

# PHARAOH



## Training Manual

Philipp Koehn  
koehn@csail.mit.edu  
MIT CSAIL

July 16, 2004

### Abstract

This manual describes the training of phrase-based statistical machine translation models. Specifically, it describes the program `train-phrase-model.perl` that generates phrase models for the statistical machine translation decoder Pharaoh.

We illustrate and commentate the training process and provide documentations for the parameters of the training script. Also, a number of additional programs are described that support data preparation, phrase table filtering, n-best list generation and parameter tuning with minimum error rate training.

The decoder is described in a different manual, please contact the author for more information.

# Contents

<b>1</b>	<b>Prelude</b>	<b>3</b>
<b>2</b>	<b>Overview</b>	<b>4</b>
2.1	Data Preparation . . . . .	5
2.2	Running GIZA++ . . . . .	6
2.3	Align Words . . . . .	7
2.4	Get Lexical Translation Table . . . . .	9
2.5	Extract Phrases . . . . .	10
2.6	Score Phrases . . . . .	11
2.7	Create Configuration File . . . . .	13
<b>3</b>	<b>Parameters for Training</b>	<b>14</b>
3.1	Basic Options . . . . .	14
3.2	Partial Training . . . . .	16
3.3	File Locations . . . . .	16
3.4	Alignment Heuristic . . . . .	17
3.5	Maximum Phrase Length . . . . .	17
3.6	GIZA++ Options . . . . .	18
<b>4</b>	<b>Additional Programs</b>	<b>19</b>
4.1	Data Preparation . . . . .	19
4.2	Filtering the Phrase Table . . . . .	19
4.3	Generating N-Best Lists . . . . .	20
4.4	Minimum Error Rate Training . . . . .	21

# 1 Prelude

This software release contains the following programs:

- `train-phrase-model.perl` training script
- `phrase-extract` called by training
- `phrase-score` called by training
- `pharaoh.2004-05-10` latest version of decoder
- `clean-corpus-n.perl` useful program for corpus generation
- `lowercase.perl` useful program for corpus generation
- `n-best-from-pharaoh.perl` n-best list generation script
- `run-filtered-pharaoh.perl` run the decoder with a filtered phrase table
- `minimum-error-rate-training.perl` program for parameter tuning

This is a very limited release of this software.

You may use this software for personal research use, but not integrate it into commercial applications, or use output of the system for commercial applications.

**DO NOT DISTRIBUTE!**

## 2 Overview

We will start with an overview of the training process. This should give a feel for what is going on and what files are produced. In the following chapter we will go into more details of the options of the training process and additional tools.

The training process takes place in 7 steps, all of them executed by the script

```
train-phrase-model.perl
```

The seven steps are

1. Prepare data (45 minutes)
2. Run GIZA++ (16 hours)
3. Align words (2:30 hours)
4. Get lexical translation table (30 minutes)
5. Extract phrases (10 minutes)
6. Score phrases (1:15 hours)
7. Create configuration file (1 second)

The run times mentioned in the steps refer to a recent training run on the 751'000 sentence, 16 million word German-English Europarl corpus, on a 3GHz Linux machine.

Training data has to be provided sentence aligned (one sentence per line), in two files, one for the foreign sentences, one for the English sentences:

```
>head -3 corpus/euro.*
==> corpus/euro.de <==
wiederaufnahme der sitzungsperiode
ich erklare die am donnerstag , den 28. maerz 1996 unterbrochene sitzungsperiode
des europaeischen parlaments fuer wiederaufgenommen .
begruessung

==> corpus/euro.en <==
resumption of the session
i declare resumed the session of the european parliament adjourned on thursday ,
28 march 1996 .
welcome
```

A few other points have to be taken care of:

- one sentence per line, no empty lines
- sentences longer than 100 words (and their corresponding translations) have to be eliminated (note that a shorter sentence length limit will speed up training)

- everything lowercased

The script may be executed as:

```
train-phrase-model.perl --root-dir . --f de --e en --corpus corpus/euro >& LOG
```

The training data should be placed in a special directory called `corpus`. This directory and all other files are in the location specified by the `--root-dir` option. Here it is the current directory (`.`), but we should always specify a full path. The foreign and English corpus file names differ only in their extension, which is specified at the command line with the switches `--f` and `--e`.

After full training, there will be four directories under `--root-dir`:

- `corpus` – the corpus files, word class files, vocabulary files
- `giza.de-en` – output of GIZA++ training foreign to English
- `giza.en-de` – output of GIZA++ training English to foreign
- `model` – the model files

## 2.1 Data Preparation

The parallel corpus has to be converted into a format that is suitable to the GIZA++ toolkit. Two vocabulary files are generated and the parallel corpus is converted into a numbered format.

