PGiza Manual

The Parallel Giza Trainer is nearly done, and the manual is here: (For example setting, you can look at /people/qing0/exp1. And the bin and script dir are set up on /people/qing0/bin and /people/qing0/script.

Download and Install

The PGiza package consists of two parts, and packed into one single package. The first part is binary, which you should compile, and the second part is scripts, which enable you set up training directory and control the whole training process.

To install the trainer, first download the package (avaiable soon), decompress it use tar, then run

```
./configure --prefix=SOMEWHERE
make
make install
```

The binary will be put into SOMEWHERE/bin, where we shall refer to as $BIN_DIR.

To install the script, copy the script files in **script** directory of the source tarball to somewhere, call $SCRIPT_DIR, and you are done.

Description of files

There are a bunch of separated utilities that makes PGIZA work, the following paragraphs will present description of these utilities. Generally they can be separated into two categories:

* Alignment tools, in charge of aligning corpora based on previously trained model, record things that need to be updated, and output them for normalization tools. Each stage of training has one particular aligner. There are three of them:
  - Model One Aligner : **model1_align**
  - HMM Aligner : **hmm_align**
  - Viterbi Aligner (for Model 3,4 and 5, currently only support model 3,4) : **model345align**

* Normalization tools. They will take the updates that alignment tools output as input, and do normalization on certain kind of table. They are specially for each kind of table, but some normalizer can also normalize several different types of table using different parameters. There are four of them, being able to normalize three kind of tables.
  - TTable Normalizer : **renomalizeModel1**
  - ATable Normalizer : **renomalizeAModel**
  - HMM Jump Combiner : **combine_hmm**
  - NTable, DTable, p0 Normalizer : **renomalizeNDP**

All the alignment tools takes only one parameter, which is the path to the configure file. In most case, you do not need to take care of generating the whole configure file, however, you should provide all the parameters you want **EXCEPT INPUT AND OUTPUT PATH OF CORPORA/MODEL**, the detailed information will be given later.

The normalizer takes more parameters, the first of all is the path to configure file, while the other parameters are all updates that will be used to update the model.

When running training the process is like this:

1. The master starts aligner on every child with models from last step, if applicable. Then the master starts watching any change in the watch directory ($WATCH_DIR).
2. The children do alignment, and put update into a collecting directory ($COLL_DIR). When it finishes, it will touch a file in the watching directory ($WATCH_DIR).
3. If all the children have finished the alignment, the master start the renomalization process, update models and goto step one if more steps needed to be taken.

In the script directory, there are several stuff that in charge of these tasks. * **defines.inc** is a sample environment configuration file, which you can use as a template for your training environment, and we will give description on it latter. * **setup_dir.sh** generate a directory where training can be performed. To run it, you must specify a environment configuration file, and the name of the directory you want to create. Type:

```
${SCRIPT_DIR}/setup_dir.sh defines.inc train_dir
```

And it will generate the directory with the following structure:

```
align  coll  data  defines.inc  log  output  snt_vcb  split  templ  tmp  watch  work
```

Then it will tell you what to do next before start training:

```
OK, the directory is set! Now you need to do the
following tasks before training:

1. Put your training data into data director, let's say your file has the full path
  /people/qing0/train_dir/data/SOURCE.RAW and
  /people/qing0/train_dir/data/TARGET.RAW
2. Find and edit a template for giza training, you should remove all
   the option relevant to input/output filenames, such as 'o' 's' 't' etc.
   Assume you put the file in /people/qing0/train_dir/TEMPLATE.gizaconf
3. Edit a machine configuration file, the file is simply a list of hostnames you
   want to run the training, each line contain one hostname, say, you added 10
   lines in the file  /people/qing0/train_dir/MC

After that, you can run :
train_ega.sh SOURCE.RAW TARGET.RAW TEMPLATE.gizaconf MC
and wait for its finishing
```

Follow the instruction, create a gizaconfigure file, the raw training data and the Multi Machine Configuration File, before going to next step, below is a sample MC file:

```
hostname1
hostname2
hostname.domain
123.45.33.95
...
```

Remember that the corpora file should be put in **data** directory.

* **train_ega.sh** the main training script. It takes 4 parameters:

```
train_ega.sh SOURCE.RAW TARGET.RAW TEMPLATE.gizaconf MC
```

The first two are source and target corpora, and the third is the template giza configure file. Last file is the machine configure file.

Other scripts are called by the main training script, but it may be good to know the detail of each script, so that if something goes wrong, you may be able to start the process without running them again. (TO-DO: Complete me)

Run Training

There are several steps to run trainning, and the following part is a tutorial of a sample training.

Setup SSH Authorization

The script is now using SSH to fire new process in remote childs, and by default SSH will need you to input password every time. So if you do not want to stay in front of the screen and type the password a thousand times, you should set up SSH Authorization. To do that, you will need to do the following work on every machine you want to run child process on:

```
ssh-keygen -d
```

Press enter until it finishes. It will generate two files in a newly create directory ~/.ssh_

```
id_dsa
id_dsa.pub
```

Fetch all the **id_dsa.pub** files, and paste their content into a single file named _~/.ssh/authorized_keys2 on the master machine. **chmod** it to 600. (A best way is to collect all the **id_dsa.pub** file into one file, including the master machine and distribute it to all the machines, so every machine will not ask for password when executing ssh command.

