LSTM で HTML、JavaScript を生成する

1　目的

- GPU 版の tensorflow(Keras) のインストール方法を学びます。
- HTML 及び、JavaScript の学習用データの取得方法を学びます。
- GPU の動作状況の確認方法を学びます。
- Keras を使った LSTM のモデルの作成方法を学びます。
- LSTM を使って、HTML 及び Javascript の構文を作成します。

2　アマゾン EC2 へのログイン

アマゾン EC2 にログインします。

`https://aws.amazon.com/jp/ec2/`

すでにアカウントがある場合

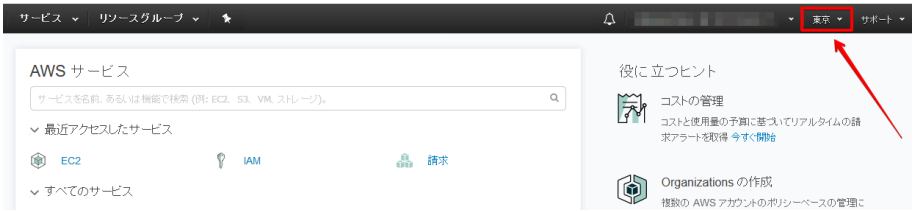
新規にアカウントをつくる場合

今回の授業では、すでにアカウントが作成されています。各自の課題提出フォルダの下に、各自の ID とパスワードが記載されたファイルがあります。そちらに記載された ID とパスワードを使ってログインして下さい。

2.1　リージョンの選択

利用するサーバのリージョンを選択します。

東京

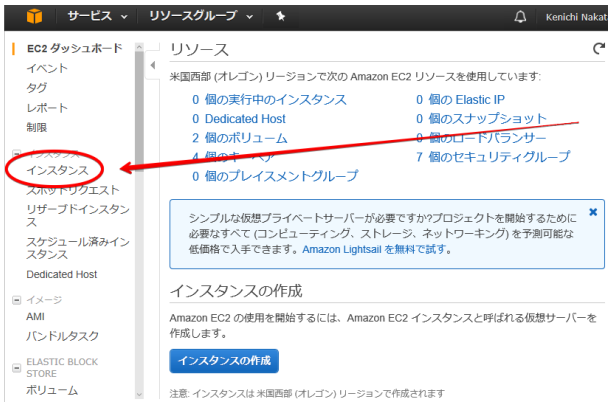


授業では、クリックして「米国東部（バージニア北部）」を選択して下さい。

EC2 サービスを選択します

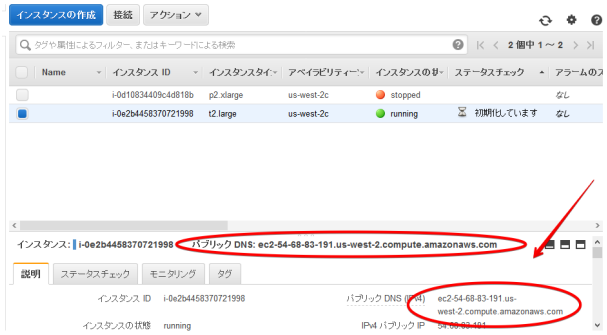
3　インスタンスの起動

以前の授業で作成した GPU 付インスタンスを作成します。授業を欠席して、作成していない人は、テキスト「アマゾン EC2 インスタンスの設定」の「インスタンスの作成」の章をみてインスタンスを作成して下さい。Putty や WinSCP のインストールや設定等を行っていない場合には、同じテキストのを参照して、インストール及び設定を行って下さい。



