Open Domain Abstractive Question Answering System using Transformers for Tamil Language

Team

- Supervisor
  - Dr. Betina Antony J
- Members
  - Abdul Azeez B 185001004
  - Aswin M 185001027
  - Balaji S 185001032

Problem Statement

- Today the world is full of articles on large variety of topics. Search becoming inevitable in everyone’s life. Most of the systems are keyword based which will most of the times not provide what we want especially for languages like Tamil which has less internet resources.
- So We aim to build a question-answering system that can understand the context and answer the questions in a meaning full way.

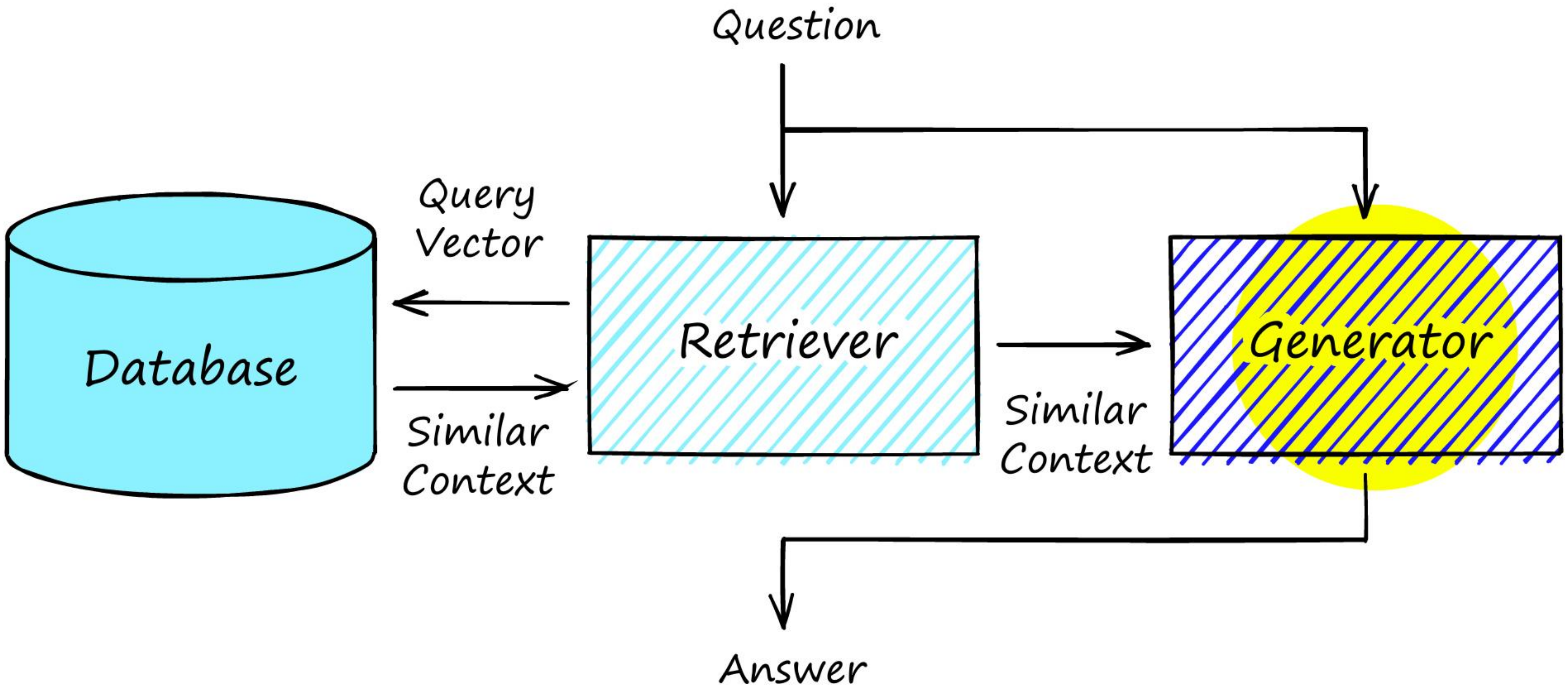
Motivation(Problem with keyword search system)

- In keyword based search system when a person knows what they are looking for and they know the keywords and terminology of the information they need, a keyword-based search is ideal.
- When the keywords and terminology of the answer are unknown, keyword search is inadequate.
- People searching for unknown answers in large repositories of documents is inefficient.
- So it will be helpful if there is a system that understands the context and respond user i.e answer lies with semantic search

Proposed system
Abstractive Open-domain Question Answering System (AODQA)

- Context : இந்தியாவில் ஒரிசா (இன்றைய ஒடிசா) மாநிலத்தில் கட்டாக் எனும் இடத்தில், 1897 ஆம் ஆண்டு சனவரி 23 ஆம் நாள், வங்காள, இந்துக் குடும்பத்தில், சுபாஷ் சந்திரபோஸ் பிறந்தார்.
- Question : நேதாஜி பிறந்த ஊர் எது?
- Answer : நேதாஜி ஒரிசா மாநிலத்தில் கட்டாக் எனும் இடத்தில் பிறந்தார்.
- AODQA is a task to generate an exact answer to a question
- Here we use transformer models(which understands the context well) to answer the questions
- Model will produce answers to factoid questions in natural language.