If home directory is set to AFS, then it will be simple, just generate **id_dsa.put** and rename it to **authorized_keys2**, then you can access all the machines using that directory as home.

Be sure to ssh each machine before start training, if you are asked for password, it is possible that the training will not work.

Share the file system

The trainer can only run on children should have access to the training directory **Using the same path**. So it will be better to put the whole training directory on NFS or AFS.

Edit environment configuration

The file defines.inc in $SCRIPT_DIR provide a sample environment definition to run giza. There are several most important definitions: * Paths to binary/script files

```
SCRIPT_DIR=/people/qing0/script # the script directory
BIN_DIR=/people/qing0/bin # the binary directory
```

Note that the package does not include **mkcls**, and you may copy it from origin GIZA to the $BIN_DIR. Without it the system can also run, but the result may not be right.

* Training rounds

```
MODEL_ONE_ROUND=5       #How many times of model one iteration
MODEL_HMM_ROUND=5       #How many times of HMM iteration
MODEL_345_ROUND=011233 #How many times of viterbi iteration
```

The first two is easy to understand, and the last one is a little complex. We use number 0,1,2,3 to indicate the

steps we want to take. 0 means transformation from HMM to model 3, 1 means model 3 iteration, 2 means transform from model 3 to model 4, and 3 means model 4 iteration. So it should be started with 0, and have several 1_s before _2, if you want model 4 training.

Setup training directory

Use the script **setup_dir.sh** to generate a new training directory, provide it will a template define.inc.

```
${SCRIPT_DIR}/setup_dir.sh defines.inc train_dir
```

Put training data into data directory

Put the training file, for example RAW and RAW into the train_dir/data subdirectory

Edit GIZA configure file

Take one of the GIZA file, modify it as you like, and remove all the input/ouput fields like "o,s,t" etc. You may put it into the training directory for future analysis.

Edit MC configure file

Decide how many machines you want to perform training, and put the hostname or IP address into an MC configure file. Each line of the file should contain one hostname or IP. If you want one machine to run more than one process, simply put more than one line of the same hostname.

It is recommend that you choose the fastest machine as the master, and it can also (and should be) be put into the MC file, because normalization is time-consuming and not parallelizable, it will run on the master. More over, when master is waiting for children to finish, it is actually idle. So using it as a child too should be a good idea.

Start training!

Now, we can start the training! Type the following script:

```
train_ega.sh SOURCE.RAW TARGET.RAW TEMPLATE.gizaconf MC
```

If nothing bad happen, the training will finish with the following output:

```
Well, everything is done
```

Then you can check the output as you defined in the **defines.inc**.

Where to find the results?

Actually, almost all the result can be specified in **defines.inc**. There are a bunch of settings, below is an example:

```
# Ouput of each model
OUTPUT_MODEL_ONE=${OUTPUT_DIR}/t1.final

# HMM output
OUTPUT_T_HMM=${OUTPUT_DIR}/thmm.final
OUTPUT_A_HMM=${OUTPUT_DIR}/ahmm.final
OUTPUT_H_HMM=${OUTPUT_DIR}/hhmm.final

# Model 3/4/5 output
OUTPUT_T_FINAL=${OUTPUT_DIR}/t4.final
```

```
OUTPUT_A_FINAL=${OUTPUT_DIR}/a4.final
OUTPUT_D_FINAL=${OUTPUT_DIR}/d4.final
OUTPUT_N_FINAL=${OUTPUT_DIR}/n4.final
OUTPUT_PZ_FINAL=${OUTPUT_DIR}/p0.final
OUTPUT_H_FINAL=${OUTPUT_DIR}/h4.final
OUTPUT_ALIGN=${OUTPUT_DIR}/align.final
```

Here, OUTPUT_MODEL_ONE is the output of TTable after finishing training of model one, and OUTPUT_T_HMM , OUTPUT_A_HMM , OUTPUT_H_HMM are TTable, ATable and HMM Jump table after the HMM alignment. The third one is not used in decoder, but it is essential for the training of next stage.

Finally, a bunch of _FINAL_s are the final models, depending on how you going to run the viterbi alignment, it may be model 3 or model 4. One important thing is that the inversed T table will only be outputed in the final stage, with the name

```
${OUTPUT_T_FINAL}.inv
```

The OUTPUT_ALIGN file contains the **full** alignment of the final step. It is still possible to find the alignment of every step, they are in another directory which is also configurable:

```
ALIGN_DIR=$PWD/align
```

The naming convention of alignments are a little complex, for example:

```
R_0_P_1_AT_islr0s10.finish.AH3
```

It has the pattern R_${ROUND}_P_${PART}_AT_${hostname}.finish.A${MODEL}. The example means that it is the first round of viterbi training, on part 1 of corpora, at islr0s10, and its model is H3 (from HMM to Model 3).

The ${MODEL} can be 1, H, H3, 3, 34, 4, corresponding to model one, HMM, HMM to Model 3, Model 3, Model 3 to Model 4 and Model 4.

We did not output combined file, and you can do it, I think it is easy:)

-- EdwardGao - 26 Nov 2007

Attachment:	Action:	Size:	Date:	Who:	Comment:
pgiza-0.1.tar.gz	action	308068	27 Nov 2007 - 23:00	EdwardGao	Download PGIZA-0.1 with Script

Revision: r1.4 - 27 Nov 2007 - 22:37 GMT - EdwardGao

ISLwiki > SmtPeople > EdwardGao > ParallelGiza