MT LAB: Phrase-based Statistical Machine Translation (SMT)

Important Dates
- Thursday September 16th: MT LAB assignment released.
- Monday September 20th: Teams Formed. Send email to instructor and TA with team name and team members.
- Friday September 29th (noon): blind test set released.
- Monday October 4th (midnight): translation of test set due. Please send email to TA.

Introduction
In this laboratory exercise, you will build a complete phrase-based statistical machine translation system from small amounts of training data, evaluate their performance, and identify ways that translation quality can be improved. Resulting systems will be evaluated on test data (released a few days before the deadline). You will build the MT system using Moses, an open-source phrase-based statistical machine translation decoder.

Teams: You should work in groups of TWO. Please contact the TA and instructor if you can’t find a group. Please decide on a team as soon as possible. We expect all teams to be formed before Monday September 20th, 2010. Select a cool name for your team. Please send the instructor an email confirming your team. Students on the same team do not have to work together later on the same project or the language-in-10-minutes task (but could). Since the language pair is Arabic-English, we are requesting that Arabic speakers do not team up, but rather work with other students. That said, it is not clear if that will in fact be an advantage 😊.

PART I: BUILDING A BASELINE

Step 1: Setup the environment and workplace
You will need a computer science account. To sign up for one, go to the CRF website and then click on "Apply for an Account". Take a look at the computer and network use policy

Setting up your environment
Moses uses a set of scripts to support training, tuning, and other tasks. You will need to set up your environment to run these scripts by issuing the following commands:
export SCRIPTS_ROOTDIR=/home/ahmed/SMT/tools/moses-scripts/scripts-20100914-1516

export PATH=/home/ahmed/SMT/tools/srilm/bin/i686-m64:/home/ahmed/SMT/tools/srilm/bin:$PATH

Setting up your workplace
Create a directory where you will work on this lab and copy the training and evaluation data to it:

mkdir MT_LAB
cd MT_LAB
cp -r /home/ahmed/SMT/data .

Notes:

- Each student has a space quota so make sure that you have enough space for the lab work.
- Avoid working on compute[01 02 03] and use the other 5 machines compute[04 05 06 07 08].
- As a good practice, please keep a copy of all the commands that you ran and all the log files. This will allow us to help you debug problems.

Step 2 - Build a SMT Model
We’ll use about 35k sentences of Arabic-English parallel data. This should be good enough for moderate quality but still be doable in a reasonable amount of time.

The Arabic data is transliterated using a modified version of Buckwalter Transliteration scheme (BW) – See also Introduction to Arabic NLP: http://dx.doi.org/10.2200/S00277ED1V01Y201008HLT010 (Table 2.1 and 2.2). This will help the non-native speakers to visualize the data and avoid dealing with different encodings. Here is an example of what Arabic text looks like naturally and what transliteration we use:

WSYNHY ALRQYS JWLT BYARP ILY TRKY.

As a Columbia student, you should have access to this online book (no need to buy it). If you have problems, let us know.
You can try first working with quarter of the data to make sure that everything is working fine, and then add more data. That would also help solve the learning curve exercise (see PART IV).

**Prepare Data**
Filter out long sentences

```
/home/ahmed/SMT/tools/moses-scripts/scripts-20100914-1516/training/clean-corpus-n.perl data/Train/Train_data_ar_en data/Train/Train_data.clean 1 100
```

This ensures that only sentences of length 1-100 are selected for training. In this case, we lose almost 882 sentences. Think about what that means <wink wink>.

**Build a Language model (LM):**
The language model captures the properties of a language and provides the probability of seeing a target language string. The language model data should include the target side of the training data in addition to much more monolingual data of the target language.

Concatenate the training data and the monolingual data to build the language model

```
cat data/LM/LM_data.en data/Train/Train_data.en > data/LM/LM_data+Train_data.en
```

We will use the SRILM toolkit to build a tri-gram language model.

```
mkdir -p work/LM
/home/ahmed/SMT/tools/srilm/bin/i686-m64/ngram-count -order 3 -interpolate -kndiscount -unk -text data/LM/LM_data+Train_data.en -lm work/LM/LM_data+Train_data.en.lm
```

**Train Phrase Model**
Moses’ toolkit does a great job of wrapping up calls to mkcls and GIZA++ inside a training script, and outputting the phrase and reordering tables needed for decoding. The script that does this is called `train-model.perl`.

