Statistical Machine Translation

Lab Exercise 1: Steps in Moses

Diagram of the Moses SMT Pipeline

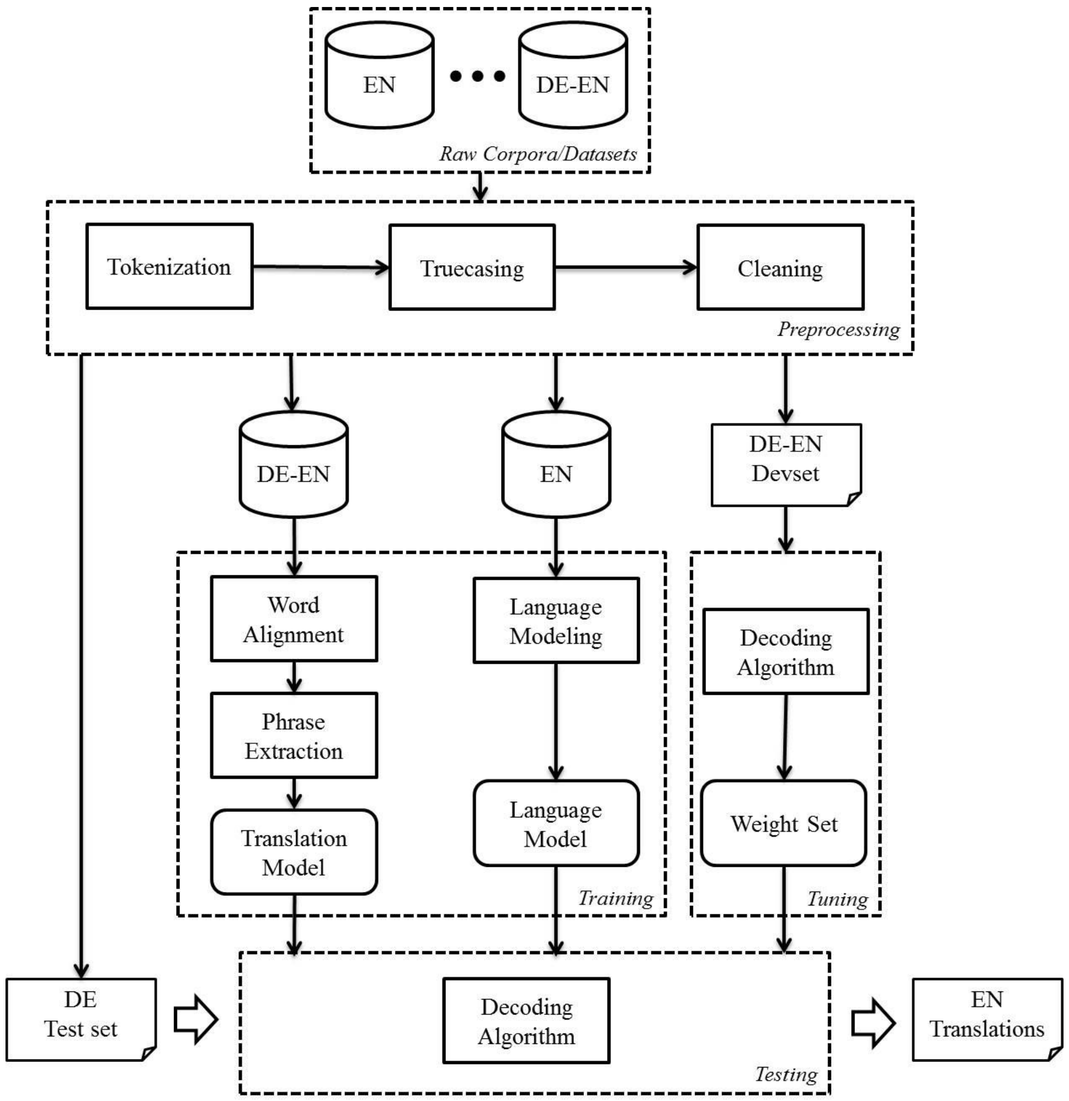
Source language: German (DE)

Target language: English (EN)

Parallel corpus: German-English (DE-EN)

Devset: development set for tuning process

Test set: test set for testing process



Reminder

If a command contains a backslash\
this means that the command is all one\
line, and the backslash should not be\
typed.

Keep the Command-line Cheat Sheet handy (accessed via Loop).

Step 0: Setting up

Download the corpus from Loop, and save it in a folder 'Corpus'.

You will can export directories as variables if you wish ($Corpus, $Moses, $Me, $KENLM) e.g.

```
export Moses="/usr/local/moses3"
export Corpus="your-home-directory-path/folder-which-contains-my-corpus"
```

If you are unsure of the exact path, move into that directory and use the pwd command.

