MOHAMMED V UNIVERSITY of Rabat

Computer Science department

Master in Data Engineering and Software Development

Entitled:
Question Answering

Prepared by:
NAIT Hamza
Supervisor:
Pr. Abderrahmane EZ-ZAHOUT

Academic year 2020-2021

Contents:

1. Introduction

Question Answering (QA) is a research area that combines research from different, but related, fields which are Information Retrieval (IR), Information Extraction (IE) and
Natural Language Processing (NLP).
Actually, what a current information retrieval system or search engine can do is just “document retrieval”, i.e. given some keywords it only returns the relevant ranked documents that contain these keywords.
Information retrieval systems do not return answers, and accordingly users are left to extract answers from the documents themselves. However, what a user really wants is often a precise answer to a question.

However, the main type of questions submitted by users in natural language are the factoid questions, such as “When did the Egyptian revolution take place?” But, the recent research trend is shifting toward more complex types of questions such as definitional questions (e.g. “Who is the President of Egypt?” or “What is SCAF?”), list questions (e.g. “List the countries that won the Cup of African Nations”), and why-type questions (e.g. “Why was Sadat assassinated?”).

2. algorithms and softwares

2.1 Simple transformers

Simple Transformer models are built with a particular Natural Language Processing (NLP) task in mind. Each such model comes equipped with features and functionality designed to best fit the task that they are intended to perform. The high-level process of using Simple Transformers models follows the same pattern.

1. Initialize a task-specific model
2. Train the model with train_model()
3. Evaluate the model with eval_model()
4. Make predictions on (unlabelled) data with predict()

However, there are necessary differences between the different models to ensure that they are well suited for their intended task. The key differences will typically be the differences in input/output data formats and any task specific features/configuration options. These can all be found in the documentation section for each task.

2.2 torshvision

This library is part of the PyTorch project. PyTorch is an open-source machine learning framework,
The torchvision package consists of popular datasets, model architectures, and common image transformations for computer vision.

2.3 WandB

Weights & Biases is the machine learning platform for developers to build better models faster. Use W&B's lightweight, interoperable tools to quickly track experiments, version and iterate on datasets, evaluate model performance, reproduce models, visualize results and spot regressions, and share findings with colleagues.

W&B can be set up in 5 minutes, then quickly iterate on machine learning pipeline with the confidence that datasets and models are tracked and versioned in a reliable system of record.

2.4 BERT

We introduce a new language representation model called BERT, which stands for Bidirectional Encoder Representations from Transformers. Unlike recent language representation models, BERT is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications.
BERT is conceptually simple and empirically powerful. It obtains new state-of-the-art results on eleven natural language processing tasks, including pushing the GLUE score to 80.5% (7.7% point absolute improvement), MultiNLI accuracy to 86.7% (4.6% absolute improvement), SQuAD v1.1 question answering Test F1 to 93.2 (1.5 point absolute improvement) and SQuAD v2.0 Test F1 to 83.1 (5.1 point absolute improvement).

2.5 Google Collab

Colaboratory, or “Colab” for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education. More technically, Colab is a hosted Jupyter notebook service that requires no setup to use, while providing free access to computing resources including GPUs.

2.6 JSON

JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language Standard ECMA-262 3rd Edition - December 1999. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.

3.Realisation

3.1 Data-set

For question answering tasks, the input data can be in JSON files or in a Python list of dictionaries in the correct format. The structure of both formats is identical, i.e. the input may be a string pointing to a JSON file containing a list of dictionaries, or it the input may be a list of dictionaries itself.
ach such dictionary contains two attributes, the "context" and "qas".

- context: The paragraph or text from which the question is asked.
- qas: A list of questions and answers (format below).

Questions and answers are represented as dictionaries. Each dictionary in qas has the following format.

- id: *(string)* A unique ID for the question. Should be unique across the entire dataset.
- question: *(string)* A question.
- is_impossible: *(bool)* Indicates whether the question can be answered correctly from the context.
- answers: *(list)* The list of correct answers to the question.

