AskBob – User Manual

This user manual is associated with the **AskBob** repository found at the following link: https://github.com/UCL-COMP0016-2020-Team-39/AskBob

AskBob is a customisable framework for developing federated, privacy-safe voice assistants designed to be operated both: within IBM's FISE ecosystem, aiming to combat social isolation, as a server integrated into the projects of teams 25 (concierge) and 38 (video conferencing); and interactively standalone on compatible low-power Windows and Linux desktop devices on which speech and data processing is performed locally to help safeguard users' privacy.

Its modular plugin architecture allows for voice assistant capabilities to be extended via additional third-party skills plugins installable at build-time, e.g. by interfacing with external services. Ask Bob is also accompanied with a React configuration generator web app to aid non-experts in designing new plugins and a skills viewer web app to inspect plugins installed on Ask Bob servers.

AskBob has two primary modes of use:

- interactive mode (where users can interact directly with the voice assistant and hear audible responses)
- server mode (where **AskBob** acts as a server responding to API calls)
    - There is also an additional "voice-enabled" server option, which allows users to upload WAV files for **AskBob** to transcribe and interpret.

When **AskBob** is installed locally on your system, both modes are available with proper configuration; however, only server mode (whether voiceless or voice-enabled) is available through Docker.

Local use (interactive and server modes)

Installation

First, ensure you have Python 3.7 and `pip` installed on your system.

On Ubuntu, `pip` may be installed using the following command:

```
$ curl https://bootstrap.pypa.io/get-pip.py | sudo python
```

On Linux, you must ensure that you have the Python 3.7 dev package installed, e.g. `python3.7-dev` on Ubuntu.

Next, ensure that your versions of `pip`, `setuptools` and `wheel` are up to date.

```
$ python -m pip install -U pip setuptools wheel
```

Dependencies shared across all **AskBob** utilisation modes (interactive mode, voiceless RESTful web API server or voice-enabled RESTful web API server) are installed by the default **AskBob** package wit no extras, as are the components required to run **AskBob** as a voiceless (i.e. with no speech-to-text capabilities) RESTful web API server.

The latest release of **AskBob** may be installed from the Python Package Index (`PyPI`) using the following command:

```
$ python -m pip install askbob
```

Alternatively, although it is recommended that **AskBob** be installed from `PyPI`, the very latest version of **AskBob** can also be installed from a cloned copy of this GitHub repository using the following commands (assuming `git` is installed):

```
$ git clone https://github.com/UCL-COMP0016-2020-Team-39/AskBob
$ cd AskBob
$ python -m pip install -e .
```

The **AskBob** Python package may be installed with multiple 'extras' depending on the use case as will be demonstrated below:

- voice
- interactive
- docs
- test

A compatible `spaCy` model must then be installed (the `en_core_web_md` model is recommended) with the following command:

```
$ python -m spacy download en_core_web_md
```

Ubuntu helper shell script

To aid users in installation, provided in `scripts/install_voiceless_server.sh` is a shell script for Ubuntu that does the following:

- installs `build-essential`
- installs `git`
- installs **AskBob** (just the base dependencies needed to run as a voiceless server)
- installs the `en_core_web_md` **SpaCy** model
- trains **AskBob** based off the plugins within the `plugins` folder and the `config.ini` file

This script must be copied to the root of an **AskBob** *project folder* and executed there with `sudo` privileges, e.g.

```
$ sudo ./install_voiceless_server.sh
```

Note: the script relies that Python 3.7 is accessible in the terminal using the command word `python`. On some systems, this is not the case and it is instead `python3` or potentially even `python3.7`, so you may have to run `alias python=python3` prior to running the script.

Voice-enabled RESTful web API server mode

To run the **AskBob** server in voice-enabled mode, where WAV files of speech can be uploaded to the **AskBob** server on the `/voicequery` endpoint from which **AskBob** will produce a JSON response, additional voice-related dependencies must be installed.

When installing **AskBob** from PyPi, use the following command:

```
$ python -m pip install askbob[voice]
```

When installing **AskBob** from a clone of this GitHub repository, instead use the following command:

```
$ python -m pip install .[voice]
```

Once the extra dependencies are installed, a mozilla\DeepSpeech-compatible model and external scorer must be downloaded into the `data` folder of an **AskBob** *project folder* (with the runtime `config.ini` file updated to reflect the correct filenames) in order for **AskBob** to use those models for speech transcription.