The vocabulary files contain words, integer word identifiers and word count information:

```
==> corpus/de.vcb <==
1      UNK      0
2      ,        928579
3      .        723187
4      die      581109
5      der      491791
6      und      337166
7      in       230047
8      zu       176868
9      den      168228
10     ich      162745
```

```
==> corpus/en.vcb <==
1      UNK      0
2      the      1085527
3      .        714984
4      ,        659491
5      of       488315
6      to       481484
7      and      352900
8      in       330156
9      is       278405
10     that     262619
```

The sentence-aligned corpus now looks like this:

```
> head -9 corpus/en-de-int-train.snt
1
3469 5 2049
4107 5 2 1399
1
10 3214 4 116 2007 2 9 5254 1151 985 6447 2049 21 44 141 14 2580 3
14 2213 1866 2 1399 5 2 29 46 3256 18 1969 4 2363 1239 1111 3
1
7179
306
```

A sentence pair now consists of three lines: First the frequency of this sentence. In our training process this is always 1. This number can be used for weighting different parts of the training corpus differently. The two lines below contain word ids of the foreign and the English sentence. In the sequence 4107 5 2 1399 we can recognize *of* (5) and *the* (2).

GIZA++ also requires words to be placed into word classes. This is done automatically by calling the `mkcls` program. Word classes are only used for the IBM reordering model in GIZA++. A peek into the foreign word class file:

```
> head corpus/de.vcb.classes
!      14
"      14
#      30
%      31
&      10
'      14
(      10
)      14
+      31
,      11
```

## 2.2 Running GIZA++

GIZA++ is a freely available implementation of the IBM Models. We need it as a initial step to establish word alignments. Our word alignments are taken from the intersection of bidirectional runs of GIZA++ plus some additional alignment points from the union of the two runs.

Running GIZA++ is the most time consuming step in the training process. It also requires a lot of memory (1-2 GB RAM is common for large parallel corpora).

GIZA++ learns the translation tables of IBM Model 4, but we are only interested in the word alignment file:

```

> zcat giza.de-en/de-en.A3.final.gz | head -9
# Sentence pair (1) source length 4 target length 3 alignment score : 0.00643931
wiederaufnahme der sitzungsperiode
NULL ({} ) resumption ({} 1 ) of ({} ) the ({} 2 ) session ({} 3 )
# Sentence pair (2) source length 17 target length 18 alignment score : 1.74092e-26
ich erkläre die am donnerstag , den 28. märz 1996 unterbrochene sitzungsperiode
des europaischen parlaments fuer wiederaufgenommen .
NULL ({} 7 ) i ({} 1 ) declare ({} 2 ) resumed ({} ) the ({} 3 ) session ({} 12 )
of ({} 13 ) the ({} ) european ({} 14 ) parliament ({} 15 )
adjourned ({} 11 16 17 ) on ({} ) thursday ({} 4 5 ) , ({} 6 ) 28 ({} 8 )
march ({} 9 ) 1996 ({} 10 ) . ({} 18 )
# Sentence pair (3) source length 1 target length 1 alignment score : 0.012128
begruessung
NULL ({} ) welcome ({} 1 )

```

In this file, after some statistical information and the foreign sentence, the English sentence is listed word by word, with references to aligned foreign words: The first word `resumption ({} 1 )` is aligned to the first German word `wiederaufnahme`. The second word of `({} )` is unaligned. And so on.

Note that each English word may be aligned to multiple foreign words, but each foreign word may only be aligned to at most one English word. This one-to-many restriction is reversed in the inverse GIZA++ training run:

```

> zcat giza.en-de/en-de.A3.final.gz | head -9
# Sentence pair (1) source length 3 target length 4 alignment score : 0.000985823
resumption of the session
NULL ({} ) wiederaufnahme ({} 1 2 ) der ({} 3 ) sitzungsperiode ({} 4 )
# Sentence pair (2) source length 18 target length 17 alignment score : 6.04498e-19
i declare resumed the session of the european parliament adjourned on thursday ,
28 march 1996 .
NULL ({} ) ich ({} 1 ) erkläre ({} 2 10 ) die ({} 4 ) am ({} 11 )
donnerstag ({} 12 ) , ({} 13 ) den ({} ) 28. ({} 14 ) märz ({} 15 )
1996 ({} 16 ) unterbrochene ({} 3 ) sitzungsperiode ({} 5 ) des ({} 6 7 )
europaischen ({} 8 ) parlaments ({} 9 ) fuer ({} ) wiederaufgenommen ({} )
. ({} 17 )
# Sentence pair (3) source length 1 target length 1 alignment score : 0.706027
welcome
NULL ({} ) begruessung ({} 1 )

```

## 2.3 Align Words

To establish word alignments based on the two GIZA++ alignments, a number of heuristics may be applied. The default heuristic `grow-diag-final` starts with the intersection of the two alignments and then adds additional alignment points (see Figure 1).