A single answer is represented by a dictionary with the following attributes.

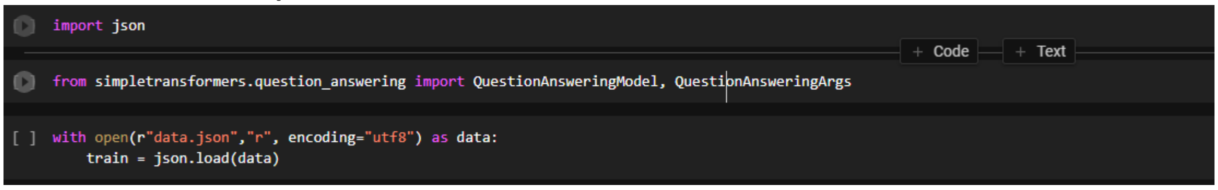
- text: *(string)* The answer to the question. Must be a substring of the context.
- answer_start: *(int)* Starting index of the answer in the context.

Train Data Format

This is a part of the data set I made my self to train the model, using some easy questions and answers in French.

```
[
  {
      "qas": [
        {
          "question": "quel est son nom?",
          "id": 0,
          "answers": [
            {
              "text": "Hamza",
              "answer_start": 11
            }
          ],
          "is_impossible": false
        },
        {
          "question": "quel age a-t-il?",
          "id": 1,
          "answers": [
            {
              "text": "21 ans",
              "answer_start": 24
            }
          ],
          "is_impossible": false
        }
      ],
      "context": "je m'appel Hamza, j'est 21 ans",
      "document_id": 0
  }
]
```

3.2 importations



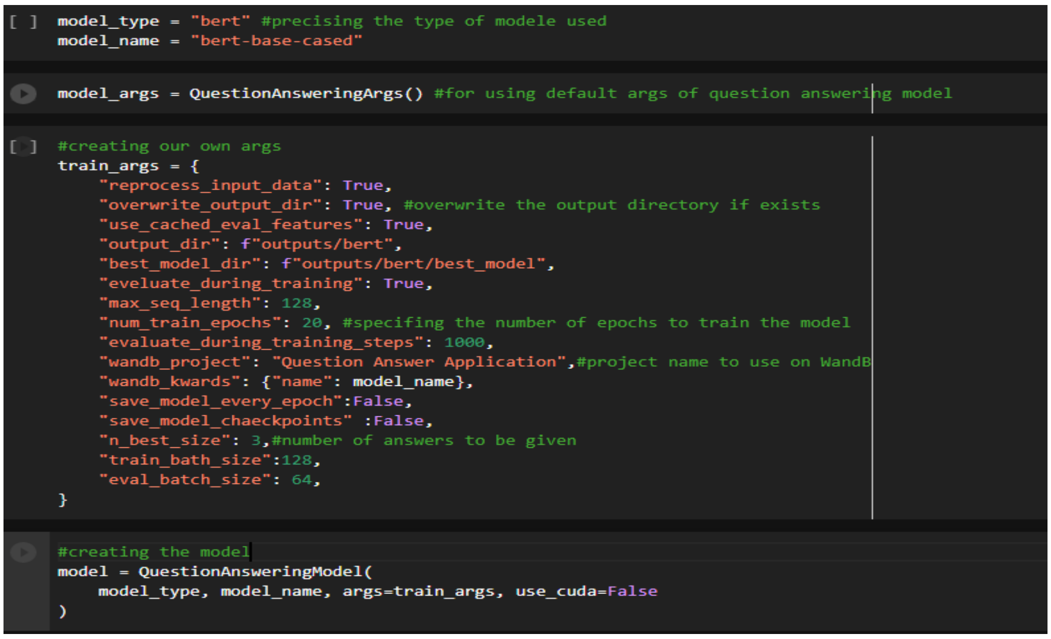
```
import json
```

```
from simpletransformers.question_answering import QuestionAnsweringModel, QuestionAnsweringArgs
```

```
[ ] with open(r"data.json","r", encoding="utf8") as data:
        train = json.load(data)
```

