Problem 1. Working with GIZA++ [15 points]

This problem will focus on using the GIZA++ alignment tool, which implements a number of alignment models including Model 1, the HMM Model and Model 4. On patas, it is installed here:

```
/NLP_TOOLS/tool_sets/giza-pp/latest/GIZA++-v2
```

Start with the README file, though you don’t need to compile. If you run GIZA++ with no parameters, it will write a list of options to stdout.

**Data preparation**

A common representation for parallel data is simple text files, with one segment or sentence per line, and the same number of lines between languages, like clean.en and clean.es from your last assignment. Before training, this data needs to be tokenized and case-normalized (lowercasing is often sufficient).

For GIZA++, however, we need to present the data in a different format, where each vocabulary item has been replaced by a word ID. Additionally, GIZA++ wants as input a list of all the words that co-occur in the same sentence pair – these will be the word pairs that may have a non-zero parameter from Model 1. The tools plain2snt.out and snt2cooc.out generate the files necessary for GIZA++ if you’d like to run them yourself. The first tool, plain2snt.out, takes in a pair of text files, and generates a vocabulary for each, as well as a file where the sentences have been replaced by word IDs. The second tool, snt2cooc.out, identifies the words that co-occur in the same sentence pair. Run them with no arguments for usage.

For the test corpus (same 10K sentences from last week), though, I’ve already done this and stored the files in /dropbox/10-11/575SMT/HW3/prob1

Make a local copy of these files. Then you should be able to run a command like the following to invoke GIZA and then dump the perplexity report.

```
/NLP_TOOLS/tool_sets/giza-pp/latest/GIZA++-v2/GIZA++ config
  -S clean.en.vcb -T clean.es.vcb -C clean.en_clean.es.snt
  -coocurrencefile enes.cooc -o enes_out && cat enes_out.perp
```

You can edit the configuration file config to change the number of iterations of each model, and swap “es” and “en” to run in the other direction.
For the problem set, use GIZA++ to train a set of word alignment models, and look at the likelihoods and perplexities of each. To indicate iteration counts of particular models, we use superscripts: $1^7 H^5 4^3$ indicates 7 iterations of Model 1, followed by 5 iterations of the HMM model, followed by 3 iterations of Model 4.

(a) [5pts] Report the perplexity of each of the following nine training regimens. (Note: this should require only 3 GIZA++ runs: in the first run, specify 7 iterations of Model 1 and 0 iterations of the other models, then report only the intermediate perplexities $1^3, 1^5, 1^7$; $1^5 H^3, 1^5 H^5, 1^5 H^7$; $1^5 H^5 4^3, 1^5 H^5 4^5, 1^5 H^5, 4^7$

(b) [5pts] As we progress through iterations with the same model, what happens to the perplexity? Why?

(c) [5pts] What happens to perplexity when we shift from one model to the next (say Model 1 to HMM, or HMM to Model 4)? Why?

Problem 2. Decoding complexity [15 points]
Say we start with the following German sentence:

Wiederaufnahme der Sitzungsperiode

for which we find the following possible phrasal translations:

<table>
<thead>
<tr>
<th>Wiederaufnahme</th>
<th>der</th>
<th>Sitzungsperiode</th>
</tr>
</thead>
<tbody>
<tr>
<td>resumption</td>
<td>of</td>
<td>session</td>
</tr>
<tr>
<td>recovery</td>
<td>the</td>
<td>of session</td>
</tr>
<tr>
<td>resumption of</td>
<td>of</td>
<td>the</td>
</tr>
</tbody>
</table>

(a) [5pts] How many possible derivations are there, allowing all possible reorderings of the input? (Note that this question is asking about derivations, not translations. Multiple derivations may lead to the same string output.)

(b) [5pts] In a monotone search with sufficiently beams to prevent pruning of any output and a bigram language model, how many hypotheses would be constructed during search without hypothesis recombination?

(c) [5pts] How many hypotheses are constructed with hypothesis recombination?

Problem 3. Decoding with a phrase based system, Moses [20 points]
A trained moses system is available at: /dropbox/10-11/575SMT/HW3/prob3

You can run the moses decoder with the command

```
/NLP_TOOLS/mt_tools/moses/latest/moses-cmd/src/moses -f model/moses.ini < evaluation/test.es > evaluation/test.en.out
```

You can evaluate the quality of the model using BLEU scores with the script

```
/NLP_TOOLS/mt_tools/moses/latest/moses-scripts/scripts-20100308-1700/generic/multi-bleu.perl
```
(a) [5pts] Decode using different distortion limits: monotone, 1, 3, 5, and unlimited. Report BLEU score and timing. (The option “-dl X” indicates the distortion limit.)

(b) [5pts] Decode with different T table limits (-ttl). Try 5, 25, and 125. Report BLEU score and timing.

(c) [5pts] Decode using different stack size limits (-s). Try 5, 25, and 125. Report BLEU score and timing.

(d) [5pts] How does the running time depend on each of these factors?