「インスタンス」をクリックします

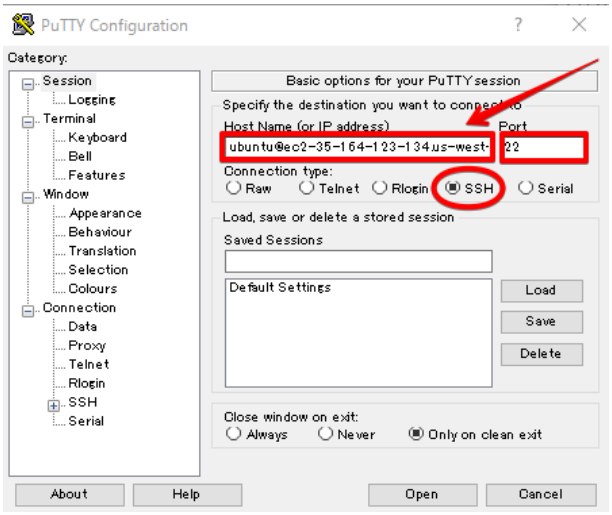
インスタンスが立ち上がっていない場合、「アクション」から「インスタンスの状態/開始」を選択し、インスタンスを立ち上げます。



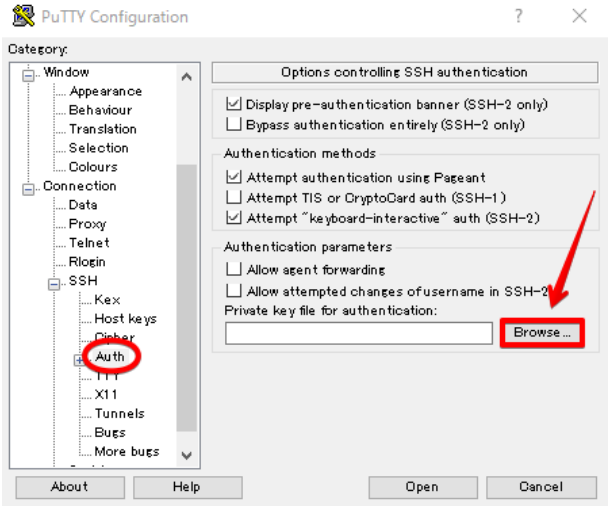
インスタンスの状態が「running」になったら、インスタンスの「パブリック DNS」を確認します。

4　PuTTY セッションの開始

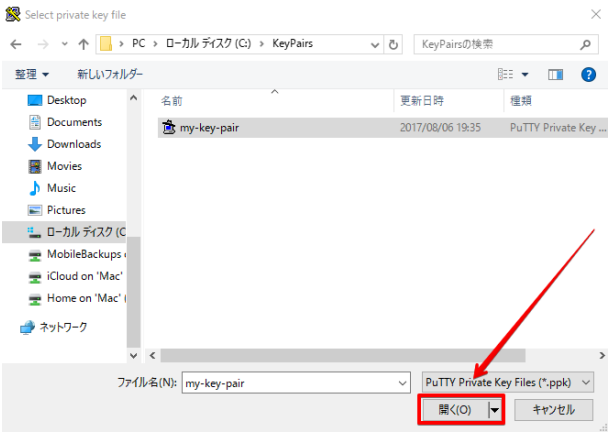
[スタート] メニューで [All Programs]-[PuTTY]-[PuTTY] を選択し、PuTTY を開始します。



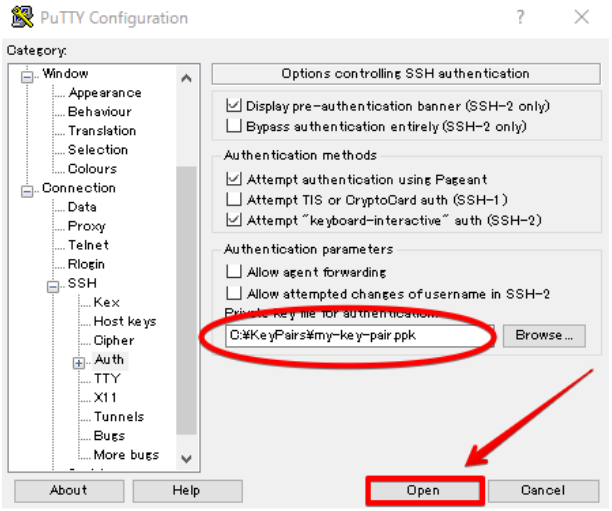
①「Host Name」に「ubuntu@(パブリック DNS)」を入力します。
②「Connection Type」を「SSH」とします。
③「Port」が「22」となっていることを確認します。