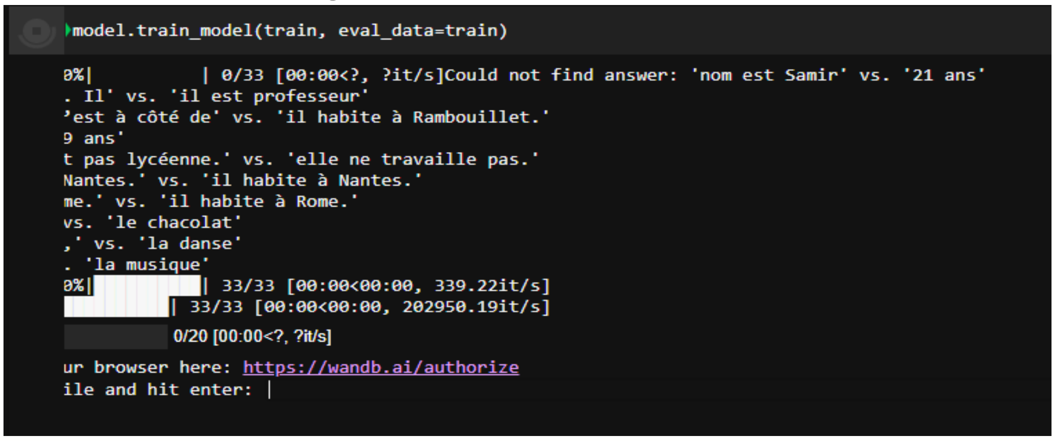
Import json to use it to read the data-set, and QuestionAnsweringModel to create and train the model

3.3 model creation



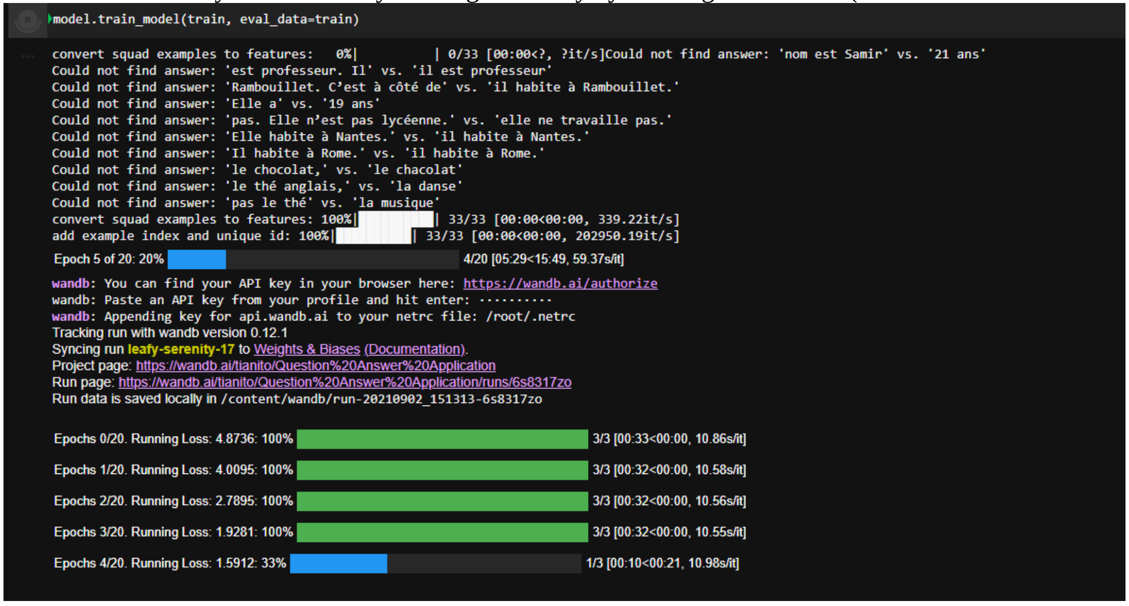
```
[ ] model_type = "bert" #precising the type of modele used
    model_name = "bert-base-cased"

    model_args = QuestionAnsweringArgs() #for using default args of question answering model

[ ] #creating our own args
    train_args = {
        "reprocess_input_data": True,
        "overwrite_output_dir": True, #overwrite the output directory if exists
        "use_cached_eval_features": True,
        "output_dir": f"outputs/bert",
        "best_model_dir": f"outputs/bert/best_model",
        "eveluate_during_training": True,
        "max_seq_length": 128,
        "num_train_epochs": 20, #specifing the number of epochs to train the model
        "evaluate_during_training_steps": 1000,
        "wandb_project": "Question Answer Application",#project name to use on WandB
        "wandb_kwards": {"name": model_name},
        "save_model_every_epoch":False,
        "save_model_chaeckpoints" :False,
        "n_best_size": 3,#number of answers to be given
        "train_bath_size":128,
        "eval_batch_size": 64,
    }

    #creating the model
    model = QuestionAnsweringModel(
        model_type, model_name, args=train_args, use_cuda=False
    )
```