Abstractive QA Model Architecture



Abstractive QA Model Methodology

- Create embedding for context using transformers
- Store embeddings in database
- Information retrieval
    - Retrieve some k embeddings with similar context to question
    - Identify the text of embeddings
- Answer generation
    - With the context and question using seq2seq model generate some answers

Dataset

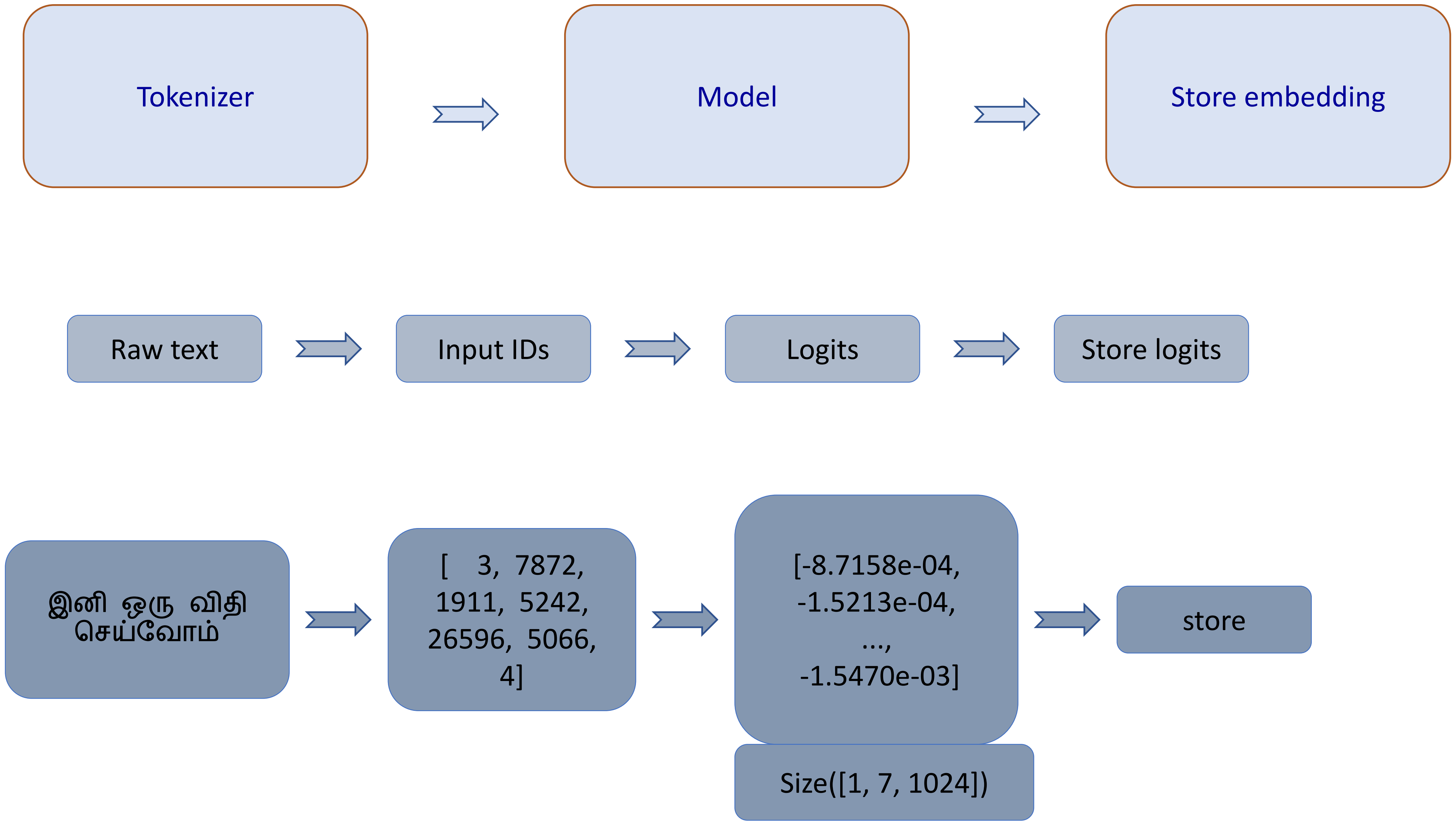
- Structure(TQA)
  - Context
  - Question
  - Answers
  - Id
- 300 TQA collected from chaii dataset
- 1000 TQA from XQA dataset
- We preparing dataset on our own also..

Components of QA system

- Dense vector generator
- Indexed data (vector database)
- Retriever
- Answer generator

