FACTOID EMBEDDING (Ver 1.0)
SETUP GUIDE

LIVING ANALYTICS RESEARCH CENTRE
2018

1 Introduction

Factoid Embedding takes in information of users from two social networks (e.g. screenname, username, links, etc.), and perform user accounts matching, i.e., finding social network accounts belonging to the same user. For a given user in source network (e.g. Facebook), the classifier returns a ranked list of top 20 most probable matching user accounts in target network (e.g. Twitter).

2 Prerequisite: Python Installation

Download and install the Python 3.5 to deploy the *Factoid Embedding*.

2.1 Docker

We strongly recommend you set up the environment in a docker container and run the program in a container.

Enter the below commands in *Terminal* or *Command Prompt* to create a container:

```
NV_GPU=4,5,6,7 nvidia-docker run -it --name test -v
local_path:container_path nvcr.io/nvidia/tensorflow:18.02-py3
bash
```

2.2 Python Packages

The following Python packages are also required for *Factoid Embedding*:

1. jellyfish (pip3 install jellyfish)
2. sklearn (pip3 install scikit-learn)
3. tensorflow (work with a specific docker)
4. keras (pip3 install keras)
5. numpy & scipy (click the left link)
6. pillow & imagehash (pip3 install pillow; pip3 install imagehash)
7. xxhash (pip3 install xxhash)
8. dispy (pip3 install dispy)

The recommended way to install the above packages is through *pip3*, which installs and manages software packages written in Python. Enter the below commands in *Terminal* or *Command Prompt* to install JellyFish:

```
pip3 install jellyfish
```

2.3 Dispy

Dispy is a python package for distributed computing. Notice, install dispy on both the server machine and the docker container. After installing dispy, run dispy on the server machine.

```
python3 /usr/local/lib/python3.5/dist-packages/dispy/dispynode.py
-i IP_ADDRESS_OF_THE_SERVER_MACHINE -c NUMBER_OF_WORKERS
```

For example:

```
python3 /usr/local/lib/python3.5/dist-packages/dispy/dispynode.py
-i 10.0.109.76 -c 32
```

3 Loading Input Data

Prior to running *Factoid Embedding*, the input files which contain information on two social networks should be placed in the *data* folder. A set of sample data (i.e. Facebook and Twitter) is provided in the *data* folder. Table 1 shows the details for each sample input files.

Table 1: List of Sample Input Files

No.	**Filename**	**Description**	**Field**
1	*fb_screen_names.txt*	Facebook users screen names	Facebook Username Facebook Screenname
2	*fb_user_names.txt*	Facebook users user name	Facebook Username Facebook Username
3	*fb_sub_network.txt*	Facebook links between users	Source User's Facebook Username Target User's Facebook Username
4	*tw_screen_names.txt*	Twitter users screen names	Twitter ID Twitter Screenname
5	*tw_user_names.txt*	Twitter users user name	Twitter ID Twitter Username
6	*tw_sub_network.txt*	Twitter links between users	Source User's Twitter ID Target User's Twitter ID

4 Deploying Factoid Embedding

4.1 Parameter Setting

Triplet Embedding has a few configurable parameters stored in *settings.cnf* file. Below are the list of parameters:

1. data
   - *path*: path to the data folder.
   - *source_prefix*: prefix of source platform (e.g. fb).
   - *target_prefix*: prefix of target platform (e.g. tw).
   - *source_col*: source column in the ground truth(e.g. 0).
   - *target_col*: target column in the ground truth (e.g. 1).
2. predicate_name
   - *concatenate*: indicate whether concatenate username and screen name (True or False).
   - *preprocess*: indicate whether preprocess the name e.g. removing non-ascii code (True or False).
   - *method*: the method to measure the name similarity (recommend tfidf) (tfidf or jaro_winkler).
   - *screen_name_exist*: indicate whether screen name exists.
3. dispy
   - *ip*: the ip address of the dispy server machine (recommend use the host machine).
   - *port*: use dispy's default port 51348.
   - *remote_path*: the path of where dispy store the intermediate data.
4. cosine_embedding
   - *pass*: indicate whether pass the embedding step to get a quick result (True or False).
   - *n_gpu*: the number of GPUs used.
   - *n_dim*: the dimension of cosine embedding.
   - *n_iter*: the number of iterations going through the data.
   - *learning_rate*: the learning rate for cosine embedding.
   - *batch_size*: the mini-batch size, i.e. number of name pairs for each mini-batch (e.g. 32*1024).
   - *partition_path*: the path to put partitioned data (in docker container).
5. triplet_embedding
   - *supervised*: indicate if learning is supervised. Default is False.
   - *bias*: transformation bias. Default is True. (See the bias vector in Section 2.3 in D3 report.)
   - *learning_rate_f*: learning rate for follow triplet.
   - *learning_rate_a*: learning rate for attribute triplet.
   - *snapshot and snapshot_gap*: for debugging, use the default setting.
   - *n_iter*: number of learning iterations.
   - *warm_up_iter*: number of warming up iterations.
   - *user_dim*: dimension of user embedding.
   - *nce_sampling*: number of negative sampling
   - *batch_size*: batch size for triplet embedding (e.g. 256).

4.2 Running Factoid Embedding

After placing the input files into the data folder and setting the parameters, run Factoid Embedding with the below command:

```
Triplet_Embedding> python3 main.py
```

Note that the above command should be executed in the *Factoid_Embedding* folder.

5 Saving Output Data

For a given user from source network (in the testing file), Factoid Embedding returns a ranked list of 20 most probable match users from target network. The output file will be save as *"top20_users.txt"* in to the folder with "result" in its name. Besides, there are also other files be generated: "user_embedding_result.npy" which stores the user embedding matrix; "sc2uid.txt" and "tg2uid.txt" store the mapping between user (from source network or target network) to the row number of user embedding matrix.