This document describes how to implement a demo to question answering task on SQuAD, Google BERT in NLP using DeepPavlov, an open-source conversational AI library built on TensorFlow and Keras.

# Setup

Create and activate a virtual environment on Mac OS:

python - m venv env

source ./env/bin/activate

Install the package inside the environment:

pip install deeppavlov

Install all required packages:

python - m deeppavlov install squad\_bert

This will download trained models:

~/.deeppavlov/downloads/bert\_models / cased\_L-12\_H-768\_A-12/

~/.deeppavlov/models/squad\_bert/

# How to run server and client demo code

Run Flask server

* open terminal and run this command: ***source ./env/bin/activate***
* then change direcroty to the server folder and run this command: ***python app.py***
* Flask server should now up and running on the localhost, port 5000: **<http://localhost:5000>**

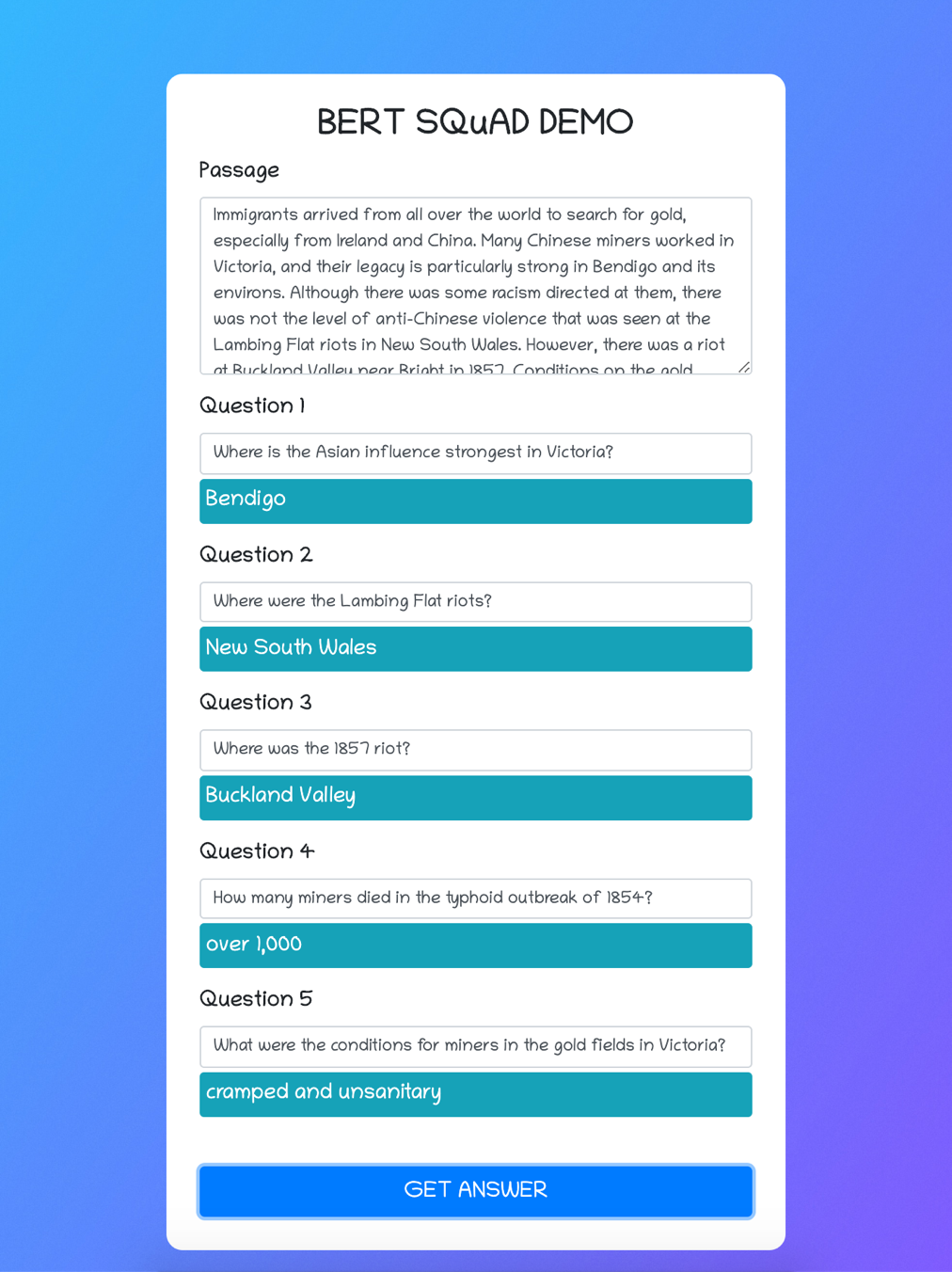
Run Vue client app

* open new terminal, change direcroty to the client folder run this command:   
  ***npm install***
* then run: ***npm run serve***
* the Vue client app should up and running on the localhost:
* **<http://localhost:8080>**

# Testing the app

Refer to this [SQuAD dataset](https://rajpurkar.github.io/SQuAD-explorer/explore/1.1/dev/Victoria_(Australia).html?model=BERT%20(ensemble)%20(Google%20AI%20Language)&version=1.1) to get the passage and the answer.

In this demo, we use Victoria\_(Australia), second passage. In our demo page, the answers were the same with the SQuAD dataset.



# Server code

*from* deeppavlov *import* build\_model, configs

*from* flask *import* Flask, jsonify, request

*from* flask\_cors *import* CORS

*import* sys

DEBUG = True

app = Flask(\_\_name\_\_)

app.config.from\_object(\_\_name\_\_)

CORS(app, resources={r'/\*': {'origin': '\*'}})

model = build\_model(configs.squad.squad\_bert, download=False)

@app.route('/answers', methods=['POST'])

def generate\_answers():

    ANSWERS = []

    post\_data = request.get\_json()

    context = post\_data.get('context')

    questions = post\_data.get('questions')

*for* question in questions:

        print(question['question'], file=sys.stdout)

        result = model([context], [question['question']])

        ANSWERS.append({

            'id': question['id'],

            'question': question['question'],

            'answer': result[0][0]

        })

    print(ANSWERS, file=sys.stdout)

*return* jsonify({

        'result': ANSWERS

    })

*if* \_\_name\_\_ == '\_\_main\_\_':

    app.run()