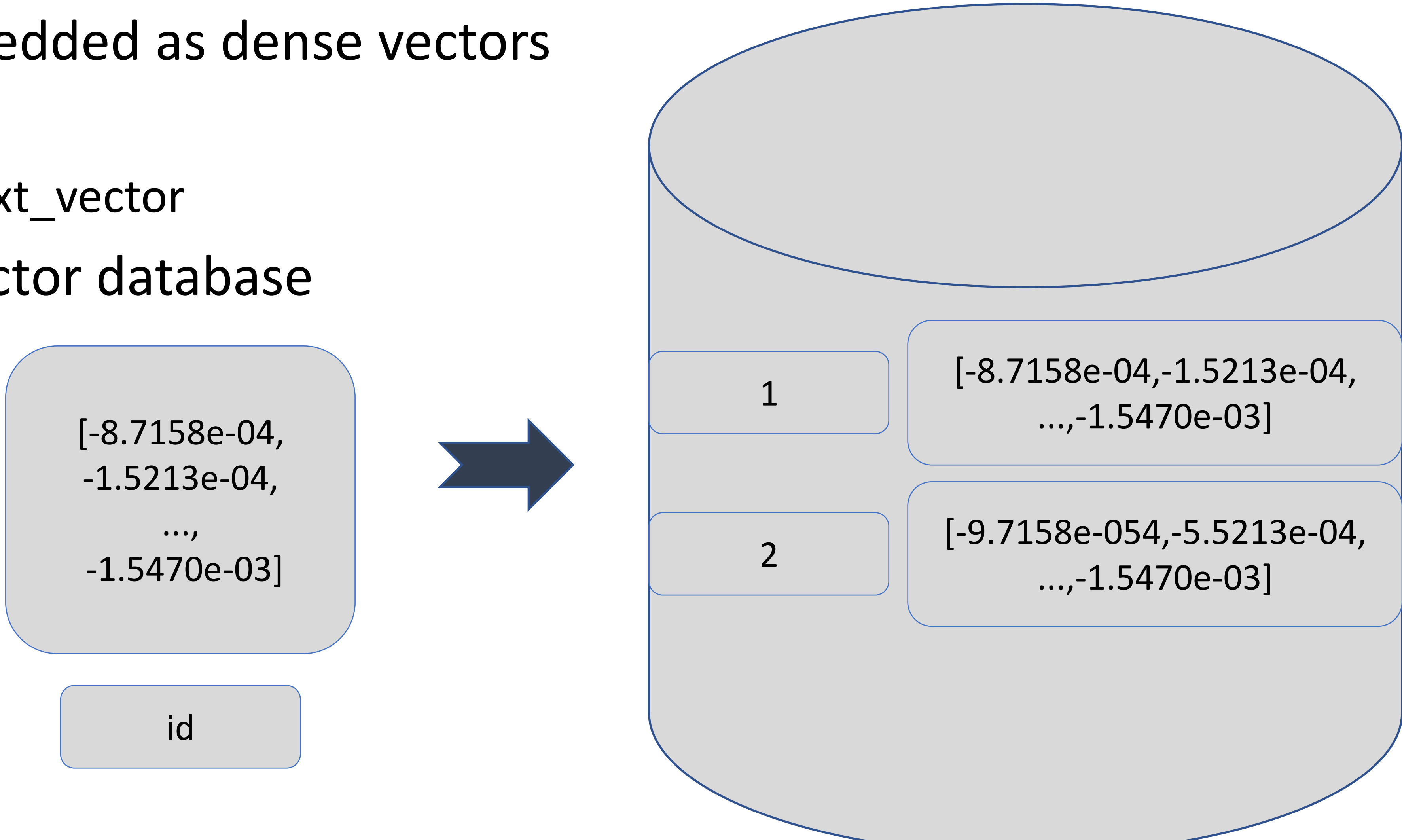
Dense vector generator model

- We use a transformer model called tamillion which is based on ELECTRA
- Tokenization
  - Tokenization methodology used is subword tokenization
- subword tokenization principle
  - Frequently used words should not be split into smaller subwords, but rare words should be decomposed into meaningful subwords
- Raw text will be given to tamillion model
- Model’s tokenizer will tokenize the raw text and return input IDS
- Model will convert the IDS into dense vector


Tokenizer
Model
Store embedding
Raw text
Input IDs
Logits
Store logits
இனி ஒரு விதி செய்வோம்
[3, 7872, 1911, 5242, 26596, 5066, 4]
[-8.7158e-04, -1.5213e-04, ..., -1.5470e-03]
Size([1, 7, 1024])
store

Indexed data (vector database)

- Contexts are embedded as dense vectors
- Database schema
  - Conext_id, context_vector
- We use Milvus vector database



Retriever IR

- Dense vectors have the advantage of enabling search via semantics.
- Question is also embedded as a dense vector and using some similarity measures some top k contexts were retrieved form database
  - max_ki=0ndense_vector_contexti * dense_vector_question
-

Methodology for IR

- Neural IR

$$h_x = E_x(x) \quad h_z = E_z(z) \quad \text{score}(x, z) = h_x^\top h_z$$

- X – question
- Z – context
- Feeding them into the language model and hx and hz is obtained
- Using similarity measures like dot product most similar document is retrieved

Answer generation(Abstractive QA)

- Abstractive QA model uses the question and context to generate an answer using a generative sequence-to-sequence (seq2seq) model
- Seq2seq models are BART, T5, GPT-2
- We will fine tune T5 seq2seq model for tamil question answering

How it is done

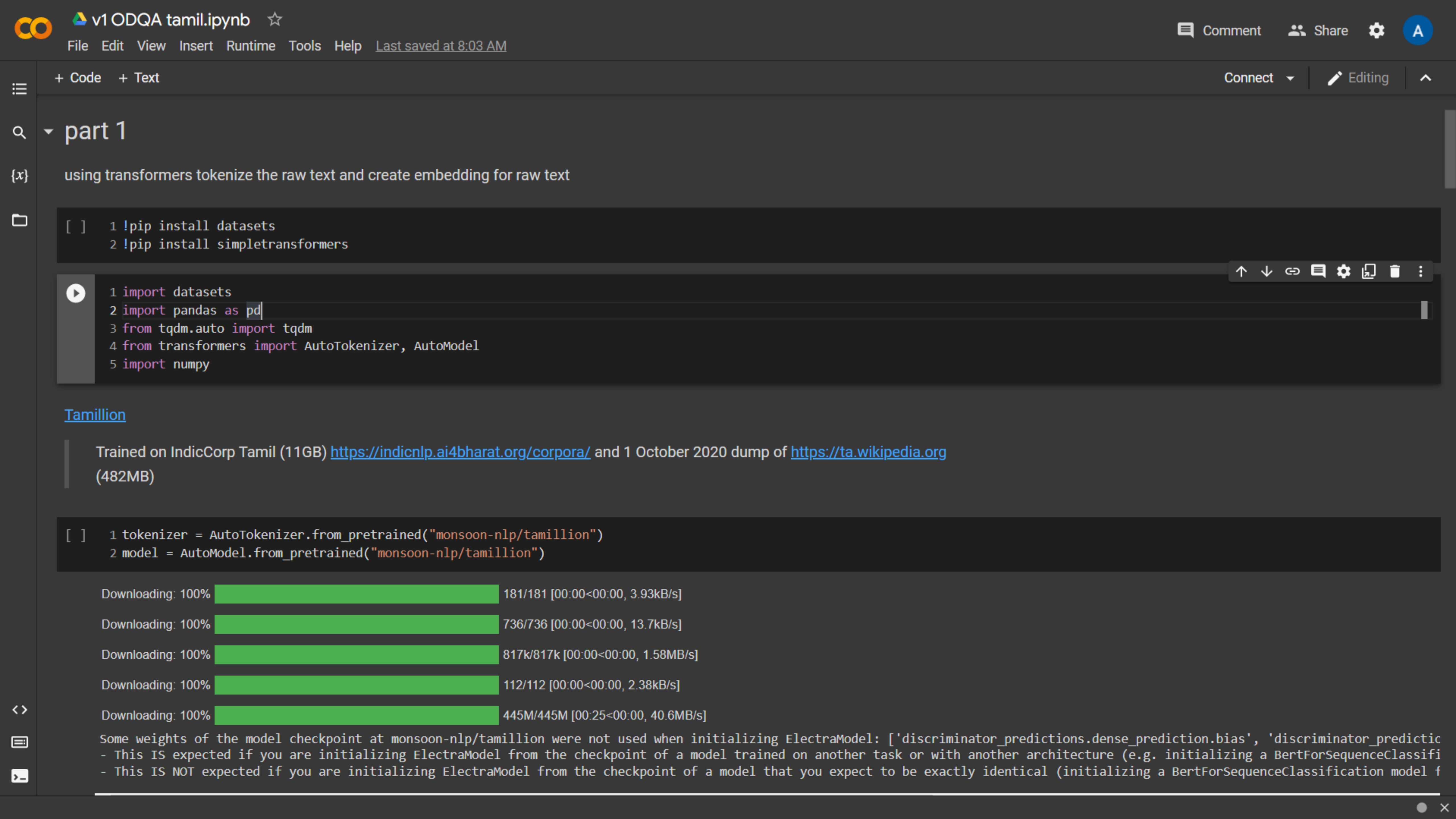
- Large transformer models store ‘representations’ of knowledge in their parameters.
- By passing relevant contexts and questions into the model, the model will use the context alongside its ‘stored knowledge’ to answer more abstract questions.
- We only focus on a single-turn QA