We'll run this in the background and nice it since it'll peg the CPU while it runs. It may take up to two hours. (Note: training on quarter of the training data will take around half an hour)

```
```
You can `tail -f work/training.out` to watch the progress of the tuning script.

Now would be a good time to look at what we've done.

```bash
ls work
corpus giza.en-ar giza.ar-en LM model
```

**Step 3 - Tuning**

Note that this step can take many hours, even days, to run on large phrase tables and tuning sets. We'll use a small tuning set to speed up the process. The training script controls for large phrase and reordering tables by filtering them to include only data relevant to the tuning set (we'll do this ourselves for the test data later).

```bash
mkdir -p work/tuning
```

**Insert weights into configuration file**

```bash
./home/ahmed/SMT/tools/scripts/reuse-weights.perl
work/tuning/mert/moses.ini < work/model/moses.ini > work/tuning/moses-tuned.ini
```

**Step 4 - Filtering Test Data**

Filtering helps in reducing memory requirements. It makes smaller phrase and reordering tables that contain only entries that will be used for a particular test set. Moses has a script that does this for us, which we'll apply to the evaluation test set:

```bash
./home/ahmed/SMT/tools/moses-scripts/scripts-20100914-1516/training/filter-model-given-input.pl work/evaluation/filtered
work/tuning/moses-tuned.ini data/Test/Test_data.mt05.src.ar
```

**Step 5 - Run Tuned Decoder on Development Test Set (dev test)**

```bash
nohup nice /home/ahmed/SMT/tools/moses/moses-cmd/src/moses -config
work/evaluation/filtered/moses.ini -input-file
data/Test/Test_data.mt05.src.ar 1> work/evaluation/Eval.tuned-filtered.output 2> work/evaluation/tuned-filtered.decode.out &
```

`work/evaluation/Eval.tuned-filtered.output` is the output file.
Step 6 - Evaluation

Wrap the output in XML

```
/home/ahmed/SMT/tools/scripts/wrap-xml.perl
data/Test/Test_data.mt05.ref.ar.xml en my-system-name <
work/evaluation/Eval.tuned-filtered.output >
work/evaluation/Eval.tuned-filtered.output.sgm
```

Score with NIST-BLEU

```
/home/ahmed/SMT/tools/evaluation-scripts/mteval-v11b.pl -s
data/Test/Test_data.mt05.src.ar.xml -r
data/Test/Test_data.mt05.ref.en.xml -t work/evaluation/Eval.tuned-
filtered.output.sgm -c
```

The BLEU score is a precision-based evaluation system to assess the quality of the translation.

PART II – SIMPLE BASELINE IMPROVEMENTS

Step 7 - Improving the system

Now we try to improve your Arabic-English translation system.

Preprocess the data

- Tokenize sets

```
/home/ahmed/SMT/tools/scripts/tokenizer.perl -l en <
data/Train/Train_data.en > data/Train/Train_data.tok.en
```

This process separates all the punctuations from the words. Repeat the same command for the tuning and monolingual English data.

- Lowercase sets

```
/home/ahmed/SMT/tools/scripts/lowercase.perl <
data/Train/Train_data.tok.en > data/Train/Train_data.tok.lowercased.en
```

Repeat the same command for the tuning and monolingual English data.

**Note:** make a copy of the Arabic files and rename them to match the counter English files to be able to run Moses commands.

Do not lowercase the Arabic text! The transliteration we use is case sensitive.