3.4 model training

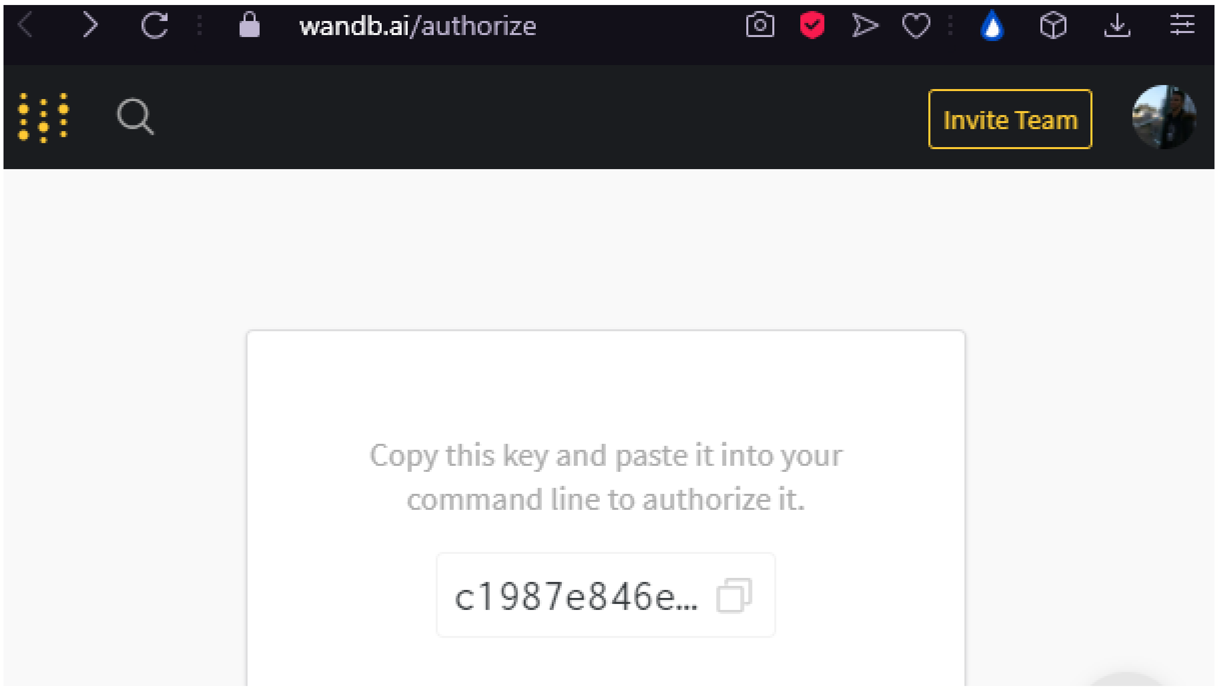


```
model.train_model(train, eval_data=train)

0%|          | 0/33 [00:00<?, ?it/s]Could not find answer: 'nom est Samir' vs. '21 ans'
. Il' vs. 'il est professeur'
'est à côté de' vs. 'il habite à Rambouillet.'
9 ans'
t pas lycéenne.' vs. 'elle ne travaille pas.'
Nantes.' vs. 'il habite à Nantes.'
me.' vs. 'il habite à Rome.'
vs. 'le chacolat'
,' vs. 'la danse'
. 'la musique'
0%|          | 33/33 [00:00<00:00, 339.22it/s]
          | 33/33 [00:00<00:00, 202950.19it/s]
          0/20 [00:00<?, ?it/s]
ur browser here: https://wandb.ai/authorize
ile and hit enter: |
```