Over all flow

- A simple transformer model that trained with large volume of text data masked language modeling.
- Given the context it is vectorized using the model and saved in the database
- Given the question it is also vectorized and using cosine similarity context some top k similar context are retrieved form database
- Another generative transformer that trained to answer the question will take the input question and most similar context and figure out an answer to user question

Evaluation

- The true answer is objective, so it is simple to evaluate model performance.s
- Test data’s answers are dense vectorized (ans1)
- Model’s answers are also dense vectorized (ans2)
- Find similarity score for ans1 and ans2
- This similarity score is evaluated as model’s performance
- Dot product of the ans1 and ans2 is similarity score



v1 ODQA tamil.ipynb

File Edit View Insert Runtime Tools Help Last saved at 8:03 AM

part 1

using transformers tokenize the raw text and create embedding for raw text

```
!pip install datasets
!pip install simpletransformers
```

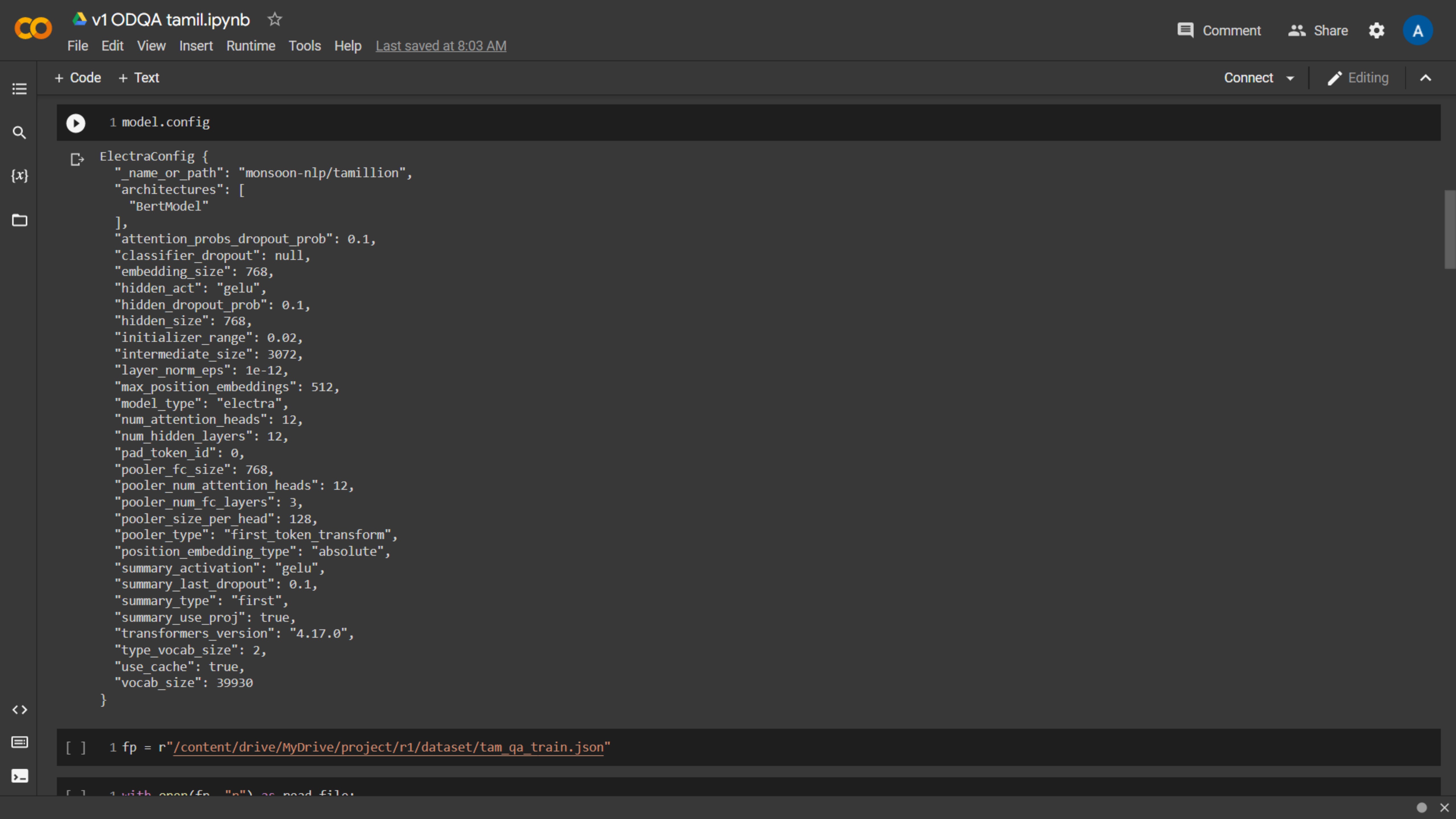
```
import datasets
import pandas as pd
from tqdm.auto import tqdm
from transformers import AutoTokenizer, AutoModel
import numpy
```