Repeat Step 2 to Step 5 on the new preprocessed data

Step 8 – Evaluating the Improved system

Train Recaser

Now we’ll train a recaser. It uses a statistical model to "translate" between lowercased and cased data.
mkdir -p work/recaser

/home/ahmed/SMT/tools/moses-scripts/scripts-20100914-1516/recaser/train-recaser.pl -train-script
/home/ahmed/SMT/tools/moses-scripts/scripts-20100914-1516/training/train-model.pl -ngram-count
/home/ahmed/SMT/tools/srilm/bin/i686-m64/ngram-count -corpus data/LM/LM_data+Train_data.tok.en -dir /home/<your account>/MT_LAB/work/recaser -scripts-root-dir

This goes through a whole GIZA and LM training run to go from lowercase sentences to cased sentences. Note that the -dir flag needs to be absolute.

Recase the output

/home/ahmed/SMT/tools/moses-scripts/scripts-20100914-1516/moses -cmd/src/moses > work/evaluation/Eval.tuned-filtered.output.recased

Detokenize the output

/home/ahmed/SMT/tools/scripts/detokenizer.pl -l en < work/evaluation/Eval.tuned-filtered.output.recased > work/evaluation/Eval.tuned-filtered.output.detokenized

Repeat Step 6 for evaluation

PART III – OPEN CHALLENGE

Now you should try to improve your Arabic-English translation system. This will be the system used for the competitive evaluation. Here are some ideas to get you started, but be as creative as you want:

- There are a variety of training options that can be modified (see Moses’ documentation).
- Preprocess the source language (Arabic). Hint: Arabic orthographic normalization or Arabic decliticization.
- Postprocess the translation output (English) to correct certain common mistakes.
- For more ideas consider papers on Arabic MT to be discussed in the class or that have been published on the MT archive.

PART IV – FINAL REPORT EXERCISES

Report on results of all four exercises below in your final report. Note that these are not all equal in the amount of effort needed. Please use the models and settings of the baseline when answering these exercises.
1. Use the alignment files to illustrate through a diagram the alignment between an Arabic and English sentence. Pick a sentence you like (not less than 10 words long though).
2. Pick the most ambiguous five words in translating from X-to-Y and the opposite? are they the same and why? (Hint: look at the phrase table and lexical translation tables).
3. Demonstrate the effect of training data size on MT quality (as judged by Bleu): create a learning curve by building 3 different systems with quarter, half and all the training data then report the evaluation BLEU score of each system instance. Beware that the language model should be the same across all systems using all monolingual data and all training data. What insights do you get from this exercise?
4. If this lab were about translating from English to Arabic, what components would remain the same and what components would change or be added/removed? [Do not implement; just describe]

**Deliverables**

The following are the deliverables for this lab. The deliverables will be graded and they will account for 10% of class grade. Although the performance on the open challenge does not affect the lab grade; members of the top three performing teams will receive a 5% bonus (of the class grade) each.

- The output of translating the dev test file, the tuned moses-tuned.ini and BLEU score for: (a) the baseline system (PART I), (b) the improved baseline system (PART II), and (c) the best system you built for the open challenge (PART III).
- The output of your best system on the blind test data (released a few days before the deadline)
- Answers to the exercises in PART IV (using the baseline system).
- Explanation of at least two attempts to improve the quality of your system in the open challenge section (PART III) including intuition and technical solution. Please report on positive and negative results (results where you did not get an improvement although you thought you could). Use an actual before-and-after (improved-baseline versus best open challenge system) example from your translation output to demonstrate the improvement.

**Delivery instructions**

**Format:** for the write-up, we highly recommend .pdf, but you’re allowed to send us a .txt file. DO NOT send us a file in .doc or .docx. You can keep the format simple (question number / answer) but write your team name and team member names and UNIs at the top.

**Submit What:** submit the write-up, and all files described above.
Submit How: zip or tar everything in a file called <YOUR_TEAM_NAME>_MTLAB.zip (or .tar.gz). For example, the team Babylon's submission would be babylon_MTLAB.zip. Go to Courseworks, 'class files', then click on 'post file', and choose the subfolder 'Shared Files'/ 'MT Lab Assignment'. Do not submit in another folder.

Late Submissions: There won't be any late submission, after the deadline we'll make the folder invisible to you.

Questions: Finally, if you have general questions concerning the homework, please use Courseworks 'discussion'.