To use WandB a key is demanded you can get this key by clicking on the link. (Create an account if needed)



```
model.train_model(train, eval_data=train)

convert squad examples to features:   0%|          | 0/33 [00:00<?, ?it/s]Could not find answer: 'nom est Samir' vs. '21 ans'
Could not find answer: 'est professeur. Il' vs. 'il est professeur'
Could not find answer: 'Rambouillet. C'est à côté de' vs. 'il habite à Rambouillet.'
Could not find answer: 'Elle a' vs. '19 ans'
Could not find answer: 'pas. Elle n'est pas lycéenne.' vs. 'elle ne travaille pas.'
Could not find answer: 'Elle habite à Nantes.' vs. 'il habite à Nantes.'
Could not find answer: 'Il habite à Rome.' vs. 'il habite à Rome.'
Could not find answer: 'le chocolat,' vs. 'le chacolat'
Could not find answer: 'le thé anglais,' vs. 'la danse'
Could not find answer: 'pas le thé' vs. 'la musique'
convert squad examples to features: 100%|██████████| 33/33 [00:00<00:00, 339.22it/s]
add example index and unique id: 100%|██████████| 33/33 [00:00<00:00, 202950.19it/s]
Epoch 5 of 20: 20% ██                  4/20 [05:29<15:49, 59.37s/it]
wandb: You can find your API key in your browser here: https://wandb.ai/authorize
wandb: Paste an API key from your profile and hit enter: ··········
wandb: Appending key for api.wandb.ai to your netrc file: /root/.netrc
Tracking run with wandb version 0.12.1
Syncing run leafy-serenity-17 to Weights & Biases (Documentation).
Project page: https://wandb.ai/tianito/Question%20Answer%20Application
Run page: https://wandb.ai/tianito/Question%20Answer%20Application/runs/6s8317zo
Run data is saved locally in /content/wandb/run-20210902_151313-6s8317zo

Epochs 0/20. Running Loss: 4.8736: 100% ████████████████████ 3/3 [00:33<00:00, 10.86s/it]

Epochs 1/20. Running Loss: 4.0095: 100% ████████████████████ 3/3 [00:32<00:00, 10.58s/it]

Epochs 2/20. Running Loss: 2.7895: 100% ████████████████████ 3/3 [00:32<00:00, 10.56s/it]

Epochs 3/20. Running Loss: 1.9281: 100% ████████████████████ 3/3 [00:32<00:00, 10.55s/it]

Epochs 4/20. Running Loss: 1.5912: 33% ███████              1/3 [00:10<00:21, 10.98s/it]
```