Tamillion

> Trained on IndicCorp Tamil (11GB) https://indicnlp.ai4bharat.org/corpora/ and 1 October 2020 dump of https://ta.wikipedia.org (482MB)

```
tokenizer = AutoTokenizer.from_pretrained("monsoon-nlp/tamillion")
model = AutoModel.from_pretrained("monsoon-nlp/tamillion")
```

```
Downloading: 100% 181/181 [00:00<00:00, 3.93kB/s]
Downloading: 100% 736/736 [00:00<00:00, 13.7kB/s]
Downloading: 100% 817k/817k [00:00<00:00, 1.58MB/s]
Downloading: 100% 112/112 [00:00<00:00, 2.38kB/s]
Downloading: 100% 445M/445M [00:25<00:00, 40.6MB/s]
Some weights of the model checkpoint at monsoon-nlp/tamillion were not used when initializing ElectraModel: ['discriminator_predictions.dense_prediction.bias', 'discriminator_predictic
- This IS expected if you are initializing ElectraModel from the checkpoint of a model trained on another task or with another architecture (e.g. initializing a BertForSequenceClassifi
- This IS NOT expected if you are initializing ElectraModel from the checkpoint of a model that you expect to be exactly identical (initializing a BertForSequenceClassification model f
```

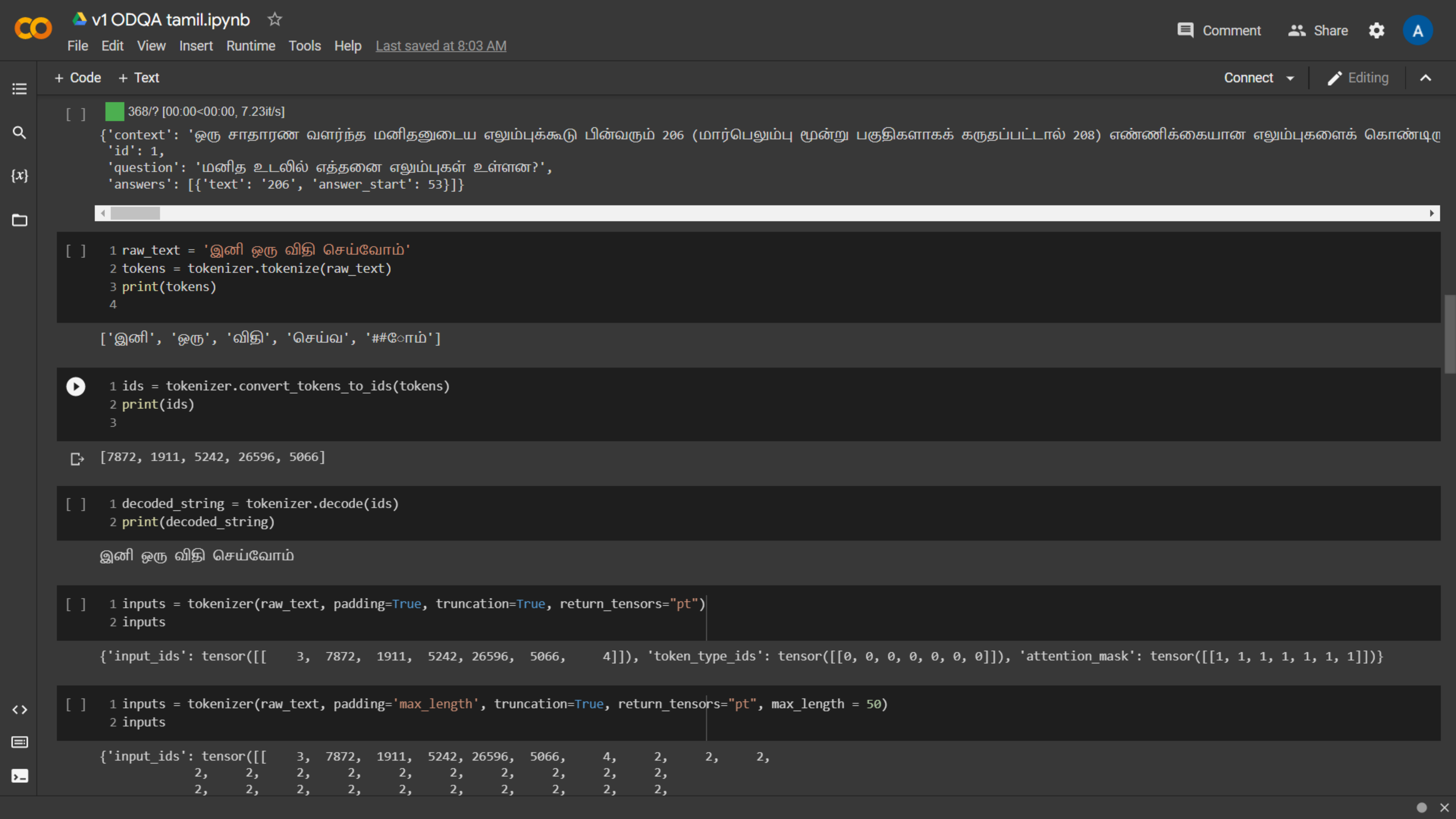

v1 ODQA tamil.ipynb
File Edit View Insert Runtime Tools Help Last saved at 8:03 AM
Comment Share
+ Code + Text
Connect Editing

```
model.config
```

```
ElectraConfig {
  "_name_or_path": "monsoon-nlp/tamillion",
  "architectures": [
    "BertModel"
  ],
  "attention_probs_dropout_prob": 0.1,
  "classifier_dropout": null,
  "embedding_size": 768,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 768,
  "initializer_range": 0.02,
  "intermediate_size": 3072,
  "layer_norm_eps": 1e-12,
  "max_position_embeddings": 512,
  "model_type": "electra",
  "num_attention_heads": 12,
  "num_hidden_layers": 12,
  "pad_token_id": 0,
  "pooler_fc_size": 768,
  "pooler_num_attention_heads": 12,
  "pooler_num_fc_layers": 3,
  "pooler_size_per_head": 128,
  "pooler_type": "first_token_transform",
  "position_embedding_type": "absolute",
  "summary_activation": "gelu",
  "summary_last_dropout": 0.1,
  "summary_type": "first",
  "summary_use_proj": true,
  "transformers_version": "4.17.0",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 39930
}
```

```
[ ] fp = r"/content/drive/MyDrive/project/r1/dataset/tam_qa_train.json"
```

```
[ ] with open(fp, "r") as read_file:
```