Ubuntu helper shell script

To aid users in installation, provided in `scripts/install_voice_server.sh` is a shell script for Ubuntu that does the following:

- installs `build-essential`
- installs `git`
- installs **AskBob** with the `voice` extra
- installs the `en_core_web_md` **SpaCy** model
- downloads a pre-trained English DeepSpeech model and external scorer
- trains **AskBob** based off the plugins within the `plugins` folder and the `config.ini` file

This script must be copied to the root of an **AskBob** *project folder* and executed there with `sudo` privileges, e.g.

```
$ sudo ./install_voice_server.sh
```

Note: the script relies that Python 3.7 is accessible in the terminal using the command word `python`. On some systems, this is not the case and it is instead `python3` or potentially even `python3.7`, so you may have to run `alias python=python3` prior to running the script. Also ensure that you have the `wget` utility program installed (it is installable with `sudo apt install wget`).

Interactive mode

To run **AskBob** as a voice assistant interactively (i.e. with speech transcription and synthesis enabled), additional interactive mode-related dependencies must be installed.

Firstly, the cross-platform audio API `portaudio` must be installed on your machine.

On Ubuntu, the `portaudio` binary can be installed with the following command:

```
$ sudo apt install portaudio19-dev python3-pyaudio
```

To use **AskBob** in interactive mode on Windows -- as this requires additional compilation -- you must have Build tools for Visual Studio 2019 installed on your system (or any other `pip`-compatible build tool). A direct download link may be found here: https://visualstudio.microsoft.com/thank-you-downloading-visual-studio/?sku=BuildTools&rel=16. You may then have to compile the `portaudio` binary used by **AskBob** from source.

Note: Christoph Gohlke maintains unofficial Windows binaries for Python extension packages, including for PyAudio, which may be installed using `pip install INSERT_BINARY_LOCATION` after installing the relevant build tools and downloading the `pyaudio` wheel for Python 3.7.

If **AskBob** is being installed on macOS and you get an `ImportError` related to `_portaudio`, then you may have to additionally run the following commands (assuming you have `git` installed):

```
$ git clone https://people.csail.mit.edu/hubert/git/pyaudio.git
$ cd pyaudio
$ sudo python setup.py install
```

With `portaudio` properly installed, you will then need to find a mozilla\DeepSpeech-compatible model and scorer to be used with **AskBob** if you have not done so already. Once downloaded, place the files in the `data` folder of an **AskBob** *project folder* and update the runtime configuration file (`config.ini`) with the correct file paths to the model and scorer, respectively.

You may also have to modify the configuration depending on the text-to-speech voices made available by your operating system to pyttsx3. These voices are listed when **AskBob** is started in interactive mode and may also be found using the following sequence of Python commands:

```
>>> import pyttsx3
>>> print(*[voice.id for voice in pyttsx3.init().getProperty('voices')])
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Speech\Voices\Tokens\TTS_MS_EN-US_DAVID_11.0
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Speech\Voices\Tokens\TTS_MS_EN-GB_HAZEL_11.0
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Speech\Voices\Tokens\TTS_MS_EN-US_ZIRA_11.0
```

A shell script containing these Python commands is found at `scripts/get_voices.sh` and is executable from this directory using the following command:

```
$ ./scripts/get_voices.sh
```

Once this is done, the additional **AskBob** `interactive` extra must be installed. When installing **AskBob** from `PyPI`, use the following command:

```
$ python -m pip install askbob[interactive]
```

When installing **AskBob** from a clone of this GitHub repository, instead use the following command:

```
$ python -m pip install .[interactive]
```

If **AskBob** is being run in interactive mode on a Linux system and the text-to-speech output fails to work, you may be missing a few additional system packages. On Ubuntu, these may be installed using the following command:

```
$ sudo apt update && sudo apt install espeak ffmpeg libespeak1
```

Training

The **AskBob** voice assistant must be trained before use using the following command:

```
$ python -m askbob --setup [optional additional configuration JSON file]
```

Running the above command with no additional configuration JSON file will build **AskBob** based off only the installed plugins placed in the `plugins` folder:

```
$ python -m askbob --setup
```

The following command is an example of how **AskBob** could be trained with such an additional build-time configuration JSON file:

```
$ python -m askbob --setup default_config.json
```

Multilingual support

AskBob may be used with any language supported for which `DeepSpeech` (for speech-to-text), `spaCy` (for natural language processing) and `pyttsx3` (for text-to-speech) models exist. To do this, the relevant models must be downloaded and installed, and then the **AskBob** `config.ini` must be changed to reflect the use of the new models.

