# BERT

BERT stands for Bidirectional Encoder Representations from Transformers.

Language modeling is to predict a probability of a word in a sentence within a given context. In natural language processing, it is an effective task for using unlabeled data to pretrain neural network. Some standard language models attempt to predict a next token in a sequence from left-to-right or from right-to-left, but not at the same time. In contrast, BERT is designed to train the language model bidirectional representations deeply from unlabeled dataset, from left-to-right and right-to-left simultaneously in all layers.

Transformer architecture is applied for encoding sentences.

BERT is trained on a next sentence prediction task to handle tasks which the relationship between two sentences are significant like question answering

# ELMo

ELMo stands for Embeddings from Language Models.

A word’s meaning is context-dependent, so ELMo uses language models to obtain embeddings for individual words while taking the entire sentence or paragraph into account. A pre-trained, multi-layer, bi-directional, LSTM-based language model is applied. ELMO then extracts the hidden state of each layer for the input sequence of words. Next, they compute a weighted sum of those hidden states to obtain an embedding for each word.

# DeepPavlov

DeepPavlov is an open-source conversational AI library built on TensorFlow and Keras.

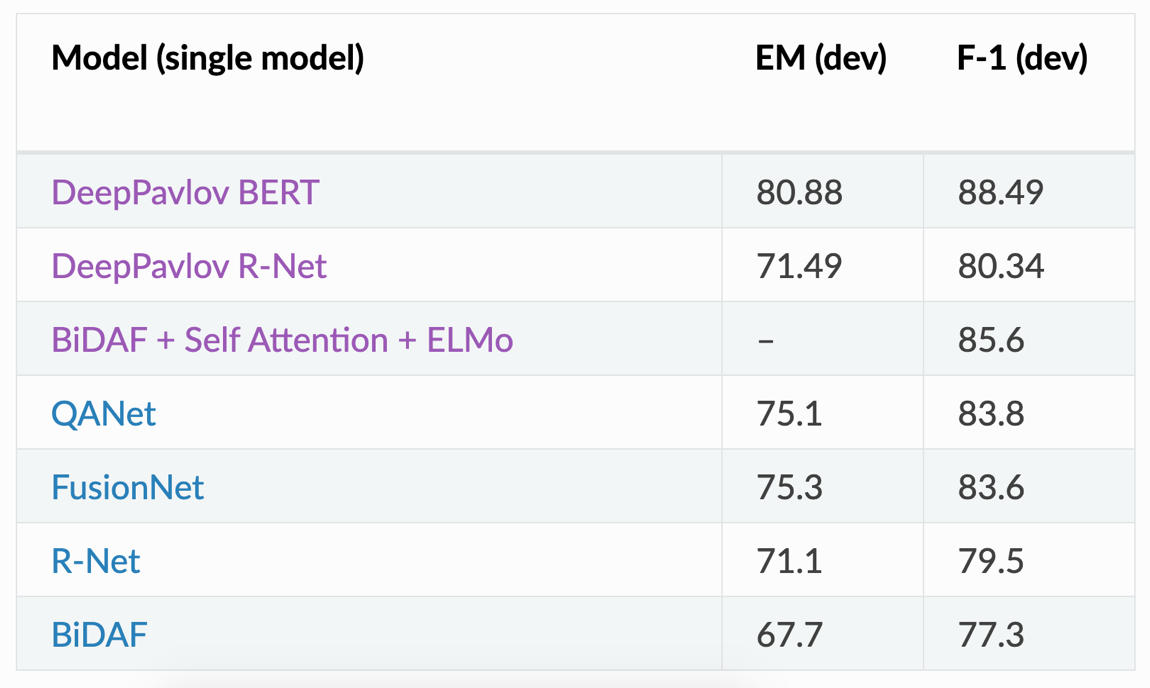
DeepPavlov is designed for:

* development of production ready chat-bots and complex conversational systems
* research in the area of NLP and, particularly, of dialog systems

