General AI Challenge

How to submit a General AI Challenge solution

Round 1: Gradual Learning

GoodAI, 04.07.2017
www.GoodAI.com
www.general-ai-challenge.org

How to submit a General AI Challenge solution

Round 1: Gradual Learning

Introduction

The evaluation will run on one of two types of Azure Virtual Machines (VMs), either CPU-bound (with no GPU) or GPU-bound.

Linux-based solutions will be run in plain Docker, or nVidia Docker in the case of GPU-bound VMs. The Docker container will have an open port for access to the CommAI environment (and no access to the internet). The solutions must be self-contained without the need for any external data.

On Windows, the situation is different since nVidia Docker is not available there. Windows-based solutions (both CPU- and GPU-bound) will be evaluated on an isolated VM running Windows 10 Enterprise N (Creators Update). The agent will not have access to the Internet and will be monitored to detect any tampering with the evaluation environment.

The following pages will guide you step-by-step through the submission process on Linux and on Windows.

Note that this is a preliminary version of the guide. It may change slightly based on your feedback and our testing. There is a possibility of minor software version changes in case of important bugfix releases.

You can learn more about the evaluation process in the Specifications of the First Round.

Testing on Azure Virtual Machines

If you would like to train or test your agent on Microsoft Azure, you can get free access from our technological partner Microsoft Czech Republic and Slovakia. To get started, please get in touch with Tomas Prokop at b-toprok@microsoft.com. Provide your team name, specify if you'd like to use VMs with GPUs (such as NV6) and attach a copy of your registration confirmation e-mail.

Note that **even those who already requested access to Azure need to additionally request access to the GPU-equipped instances**, if GPU use is intended.

Enabling access to GPU-equipped VMs can take several workdays. We recommend to ask for the access soon after you realize you or your team may need it.

How-to for Linux-based solutions

The evaluation VMs will have the following software installed:

- Ubuntu 16.04.2 LTS, 64-bit
- Linux kernel 4.4 series (currently 4.4.0-78-generic)
- Docker 17.03+ (currently 17.03.1-ce, API version: 1.27 (min. version 1.12))

There is a possibility of bugfix updates, but there will be no older versions. Your solution will need to be compatible with the Linux kernel version and the Docker version listed above. Optionally, you may also match the Linux distribution (Ubuntu 16.04). For your information, we will run the evaluation environment on Python 3.5 series (currently 3.5.2).

1. Prepare Docker image (mandatory)

This section describes how to to create a Docker image containing the agent on your system. If you also want to test-run the image against the CommAI environment, see section *Test-run an agent in Docker on an Azure VM* below.

1. Install Docker Community Edition:
    a. http://docs.master.dockerproject.org/engine/installation/linux/ubuntulinux/
2. If you target the GPU-bound test VM, install also NVidia Docker:
    a. https://github.com/NVIDIA/nvidia-docker/wiki/Installation
3. Create a Dockerfile and prepare your agent for integration into a Docker image. The image must contain the agent and everything it needs to run, such as libraries, pretrained data, and any other dependencies.You can use the template that we provide below, which contains a Dockerfile and other files necessary for creating a Docker image.
    a. Download the Docker template (in this document we use `challenge-local` and `challenge-remote` directories within the home directory to make clear on which computer you currently are):
        - `mkdir ~/challenge-local`
        - `cd ~/challenge-local`
        - `git clone https://github.com/general-ai-challenge/Round1.git`
        - (or `git clone git@github.com:general-ai-challenge/Round1.git` for ssh access)
        - `cd Round1/evaluation`
    b. The template contains an example Python agent (and an example C++ agent) that you should replace with your own implementation, in directory `~/challenge-local/Round1/evaluation/examples/linux-python/agent`
    c. You can modify the example build script `examples/linux-python/build.sh` to create an image with your agent when submitting your solution.
    d. Additional info:
        - The environment will listen on the IP address `172.18.0.1`, TCP port `5556`. This is where the agent should connect to. Remember to set this address in your solution before wrapping it into the docker image.
4. Build your docker image:
    a. If you are using the template, run:
        - `cd ~/challenge-local/Round1/evaluation/examples/linux-python`
        - `sudo ./build.sh`
    b. The image name must be **`learner.tar.gz`** (the example build script respects this). This is the file that you will submit.

5. The method how to submit the file for the final evaluation is specified in the section *How-to upload the solution* below.

2. Test-run an agent in Docker on an Azure VM (optional)

This section describes how to run the containered agent on an Azure VM (or in a similar environment). It is not necessary for the submission. However, you may want to run this test, because we will use the same procedure when evaluating your submission in the quantitative evaluation.

