Homework #2: CMPT-825

Anoop Sarkar – anoop@cs.sfu.ca

(1) **Statistical Machine Translation with EGYPT/GIZA++**: In this homework you will learn how to train a statistical machine translation system, view the alignments it learns from the training data and evaluate the output of the system after decoding some input text.

The particular system we will be using is the EGYPT system and the GIZA++ system which is an implementation of IBM model 4 (with some extensions).

The EGYPT toolkit was developed during a summer research workshop in 1999 at the Center for Language and Speech Processing at Johns Hopkins University (CLSP/JHU).

GIZA++ is an extension of the program GIZA (which was part of the SMT toolkit EGYPT). The extensions of GIZA++ were designed and written by Franz Josef Och.

These programs only run on linux.

First do the following steps to get the machine translation system running:

1. *Setting up your environment*: Put `/cs/natlang-a/packages/smt/tools/bin` in your PATH environment variable.
   - If using `tcsh`: `setenv PATH /cs/natlang-a/packages/smt/tools/bin:$PATH; rehash`
   - If using `bash`: `export PATH=/cs/natlang-a/packages/smt/tools/bin:$PATH`
2. *Training data*: Now to create the training data (a parallel corpus of aligned english text e and foreign text f). The particular translation task we will consider is translating from the archaic form of English used in some versions of the Bible to a more modern form of English:

   f: Abraham begat Isaac ; and Isaac begat Jacob ; and Jacob begat Judas and his brethren

   e: Abraham was the father of Isaac , and Isaac the father of Jacob , and Jacob the father of Judah and his brothers

   Other f languages available in the above directory are French, Spanish, Swedish and Tetun (East Timorese). As shown above, we will be using two forms of English as source text and foreign text. This is useful since this will allow you to understand the kinds of decisions and the kinds of errors made by MT systems. Since training is computationally expensive we'll create a subset of our training data (note that the system performance will be better with more training data). First, we create the training data for GIZA++ in your home directory or in `/tmp`:

   ```
   mkdir test
   cd test
   cp -r /cs/natlang-a/packages/smt/giza-lab/english .
   cd english
   head -3000 bible.train.english > run/bible.e
   head -3000 bible.train.foreign > run/bible.f
   ```

3. *Training GIZA++*: Statistical machine translation is modelled as follows:

$$P(\mathbf{e} \mid \mathbf{f}) = P(\mathbf{e}) \times P(\mathbf{f} \mid \mathbf{e}) \tag{1}$$

We will train the language model $P(\mathbf{e})$ separately from the translation model $P(\mathbf{f} \mid \mathbf{e})$. We will use GIZA++ to train the translation model from a parallel English-Foreign text corpus. GIZA++ uses IBM Model 4 (actually with some extensions) to train the translation model.

The trainer for GIZA++ takes three arguments, the source text, the *foreign* text and the directory where the output translation model is written. The trainer first runs a tool called `whittle` which pre-processes the training data in a form suitable for GIZA++. The trainer then runs the iterative EM training procedure (this will take quite some time to terminate, depending on the machine specs). Take a look at the printed output of the trainer to observe the various steps taken by the training process. In particular, observe the iterations of the EM algorithm over the training data.

```
cd run
runGIZA++.pl bible.e bible.f linux
```

This will create the output of training in the directory `linux`.

4. *Viewing trained alignments using Cairo*: We will visualize the alignments induced by GIZA++ using Cairo (written by Mike Jahr and Noah Smith, also part of the EGYPT system). First go to the `run` directory from the previous step. Now we can "cairoize" the statistical MT model (notice the `"."` after the command)

   ```
   sh do.cairoize .
   ```

   Then run the Java based word alignment viewer, Cairo (ignore any warnings):

   ```
   /cs/natlang-a/packages/smt/tools/cairo/run_cairo
   ```

   Use `File|Open` and go to the directory where you ran `do.cairoize` above. The alignments are stored in this directory in filenames of the type `username.date.pid` – open up the file that corresponds to your username. You will be able to see the alignments learned from the training data by GIZA++.