1. First, download new `DeepSpeech` models into the `data` folder and adjust the `model` and `scorer` parameters to use these new models:

```
[Listener]
model = data/deepspeech-0.9.1-models.pbmm
scorer = data/deepspeech-0.9.1-models.scorer
```

2. Then, download a new `spaCy` model for the language and update the language configuration (e.g. for French `fr`):

```
$ python -m spacy download fr_core_news_md
```

```
[Rasa]
language = fr
spacy_model = fr_core_news_md
```

3. Modify the `voice_id` parameter in the text-to-speech settings to that of an installed TTS voice for that language. Supported voices are listed when **AskBob** is run in interactive mode.

```
[TTS]
voice_id =
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Speech\Voices\Tokens\TTS_MS_EN-GB_HAZEL_11.0
```

Usage

Once installed on your system, **AskBob** should be run from an **AskBob** *project folder* containing at a bare minimum:

- `/plugins` - a folder of installed plugins
- `config.ini` - a runtime configuration file (as described below)

An example of a *skeleton project folder* may be found at our `askbob-plugin-skeleton` repository: https://github.com/UCL-COMP0016-2020-Team-39/askbob-plugin-skeleton

This *skeleton project repository* can serve as a base for third-party developers to work on their own **AskBob** plugins.

All of the following commands must be executed from the base of an **AskBob** *project folder*.

AskBob may be run interactively with the following command:

```
$ python -m askbob
```

You can specify your own runtime config.ini file with the -c flag:

```
$ python -m askbob -c config.ini
```

AskBob can also be run as a standalone server with the following command:

```
$ python -m askbob -s
```

The `/voicequery` endpoint where a single-channel (preferably 16kHz) WAV file (<10MiB in size) may be uploaded is enableable by starting **AskBob** with the `-v` flag in addition to the `-s` flag, i.e.

```
$ python -m askbob -s -v
```

Further help is available using the `--help` flag.

```
$ python -m askbob --help
usage: __main__.py [-h] [-c CONFIG] [-w SAVEPATH] [-f FILE] [-d DEVICE]
                   [-r RATE] [-s] [-v] [--setup [SETUP]]

**AskBob**: a customisable voice assistant.

optional arguments:
  -h, --help            show this help message and exit
  -c CONFIG, --config CONFIG
                        The configuration file.
  -w SAVEPATH, --savepath SAVEPATH
                        Save .wav files of utterences to a given directory.
  -f FILE, --file FILE  Read from a .wav file instead of the microphone.
  -d DEVICE, --device DEVICE
                        The device input index (int) as given by
                        pyaudio.PyAudio.get_device_info_by_index(). Default:
                        pyaudio.PyAudio.get_default_device().
  -r RATE, --rate RATE  The input device sample rate (your device might
                        require 44100Hz). Default: 16000.
  -s, --serve           Run AskBob as a server instead of interactively.
```

```
21    -v, --voice           Enable speech transcription in server mode.
22    --setup [SETUP]       Setup AskBob from the configuration JSON file
23                          provided.
```

Docker (for use as a server only)

AskBob may be built and run as a server headlessly (with support for voice queries either enabled or disabled) using Docker.

The Docker Compose configurations contained within *this repository* are for building, testing, developing and contributing to the **AskBob** codebase contained within this repository -- not creating your own **AskBob** builds.

The plugin skeleton repository contains an example Docker configuration that you can use to run **AskBob** using Docker within an **AskBob** *project folder* containing your own personal project.

Note: it is highly recommended that for your own personal projects, you do install **AskBob** locally or use the example skeleton project Docker setup within an **AskBob** *project folder* (containing the plugins and configuration specific to your project), rather than attempting to clone *this repository* and develop your plugins there (while that may be possible).