v1 ODQA tamil.ipynb
File Edit View Insert Runtime Tools Help Last saved at 8:03 AM
Comment Share
+ Code + Text
Connect Editing

```
🟩 368/? [00:00<00:00, 7.23it/s]
{'context': 'ஒரு சாதாரண வளர்ந்த மனிதனுடைய எலும்புக்கூடு பின்வரும் 206 (மார்பெலும்பு மூன்று பகுதிகளாகக் கருதப்பட்டால் 208) எண்ணிக்கையான எலும்புகளைக் கொண்டி
 'id': 1,
 'question': 'மனித உடலில் எத்தனை எலும்புகள் உள்ளன?',
 'answers': [{'text': '206', 'answer_start': 53}]}
```

```
raw_text = 'இனி ஒரு விதி செய்வோம்'
tokens = tokenizer.tokenize(raw_text)
print(tokens)
```

```
['இனி', 'ஒரு', 'விதி', 'செய்வ', '##ோம்']
```

```
ids = tokenizer.convert_tokens_to_ids(tokens)
print(ids)
```

```
[7872, 1911, 5242, 26596, 5066]
```

```
decoded_string = tokenizer.decode(ids)
print(decoded_string)
```

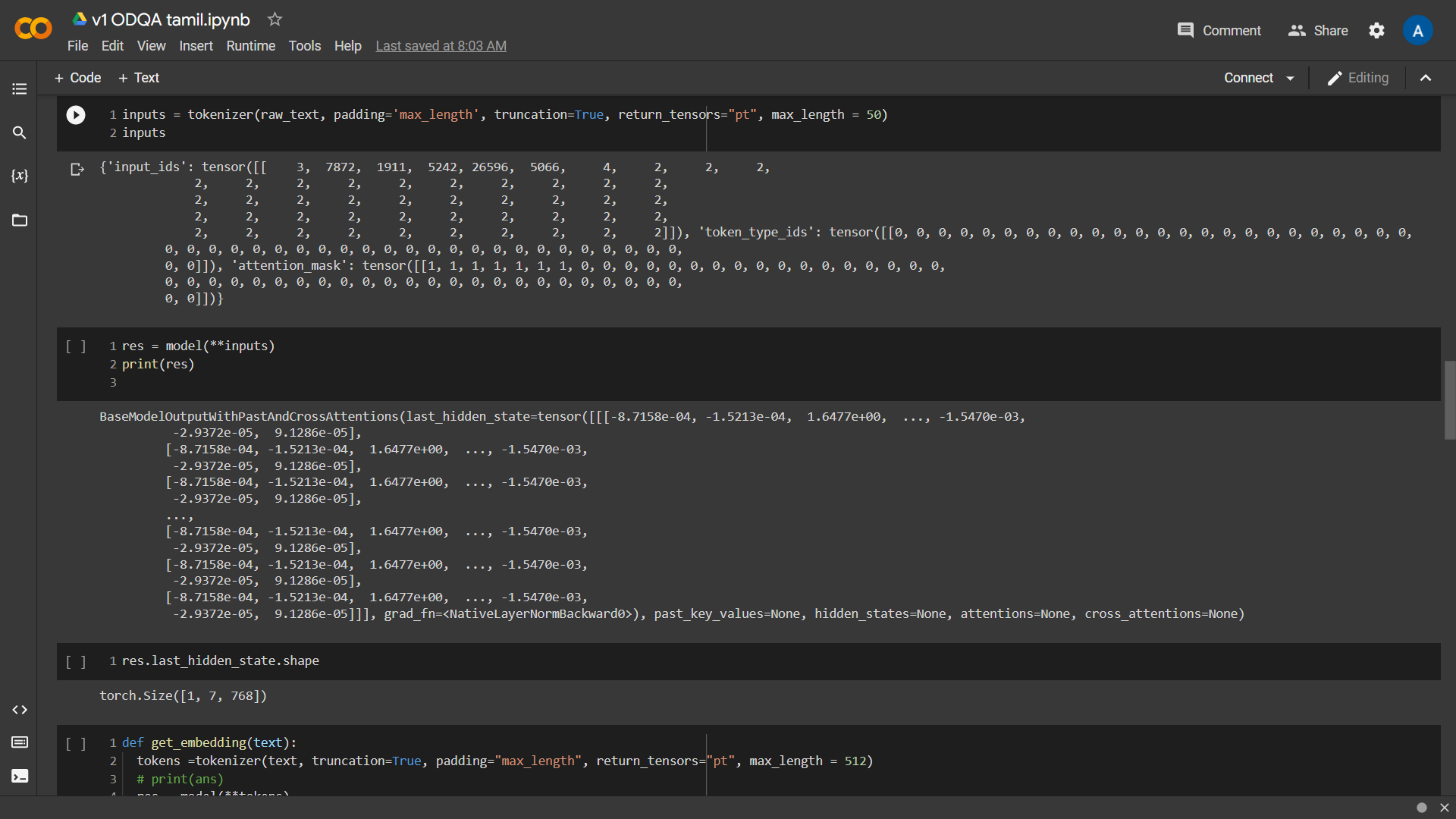
```
இனி ஒரு விதி செய்வோம்
```

```
inputs = tokenizer(raw_text, padding=True, truncation=True, return_tensors="pt")
inputs
```

```
{'input_ids': tensor([[    3,  7872,  1911,  5242, 26596,  5066,     4]]), 'token_type_ids': tensor([[0, 0, 0, 0, 0, 0, 0]]), 'attention_mask': tensor([[1, 1, 1, 1, 1, 1, 1]])}
```

```
inputs = tokenizer(raw_text, padding='max_length', truncation=True, return_tensors="pt", max_length = 50)
inputs
```

```
{'input_ids': tensor([[    3,  7872,  1911,  5242, 26596,  5066,     4,     2,     2,     2,
             2,     2,     2,     2,     2,     2,     2,     2,     2,     2,
             2,     2,     2,     2,     2,     2,     2,     2,     2,     2,
```