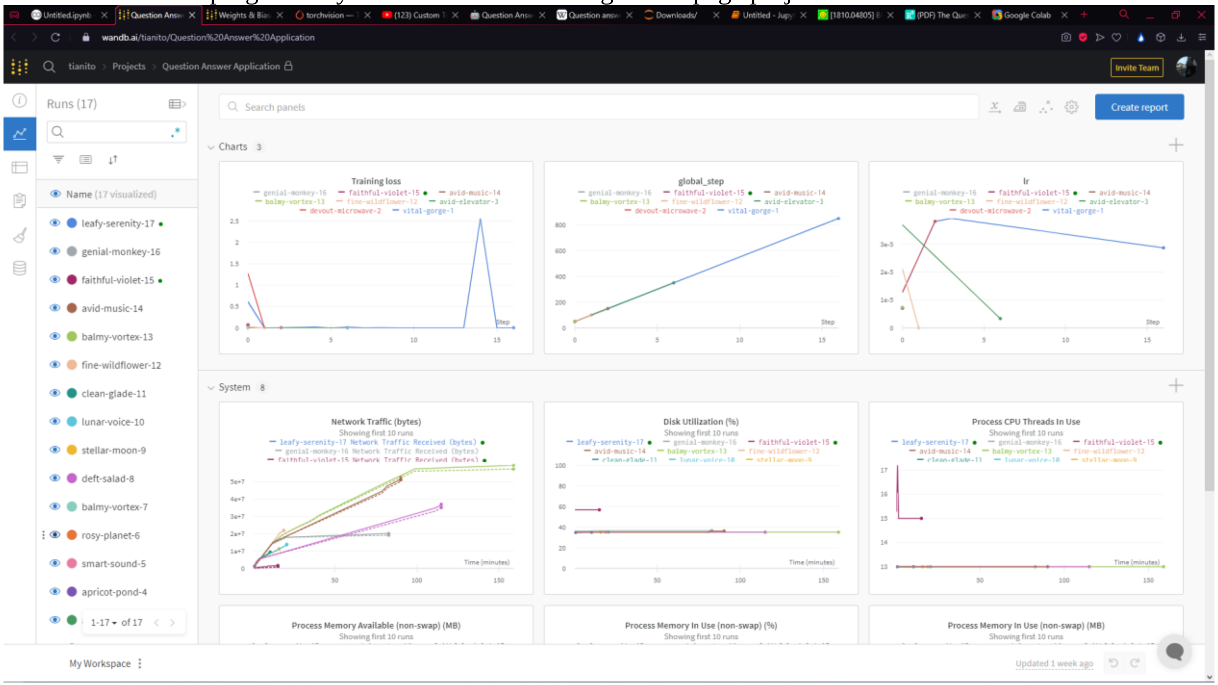

wandb.ai/authorize

Invite Team

Copy this key and paste it into your command line to authorize it.

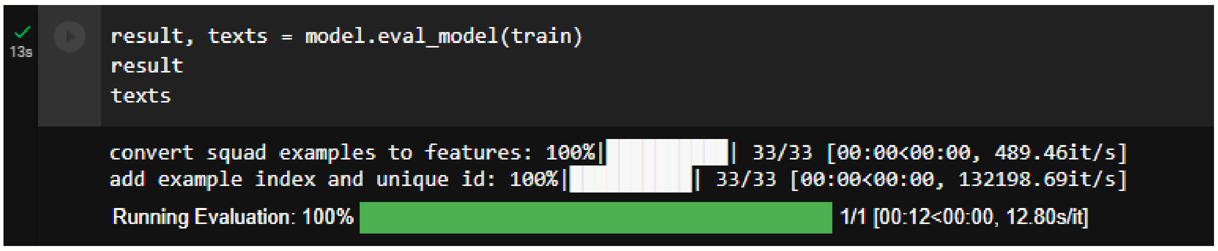
c1987e846e...

You can watch the progress of your model while training on the page project in WandB


wandb.ai/tianito/Question%20Answer%20Application
tianito > Projects > Question Answer Application
Runs (17)
Charts 3
Training loss
global_step
lr
System 8
Network Traffic (bytes)
Disk Utilization (%)
Process CPU Threads In Use
Create report

The model will train for 20 epochs (can be changed in model args) which seemed enough for the small data-set I created