- Create an Azure VM from your Azure portal. The following options are recommended:
    - Select **Ubuntu Server 16.04 LTS** (click the *Create* button)
    - Enter a name, VM disk type **HDD**, some login details, a new resource group and location **East US** (this may change, we'll let you know in such case), (hit *OK* to continue)
    - Choose machine size: **F16** for CPU-bound solutions or **NV6** for GPU-bound ones (you must choose to see all options, these are not among the recommended ones), create the machine. A smaller machine, such as F4, would also suffice for testing in most cases. (Click *Select* to continue)
    - You can keep the default values for the settings on the next page. Click *OK* to continue, and after a successful validation once again.
- Connect to your Azure VM remotely using ssh
- Install Docker and optionally also NVidia Docker (use the instructions from the section *Prepare Docker image*)
- Set up docker network
    - `sudo docker network create -d bridge --subnet 172.18.0.0/16 --gateway 172.18.0.1 challengenetwork`
- Install and run CommAI-env (pretend it's the evaluation version)
    - Checkout the Round1 repository:
        - `mkdir ~/challenge-remote`
        - `cd ~/challenge-remote`
        - `git clone https://github.com/general-ai-challenge/Round1`
        - (or `git clone git@github.com:general-ai-challenge/Round1.git` for ssh access)
        - `cd Round1`
    - Set up the environment using instructions in the repository's Readme file.
        - Tip: To save time, You can use the script `install-py3.sh` which installs python3-pip and Python 3 requirements of CommAI-env:
        - `./install-py3.sh`
    - Run the evaluation environment:
        - `./run-remote.sh`
- Upload the docker image containing the agent (**`learner.tar.gz`**).
    - For testing purposes, you can use scp (or WinSCP). For the final submission, the method will be different.
    - Put the image into the `~/challenge-remote/Round1/evaluation` directory.
- Open a second ssh connection to the Azure VM and run the docker image.
    - Use `run.sh` inside the `evaluation` directory:
        - `cd ~/challenge-remote/Round1/evaluation`
        - `sudo run.sh #(or sudo run_gpu.sh)`
        - Done. You should see your solution communicating with the evaluation environment!

How-to for Windows-based solutions

The evaluation VMs will have the following software installed:

- Windows 10 Enterprise N (Creators Update), 64-bit
- Python 3.5 series (currently 3.5.3)
- Java 8 (currently update 131)
- CUDA Toolkit 8 (currently 8.0.61), *only on the VM with GPU*

There is a possibility of a bugfix update, but there will be no older versions.

We encourage you to make the solution as easy to install as possible. The prefered way is to provide an installation-free package (a zipped directory) that can be just unpacked and then run using a single command.

If installation of some prerequisites is necessary, please provide clear instructions how to do it. The required format is described below.

1. Prepare solution package (mandatory)

This section describes how to to create a package containing the agent (on your system). If you want to also test-run the image against the CommAI environment, see section *Test-run an agent on an Azure VM* below.

The format of the solution that you will submit is a zip file (compatible with the Windows 10 built-in unzip) named `Learner.zip` with at least the following contents:

- `Learner`
  - `Learner.cmd`

In words, the zip archive must contain a single top level folder called `Learner` and inside it, there must be a Windows shell script `Learner.cmd` that can run any other program contained in the package. There can also be any number of other files and folders that the solution needs.

In the case the solution requires any other software installed in addition to that listed above, the package may also contain a `Setup` folder inside the `Learner` folder with installer programs renamed to `Setup1.exe` up to `Setup9.exe` (so that the order of running them is clear). The fewer programs the better. The installers may also be `msi` files or `cmd` scripts.

Please provide clear instructions if any installer option requires a non-default value. Write it down into a file called Readme.txt inside the Setup folder. Note the name of the installer, name of the section/page, name of the option, the default value and the desired value.

The archive contents with some installer files would look like this:

- `Learner`
  - `Setup`
    - `Readme.txt`
    - `Setup1.exe`
    - `Setup2.exe`
    - `Setup3.msi`
    - `Setup4.cmd`
  - `Learner.cmd`

The steps to create a solution package are the following:

1. Create a directory called `Learner`
2. Copy your agent and any programs, libraries and other data that the agent needs into the `Learner` directory

        a. The environment will listen on the IP address `127.0.0.1` (localhost) TCP port `5556`. This is where the agent should connect to.
3. (optional) If your agent also needs some software to be installed, add a subfolder named `Setup` inside the `Learner` directory
    a. Add one or more setup files into the `Setup` folder in the format described above
    b. Note any options of the installers that we should change inside a `Readme.txt` file
4. Pack the folder using the standard zip algorithm, one way to do it:
    a. Right-click the `Learner` folder and choose *Send to > Compressed (zipped) folder*
5. The resulting archive **`Learner.zip`** is the file that you will submit.
6. The method how to submit the file for the final evaluation is specified in the section *How-to upload the solution* below.

The agent must not access the internet and must not tamper with the evaluation environment. It will be monitored to detect any attempts to do so. We may also study the source code before we accept the results as valid.

2. Test-run an agent on an Azure VM (optional)

This section describes how to install and run the CommAI environment and the agent on an Azure VM (or in a similar environment). It is not necessary for the submission. However, you may want to run this test, because we will use the same procedure when evaluating your submission in the quantitative evaluation.