v1 ODQA tamil.ipynb
File Edit View Insert Runtime Tools Help Last saved at 8:03 AM
Comment Share
+ Code + Text
Connect Editing

```
inputs = tokenizer(raw_text, padding='max_length', truncation=True, return_tensors="pt", max_length = 50)
inputs
```

```
{'input_ids': tensor([[    3,  7872,  1911,  5242, 26596,  5066,     4,     2,     2,     2,
             2,     2,     2,     2,     2,     2,     2,     2,     2,     2,
             2,     2,     2,     2,     2,     2,     2,     2,     2,     2,
             2,     2,     2,     2,     2,     2,     2,     2,     2,     2,
             2,     2,     2,     2,     2,     2,     2,     2,     2,     2]]), 'token_type_ids': tensor([[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0]]), 'attention_mask': tensor([[1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0]])}
```

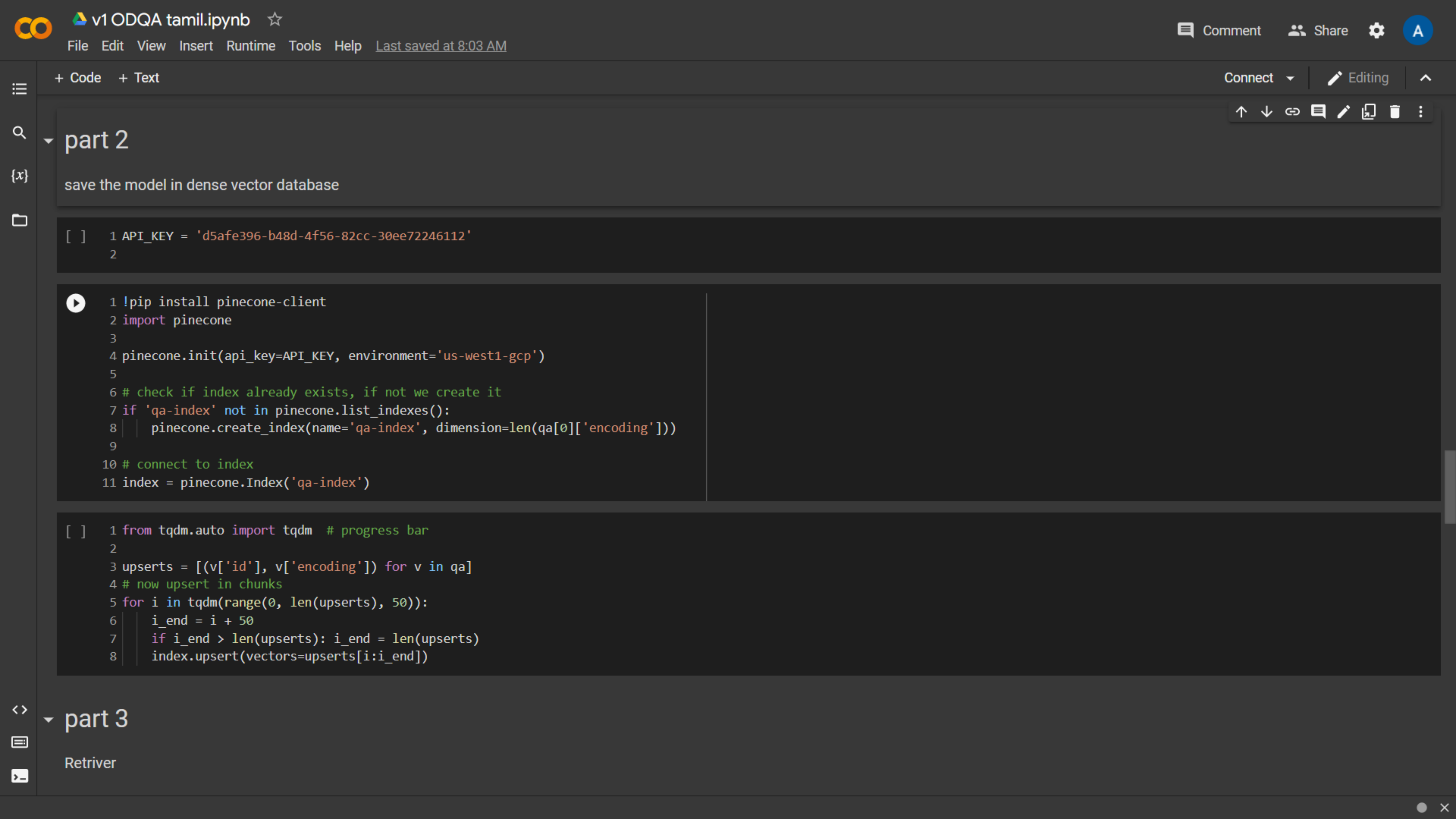
```
res = model(**inputs)
print(res)