Here, the pseudo code for the default heuristic:

```
GROW-DIAG-FINAL(e2f , f2e):
```

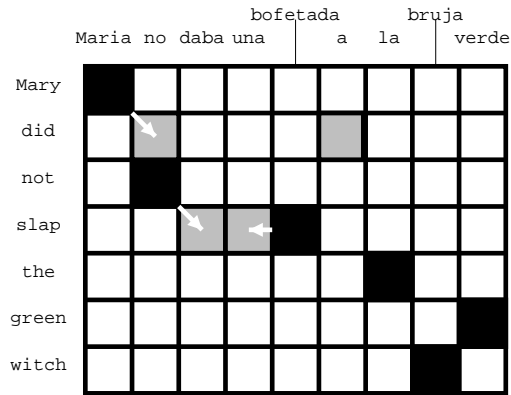


Figure 1: Intuition behind the growing heuristic for word alignment: Start with reliable alignment points, and add some less reliable neighboring points

```

neighboring = ((-1,0),(0,-1),(1,0),(0,1),(-1,-1),(-1,1),(1,-1),(1,1))
alignment = intersect(e2f,f2e);
GROW-DIAG(); FINAL(e2f); FINAL(f2e);

GROW-DIAG():
  iterate until no new points added
  for english word e = 0 ... en
    for foreign word f = 0 ... fn
      if ( e aligned with f )
        for each neighboring point ( e-new, f-new ):
          if ( ( e-new not aligned and f-new not aligned ) and
              ( e-new, f-new ) in union( e2f, f2e ) )
            add alignment point ( e-new, f-new )

FINAL(a):
  for english word e-new = 0 ... en
    for foreign word f-new = 0 ... fn
      if ( ( e-new not aligned or f-new not aligned ) and
          ( e-new, f-new ) in alignment a )
        add alignment point ( e-new, f-new )

```

To illustrate this heuristic, see the example in Figure 2 on the following page. We with the intersection of the two alignments for the second sentence in the corpus above (left side) and then add some additional alignment points that lie in the union of the two alignments (right side).

This alignment has a blatant error: the alignment of the two verbs is mixed up. resumed is aligned to unterbrochene, and adjourned is aligned to wiederaufgenommen, but it should be the other way around.

To conclude this section, a quick look into the files generated by the word alignment process:

```

==> model/aligned.de <==
wiederaufnahme der sitzungperiode
ich erklare die am donnerstag , den 28. maez 1996 unterbrochene sitzungperiode

```

<pre> #----- i -#----- declare ----- resumed --#----- the -----#----- session -----#----- of ----- the -----#----- european -----#----- parliament ----- adjourned ----- on ---#----- thursday ---#----- , -----#----- 28 -----#----- march -----#----- 1996 -----#----- . iedad,d2m1usdepfw. crimo e8a9nieuau hke n n.r9ttsrrre l n z6ez ol d a e ru pa e r r bn am r e s rg ie a t os sn u </pre>	<pre> #----- i -#----- declare -----#----- resumed --#----- the -----#----- session -----#----- of -----#----- the -----#----- european -----#----- parliament -----##----- adjourned ---#----- on ---##----- thursday ---#----- , -----#----- 28 -----#----- march -----#----- 1996 -----#----- . iedad,d2m1usdepfw. crimo e8a9nieuau hke n n.r9ttsrrre l n z6ez ol d a e ru pa e r r bn am r e s rg ie a t os sn u </pre>
---	---

Figure 2: Intersection (left) and grow-align-final (right) on a sentence pair from the Europarl corpus

```

des europaeischen parlaments fuer wiederaufgenommen .
begruessung

==> model/aligned.en <==
resumption of the session
i declare resumed the session of the european parliament adjourned on
thursday , 28 march 1996 .
welcome

==> model/aligned.grow-diag-final <==
0-0 0-1 1-2 2-3
0-0 1-1 2-3 3-10 3-11 4-11 5-12 7-13 8-14 9-15 10-2 11-4 12-5 12-6 13-7
14-8 15-9 16-9 17-16
0-0

```

The third file contains alignment information, one alignment point at a time, in form of the position of the foreign and English word.

## 2.4 Get Lexical Translation Table

Given this alignment, it is quite straight-forward to estimate a maximum likelihood lexical translation table. We estimate the  $w(e|f)$  as well as the inverse  $w(f|e)$  word translation table. Here are the top translations for europa into English:

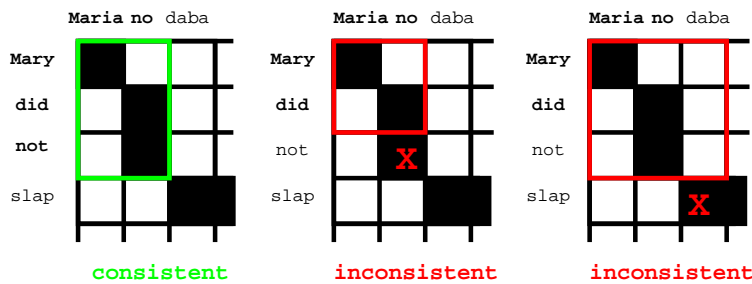
```

grep ' europa ' model/lex.f2n | sort -nrk 3 | head
europe europa 0.8874152
european europa 0.0542998
union europa 0.0047325
it europa 0.0039230
we europa 0.0021795
eu europa 0.0019304
europeans europa 0.0016190
euro-mediterranean europa 0.0011209
europa europa 0.0010586
continent europa 0.0008718

```

## 2.5 Extract Phrases

All phrase pairs that are consistent with the word alignment are collected. The definition of *consistent* can be best explained with an example:



In this graph, a phrase alignment is a box, and a word is either a row (English word) or a column (foreign word). For a phrase alignment to be consistent with the word alignment, all alignment points for rows and columns that are touched by the box have to be in the box, and not outside.

