Minimum Error Rate Training Lab

Theory

- Friday's lecture
- Philipp Koehn's book (Chapter 9)
- Minimum Error Rate Training in Statistical Machine Translation by Franz Och - The original MERT paper.
- Improved Minimum Error Rate Training in Moses by Nicola Bertoldi, Barry Haddow, Jean-Baptiste Fouet, from this week's workshop. A description of the new implementation in moses.

Practical

Inner Loop

The new mert implementation is in the mosesdecoder svn project, in the subdirectory mert. To check out and build a copy run

```
svn co https://mosesdecoder.svn.sourceforge.net/svnroot/mosesdecoder/t
cd mert
```

This builds the `extractor` (for score statistics precomputation and feature extraction) and `mert` (which does the optimisation). Running each of these programs with a `--help` option will show the possible command-line arguments.

The `example/` directory contains some toy data - change to that directory and have a look at the different files. The file `NBEST` contains the decoder output, and the `REF*` files contain different versions of the references.

To run the `extractor` from the `example` directory, use the command

This creates two files: `FEATSTAT` contains the feature values, extracted from the n-best list, and `SCORESTAT` contains the statistics required to compute bleu (ngram matches).

The optimiser can then be run with

```
../mert --ifile init.opt --scfile SCORESTAT --ffile FEATSTAT -d 15
```

The command-line arguments specify the initial feature weights, the input files, and the number of dimensions. The optimiser will output the best weight set to the file `weights.txt` as well as to `stdout`. To see more detail about what the optimiser is doing, try increasing the verbosity level (using the `-v` switch) up to a maximum of 10. Also worth experimenting with is the `-n` switch, which specifies the number of random restarts to try. To change the scorer (i.e. the automatic translation metric), you'll need to rerun both the `extractor` and `mert` with the new `--sctype` options (try `PER`).

To see mert working with a larger data set, change to the mert/tests directory. This script relies on data stored on the statmt server, and runs several iterations of the outer loop using pre-prepared decoder output. To run one of the tests use the following commands

```
./testmert.py list
./testmert.py run fr-en-100
```

Whilst the test is running (the five iterations will take at least half-an-hour so you won't have time for the whole lot), look at the files in the data directory and try to follow the mert debug to see what it's doing.

Outer Loop

The outer loop is controlled using a perl script (`mert-moses-new.pl`) which lives in the `scripts/training` directory inside moses. Running it with the `--help` option displays the long list of command-line arguments. This script deals with the running of the decoder to create the n-best lists, and also with parallelisation of the decoder runs. A typical command line for the outer loop script looks like this:

```
./mert-moses-new.pl input reference $MOSES_HOME/moses-cmd/src/moses mo
```

The `$MOSES_HOME` environment variable points to the moses decoder directory, and `$SCRIPTS_ROOTDIR` is the directory in which the scripts were installed by the moses scripts `make release` command. The four positional arguments of the script specify the tuning set source sentences, the tuning set target sentences, the moses decoder executable and the moses config file, respectively. Also required are the locations of the moses scripts, the directory to use for intermediate files the command used to filter the phrase table and the location of mert. Without parallelisation tuning will take several days on a realistic size data set (eg. wmt shared task) and even with parallelisation of the decoder it can take of the order of a day.