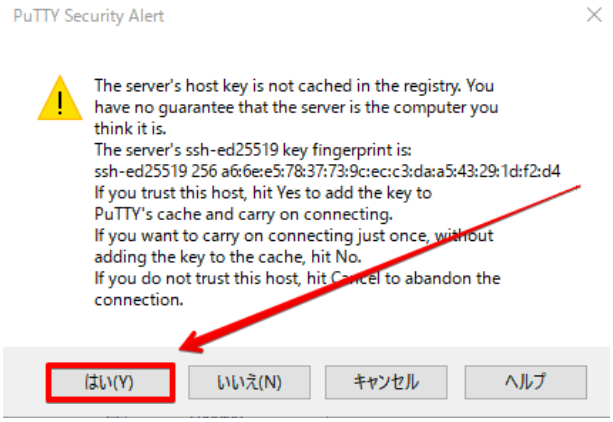
① [Category] ペインで、[Connection]、[SSH] の順に展開し、[Auth] を選択します。
②「Browse」をクリックします。



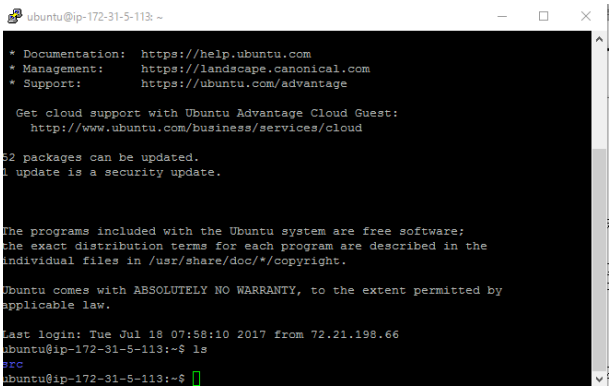
①先ほど保存した ppk ファイルを選択します。
②「開く」を押します。



①選択した ppk ファイルのパスが入力されていることを確認します。
②「Open」を押します。



「はい (Y)」を押します。



SSH クライアントでインスタンスに接続しました

5 環境設定

Amazon EC2 の GPU 付インスタンスにログインします。tensorflow（GPU 版）と、keras をインストールします。

```
conda create --name=tensorenv python=2.7
source activate tensorenv
conda install tensorflow-gpu
conda install keras-gpu
```

keras のバックエンドを mxnet から tensorflow に変更します。

```
nano ~/.keras/keras.json
```

```
{
    "image_dim_ordering": "tf",
    "epsilon": 1e-07,
    "floatx": "float32",
    "backend": "tensorflow"
}
```

「backend:」が最初から”tensorflow”になっている場合には、変更せずにそのままで結構です。

6 作業用ディレクトリの作成

6.1 作業用ディレクトリの作成、ツールの取得

```
cd ~
mkdir Lesson
cd Lesson
git clone https://github.com/k1nk/tools.git
mkdir LSTM
cd LSTM
```

7 データの取得

github(`https://github.com/explore`) 等で公開されているプロジェクトから、ファイルを抽出します。この例では、angular.js というプロジェクトからファイルを抽出しています。

```
git clone https://github.com/angular/angular.js.git
cp ../tools/mkextdata.py .
```

mkextdata.py を実行すると、指定したプロジェクトから、”js”,”html”,”css”,”md”,”json”の拡張子をもったファイルを抽出し、それぞれ 1 つのファイルにまとめて出力します。Javascript のファイルの場合には、「all.js」というファイルに出力します。「all.js」の先頭の 1 万行を学習用データとして利用します。Javascript のファイルの場合には、「all.html」というファイルに出力します。「all.html」は、先頭の２０００行を学習用データとして利用します。