In the second example the alignment point no-not is outside the box, although the word no is touched by the box. Hence, the box is an inconsistent word alignment.

Mathematically, we can define the set of extracted bilingual phrase pairs  $BP$  as:

$$\begin{aligned}
 (\bar{e}, \bar{f}) \in BP \Leftrightarrow & \quad \forall e_i \in \bar{e} : (e_i, f_j) \in A \rightarrow f_j \in \bar{f} \\
 \text{AND} \quad \forall f_j \in \bar{f} : & \quad (e_i, f_j) \in A \rightarrow e_i \in \bar{e}
 \end{aligned}$$

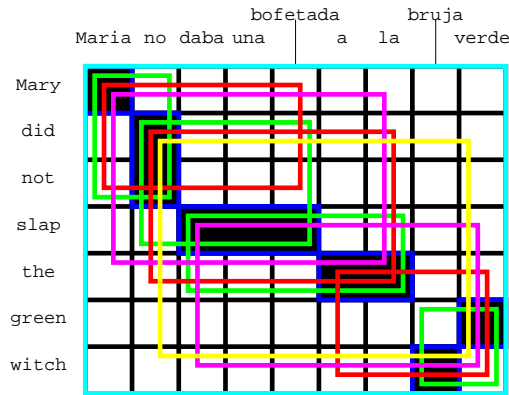
For an example, which phrases may be extracted, see Figure 3 on the next page.

In the phrase extraction step, all phrases are dumped into one big file. Here is the top of that file:

```

> head model/extract
wiederaufnahme ||| resumption ||| 0-0
wiederaufnahme der ||| resumption of the ||| 0-0 1-1 1-2
wiederaufnahme der sitzungperiode ||| resumption of the session ||| 0-0 1-1 1-2 2-3
der ||| of the ||| 0-0 0-1

```



(Maria, Mary), (no, did not), (slap, daba una bofetada), (a la, the), (bruja, witch), (verde, green), (Maria no, Mary did not), (no daba una bofetada, did not slap), (daba una bofetada a la, slap the), (bruja verde, green witch), (Maria no daba una bofetada, Mary did not slap), (no daba una bofetada a la, did not slap the), (a la bruja verde, the green witch), (Maria no daba una bofetada a la, Mary did not slap the), (daba una bofetada a la bruja verde, slap the green witch), (no daba una bofetada a la bruja verde, did not slap the green witch), (Maria no daba una bofetada a la bruja verde, Mary did not slap the green witch)

Figure 3: Phrase pairs extracted from an example

```

der sitzungperiode ||| of the session ||| 0-0 0-1 1-2
situngsperiode ||| session ||| 0-0
ich ||| i ||| 0-0
ich erklre ||| i declare ||| 0-0 1-1
erklre ||| declare ||| 0-0
situngsperiode ||| session ||| 0-0

```

The content of this file is for each line: foreign phrase, English phrase, and alignment points. Alignment points are pairs (foreign,english). Also, an inverted alignment file is generated (`extract.inv`).

## 2.6 Score Phrases

Subsequently, a translation table is created from the stored phrase translation pairs. The two steps are separated, because for larger translation models, the phrase translation table does not fit into memory. Fortunately, we never have to store the phrase translation table into memory — we can construct it on disk.

To estimate the phrase translation probability  $\phi(e|f)$  we proceed as follows: First, the extract file is sorted. This ensures that all English phrase translations for an foreign phrase are next to each other in the file. Thus, we can process the file, one foreign phrase at a time, collect counts and compute  $\phi(e|f)$  for that foreign phrase  $f$ . To estimate  $\phi(f|e)$ , the inverted file is sorted, and then  $\phi(f|e)$  is estimated for an English phrase at a time.

Next to phrase translation probability distributions  $\phi(f|e)$  and  $\phi(e|f)$ , additional phrase translation scoring functions can be computed, e.g. lexical weighting, word penalty, phrase penalty, etc. Currently, lexical weighting is added for both directions and a fifth score is the phrase penalty.

