[===================>]   2.11M  --.-KB/s    in 0.01s

2024-09-20 12:36:38 (164 MB/s) - ‘1706.03762’ saved [2215244/2215244]
```

Indexing

In [ ]:
```
RAG.index(
    input_path="./docs/attention_is_all_you_need.pdf",
    index_name="attention_is_all_you_need",
    store_collection_with_index=True, # set this to false if you don't want t
    overwrite=True
)
```

```
Starting from v4.46, the `logits` model output will have the same type as the m
odel (except at train time, where it will always be FP32)
Added page 1 of document 0 to index.
Added page 2 of document 0 to index.
Added page 3 of document 0 to index.
Added page 4 of document 0 to index.
Added page 5 of document 0 to index.
Added page 6 of document 0 to index.
Added page 7 of document 0 to index.
Added page 8 of document 0 to index.
Added page 9 of document 0 to index.
Added page 10 of document 0 to index.
Added page 11 of document 0 to index.
```

```
Added page 12 of document 0 to index.
Added page 13 of document 0 to index.
Added page 14 of document 0 to index.
Added page 15 of document 0 to index.
Index exported to .byaldi/attention_is_all_you_need
Index exported to .byaldi/attention_is_all_you_need
```

Out[ ]: {0: 'docs/attention_is_all_you_need.pdf'}

Query time

In [ ]:
```
query = "What is the BLEU score of the Transformer (base model)?"
```

In [ ]:
```
results = RAG.search(query, k=1)
```

Actual image data

In [ ]:
```
image_bytes = base64.b64decode(results[0].base64)
```

In [ ]:
```
filename = 'image.jpg'  # I assume you have a JPG file
with open(filename, 'wb') as f:
  f.write(image_bytes)
```

In [ ]:
```
from IPython.display import Image

display(Image(filename))
```

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

Residual Dropout We apply dropout [33] to the output of each sub-layer, before it is added to the sub-layer input and normalized. In addition, we apply dropout to the sums of the embeddings and the positional encodings in both the encoder and decoder stacks. For the base model, we use a rate of $P_{drop} = 0.1$.

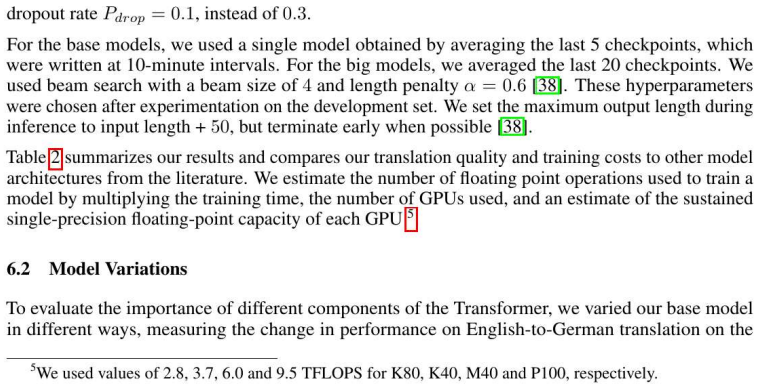
Label Smoothing During training, we employed label smoothing of value $\epsilon_{ls} = 0.1$ [36]. This hurts perplexity, as the model learns to be more unsure, but improves accuracy and BLEU score.

6 Results

6.1 Machine Translation

On the WMT 2014 English-to-German translation task, the big transformer model (Transformer (big) in Table 2) outperforms the best previously reported models (including ensembles) by more than 2.0 BLEU, establishing a new state-of-the-art BLEU score of 28.4. The configuration of this model is listed in the bottom line of Table 3. Training took 3.5 days on 8 P100 GPUs. Even our base model surpasses all previously published models and ensembles, at a fraction of the training cost of any of the competitive models.

On the WMT 2014 English-to-French translation task, our big model achieves a BLEU score of 41.0, outperforming all of the previously published single models, at less than $1/4$ the training cost of the previous state-of-the-art model. The Transformer (big) model trained for English-to-French used



dropout rate $P_{drop} = 0.1$, instead of 0.3.

For the base models, we used a single model obtained by averaging the last 5 checkpoints, which were written at 10-minute intervals. For the big models, we averaged the last 20 checkpoints. We used beam search with a beam size of 4 and length penalty $\alpha = 0.6$ [38]. These hyperparameters were chosen after experimentation on the development set. We set the maximum output length during inference to input length + 50, but terminate early when possible [38].

Table 2 summarizes our results and compares our translation quality and training costs to other model architectures from the literature. We estimate the number of floating point operations used to train a model by multiplying the training time, the number of GPUs used, and an estimate of the sustained single-precision floating-point capacity of each GPU [5].

6.2 Model Variations

To evaluate the importance of different components of the Transformer, we varied our base model in different ways, measuring the change in performance on English-to-German translation on the

[5] We used values of 2.8, 3.7, 6.0 and 9.5 TFLOPS for K80, K40, M40 and P100, respectively.

8

Test using gemini-1.5-flash

In [ ]:
```
import google.generativeai as genai

genai.configure(api_key='your-api-key')
model = genai.GenerativeModel(model_name="gemini-1.5-flash")
```

In [ ]:
```
from PIL import Image
image = Image.open(filename)
```

In [ ]:
```
response = model.generate_content([image, query])
print(response.text)
```

```
The BLEU score of the Transformer (base model) is 27.3.
```