```

```
BaseModelOutputWithPastAndCrossAttentions(last_hidden_state=tensor([[[-8.7158e-04, -1.5213e-04,  1.6477e+00,  ..., -1.5470e-03,
          -2.9372e-05,  9.1286e-05],
         [-8.7158e-04, -1.5213e-04,  1.6477e+00,  ..., -1.5470e-03,
          -2.9372e-05,  9.1286e-05],
         [-8.7158e-04, -1.5213e-04,  1.6477e+00,  ..., -1.5470e-03,
          -2.9372e-05,  9.1286e-05],
         ...,
         [-8.7158e-04, -1.5213e-04,  1.6477e+00,  ..., -1.5470e-03,
          -2.9372e-05,  9.1286e-05],
         [-8.7158e-04, -1.5213e-04,  1.6477e+00,  ..., -1.5470e-03,
          -2.9372e-05,  9.1286e-05],
         [-8.7158e-04, -1.5213e-04,  1.6477e+00,  ..., -1.5470e-03,
          -2.9372e-05,  9.1286e-05]]], grad_fn=<NativeLayerNormBackward0>), past_key_values=None, hidden_states=None, attentions=None, cross_attentions=None)
```

```
res.last_hidden_state.shape
```

```
torch.Size([1, 7, 768])
```

```
def get_embedding(text):
  tokens =tokenizer(text, truncation=True, padding="max_length", return_tensors="pt", max_length = 512)
  # print(ans)
```


v1 ODQA tamil.ipynb
File Edit View Insert Runtime Tools Help Last saved at 8:03 AM
Comment Share
+ Code + Text
Connect Editing

part 2

save the model in dense vector database

```
API_KEY = 'd5afe396-b48d-4f56-82cc-30ee72246112'
```

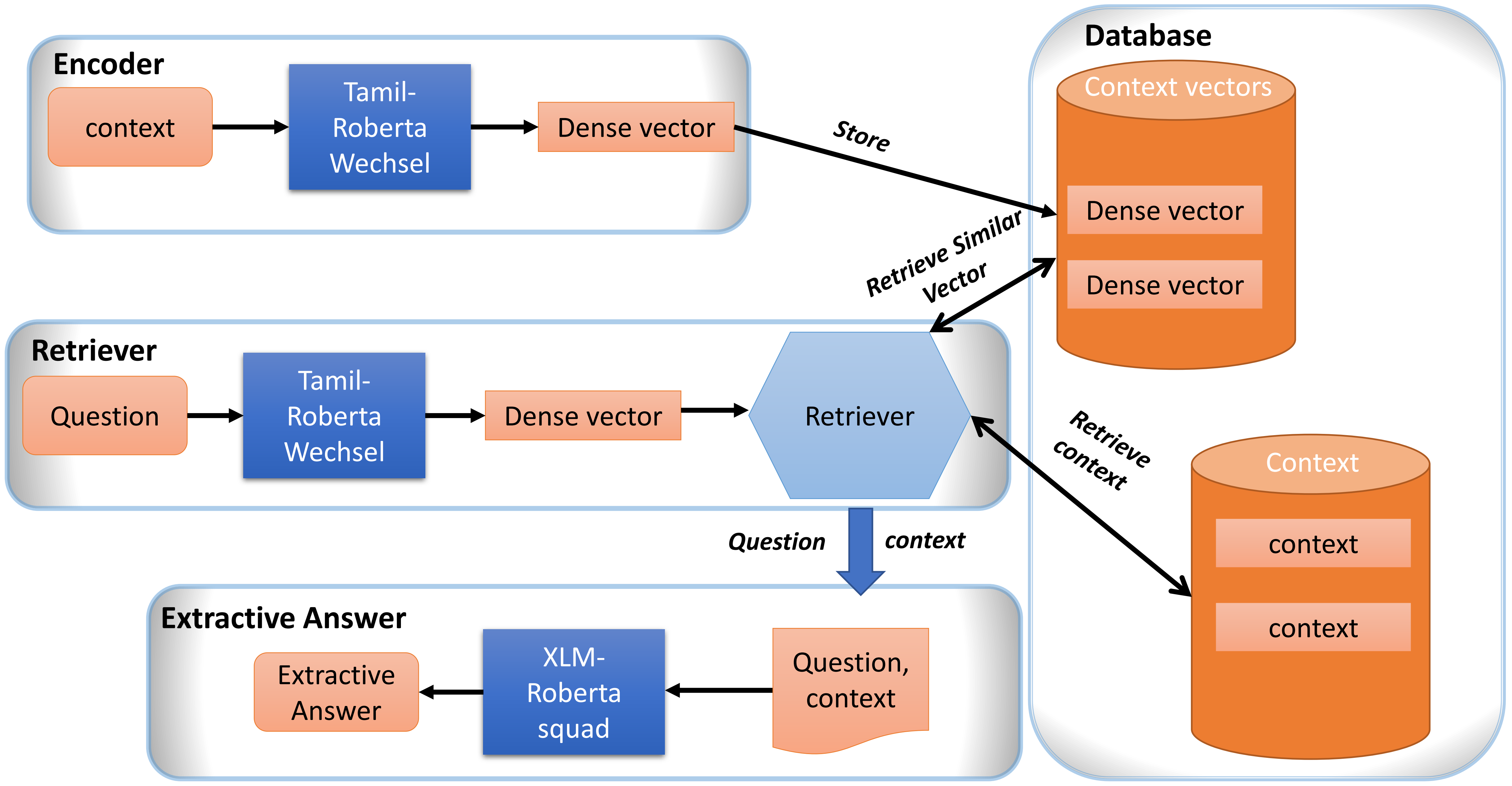
```
!pip install pinecone-client
import pinecone

pinecone.init(api_key=API_KEY, environment='us-west1-gcp')

# check if index already exists, if not we create it
if 'qa-index' not in pinecone.list_indexes():
    pinecone.create_index(name='qa-index', dimension=len(qa[0]['encoding']))

# connect to index
index = pinecone.Index('qa-index')
```

```
from tqdm.auto import tqdm  # progress bar

upserts = [(v['id'], v['encoding']) for v in qa]
# now upsert in chunks
for i in tqdm(range(0, len(upserts), 50)):
    i_end = i + 50
    if i_end > len(upserts): i_end = len(upserts)
    index.upsert(vectors=upserts[i:i_end])
```

part 3

Retriver

Thank You


Encoder
context
Tamil-Roberta Wechsel
Dense vector
Store
Database
Context vectors
Dense vector
Dense vector
Retrieve Similar Vector
Retriever
Question
Tamil-Roberta Wechsel
Dense vector
Retriever
Retrieve context
Context
context
context
Question
context
Extractive Answer
Extractive Answer
XLM-Roberta squad
Question, context