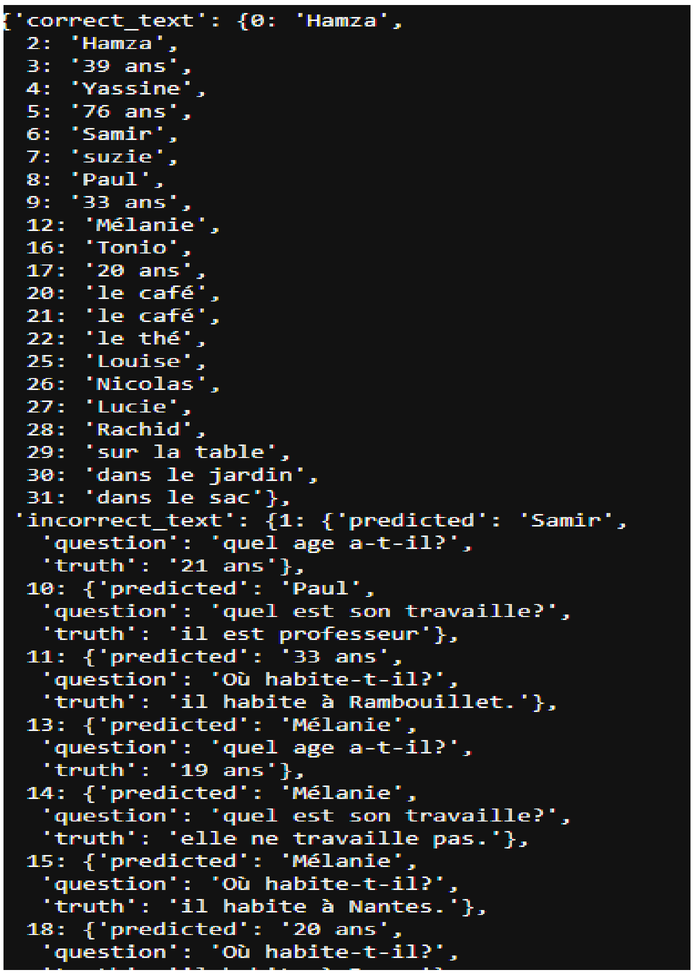
3.5 evaluate model



```
result, texts = model.eval_model(train)
result
texts
```

```
convert squad examples to features: 100%|██████████| 33/33 [00:00<00:00, 489.46it/s]
add example index and unique id: 100%|██████████| 33/33 [00:00<00:00, 132198.69it/s]
Running Evaluation: 100% ████████████████████ 1/1 [00:12<00:00, 12.80s/it]
```