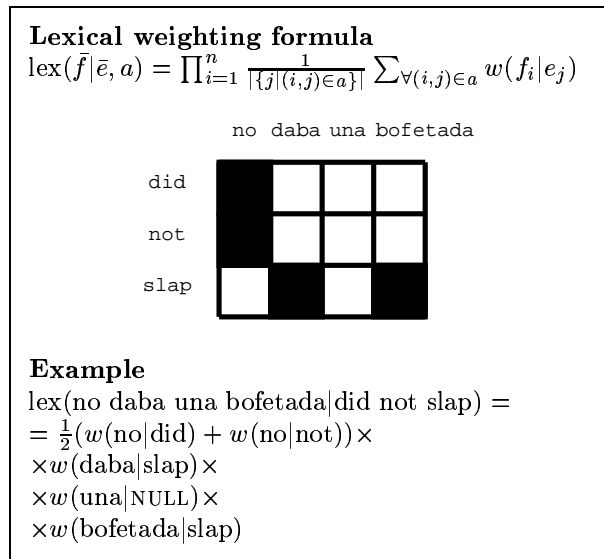


Figure 4: Lexical weighting as one method to score phrase translation pairs

Here some phrase translations for in europe:

```
> grep '| in europe |' model/phrase-table | sort -nrk 7 -t\| | head
in europa ||| in europe ||| 0.829007 0.207955 0.801493 0.492402 2.718
europas ||| in europe ||| 0.0251019 0.066211 0.0342506 0.0079563 2.718
in der europischen union ||| in europe ||| 0.018451 0.00100126 0.0319584 0.0196869 2.718
in europa , ||| in europe ||| 0.011371 0.207955 0.207843 0.492402 2.718
europischen ||| in europe ||| 0.00686548 0.0754338 0.000863791 0.046128 2.718
im europischen ||| in europe ||| 0.00579275 0.00914601 0.0241287 0.0162482 2.718
fr europa ||| in europe ||| 0.00493456 0.0132369 0.0372168 0.0511473 2.718
in europa zu ||| in europe ||| 0.00429092 0.207955 0.714286 0.492402 2.718
an europa ||| in europe ||| 0.00386183 0.0114416 0.352941 0.118441 2.718
der europischen ||| in europe ||| 0.00343274 0.00141532 0.00099583 0.000512159 2.718
```

Currently, five different phrase translation scores are computed:

1. phrase translation probability  $\phi(e|f)$
2. lexical weighting  $\text{lex}(e|f)$
3. phrase translation probability  $\phi(f|e)$
4. lexical weighting  $\text{lex}(f|e)$
5. phrase penalty (always  $\exp(1) = 2.718$ )

The formula for the lexical weighting that is used, is displayed in Figure 4. It uses word translation probabilities computed from a step 4.

## 2.7 Create Configuration File

As a final step, a configuration file for the decoder is generated with all the correct paths for the generated model and a number of default parameter settings.

This file is called

```
model/pharaoh.ini
```

You will also need to train a language model. This is described in the decoder manual.

## 3 Parameters for Training

This section is the core of the manual. We described the process of training a statistical phrase translation model, which was illustrated in the previous section. Now we will go into different options of running the training script `train-translation-model.perl`.

A number of support programs are required:

- `phrase-extract` and `phrase-score` are binaries that implement part of the training efficiently in C++. These programs are part of the distribution of this script.
- `GIZA++` is an implementation of the training method for various IBM Models. It is available at <http://www-i6.informatik.rwth-aachen.de/web/Software/GIZA++.html>
- `mkcls` and `snt2cooc.out` are support programs for `GIZA++`. They are available on the same web page.

You may have to adjust the hard coded paths to these scripts in the header of the Perl script.

The most expensive step in terms of both time and use of computer main memory is `GIZA++`. For current corpus sizes of roughly 100 million words, you will need 2-4 GB of RAM and up to a few days of CPU time.

### 3.1 Basic Options

A number of parameters are required to point the training script to the correct training data. We will describe them in this section. Other options allow for partial training runs and alternative settings.

As mentioned before, you want to create a special directory for training. The path to that directory has to be specified with the parameter `--root-dir`.

The root directory has to contain a sub directory (called `corpus`) that contains the training data. The training data is a parallel corpus, stored in two files, one for the English sentences, one for the foreign sentences. The corpus has to be sentence-aligned, meaning that the 1624th line in the English file is the translation of the 1624th line in the foreign file.

Typically, the data is lowercased, no empty lines are allowed, and having multiple spaces between words may cause problems. Also, sentence length is limited to 100 words per sentence. The sentence length ratio for a sentence pair can be at most 9 (i.e, having a 10-word sentence aligned to a 1-word sentence is disallowed). These restrictions on sentence length are caused by `GIZA++` and may be changed (see below).

The two corpus files have a common file stem (say, `euro`) and extensions indicating the language (say, `en` and `de`). The file stem (`--corpus-file`), and the language extensions (`--e` and `--f`) have to be specified to the training script.

