str2vec

str2vec is a toolkit for computing vector-space representations for variable-length phrases using recursive autoencoders (RAE). In this document, we demostrate

- How to train RAEs in an unsupervised and parallelized manner;
- And how to compute vector-space representations for phrases once the RAE is trained.

For more information about recursive auto encoders, please refer to:

> Richard Socher, Jeffrey Pennington, Eric Huang, Andrew Y. Ng, and Christopher D. Manning. *Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions*. Proc. of Conference on Empirical Methods in Natural Language Processing (EMNLP 2011), pp. 151-161.

Prerequisites

This toolkit has been tested on Ubuntu 14.04 and Mac OS 10.9. But it should work on other platforms which supported by the following softwares:

- Python 2.7.8 or later (Python 3.x is not supported)
- open-mpi 1.8.1 or later (other MPI implementation supported by mpi4py should also be OK)
- Numpy 1.8.1 or later
- Scipy 0.14.0 or later
- mpi4py 1.3.1 or later

Python is easy to install. open-mpi is usually available on most Linux platforms. Numpy, Scipy and mpi4py are available from pip. Alternatively, you can install all these softwares from source code if you like or you do care about efficiency.

We use the following commandlines to install the above softwares on Ubuntu 14.04:

```
# install python, python development files and pip
sudo apt-get install python python-dev python-pip
# install open-mpi runtime and development files and tools
sudo apt-get install openmpi-bin libopenmpi-dev
# install mpi4py
sudo pip install mpi4py
# install numpy and scipy
sudo apt-get install g++ gfortran build-essential
sudo apt-get install libopenblas-dev liblapack-dev
sudo pip install numpy scipy
```

Recursive Autoencoders Training

We have provided a toy demo on ********. We assume you have already downloaded the demo and use $DEMODIR to refer the root directory of the demo

Inputs

You need to provide two input files:

- word vectors file
- training phrases file

The word vectors file should look like this

```
10 3
word1 -0.049524 0.033159 0.008865
word2 -0.049524 0.033159 0.008865
word3 -0.049524 0.033159 0.008865
.....
word10 -0.049524 0.033159 0.008865
```

The first line is `vocabulary_size word_embedding_size` and is optional. The reset of the file are `vocabulary_size` lines which look like `word value1 value2 ... value3`. These word vectors can be trained using any toolkit, e.g. word2vec.

The training phrases file should look like this

```
first phrase ||| 10
second phrase ||| 1
...
nth phrase ||| 8
a phrase without frequency
```

`|||` is a separator between the phrase and its frequency (an integer). The frequency is optional, in which case it is assumed to be 1. As we use L-BFGS, which is a batch mode algorithm, to train the recursive autoencoders, the frequency can help us to save computation time (otherwise we need to the same computation `N` times if a phrase occurs `N` times).

Training

We have provided a demo training script `mpi-train.sh` as following

```
#!/bin/bash
if [ $# -ne 1 ]
then
  echo "Usage: $0 coreNum"
  exit -1
fi

N=$1
DEMODIR=`pwd -P`
export PYTHONPATH=$DEMODIR/bin/str2vec/src
mpirun -n $1 python $PYTHONPATH/nn/lbfgstrainer.py\
  -instances $DEMODIR/input/sample-training-file.txt\
  -model $DEMODIR/output/sample-training-file.mpi-$N.model.
gz\
  -word_vector $DEMODIR/input/sample-word-vectors-trained-b
y-word2vec.txt\
  -lambda_reg 0.01\
  -m 200
```

**************处理lambda_reg

You can use `./mpi-train.sh 2` to start 2 processes to train the RAE. The output looks like

```
Instances file: /Users/lpeng/exp/str2vec/demo/input/sample-
training-file.txt
Model file: /Users/lpeng/exp/str2vec/demo/output/sample-tra
ining-file.mpi-2.model.gz
Word vector file: /Users/lpeng/exp/str2vec/demo/input/sampl
e-word-vectors-trained-by-word2vec.txt
lambda_reg: 0.010000000000000000
Max iterations: 200

load word vectors...
preparing data...
init. RAE parameters...
seed: None
shape of theta0 430
optimizing...
saving parameters to /Users/lpeng/exp/str2vec/demo/output/s
ample-training-file.mpi-2.model.gz
Init. theta0  :       0.00 s
Optimizing    :       3.16 s
Saving theta  :       0.01 s
Done!
```

There are 5 parameters for the python program `$PYTHONPATH/nn/lbfgstrainer.py`

- `-instances` : the training phrase file
- `-model` : output model file
- `-word_vector` : word vectors file
- `-m` : max number of iterations

Output

The model files will be output into the directory if you use the above script

- `sample-training-file.mpi-2.model.gz` : the binary model file used by the toolkit
- `sample-training-file.mpi-2.model.gz.txt` : for human reading

Computing Vector Space Representations

We have provided a demo script `$DEMODIR/compute-vector.sh` for computing vector space representations for the phrases in `$DEMODIR/input/sample-training-file.txt` as following:

```
#!/bin/bash
DEMODIR=`pwd -P`
export PYTHONPATH=$DEMODIR/bin/str2vec/src
python $PYTHONPATH/nn/rae.py\
  $DEMODIR/input/sample-training-file.txt\
  $DEMODIR/input/sample-word-vectors-trained-by-word2vec.tx
t\
  $DEMODIR/output/sample-training-file.mpi-2.model.gz\
  $DEMODIR/output/sample-training-file.vec.txt
```

There are 4 parameters for the python program `$PYTHONPATH/nn/rae.py`

- 1st: input file contains phrases
- 2nd: word vectors file
- 3rd: binary model file
- 4th: output file

The phrase vectors will be output into the `output file`, and you will see the following command output which reports reconstruction errors.

```
load word vectors...
load RAE parameters...
============================================================
====
                 all             avg/node        internal n
ode
------------------------------------------------------------
----
          0.00000000,          0.00000000,
  0
          1.34299144,          1.34299144,
  1
         12.00603426,         12.00603426,
  1
          0.00000000,          0.00000000,
  0
          1.90475235,          0.63491745,
  3
          9.15740678,          4.57870339,
  2
         14.71414328,          2.94282866,
  5
          0.00000000,          0.00000000,
  0
          7.98703479,          7.98703479,
  1
         15.69007530,          2.24143933,
  7
         33.84378985,          8.46094746,
  4
         21.16908221,          3.02415460,
  7
------------------------------------------------------------
----
average reconstruction error per instance:          9.8179
4252
average reconstruction error per node:              3.8004
9388
============================================================
====
```