1. Create an Azure VM from your Azure portal. The following options are recommended:
    a. Select **Windows Client -> Windows 10 Enterprise N** (click the *Create* button)
    b. Enter a name, VM disk type **HDD**, some login details, a new resource group and location **East US** (this may change, we'll let you know in such case), (hit *OK* to continue)
    c. Choose machine size: **F16** for CPU-bound solutions or **NV6** for GPU-bound ones (you must choose to see all options, these are not among the recommended ones), create the machine. A smaller machine, such as F4, would also suffice for testing in most cases. (Click *Select* to continue)
    d. You can keep the default values for the settings on the next page. Click *OK* to continue, and after a successful validation once again.
    e. When the machine is deployed, connect to it using the "Connect" button. An .rdp file will be downloaded – run it to open an RDP session.
2. Install Git (it is required by pip, the Python package manager)
    a. Any recent version is OK, we tested Git for Windows 2.13.0 64-bit
        ○ https://git-scm.com/download/win
        ○ The default options selected by the installer should work fine.
3. Install Python
    a. We tested Python version 3.5.3 for Windows x86-64
    b. You can find the web based installer here:
        ○ https://www.python.org/downloads/release/python-353/
        ○ On the first screen, check "Add Python 3.5 to PATH"
        ○ The installation includes pip
4. Checkout Round1
    a. Run Git Bash (or use Git client of you choice)
        ○ You should be in your home directory (~)
        ○ `git clone https://github.com/general-ai-challenge/Round1`
        ○ `cd Round1`
5. Install CommAI requirements

   a. You can stay in Git Bash or you can use any other terminal such as Windows PowerShell
   b. Run the following command (you can also use backslashes "\" on Windows):
      - `cd Round1`   # (unless you are already there)
      - `pip install -r src/requirements/py3.txt`
6. Run CommAI with remote learner
   a. We are still inside the same `Round1` directory
   b. Run the following command (it's a single line command):
      - `python src/run.py src/tasks_config.challenge.json -l learners.base.RemoteLearner`
7. Run your agent
   a. Make sure it connects to `127.0.0.1` (localhost), port `5556`, as mentioned above.
   b. If you want to test the setup with a different agent, you can use the Python test script from the Linux Docker example. Run the following commands in a new terminal:
      - `cd Round1/evaluation/examples/linux-python/agent`
      - `python example_learner.py -a 127.0.0.1 -p 5556`
   c. Done. You should see the agent communicating with the evaluation environment!

How to upload the solution

This section describes how to upload the solution and the sources of your agent. We will ask you to upload the data to Microsoft Azure Blob Storage. The method is the same for both Linux-based and Windows-based solutions.

When you are ready to submit your solution, send us an email to solved@general-ai-challenge.org from the address you provided at the registration *using the following form:*

1) Author:
   - ***Name***
   - ***Team name***
   - ***Contact email/ phone number***

2) Evaluation hardware: ***GPU-bound / CPU-bound***
3) Target operating system: ***Linux / Windows***
4) *I would like to open source the solution/to share the solution with organizers only* (choose one option).

In the topic of your email, please indicate the submission category: ***quantitative prize / qualitative prize / both.***

The participants must attach a white paper (description/explanation of the agent) to this email. If the contestants decide to compete for the qualitative prize only (best idea award), they can submit the white paper only.

The white paper should explain the main principles and motivations behind the agent's design in a brief, structured manner (2 pages max.; in case the whitepaper exceeds the 2-page limit, participants must include a one-page summary of the paper at the beginning).

The paper must also include the instructions on how to compile and run the agent.

The submission deadline is August 14th 2017 (23:59 CET). Only the uploads done before the submission deadline will be considered for evaluation. The participants may also send just a request for the submission link and provide other information later, but before the submission deadline.

After receiving the request, the organizers will reply with an email containing a link. We will create a new block container for each participant in Azure Blob Storage and send you a *shared access signature* (SAS) URI with write access to the container.

You should submit:

- the source code of the agent (in any programming language); participants should use the standard file name source.zip;
- pre-trained agent ready for evaluation, according to the submission guide;
- only upon request from the organizers, the participants should provide the training data used for training the agent. Without the special request from the organizers, the training data should not be provided.

How to upload the solution:

1. Download MS Azure Storage Explorer and connect to Azure using the SAS URI received from the organizers.
    a. Look for a node named *(SAS-Attached Services)* in the tree view.
    b. Expand the node and expand also its child node *Blob Containers*.
    c. Select the target container that you should find there.
2. (Alternatively, you can use a custom script instead of Storage Explorer to perform the upload using the Azure API.)
3. Upload the archive with the solution and a second archive with the sources to the blob container.
    a. Upload the sources in an archive named **`source.zip`**
    b. Upload the solution with the standard file name:
    `Learner.zip` (Windows-based solution) *or* **`learner.tar.gz`** (Linux-based).
4. Done.

Note that depending on the size of your submission, the upload can take some significant time. You will also need a stable connection since the Storage Explorer does not support continued upload in case of an interruption (small outages will likely be handled by TCP/IP). Please ask for the SAS URI and start with the upload sufficiently in advance of the deadline.

Good luck!