In summary, the training script may be invoked as follows:

```
train-phrase-model.perl --root-dir . --f de --e en --corpus corpus/euro >& LOG
```

After training, typically the following files can be found in the root directory:

```

>ls -lh *
-rw-rw-r--  1 koehn  user          110K Jul 13 21:49 LOG

corpus:
total 399M
-rw-rw-r--  1 koehn  user          104M Jul 12 19:58 de-en-int-train.snt
-rw-rw-r--  1 koehn  user          4.2M Jul 12 19:56 de.vcb
-rw-rw-r--  1 koehn  user          3.2M Jul 12 19:42 de.vcb.classes
-rw-rw-r--  1 koehn  user          2.6M Jul 12 19:42 de.vcb.classes.cats
-rw-rw-r--  1 koehn  user          104M Jul 12 19:59 en-de-int-train.snt
-rw-rw-r--  1 koehn  user          1.1M Jul 12 19:56 en.vcb
-rw-rw-r--  1 koehn  user          793K Jul 12 19:56 en.vcb.classes
-rw-rw-r--  1 koehn  user          614K Jul 12 19:56 en.vcb.classes.cats
-rw-rw-r--  1 koehn  user           94M Jul 12 18:08 euro.de
-rw-rw-r--  1 koehn  user           84M Jul 12 18:08 euro.en

giza.de-en:
total 422M
-rw-rw-r--  1 koehn  user          107M Jul 13 03:57 de-en.A3.final.gz
-rw-rw-r--  1 koehn  user          314M Jul 12 20:11 de-en.cooc
-rw-rw-r--  1 koehn  user           2.0K Jul 12 20:11 de-en.gizacfg

giza.en-de:
total 421M
-rw-rw-r--  1 koehn  user          107M Jul 13 11:03 en-de.A3.final.gz
-rw-rw-r--  1 koehn  user          313M Jul 13 04:07 en-de.cooc
-rw-rw-r--  1 koehn  user           2.0K Jul 13 04:07 en-de.gizacfg

model:
total 2.1G
-rw-rw-r--  1 koehn  user           94M Jul 13 19:59 aligned.de
-rw-rw-r--  1 koehn  user           84M Jul 13 19:59 aligned.en
-rw-rw-r--  1 koehn  user          90M Jul 13 19:59 aligned.grow-diag-final
-rw-rw-r--  1 koehn  user         214M Jul 13 20:33 extract.gz
-rw-rw-r--  1 koehn  user         212M Jul 13 20:35 extract.inv.gz
-rw-rw-r--  1 koehn  user           78M Jul 13 20:23 lex.f2n
-rw-rw-r--  1 koehn  user           78M Jul 13 20:23 lex.n2f
-rw-rw-r--  1 koehn  user          862 Jul 13 21:49 pharaoh.ini
-rw-rw-r--  1 koehn  user         1.2G Jul 13 21:49 phrase-table

```

### Parameters

```

--root-dir - root directory, where output files are stored
--corpus - corpus, expected in $ROOT/corpus
--e - extension of the English corpus file
--f - extension of the foreign corpus file
--lm - language model file

```

## 3.2 Partial Training

You may have better ideas how to do word alignment, extract phrases or score phrases. Since the training is modular, you can start training at any of the seven training steps (`--first-step`) and end it at any subsequent step (`--last-step`).

Again, the seven training steps are:

1. Prepare data
2. Run GIZA++
3. Align words
4. Get lexical translation table
5. Extract phrases
6. Score phrases
7. Create configuration file

For instance, if you may have your own method to generate a word alignment, you want to skip these training steps and start with lexical translation table generation, you may specify this by

```
train-phrase-model.perl [...] --first-step 4
```

### Parameters

- `--first-step` – first step in the training process (default 1)
- `--last-step` – last step in the training process (default 7)

## 3.3 File Locations

A number of parameters allow you to break out of the rigid file name conventions of the training script. A typical use for this is that you want to try alternative training runs, but there is no need to repeat all the training steps.

For instance, you may want to try an alternative alignment heuristic. There is no need to rerun GIZA++. You could copy the necessary files from the `corpus` and the `giza.*` directories into a new root directory, but this takes up a lot of additional disk space and makes the file organization unnecessarily complicated.

Since you only need a new model directory, you can specify this with the parameter `--model-dir`, and stay within the precious root directory structure:

```
train-phrase-model.perl [...] --first-step 3 --alignment union --model-dir model-union
```

The other parameters for file and directory names fulfill similar purposes.