mkextdata.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
#

import sys
import os
import io

def concat_file(src,dstf,ext):
    dstfilename = dstf+ext
    df=io.open(dstfilename,'a',encoding="utf-8")
    with io.open(src,'r',encoding="utf-8") as sf:
        for row in sf:
            df.write(row)
    df.close()

def concat_if_file_with_ext(src,dstf,extensions):
    root, ext = os.path.splitext(src)
    for target_ext in extensions:
        if ext == "." + target_ext :
            print(src)
            concat_file(src,dstf,ext)

def transtree(src, dstf,extensions):
    names = os.listdir(src)
    for name in names:
        srcname = os.path.join(src, name)
        #dstname = os.path.join(dst, name)
        try:
            if os.path.isdir(srcname):
                transtree(srcname, dstf,extensions)
            else:
                concat_if_file_with_ext(srcname, dstf,extensions)
```

```
        except (IOError, os.error) as why:
            print ("Can't translate %s to %s: %s" % (srcname, dstname,
                str(why)))

if __name__ == '__main__':
    source_dir = sys.argv[1]
    dest_filename = "all"
    extensions = ["js","html","css","md","json"]
    transtree(source_dir,dest_filename,extensions)
```

```
python mkextdata.py angular.js
head -n 10000 all.js > input.js
head -n 2000 all.html > input.html
```

8 学習の実行

JavaScript と HTML のファイルイ対して、ＬＳＴＭにより学習を行います。

lstm_text_generation.py

```
# -*- coding: utf-8 -*-

'''Example script to generate text from Nietzsche's writings.
At least 20 epochs are required before the generated text
starts sounding coherent.
It is recommended to run this script on GPU, as recurrent
networks are quite computationally intensive.
If you try this script on new data, make sure your corpus
has at least ~100k characters. ~1M is better.
'''

from __future__ import print_function
from keras.models import Sequential
from keras.layers import Dense, Activation
from keras.layers import LSTM
from keras.optimizers import RMSprop
from keras.utils.data_utils import get_file
from keras.callbacks import ModelCheckpoint
import numpy as np
import random
import sys
```

```
22
23 path = sys.argv[1]
24 text = open(path).read().lower()
25 print('corpus length:', len(text))
26
27 chars = sorted(list(set(text)))
28 print('total chars:', len(chars))
29 char_indices = dict((c, i) for i, c in enumerate(chars))
30 indices_char = dict((i, c) for i, c in enumerate(chars))
31
32 # cut the text in semi-redundant sequences of maxlen characters
33 maxlen = 40
34 step = 3
35 sentences = []
36 next_chars = []
37 for i in range(0, len(text) - maxlen, step):
38     sentences.append(text[i: i + maxlen])
39     next_chars.append(text[i + maxlen])
40 print('nb sequences:', len(sentences))
41
42 print('Vectorization...')
43 X = np.zeros((len(sentences), maxlen, len(chars)), dtype=np.bool)
44 y = np.zeros((len(sentences), len(chars)), dtype=np.bool)
45 for i, sentence in enumerate(sentences):
46     for t, char in enumerate(sentence):
47         X[i, t, char_indices[char]] = 1
48     y[i, char_indices[next_chars[i]]] = 1
49
50
51 # build the model: a single LSTM
52 print('Build model...')
53 model = Sequential()
54 model.add(LSTM(128, input_shape=(maxlen, len(chars))))
55 model.add(Dense(len(chars)))
56 model.add(Activation('softmax'))
57
58 optimizer = RMSprop(lr=0.01)
59 model.compile(loss='categorical_crossentropy', optimizer=optimizer)
60 model.summary()
61
```