5. *Creating a Language Model with the CMU-Cambridge LM Toolkit*: Now that we have a trained translation model we can try to decode sentences from the foreign text into the source English text. Before we can run a decoder, we need to build a language model $P(\mathbf{e})$ in equation (1).

   We will use the CMU-Cambridge Language Modeling Toolkit to construct a LM. First, we create a vocabulary list which will be used as input to the LM toolkit (the backslash indicates the command extends across several lines but is actually one single command).

   ```
   cd test/english/run/lm
   perl voc.pl < \
     /cs/natlang-a/packages/smt/giza-lab/english/bible.train.english > \
     english.bible.voc
   ```

   Now we create a language model (an n-gram model of the text) using the CMU-Cambridge LM toolkit:

   ```
   perl /cs/natlang-a/packages/smt/tools/bin/runCMUToolkit.pl \
     /cs/natlang-a/packages/smt/giza-lab/english/bible.train.english \
      -d . -v english.bible.voc
   ```

6. *Translation from source to target using the ISI ReWrite Decoder*: Now we can translate from input f text to output e text using a decoder. First go to the `decode` directory:

   ```
   cd test/english/run/decode
   ```

The config file `isi-decoder.config` has to be edited. The top two lines of the config file should be modified as follows (note that you have to supply to full path to your home directory and you cannot use ~ instead):

```
LanguageModelFile = YOUR_HOME_DIR/test/english/run/lm/bible.train.english.binlm
TranslationModelConfigFile = YOUR_HOME_DIR/test/english/run/linux/tmconfig.cfg
```

Now we can translate from archaic English to modern English. Note that the input to the decoder is the "foreign" text (archaic English) and the output is modern English. To run the decoder and save the results in a file `output` run:

```
head -100 /cs/natlang-a/packages/smt/giza-lab/english/bible.eval.foreign | \
  isi-decoder.linux --config isi-decoder.config > english.out
```

This translates the first hundred sentences from the file `bible.eval.foreign`. Note that this will take a considerable amount of time (it's solving an NP-Hard problem, what do you expect). Examine the input to the translation and the output.

7. *Evaluating Translations with the BLEU metric*: We will now evaluate the output of the decoder using the *BLEU* metric. It provides a numerical estimate of the quality of the translation by comparing it with a *reference translation*. The comparison is done at the level of *n*-grams. The higher the BLEU score, the better the translation (according to the BLEU metric).

   First go the eval directory:

   ```
   cd test/english/run/eval
   ```

   As a sanity check, find the score of not translating the input at all.

   ```
   perl mteval-v09c.pl -r ref.sgm -s src.sgm -t src1.sgm -b
   ```

   Now score the translation output produced in the previous step.

   ```
   perl trg2sgml.pl ../decode/english.out > trg.sgm
   perl mteval-v09c.pl -r ref.sgm -s src.sgm -t trg.sgm -b
   ```

At last you are done with the entire process of statistical machine translation. Now to the questions:

a. Using Cairo, find at least three (3) distinct kinds of alignments found by the model in the training data that you think are incorrect. Explain for each one why you think it is an incorrect alignment.

b. Improve the quality of the output translations. You will need to create a new training set which based on some *transformation* that you think will improve translation quality. Then you will need to retrain the GIZA++ models, retrain the language models, and then decode to get new translations. Report your new improved BLEU scores. Here are some ideas to go on – you are encouraged to try your own ideas:

   - Based on alignments you saw using Cairo, merge certain multi-word Modern English phrases (e.g. `the father of` $\rightarrow$ `the_father_of`). Notice how this transformation takes care of the many to many mappings between e and f that are not natural in Model 4. Before evaluating the BLEU scores, you will need to remove the "_" from the decoder output.
   - Lowercase the training data to reduce vocabulary size. Note that your test set has to be lowercased as well.
   - Find the Porter stemmer from the web. Apply it to the archaic English in the training data. Again note that your test set needs to be consistent.