#### Parameters

```
--corpus-dir – corpus directory (default $ROOT/corpus)
--lexical-dir – lexical translation probability directory (default $ROOT/model)
--model-dir – model directory (default $ROOT/model)
--extract-file – extraction file (default $ROOT/model/extract)
--giza-f2e – GIZA++ directory (default $ROOT/giza.$F-$E)
--giza-e2f – inverse GIZA++ directory (default $ROOT/giza.$E-$F)
```

### 3.4 Alignment Heuristic

A number of different word alignment heuristics are implemented, and can be specified with the parameter `--alignment`. The options are:

- `intersect` – the intersection of the two GIZA++ alignments is taken. This usually creates a lot of extracted phrases, since the unaligned words create a lot of freedom to align phrases.
- `union` – the union of the two GIZA++ alignments is taken
- `grow-diag-final` – the default heuristic described in Section 2.3 on page 7
- `grow-diag` – same as above, but without a call to function `FINAL()` (see Section 2.3).
- `grow` – same as above, but with a different definition of `neighboring`. Now diagonally adjacent alignment points are excluded.
- `grow` – no diagonal neighbors, but with `FINAL()`

Different heuristic may show better performance for a specific language pair or corpus, so some experimentation may be useful.

#### Parameters

```
--alignment – heuristic used for word alignment: intersect, union, grow, grow-final, grow-diag, grow-diag-final (default)
```

### 3.5 Maximum Phrase Length

The maximum length of phrases is limited to 7 words. The maximum phrase length impacts the size of the phrase translation table, so shorter limits may be desirable, if phrase table size is an issue. Previous experiments have shown that performance increases only slightly when including phrases of more than 3 words.

#### Parameters

```
--max-phrase-length – maximum length of phrases entered into phrase table (default 7)
```

### 3.6 GIZA++ Options

GIZA++ takes a lot of parameters to specify the behavior of the training process and limits on sentence length, etc. Please refer to the corresponding documentation for details on this.

Parameters can be passed on to GIZA++ with the switch `--giza-option`.

For instance, if you want to change the number of iterations for the different IBM Models to 4 iterations of Model 1, 0 iterations of Model 2, 4 iterations of the HMM Model, 0 iterations of Model 3, and 3 iterations of Model 4, you can specify this by

```
train-phrase-model.perl [...] --giza-option m1=4,m2=0,mh=4,m3=0,m4=3
```

#### **Parameters**

`--giza-option` – additional options for GIZA++ training

## 4 Additional Programs

Finally, we describe a few additional scripts that perform useful task for training or decoding.

### 4.1 Data Preparation

The script `clean-corpus-n.perl` is small script that cleans up a parallel corpus, so it works well with the training script.

It performs the following steps:

- removes empty lines
- removes redundant space characters
- drops lines (and their corresponding lines), that are empty, too short, too long or violate the 9-1 sentence ratio limit of GIZA++

The command syntax is:

```
clean-corpus-n.perl CORPUS L1 L2 OUT MIN MAX
```

For example:

```
clean-corpus-n.perl raw de en clean 0 50
```

takes the corpus files `raw.de` and `raw.en`, deletes lines longer than 50, and creates the output files `clean.de` and `clean.en`.

### 4.2 Filtering the Phrase Table

The phrase translation tables has easily the size of a few giga bytes, which may make it impossible to load it into memory. Fortunately, for a given test set, only phrase pairs that have foreign phrases that actually occur in the test set, have to be considered by the decoder.

The script `run-filtered-pharaoh.perl` filters the phrase table and then runs the decoder on the reduced table.

The command syntax is:

```
run-filtered-pharaoh.perl FILTER-DIR PHARAOH CONFIG INPUT PARAMETERS
```

The script first creates a directory `FILTER-DIR` that contains the filtered phrase table, and a modified configuration file. The decoder binary has to be specified by `PHARAOH`, its configuration file by `CONFIG` and additional decoder parameters by `PARAMETERS`. The input file is `INPUT`.

For example:

```
run-filtered-pharaoh.perl fdir pharaoh pharaoh.ini test.f -dl 4
```

produces identical output to the call `pharaoh -f pharaoh.ini <test.f -dl 4` with less use of memory.

### 4.3 Generating N-Best Lists

Generating an n-best list of top translations for an input sentence, opposed to just the top 1 translations is useful for tasks as reranking (with additional features) and parameter tuning (see also below).

Unfortunately, n-best generation is a bit convoluted with the decoder Pharaoh, but this is remedied by the script `n-best-from-pharaoh.perl`. To run this script, you will also need the finite state machine toolkit `carmel`, which is available from <http://www.isi.edu/licensed-sw/carmel/>.

The path names pointing to Carmel and Pharaoh at the beginning at the script may have to be adapted.

The syntax of the command is:

```
n-best-from-pharaoh.perl CONFIG FILTERED-DIR WORKING-DIR DEV-F N PARAMETER OUTFILE
```

The script also invokes the previous script for filtering the translation table for a given test set. Directory paths for the filtered translation model `FILTERED-DIR` and a working directory `WORKING-DIR` have to be provided, in addition to the decoder configuration file `CONFIG`, additional parameters for the decoder `PARAMETER` the maximum size of the n-best list `N`, the set of foreign input sentences `DEV-F`, and the file name of the resulting file that contains the n-best list.

