Files generated by traingiza

These are the files generated by our script traingiza in the output directory. Click on a file name for more information. Most of the information is excerpted from the official README for the GIZA++ package from the SMT toolkit Egypt. Some of the files may appear with a .bz2 extension, meaning that they're compressed using bzip2. Use bunzip2 to decompress them or bzcat to do so on the fly.

Warning Since this documentation is largely based on that of GIZA++, we use their terminology. This means that the **source language** is the language you plan to translate TO and the **target language** is the language you plan to translate FROM.

```
./ibm1
./ibm1/giza.A1.5
./ibm1/giza.t1.5
./ibm2
./ibm2/giza.A2.5
./ibm2/giza.a2.5
./ibm2/giza.t2.5
./classesNN
./classesNN/ibm3
./classesNN/ibm3/giza.A3.5
./classesNN/ibm3/giza.a3.5
./classesNN/ibm3/giza.d3.5
./classesNN/ibm3/giza.n3.5
./classesNN/ibm3/giza.p0_3.5
./classesNN/ibm3/giza.t3.5
./classesNN/ibm3/giza.gizacfg
./classesNN/ibm3/zerofert
./classesNN/ibm4
./classesNN/ibm4/giza.A3.final
./classesNN/ibm4/giza.D4.final
./classesNN/ibm4/giza.a3.final
./classesNN/ibm4/giza.d3.final
./classesNN/ibm4/giza.d4.final
./classesNN/ibm4/giza.actual.ti.final
./classesNN/ibm4/giza.n3.final
./classesNN/ibm4/giza.p0_3.final
./classesNN/ibm4/giza.t3.final
./classesNN/ibm4/giza.ti.final
./classesNN/ibm4/giza.gizacfg
./classesNN/ibm4/zerofert
./classesNN/corpusname.1.vcb.classes
./classesNN/train.f.oneline.vcb.classes
./classesNN/corpusname.1.vcb.classes.cats
./classesNN/train.f.oneline.vcb.classes.cats
./traindata
./traindata/giza.gizacfg
./traindata/giza.perp
./traindata/zerofert
./traindata/corpusname.1.vcb
./traindata/corpusname.2.vcb
./traindata/corpusname.1_corpusname.2.snt
./traindata/corpusname.2_corpusname.1.snt
./traingizalog.NNNN

./links
./links/ibm1.ali -> ../ibm1/giza.A1.5
./links/ibm1.t -> ../ibm1/giza.t1.5
./links/ibm2.a -> ../ibm2/giza.a2.5
./links/ibm2.ali -> ../ibm2/giza.A2.5
./links/ibm2.t -> ../ibm2/giza.t2.5
./links/src.vcb -> ../traindata/corpusname.2.vcb
./links/trg.vcb -> ../traindata/corpusname.1.vcb
./links/classesNN
```

```
./links/classesNN/ibm3.a -> ../../classesNN/ibm3/giza.a3.5
./links/classesNN/ibm3.ali -> ../../classesNN/ibm3/giza.A3.5
./links/classesNN/ibm3.dis -> ../../classesNN/ibm3/giza.d3.5
./links/classesNN/ibm3.fer -> ../../classesNN/ibm3/giza.n3.5
./links/classesNN/ibm3.t -> ../../classesNN/ibm3/giza.t3.5
./links/classesNN/ibm4.a -> ../../classes80/ibm4/giza.a3.final
./links/classesNN/ibm4.ali -> ../../classes80/ibm4/giza.A3.final
./links/classesNN/ibm4.dis -> ../../classes80/ibm4/giza.d3.final
./links/classesNN/ibm4.fer -> ../../classes80/ibm4/giza.n3.final
./links/classesNN/ibm4.t -> ../../classes80/ibm4/giza.t3.final
./links/classesNN/ibm4.ti -> ../../classes80/ibm4/giza.ti.final
./links/classesNN/src.vcb -> ../../traindata/corpusname.2.vcb
./links/classesNN/trg.vcb -> ../../traindata/corpusname.1.vcb
```

Directory ibmn

Directory where ibm *n* model files are stored.