Installation & training

If you decide to use the `Dockerfile` examples provided in this repository (or the skeleton project folder repository), note that the **AskBob** is both installed and trained when the containers are built in one step.

AskBob can be built without support for voice queries with `docker-compose` using the following command:

```
1  $ docker-compose build voiceless
```

Similarly, **AskBob** can be built with such support using the following command:

```
1  $ docker-compose build voice
```

An additional build-time configuration (JSON) used in training may be specified using a build argument as in the following command:

```
1  $ docker-compose build --build-arg ASKBOB_SETUP_CONFIG=default_config.json
   voice
```

Usage

The **AskBob** server can then be launched using the following commands:

- voice mode

```
1  $ docker-compose up voice
2
```

- voiceless mode

```
1  $ docker-compose up voiceless
```

Demo AskBob project folder

An **AskBob** project folder for demonstrative purposes is found under `examples/demo`. Once **AskBob** is installed locally on your machine, you can change into that directory and run the **AskBob** demonstration, e.g. as a server with `python -m askbob -s`. If you download DeepSpeech model/scorer pair into the `data` folder (`examples/demo/data`) and update the `config.ini` file if the model/scorer filenames require updating, you may also run `python -m askbob -s -v`. If the relevant dependencies for interactive mode are installed, the interactive demo may also be run with `python -m askbob` from that folder.

A Docker Compose configuration is also provided as a part of the demo project folder to run the **AskBob** as a server.

AskBob can be built without support for voice queries with `docker-compose` using the following command:

```
$ docker-compose build voiceless
```

Similarly, **AskBob** can be built with such support using the following command (provided you add DeepSpeech models into a `data` folder in `examples/demo`):

```
$ docker-compose build voice
```

Developing new AskBob plugins

An **AskBob** plugin consists of a plugin folder containing at a very minimum a `config.json` file containing all of the data needed to train a Rasa model. **AskBob** is accompanied with a configuration generator web app (GitHub), which simplifies the drafting of these `config.json` files to aid non-experts in designing new **AskBob** plugins.

A specification of all the supported JSON options for `config.json` files may be found in the `specification.json` file within this repository.

Plugins may also contain Rasa custom action code, which allows **AskBob** plugins to run arbitrary Python code when certain spoken intents are triggered by the user (in either interactive or server modes).

Files containing such custom action code must be listed within a Python list inside the `__init__.py` file at the root of the plugin folder in the following way (e.g. for a file containing custom action code called `actions.py`):

```
__all__ = ["actions"]
```

Within `actions.py`, custom action code takes the following form:

```
from typing import Any, Text, Dict, List
from rasa_sdk import Action, Tracker
from rasa_sdk.executor import CollectingDispatcher

import askbob.plugin


@askbob.plugin.action("plugin_name", "action_name")
class ActionHelloWorld(Action):
```

```
10
11      def run(self, dispatcher: CollectingDispatcher,
12              tracker: Tracker,
13              domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
14          from datetime import datetime
15
16          dispatcher.utter_message(text="Hello, world")
17
18          return []
19
```

More detail on Rasa custom actions may be found at the Rasa SDK documentation.

Plugin example

An example of a plugin that does not contain any custom action code may be found under `examples/plugins/main`. If you are developing your own custom plugin, you may want to include `examples/plugins/main` within `plugins` in your **AskBob** *project folder*. It provides simple responses to "hello" and "goodbye", as well as an example on how to provide a natural language understanding fallback response triggered when **AskBob** cannot determine a user's intent.

An example of a plugin that does use custom action code is the `time` plugin found at `examples/plugins/time`. It has the following `config.json` file contents:

```
1  {
2      "plugin": "time",
3      "intents": [
4          {
5              "intent_id": "ask_time",
6              "examples": [
7                  "What time is it?",
8                  "What time is it right now?",
9                  "What time is it now?",
10                 "Tell me the time",
11                 "Tell me the time right now",
12                 "Tell me the time now"
13             ]
14         }
15     ],
16     "actions": [
17         "fetch_time"
18     ],
19     "skills": [
20         {
21             "description": "give the system time",
22             "intent": "ask_time",
23             "actions": [
24                 "fetch_time"
25             ]
26         }
27     ]
28 }
29
```

The `__init__.py` file within `examples/plugins/time` contains `__all__ = ["actions"]` and `actions.py` contains the following Python code:

```
from typing import Any, Text, Dict, List
from rasa_sdk import Action, Tracker
from rasa_sdk.executor import CollectingDispatcher

import askbob.plugin


@askbob.plugin.action("time", "fetch_time")
class ActionFetchTime(Action):

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
        from datetime import datetime

        dispatcher.utter_message(text="The time is {0}.".format(
            datetime.now().strftime("%H:%M")))

        return []

```

Further examples of **AskBob** plugins may be found under the `examples/plugins` folder.