The script generates a file that looks like this:

```
0 0 ||| the events of the difficulties of the euro steile rise in oil prices
and at another level hope for democratic change in yugoslavia . ||| 0-0,0-1
1-1,2-2 2-2,3-3 3-6,4-6 7-7,7-9 8-8,10-10 9-12,11-13 13-13,14-14 14-15,15-16
16-16,17-18 17-18,19-21 19-21,22-24 22-23,25-26 ||| 0 -105.57 -48.9746
-55.0231 -14.4686 -27.5596 11.9988 -24
0 1 ||| the events of the difficulties of the euro steile rise in oil prices
and at another level hope for democratic change in yugoslavia . ||| 0-0,0-1
1-1,2-2 2-2,3-3 3-6,4-6 7-7,7-9 8-8,10-10 9-12,11-13 13-13,14-14 14-15,15-16
16-16,17-18 17-18,19-21 19-21,22-24 22-22,25-25 23-23,26-26 ||| 0 -105.57
-49.0295 -55.0231 -14.4612 -27.5596 12.9987 -24
0 2 ||| the events of the difficulties of the euro steile rise in oil prices
and other level hope for democratic change in yugoslavia . ||| 0-0,0-1
1-1,2-2 2-2,3-3 3-6,4-6 7-7,7-9 8-8,10-10 9-12,11-13 13-13,14-15
14-14,16-16 15-15,17-18 16-17,19-21 18-20,22-24 21-22,25-26 ||| 0 -103.535
-57.5445 -51.9465 -17.502 -23.1638 11.9988 -23
```

Each line contains a candidate translation, with information items separated by three bars (|||):

- the foreign sentence id and candidate translation id
- the text of the candidate translation
- the phrase alignment between foreign input sentence and candidate translation: this is specified as a list of span pairs (from-english – to-english , from-foreign – to-foreign)
- the values for different component model scores: distortion model, language model(s), translation models, and word penalty

## 4.4 Minimum Error Rate Training

The translation model has a number of model components (reordering model, language model, different phrase translation scoring methods, word penalty). Setting weights for these components is hard to do by hand, while finding good weights is essential for optimal translation performance.

The script `minimum-error-rate-training.perl` is an implementation of Och's minimum error rate training using the BLEU score. The Perl implementation is inefficient, slow, but it basically works. While we provide it with this software release, we encourage the implementation of alternative parameter tuning methods.

The syntax of the script is

```
minimum-error-rate-training.perl CONFIG WORKING-DIR TRACE DEV-F DEV-E N-BEST-LIST-SIZE  
BEAM PARAMETERS LAMBDA
```

The parameters are:

CONFIG Pharaoh configuration file

WORKING-DIR a sub directory to store intermediate files

TRACE translation output at each iteration, typically set to `$WORKING-DIR/trace`

DEV-F foreign development corpus, typically a few hundred sentences

DEV-E English reference translations for development corpus: if multiple reference translations exist, these have to split up into files `${DEV-E}0`, `${DEV-E}1`, `${DEV-E}2`, etc.

N-BEST-LIST-SIZE length of the n-best list generated at each iteration, typically 100-1000

BEAM beam size for the decoder, typically 100

PARAMETERS additional parameters for the decoder: these have to be provided in the form `'-d1 4 -b 0.03'`, i.e. in quotes

LAMBDA starting point and ranges for the weights, format is `(WEIGHT-NAME:(INITIAL:MIN-MAX)+)`, e.g.  
`"d:1,0.5-1.5 lm:1,0.5-1.5 tm:0.3,0.25-0.75;0.2,0.25-0.75;0.2,0.25-0.75;0.3,0.25-0.75;  
0,-0.5-0.5 w:0,-0.5-0.5"`

A very rich log file (to `STDOUT`) is generated by the script detailing BLEU scores at various stages of the training. The script terminates after a number of iterations (typically 10-20 iterations). The final best parameter setting can be found in the log file, or in the file name of the last trace file.

## Appendix: All Training Parameters

### Parameters

`--root-dir` – root directory, where output files are stored  
`--corpus` – corpus, expected in `$ROOT/corpus`  
`--e` – extension of the English corpus file  
`--f` – extension of the foreign corpus file  
`--lm` – language model file  
`--first-step` – first step in the training process (default 1)  
`--last-step` – last step in the training process (default 7)  
`--corpus-dir` – corpus directory (default `$ROOT/corpus`)  
`--lexical-dir` – lexical translation probability directory (default `$ROOT/model`)  
`--model-dir` – model directory (default `$ROOT/model`)  
`--extract-file` – extraction file (default `$ROOT/model/extract`)  
`--giza-f2e` – GIZA++ directory (default `$ROOT/giza.$F-$E`)  
`--giza-e2f` – inverse GIZA++ directory (default `$ROOT/giza.$E-$F`)  
`--alignment` – heuristic used for word alignment: intersect, union, grow, grow-final, grow-diag, grow-diag-final (default)  
`--max-phrase-length` – maximum length of phrases entered into phrase table (default 7)  
`--giza-option` – additional options for GIZA++ training