Alignment file (giza.A*, *.ali)

```
In each iteration of the training, and for each sentence pair in the
training set, the best alignment (viterbi alignment) is written to the
alignment file (if the dump parameters are set accordingly). The
alignment file is named prob_table.An.i, where n is the model number
({1,2, 2to3, 3 or 4}), and i is the iteration number. The format of
the alignments file is illustrated in the following sample:

# Sentence pair (1)
il s' agit de la même société qui a changé de propriétaires
NULL ({ }) UNK ({ }) UNK ({ }) ( ({ }) this ({ 4 11 }) is ({ }) the ({ }) same ({ 6 }) agency ({ }) which ({ 8 }) has ({
# Sentence pair (2)
UNK UNK , le propriétaire , dit que cela s' est produit si rapidement qu' il n' en connaît pas la cause exacte
NULL ({ 4 }) UNK ({ 1 2 }) UNK ({ }) , ({ 3 }) the ({ }) owner ({ 5 22 23 }) , ({ 6 }) says ({ 7 8 }) it ({ }) happened

The alignment file is represented by three lines for each sentence
pair. The first line is a label that can be used, e.g., as a caption
for alignment visualization tools.  It contains information about the
sentence sequential number in the training corpus, sentence lengths,
and alignment probability. The second line is the target sentence, the
third line is the source sentence. Each token in the source sentence
is followed by a set of zero or more numbers. These numbers represent
the positions of the target words to which this source word is
connected, according to the alignment.
```

Translation table (giza.t*, *.t)

```
Each line is of the following format:

s_id t_id P(t_id/s_id)

where:
 s_id: is the unique id for the source token
 t_id: is the unique id for the target token
 P(t_id/s_id) the probability of translating s_id as t_id

sample part of a file:

3599 5697 0.0628115
2056 10686 0.000259988
8227 3738 3.57132e-13
5141 13720 5.52332e-12
10798 4102 6.53047e-06
8227 3750 6.97502e-14
7712 14080 6.0365e-20
7712 14082 2.68323e-17
7713 1083 3.94464e-15
7712 14084 2.98768e-15
```

Directory classesNN

Before launching any giza training, word classes must be created (clustering) with the `mkcls` utility. These classes will influence the result of the ibm*n* models, where *n* > 2. Therefore, models 3 and up are stored in a directory classes*NN* where *NN* is the number of classes created with `mkcls`. There is one directory classes*NN* for each number *NN* of classes created to launch a training.

a table (giza.a*, *.a)

```
The format of each
```

```
line is as follows:

i j l m p(i | j, l, m)

where i, j, l, m are all integers and
 j = position in target sentence
 i = position in source sentence
 l = length of source sentence
 m = length of target sentence
and p(i/j,l,m) is the probability that a source word in position i is
moved to position j in a pair of sentences of length l and m.

sample:

15 14 15 14 0.630798
15 14 15 15 0.414137
15 14 15 16 0.268919
15 14 15 17 0.23171
15 14 15 18 0.117311
15 14 15 19 0.119202
15 14 15 20 0.111369
15 14 15 21 0.0358169
```

Distortion table (giza.d*, *.dis)

```
The format is similar to the a table
        with a slight difference -- the
position of i & j are switched:

j i l m p(j/i,l,m)

sample:

15 14 14 15 0.286397
15 14 14 16 0.138898
15 14 14 17 0.109712
15 14 14 18 0.0868322
15 14 14 19 0.0535823
```

Fertility table (giza.n*, *.fer)

```
Each line in this file is of the following format:

source_token_id p0 p1 p2 .... pn

where p0 is the probability that the source token has zero fertility;
p1, fertility one, ...., and n is the maximum possible fertility as
defined in the program.

sample:

1 0.475861 0.282418 0.133455 0.0653083 0.0329326 0.00844979 0.0014008
10 0.249747 0.000107778 0.307767 0.192208 0.0641439 0.15016 0.0358886
11 0.397111 0.390421 0.19925 0.013382 2.21286e-05 0 0
12 0.0163432 0.560621 0.374745 0.00231588 0 0 0
13 1.78045e-07 0.545694 0.299573 0.132127 0.0230494 9.00322e-05 0
14 1.41918e-18 0.332721 0.300773 0.0334969 0 0 0
15 0 5.98626e-10 0.47729 0.0230955 0 0 0
17 0 1.66346e-07 0.895883 0.103948 0 0 0
```

P0 table (giza.p0*)

```
(1 - P0 is the probability of inserting a null after a
   source word.)

This file contains only one line with one real number which is the
value of P0, the probability of not inserting a NULL token.
```

Zero fertility file (zerofert)

File with zero fertility words. File format: one word per line. This file is created for the ISI ReWrite Decoder , using a utility provided with that decoder. The words belong to the language you wish to translate to.

Configuration file (giza.gizacfg)

All configuration parameters used when running giza++ for that model.

Actual inverse probability table (giza.actual.ti.final)

Similar to translation table, but with source id and target id reversed on each line. Moreover, each source id and target id is replaced by the actual token.

Inverse probability table (giza.ti.final, *.ti)

Similar to translation table, but with source id and target id reversed on each line.