```
62 def sample(preds, temperature=1.0):
63     # helper function to sample an index from a probability array
64     preds = np.asarray(preds).astype('float64')
65     preds = np.log(preds) / temperature
66     exp_preds = np.exp(preds)
67     preds = exp_preds / np.sum(exp_preds)
68     probas = np.random.multinomial(1, preds, 1)
69     return np.argmax(probas)
70
71 # train the model, output generated text after each iteration
72 for iteration in range(1, 60):
73     print()
74     print('-' * 50)
75     print('Iteration', iteration)
76     history = model.fit(X, y,
77               batch_size=128,
78               epochs=1,
79               validation_split=0.1,
80               callbacks=[ModelCheckpoint('simple_lstm_model.h5',
                     monitor='val_loss', verbose=0, save_best_only=True,
                     save_weights_only=False, mode='auto', period=1)])
81     print(history.history)
82
83     start_index = random.randint(0, len(text) - maxlen - 1)
84
85     for diversity in [0.2, 0.5, 1.0, 1.2]:
86         print()
87         print('----- diversity:', diversity)
88
89         generated = ''
90         sentence = text[start_index: start_index + maxlen]
91         generated += sentence
92         print('----- Generating with seed: "' + sentence + '"')
93         sys.stdout.write(generated)
94
95         for i in range(400):
96             x = np.zeros((1, maxlen, len(chars)))
97             for t, char in enumerate(sentence):
98                 x[0, t, char_indices[char]] = 1.
99
```

```
            preds = model.predict(x, verbose=0)[0]
            next_index = sample(preds, diversity)
            next_char = indices_char[next_index]

            generated += next_char
            sentence = sentence[1:] + next_char

            sys.stdout.write(next_char)
            sys.stdout.flush()
        print()
```

diversity（temperature）は、次の文字を選択する際に、確率の高い文字をより選択しやすくするか、あるいは、確率の低い文字を選択しやすくするかを調整するパラーメタです。予測を行うと、次の文字の候補としてそれぞれの文字毎の確率が計算されます。emperature=1.0 が調整をしない状態です。temperature が低いと、確率の高い文字が調整しない状態よりもより選択されやすくなります。temperature が高いと、確率の低い文字も、調整しない状態と比べて選択されやすくなります。このプログラムは、temperature を変えて、文字の選択を行っています。

```
cp ../tools/lstm_text_generation.py .
python lstm_text_generation.py input.js
python lstm_text_generation.py input.html
```

GPU の動作状況は、新たに Putty のターミナルを開いて、

```
nvidia-smi
```

とすることにより確認できます。

```
(tensorenv) ubuntu@ip-172-31-43-243:~$ nvidia-smi
Wed Sep 20 11:57:46 2017
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 375.66                 Driver Version: 375.66                    |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla K80           On   | 0000:00:1E.0     Off |                    0 |
| N/A   43C    P0    84W / 149W |  10915MiB / 11439MiB |     59%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID  Type  Process name                               Usage      |
|=============================================================================|
|    0     15881    C   python                                     10911MiB |
+-----------------------------------------------------------------------------+
```

５３回学習後（JavaScript）

```
----- diversity: 0.2
----- Generating with seed: "rmatsymbols_is_is');
goog.provide('goog."
rmatsymbols_is_is');
goog.provide('goog.i18n.numberformatsymbols_en_dg');
goog.provide('goog.i18n.numberformatsymbols_en_pr');
```

```
goog.provide('goog.i18n.numberformatsymbols_en_dg');
goog.provide('goog.i18n.numberformatsymbols_en_pr');
goog.provide('goog.i18n.numberformatsymbols_en_nu'ts');
goog.provide('goog.i18n.numberformatsymbols_en_gm');
goog.provide('goog.i18n.numberformatsymbols_en_pr');
goog.provide('goog.i18n.numberformatsymbols_b