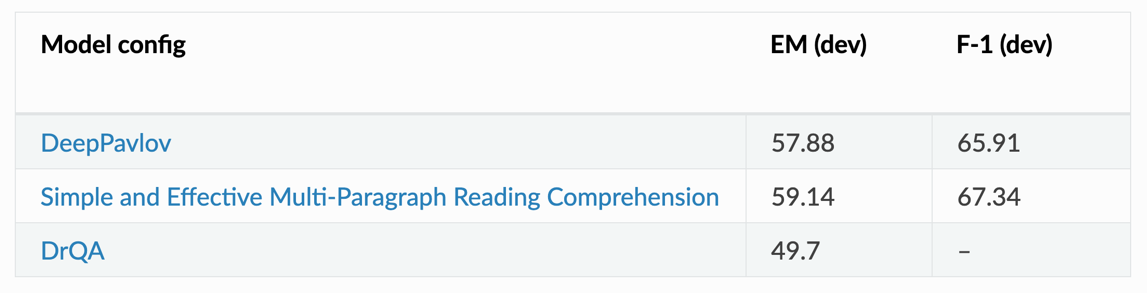
## 1. Using BERT for Context Question Answering Model on SQuAD dataset

This is a task to find an answer on question in a given context, where the answer’s start and end position can be looked for in the given context.

BertSQuADModel uses two linear transformations to predict probability that current sub-token is start/end position of an answer.



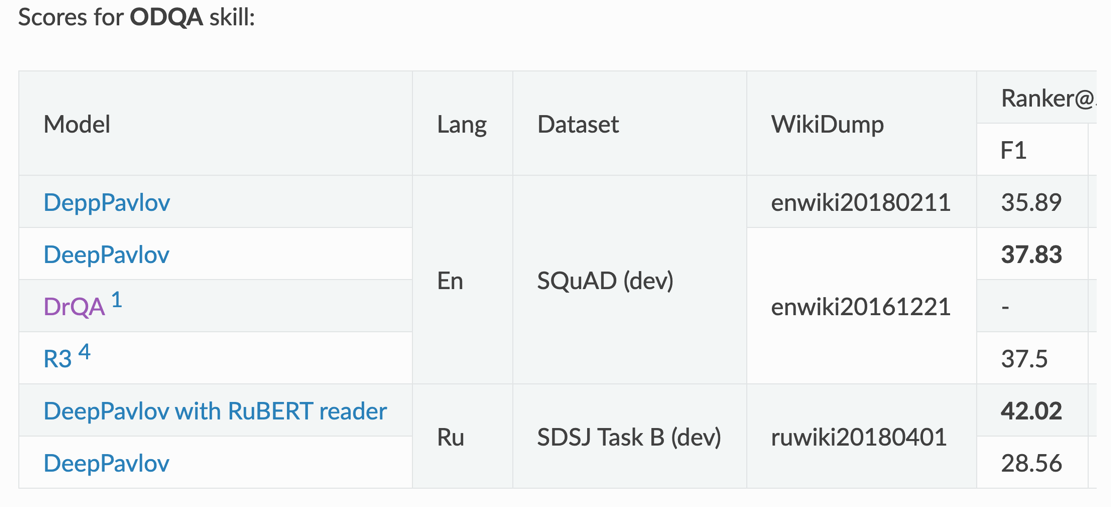
SQuAD with contexts without correct answers



## 2. Open Domain Question Answering (ODQA) on Wikipedia

ODQA is a task to look for an answer to any question in Wikipedia articles by providing only a question, the system outputs the best answer it can find.

ODQA’s combines two models, a ranker and a reader. The ranker is based on DrQA proposed by Facebook Research, and the reader is based on R-NET proposed by Microsoft Research Asia.