Plugin installation

Plugins are installed by copying the plugin folder into the `plugins` folder of an **AskBob** *project folder*. **AskBob** must be retrained when new plugins are installed (`python -m askbob --setup`).

A more visual way to view a summary of all the skills installed within a particular **AskBob** *project folder* than the `GET /skills` raw JSON endpoint is to use the skills viewer web app (GitHub). Run **AskBob** in server mode (`python -m askbob -s`) and provide a URL to the skills endpoint to the web app, e.g. `http://localhost:8000/skills`.

Runtime Configuration Options

Listener

```
[Listener]
model = data/deepspeech-0.9.1-models.pbmm
scorer = data/deepspeech-0.9.1-models.scorer
aggressiveness = 1
```

- `model` is the location to the DeepSpeech model
- `scorer` is the optional location to an external DeepSpeech scorer
- `aggressiveness` is an optional integer (1 if unspecified) between 0 and 3 (inclusive) determining how aggressive the voice activity detector is at filtering out non-speech (0 is the least aggressive, 3 is the most aggressive).

TTS

```
[TTS]
voice_id =
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Speech\Voices\Tokens\TTS_MS_EN-GB_HAZEL_11.0
```

- `voice_id` is the ID of the voice to be used by the pyttsx3 text-to-speech library. If this voice cannot be found, the default voice is used. You may need to install voices in a manner consistent with your operating system before this may work.

Rasa

```
[Rasa]
config = data/rasa/config
model = data/rasa/models
language = en
spacy_model = en_core_web_md
```

- `config` is the location where **AskBob** will generate a set of Rasa YAML config files
- `model` is the location where **AskBob** will place trained Rasa models
- `language` is a 2-letter language code representing the language to be used, e.g. `en` or `fr`.
- `spacy_model` is the name of the `spaCy` model to be used, e.g. `en_core_web_md`

It is highly recommended that you do not change either of these values.

Server

```
[Server]
host = 0.0.0.0
port = 8000
cors_origins =
```

- `host` is the host **AskBob** will bind to when being run in server mode (0.0.0.0 if unspecified)
- `post` is the post **AskBob** will bind to when being run in server mode (8000 if unspecified)
- `cors_origins` is the set of origins used for CORS (cross-origin resource sharing) - leave blank as above to disable CORS, otherwise something such as `cors_origins = *` would be typical for this configuration option.

Plugins

```
[Plugins]
location = plugins
action_server_port = 5055
summary = data/summary.json
```

- `plugins` is the location of the plugins folder (this cannot contain any spaces and must be a valid python module name)
- `action_server_port` is the port used by the internal Rasa custom action server (5055 if unspecified)
- `summary` is the location at which a summary of installed voice assistant skills is stored at build-time

It is highly recommended that you do not change any of these values.

Server endpoints

The following is a description of the endpoints available when running **AskBob** in server mode.

GET /

This is just a landing page that returns the message below:

> "Hi, there! I think you might be in the wrong place... Bob."

POST /query

Request (encoded as `x-www-form-urlencoded`):

- `sender` - a unique identifier for the current user
- `message` - a query for **AskBob** to interpret

Response (JSON):

```
{
    "query": "the original query",
    "messages": [
        // Ask Bob messages
    ]
}
```

For example, with the `sender` as `"askbob"` and the `message` as `"tell me a joke"`, **AskBob** using the example `puns` plugin could produce the following response:

```
{
    "query": "tell me a joke",
    "messages": [
        {
            "text": "One joke, coming right up!"
        },
        {
            "text": "Without geometry life is pointless."
        }
    ]
}
```

In addition to `text`-type messages, **AskBob** plugins implementing `Rasa` custom actions may also pass back arbitrary JSON as `custom`-type messages, e.g. **AskBob** using the FISE video conferencing integration plugin:

```
{
    "query": "call John",
    "messages": [
        {
            "text": "Calling John."
        },
        {
            "custom": {
                "type": "call_user",
                "callee": "John"
            }
        }
    ]
}
```

There are also `image`-type messages available for third-party developers, which take the following form:

```
{
  "image": "http://example.com/image_url.png"
}
```

GET /query

This endpoint is identical to `POST /query` above, other than `sender` and `message` may be passed as query string parameters, e.g. `GET /query?sender=askbob&message=hello`.