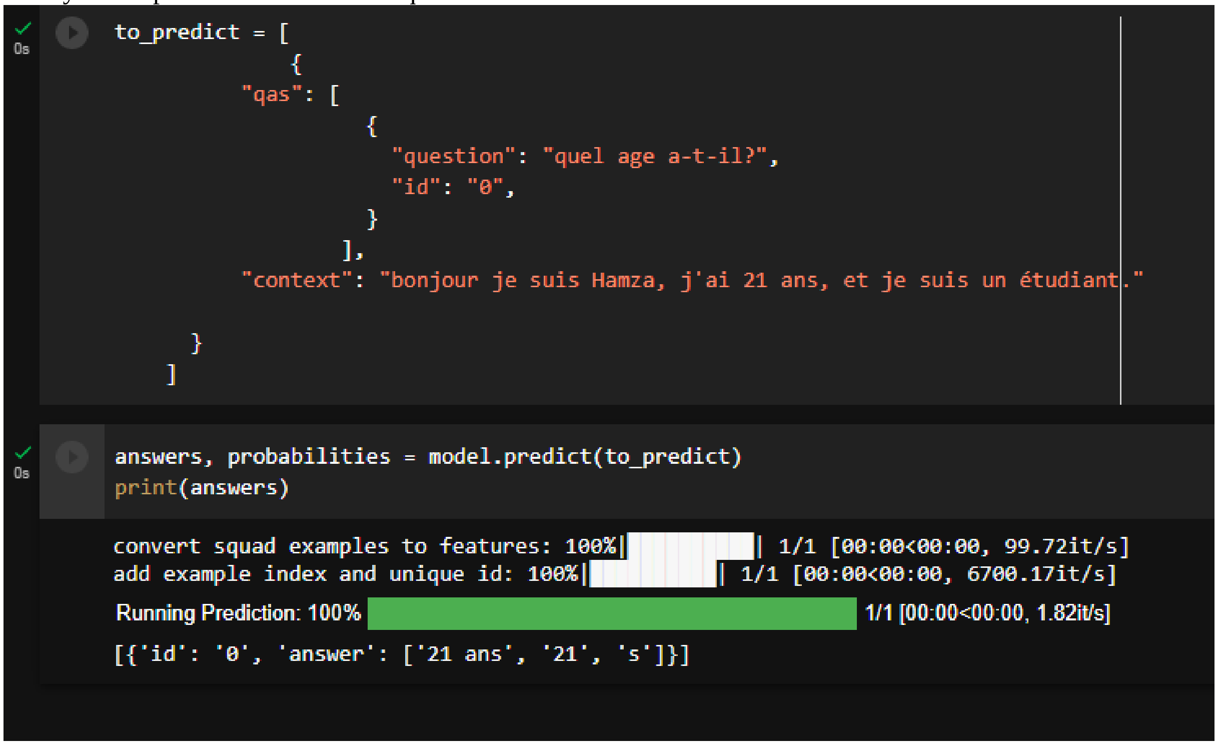
Show the results to see the answers your model was able to predict correctly and the answers he could not.



```
{'correct_text': {0: 'Hamza',
  2: 'Hamza',
  3: '39 ans',
  4: 'Yassine',
  5: '76 ans',
  6: 'Samir',
  7: 'suzie',
  8: 'Paul',
  9: '33 ans',
  12: 'Mélanie',
  16: 'Tonio',
  17: '20 ans',
  20: 'le café',
  21: 'le café',
  22: 'le thé',
  25: 'Louise',
  26: 'Nicolas',
  27: 'Lucie',
  28: 'Rachid',
  29: 'sur la table',
  30: 'dans le jardin',
  31: 'dans le sac'},
 'incorrect_text': {1: {'predicted': 'Samir',
   'question': 'quel age a-t-il?',
   'truth': '21 ans'},
  10: {'predicted': 'Paul',
   'question': 'quel est son travaille?',
   'truth': 'il est professeur'},
  11: {'predicted': '33 ans',
   'question': 'Où habite-t-il?',
   'truth': 'il habite à Rambouillet.'},
  13: {'predicted': 'Mélanie',
   'question': 'quel age a-t-il?',
   'truth': '19 ans'},
  14: {'predicted': 'Mélanie',
   'question': 'quel est son travaille?',
   'truth': 'elle ne travaille pas.'},
  15: {'predicted': 'Mélanie',
   'question': 'Où habite-t-il?',
   'truth': 'il habite à Nantes.'},
  18: {'predicted': '20 ans',
   'question': 'Où habite-t-il?',
```

3.6 predict answer

Now you can predict the answer of a question not far from the contexts that exist on the data-set:



```
to_predict = [
              {
          "qas": [
                    {
                      "question": "quel age a-t-il?",
                      "id": "0",
                    }
                  ],
          "context": "bonjour je suis Hamza, j'ai 21 ans, et je suis un étudiant."

        }
      ]
```

```
answers, probabilities = model.predict(to_predict)
print(answers)
```

```
convert squad examples to features: 100%|██████████| 1/1 [00:00<00:00, 99.72it/s]
add example index and unique id: 100%|██████████| 1/1 [00:00<00:00, 6700.17it/s]
Running Prediction: 100% 1/1 [00:00<00:00, 1.82it/s]
[{'id': '0', 'answer': ['21 ans', '21', 's']}]
```

4 Conclusion

As much as it is trained our model can answer more and more questions, which means that in order to create an Artificial Intelligence capable of answering most questions we will need huge amounts of data, which explains the race that huge companies such as Google, Amazon and Facebook to collecting data, so that they can analyze the market and the bombard it with adds, in this field who has more data wins.