Word class file (*.classes)

Class file generated by `mkcls` after clustering. The format is not documented, but seems to be a list of words in lexicographical order, each one on a line followed by the class number to which each word belongs. Example:

```
19 15
? 16
a 17
a-la-carte 18
a-la-mode 19
a.m. 20
aa 21
aaah 22
abalone 23
abated 24
```

Word category file (*classes.cats)

Class file generated by `mkcls` after clustering. The format is not documented, but seems to be, for each line, a category number, and a comma-separated list of words belonging to that category. There are some cryptic comments too. Example:

```
;KategProblem:cats: 80   words: 1338
0:$,
1:
2:absolutely,attended,brought,call,cluases,come,concerned,deleting,demonstrated,
done,introduced,lacking,left,mandatory,mentioning,misled,moved,participate,
passing,protected,put,registered,remained,signed,still,struck,tabled,taken,
testing,trade,waiting,worked,
```

...

```
81:11,happy,his,its,merely,politicians,racial,several,this,visible,
;I have 81 categories used.

Problem(1,0,208)
      #value: 6
#valueChange: 31870
   #doChange: 15863
```

Traindata directory (traindata)

This directory contains some results from the training as well as the results of the preprocessing of the corpora needed by giza++. If this directory exists and is intact, then the script `traingiza` will use the results found there and skip the preprocessing of the corpora given as arguments.

Perplexity file (giza.perp)

```
This file will be generated at the end of training. It summarizes
perplexity values for each training iteration.  Here is a sample
perplexity file that illustrates the format. The format is the same
for cross entropy. If no test corpus was provided, the values for it
will be set to "N/A".

# train-size test-size iter. model train-perplexity test-perplexity final(y/n) train-viterbi-perp test-viterbi-perp
        447136 9625 0 1 187067 186722 n 3.34328e+06 3.35352e+06
        447136 9625 1 1 192.88 248.763 n 909.879 1203.13
        447136 9625 2 1 99.45 139.214 n 316.363 459.745
        447136 9625 3 1 83.4746 126.046 n 214.612 341.27
        447136 9625 4 1 78.6939 124.914 n 179.218 303.169
        447136 9625 5 2 76.6848 125.986 n 161.874 286.226
        447136 9625 6 2 50.7452 86.2273 n 84.7227 151.701
        447136 9625 7 2 42.9178 74.5574 n 63.6644 116.034
        447136 9625 8 2 40.0651 70.7444 n 56.3186 104.274
        447136 9625 9 2 38.8471 69.4105 n 53.1277 99.6044
        447136 9625 10 2to3 38.2561 68.9576 n 51.4856 97.4414
        447136 9625 11 3 129.993 248.885 n 86.6675 165.012
        447136 9625 12 3 79.2212 169.902 n 86.4842 171.367
        447136 9625 13 3 75.0746 164.488 n 84.9647 172.639
        447136 9625 14 3 73.412 162.765 n 83.5762 172.797
        447136 9625 15 3 72.6107 162.254 y 82.4575 172.688
```

Vocabulary file (*.vcb)

In the `links` directory, `src.vcb` is the vocabulary file for the source language and `trg.vcb` is the vocabulary file for the target language.

Each entry is stored on one line as follows:

```
 uniq_id1 string1 no_occurrences1
 uniq_id2 string2 no_occurrences2
 uniq_id3 string3 no_occurrences3
 ....
```

Here is a sample from an English vocabulary file:

```
627 abandon 10
628 abandoned 17
629 abandoning 2
630 abandonment 12
631 abatement 8
632 abbotsford 2
```

uniq_ids are sequential positive integer numbers. 0 is reserved for the special token NULL.

Bitext file (*.snt)

Each sentence pair is stored in three lines. The first line is the number of times this sentence pair occurred. The second line is the source sentence where each token is replaced by its unique integer id from the vocabulary file and the third is the target sentence in the same format.

Here's a sample of 3 sentences from English/french corpus:

```
1
1 1 226 5008 621 6492 226 6377 6813 226 9505 5100 6824 226 5100 5222 0 614 10243 613
2769 155 7989 585 1 578 6503 585 8242 578 8142 8541 578 12328 6595 8550 578 6595 6710 1
1
1 1 226 6260 11856 11806 1293
11 1 1 11 155 14888 2649 11447 9457 8488 4168
1
1 1 226 7652 1 226 5337 226 6940 12089 5582 8076 12050
1 1 155 4140 6812 153 1 154 155 14668 15616 10524 9954 1392
```

Train log (traingizalog.*)

Log created by traingiza for each training launched in that directory.

Links directory

This directory is provided for convenience only. It contains symbolic links to all the important model files, using a simple naming scheme. The files the links point to may or may not be compressed (with bzip2).

Fabrizio Gotti, 2005

Contact Us
Syndication
Site map
Webmaster

Login or Register