GET /skills

This endpoint produces a JSON index that describes of all the plugins installed on an **AskBob** server. It returns data in the following format:

```
{
    "plugins": [
        {
            "plugin": "miscellaneous",
            "description": "",
            "author": "",
            "icon": ""
        },
        {
            "plugin": "puns",
            "description": "",
            "author": "",
            "icon": ""
        }
    ],
    "skills": [
        {
            "plugin": "miscellaneous",
            "category": "miscellaneous",
            "description": "Greet the user",
            "examples": [
                "Hello",
                "Hi",
                "Hey",
                "Howdy",
                "Howdy, partner"
            ]
        },
        {
            "plugin": "miscellaneous",
            "category": "miscellaneous",
            "description": "Say goodbye to the user.",
            "examples": [
                "Goodbye",
                "Good bye",
                "Bye",
                "Bye bye",
                "See you",
                "See you later",
```

```
                "In a bit!",
                "Catch you later!"
            ]
        },
        {
            "plugin": "puns",
            "category": "miscellaneous",
            "description": "tell a joke",
            "examples": [
                "tell me a joke",
                "tell me a joke please",
                "tell me a dad joke",
                "tell me a dad joke please",
                "tell me a pun",
                "tell me a pun please",
                "give me a joke",
                "give me a joke please",
                "give me a dad joke",
                "give me a dad joke please",
                "give me a pun",
                "give me a pun please",
                "could you please tell me a joke?",
                "could you tell me a joke?",
                "I'd love to hear a joke",
                "I'd love to hear a dad joke",
                "I'd love to hear a pun",
                "give me your cheesiest joke",
                "give me a cheesy joke",
                "tell me a cheesy joke",
                "tell me something funny"
            ]
        },
        {
            "plugin": "puns",
            "category": "miscellaneous",
            "description": "assure the user of the quality of dad jokes",
            "examples": [
                "Are you funny?",
                "How funny are you?",
                "How good are your puns?",
                "How good are your dad jokes?",
                "How cheesy are your jokes?"
            ]
        }
    ]
}
```

POST /voicequery

When **AskBob** is being run as a voice-enabled server, an additional `/voicequery` endpoint becomes available.

Request:

- `sender` - a unique identifier for the current user (encoded as `x-www-form-urlencoded`)
- `speech` - a 16000Hz single-channel WAV file under 10MiB containing a single speech utterance for **AskBob** to interpret (encoded as a form file upload)

Response:

- as in `POST /query` and `GET /query`

Documentation

Before attempting to generate the documentation, ensure that the documentation tools are installed.

```
$ python -m pip install askbob[docs]
```

The documentation can be generated using the following commands on Linux:

```
$ cd docs
$ make html
```

On Windows in a bash-style terminal, use the `make.bat` file provided in the following way:

```
> cd docs
> ./make.bat html
```

The generated documentation will be found at `docs/_build/html`.

In order to generate the full documentation, ensure all project dependencies are installed (all extras, i.e. voice, interactive, etc).

Tests

Before running any tests, ensure that **AskBob** was installed with the `test` extra to additionally include the testing suite.

```
$ python -m pip install askbob[test]
```

Tests (with code coverage report generation) may be run with the following command:

```
$ python -m pytest tests/
```

The coverage report may be viewed afterwards by running the following command:

```
$ cd cov_test && python -m http.server 80
```

And then navigating to `http://localhost/` in a browser.

Note: the spaCy `en_core_web_md` model must be installed to run the test suite, as well as all dependencies (including interactive mode dependencies).

The DeepSpeech model `data/deepspeech-0.9.1-models.pbmm` and scorer `data/deepspeech-0.9.1-models.scorer` must exist to run the test suite.