----- diversity: 0.5
----- Generating with seed: "rmatsymbols_is_is');
goog.provide('goog."
rmatsymbols_is_is');
goog.provide('goog.i18n.numberformatsymbols_fr_gn');
goog.provide('goog.i18n.numberformatsymbols_en_ph');
goog.provide('goog.i18n.numberformatsymbols_en_ph');
goog.provide('goog.i18n.numberformatsymbols_en_mo');
goog.provide('goog.i18n.numberformatsymbols_en_tz');
goog.provide('goog.i18n.numberformatsymbols_fr_gn');
goog.provide('goog.i18n.numberformatsymbols_en_nu');
goog.provide('goog.i18n.numberformatsymbols_en_g

----- diversity: 1.0
----- Generating with seed: "rmatsymbols_is_is');
goog.provide('goog."
rmatsymbols_is_is');
goog.provide('goog.i18n.numberformatsymbols_bo_mn');
goog.provide('goog.i18n.numberformatsymbols_mn_cn');
goog.provide('goog.i18n.numberformatsymbols_ug',
  decimal_sep: '.',
  group_sep: ',',
  percent: '%',
  zero_digit: '0',
  plus_sign: '+',
  minus_sign: '-',
  exp_symbol: 'e',
  permill: '‰',
  infinity: '∞',
  nan: 'nan',
  decimal_pattern: '#,##,#,##0.###',
  scientific_pattern: '#e0',
  percent_pattern:
```

```
----- diversity: 1.2
----- Generating with seed: "rmatsymbols_is_is');
goog.provide('goog."
rmatsymbols_is_is');
goog.provide('goog.i18n.numberformatsymbols_en_us');
goog.provide('goog.i18n.numberformatsymbols_bu_ke');
goog.provide('goog.i18n.numberformatsymbols_bo_mr');
goog.provide('goog.i18n.numberformatsymbncy*igh']; i,
  gn', '$', 'mullbow ] ☒
t', ''];
  .elvalide: [0, 'udr.beif 'ge',
        '{
    build: prutter equery'[rangular-bancy.se,
        k(zy/'testart
  reg
  gruntes-vanij            apalipation: {
        dops:

--------------------------------------------------
```

60回学習後（HTML）

```
----- diversity: 1.0
----- Generating with seed: "below the carousel -->
  <div class="row"
below the carousel -->
  <div class="row">
   <bodt closs="paraps" value="orderby:na in blocauri'sepes</a>
                      <br>
                                          <input type="navider">
        <h1>one benchud</h2>
            <div ng-repeat="(rode: 1tath'"> = <brdion></div>


<div>
  <div class="page" role="page" vinput="row.obumbj"></span>

                     <a class="navbar-brondencexatems">
      <li class="fang-heagin"
```

モデルを

```
model = Sequential()
model.add(LSTM(128, return_sequences=True, input_shape=(maxlen, len(
    chars))))
model.add(LSTM(128, return_sequences=True))
model.add(LSTM(128, return_sequences=True))
model.add(LSTM(128, return_sequences=True))
model.add(LSTM(128))
model.add(Dense(len(chars)))
model.add(Activation('softmax'))
```

とすると、LSTM を重ねたモデルとなります。より複雑な表現が可能となります。

9 （オプション）CSS/md/json の学習

mkextdata.py では、CSS/md/json のファイルも取得しています。同様に、学習させてみましょう。

```
python lstm_text_generation.py all.css
python lstm_text_generation.py all.md
python lstm_text_generation.py all.json
```

10 （参考）他のプロジェクトからのデータの取得

今回のプロジェクト以外にも、多くのプロジェクトがあります。github.com `https://github.com/explore` などから探して、データを取得することもできます。

11 （参考）wget によるデータの取得

wget コマンドを使って特定の URL を起点に、再帰的にデータを取得する方法もあります。そして、取得したデータを 1 つのファイルにまとめて出力します。例

```
wget --recursive --level inf --random-wait --no-parent -Q1m --output-
    document=msn1m.html www.msn.com
wget --recursive --level inf --random-wait --no-parent -Q1m --no-check-
    certificate --output-document=yahoo_com.html www.yahoo.com
headlines?hl=ja&ned=jp
```