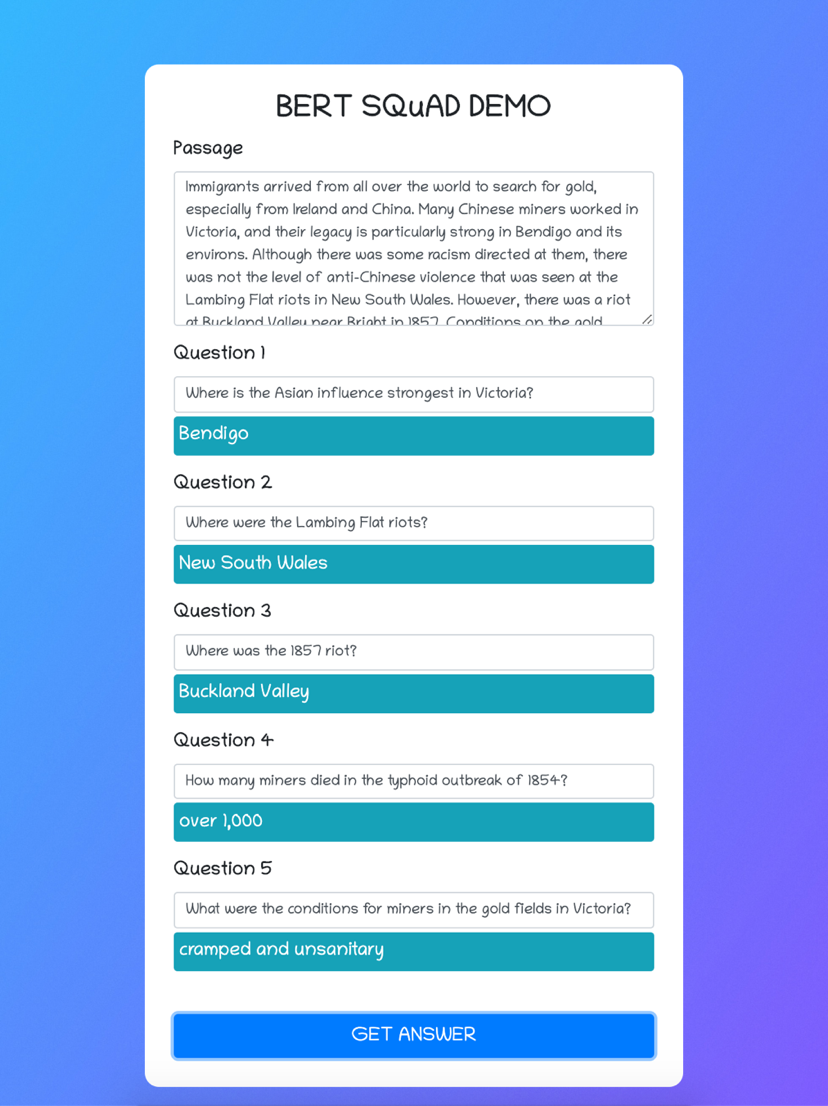


# Demo

User inputs a content of passage and five questions, then click the button. The system will find answer for each question and output the result like the image below. For more detail how to setup and run the demo, please refer to the next section.

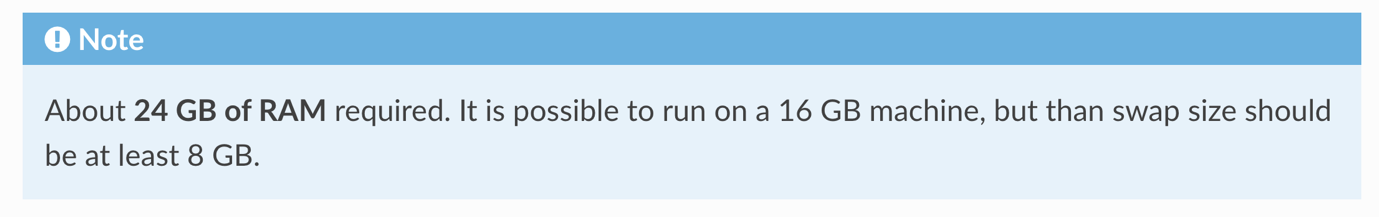
BertSQuADModel is used to find the answer for the question:

* if the model can find the answer, the system sends the output to a client
* if there is no answer, the model outputs the empty string. Then, ODQA tasks is used to search for the answer from Wikipedia (local database) and the system outputs the answer to client



# How to setup and running

This section 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

python - m deeppavlov install en\_odqa\_infer\_wiki

This will download trained models and wikipedia as a database:

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

~/.deeppavlov/models/squad\_bert/

~/.deeppavlov/models/odqa/

~/.deeppavlov/models/multi\_squad\_model\_noans/

~/.deeppavlov/embeddings

~/.deeppavlov/odqa

# 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**](http://localhost:5000)

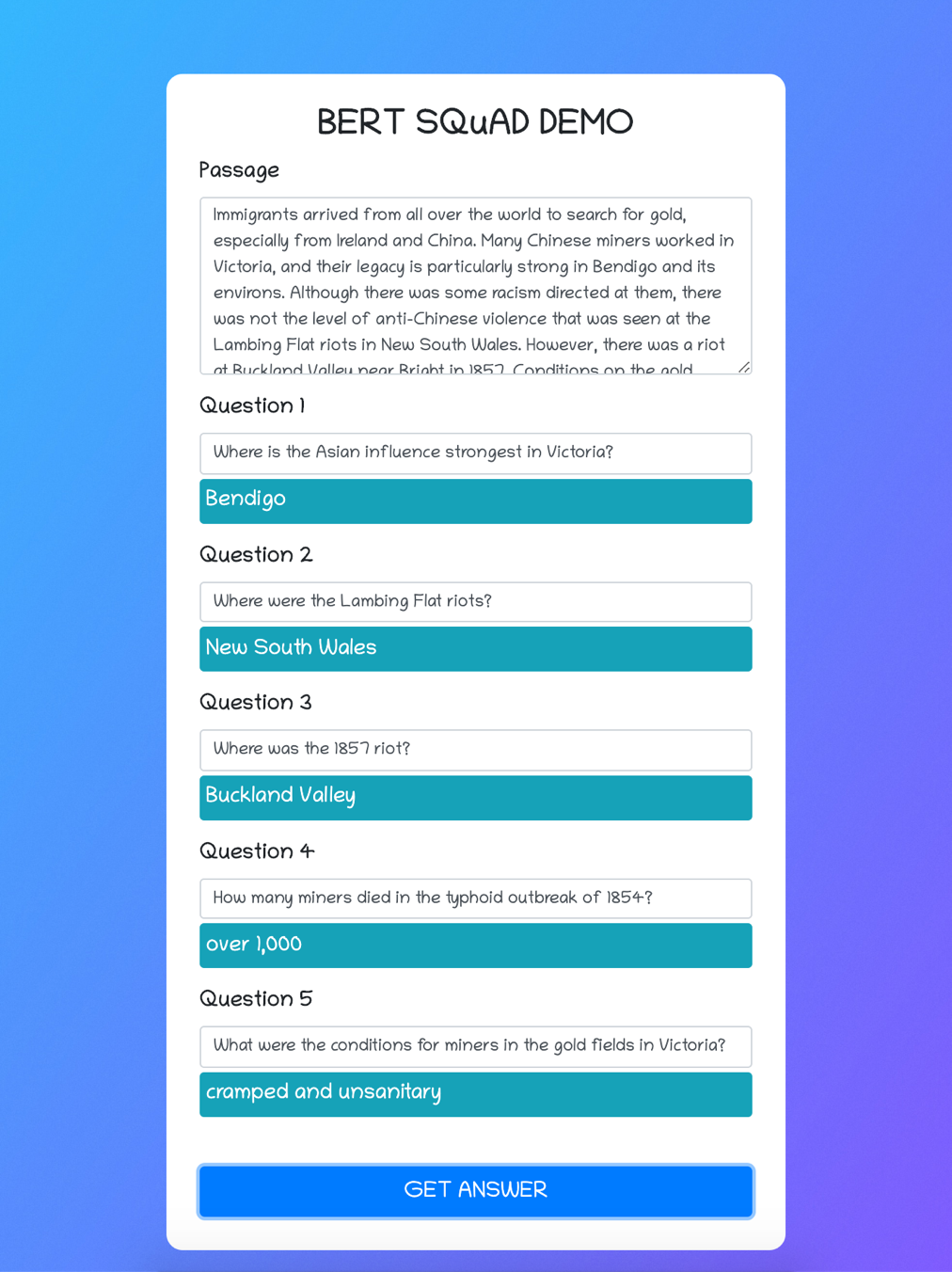
Run Vue client app

* 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**](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.



# References:

<http://docs.deeppavlov.ai/>

<https://github.com/deepmipt/DeepPavlov>