Step 1: Preprocessing

1.1: Tokenisation

The following code segment tokenises the English corpus. Tokenisation is a process in which spaces are inserted between words and punctuation. E.g. after tokenisation "Hello, World!" would become " Hello , World ! "

```
$Moses/scripts/tokenizer/tokenizer.perl -l en\
< $Corpus/train-data.en > $Corpus/train-data-token.en
```

The **-l** flag allows the user to specify which language is being modelled, in this case English (en). These 2-letter language codes used are knows as ISO 639-1 Codes, and can be viewed here:

https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes

Investigate if this step is successful by viewing the contents of train-data-token.en, and comparing them to the untokenised version (train-data.en)

```
head $Corpus/train-data.en $Corpus/train-data-token.en
```

Repeat the tokenisation steps using train-data.fr, and the test and tune data, so that all our data is tokenised.

Q: Some languages (Chinese, Farsi, Arabic, Turkish etc.) have different writing systems and different alphabets. How should we tokenize such languages?

More information:

http://www.statmt.org/moses/?n=FactoredTraining.PrepareTraining

1.2: Cleaning

The following code segment 'cleans' the tokenised data, i.e. makes it fit for use in Moses. The script clean-corpus-n.perl removes empty lines (and their corresponding lines) and unnecessary spaces. It also removes lines that are too long or violate the 9:1 sentence ratio limit. (i.e. because it is unlikely that a sentence containing 9 words is a translation of a sentence containing 1 word)

```
$Moses/scripts/training/clean-corpus-n.perl\
$Corpus/train-data-token fr en \
$Corpus/train-data-clean   1 50
```

As in the tokenisation step fr and en refer to the languages in question. The number 1 is the minimum length a sentence can be, and 50 is the maximum. If a sentence is less than 1 word or more than 50 words in length it (and its corresponding translation) will be removed.

Q: Why do we need to clean the parallel corpus before training step?

Step 2: Language Modelling

A language model (LM) is used for modelling the target language. It does not learn any new translations, instead it attempts to represent the grammar of the target language, based on the monolingual data.

Some famous language modelling toolkits:

1. IRSTLM: https://github.com/luispedro/irstlm
2. SRILM: http://www.speech.sri.com/projects/srilm/
3. KenLm: https://kheafield.com/code/kenlm/

A LM is always trained for the target language, i.e. the language we want to translate into. In this case, it is English. In order to train a language model (using KENLM) the following code segment is used:

```
$Moses/bin/lmplz -o 3 < $Corpus/train-data-token.en > lm.arpa.en
$Moses/bin/build_binary lm.arpa.en lm.binary.en
```

The first line builds the language model. `-o 3` indicates which we are training a 3-gram model. The second line binarises it for speed.

Q: In our experiment we trained a tiny language model which is not the case for real-world problems. How can we cope with the space and speed issues in training huge language models?

More information:

https://kheafield.com/papers/avenue/kenlm.pdf

Step 3: Training a Translation Model

A translation model (TM) is used to figure out the translations from our source language (in this case French) to our target language (in this case, English). To create a TM for a specific language pair, we say that we 'train' a TM. In order to train a translation model in Moses we need a parallel corpus (i.e. a collection of sentences and their exact translations).

To train a TM using our train data, the following code segment is used:

```
nohup nice $Moses/scripts/training/train-model.perl \
-root-dir train -corpus $Corpus/train-data-clean -f fr -e en\
-alignment grow-diag-final-and -reordering msd-bidirectional-fe\
-lm 0:3:$Corpus/lm/lm.arpa.en:8 -external-bin-dir $Moses/training-tools\
>& training.out
```

`-f` is the flag for the source language (language we a translating *from*)

`-e` is the flag for the target language (language we a translating *into*)

`-external-bin-dir` is where directory where the alignment tools are stored

The trained model will be stored in this newly created directory: `train`

After training, the `train` directory will include the `corpus`, `giza.fr-en`, `giza.en-fr` and `model` folders. The first three folders cd ../are related to the alignment phase. The `model` folder includes tables and the Moses config file (`moses.ini`).

Q: Try to understand the format and content of `lex` files, `phrase-table.gz` and `moses.ini` in `model`.

Q: If you were to train a real translation engine for your language what challenges might you have?

Q: Compare the BLEU score achieved by your trained system to the BLEU score achieved by Google Translate for the same data. Manually compare the actual translations, i.e. look at them!

More information:

http://www.statmt.org/moses/?n=FactoredTraining.TrainingParameters

Reference:

http://www.statmt.org/moses/manual/manual.pdf