Using TM Cleaner with Fastalign

In this modality the software aligns the source and target segments at word level using Fastalign and computes a set of features based on alignments.
Warning: Your test file should not contain tags (e.g. sequences like <something>). If it has tags then first replace (< with < or nothing and > to > or nothing)

Prerequisites

To use TM Cleaner in this modality you need to download and install MMT (Modern Machine Translation- www.modernmt.eu). The installing instructions are very easy: practically you need to download the distribution, check you have java 8 , un-archive the distribution and set some variables
This version of TM Cleaner works with MMT 0.13:
https://github.com/ModernMT/MMT/releases/download/v0.13/mmt-0.13-ubuntu14_04.tar.gz

1. Download this version and un-archive it :
    a. tar -xvzf mmt-0.13-ubuntu14_04.tar.gz

A directory called mmt will be created.

2. Export the classpath to mmt jar file inside the mmt distribution (e.g. in .bash_profile add) :
    a. export CLASSPATH=.:/PATH_TO_MMT/build/mmt-0.13.jar

From the MMT we use a Fastalign version that allows training a model on parallel corpora. The trained model can then be used to align parallel texts.
From MMT we also use a very fast java tokenizer that has support for 45 languages.

Training Fastalign models

Here I show how to train the forward and backward Fastalign models. For this tutorial we provide a Fastalign model trained on 100 millions words parallel corpus (see Resources for details)

Assume you have an English and Italian parallel corpus:

1. The English file is "input.en.txt"
2. The Italian file is "input.it.txt"

The format of each text files is one sentence per line. The sentence in English at line "*i*" and the sentence in Italian at the line "*i*" are translations.

First: tokenize each file. After tokenization the tokens will be separated by space. For this you can use the tokenizer in MMT.

1. Tokenize the English and Italian files with MMT tokenizer:
    a. *java -cp build/mmt-0.13.jar -Dmmt.home=/path/to/mmt eu.modernmt.cli.PreprocessorMain --no-tags --print-placeholders --lang en input.en.txt > tokenized.en*
    b. *java -cp build/mmt-0.13.jar -Dmmt.home=/path/to/mmt eu.modernmt.cli.PreprocessorMain --no-tags --print-placeholders --lang it input.it.txt > tokenized.it*

2. Obtain the input file for Fastalign (*en-it-FA.txt*). Basically you merge the tokenized files above, taking care that the merged file will have the following format (please notice the space separating the separator "|||" from the last source and the first target token):
   First English : Tokenized Sentence . ||| Parallel Italian : Tokenized Sentence .
   Second English : Tokenized Sentence . ||| Parallel Italian : Italian Tokenized Sentence .

3. Run Fastalign and obtain the forward and backward models.
    a. Forward Model (the model to be obtained is called *model.fwd.bin*)
       fast_align -d -v -o -B -p model.fwd.bin –i en-it-FA.txt
    b. Backward Model (the model to be obtained is called *model.bckp.bin*)

```
fast_align –r -d -v -o -B -p  model.bckp.bin –i en-it-FA.txt
```

Configuration Files

1. *Parameters/p-FastAlign.txt*. The file contains three parameters:
    a. The full path to the Fastalign executable. The Fastalign executable is under /PATH_TO_MMT/opt/bin/fastalign-maurobuild
    b. The flags to be passed to Fastalign for forward alignment
    c. The flags to be passed to Fastalign for backward alignment
2. *Parameters/p-tokenizer.txt*. The file contains three parameters:
    a. *mmt*.home is the path to the your mmt installation. Please, edit this parameter
    b. *tok*.class is the class of the MMT tokenizer
    c. *tok.flags* are some flags to be passed to the MMT tokenizer

To run the tutorial you should provide the full path to your Fastalign executable and the path to your mmt installation.

Resources

The resources that can be used with this tutorial are following ones:

1. An *English-Italian* scikit-learn model trained on a sample of English-Italian positive and negative bi-segments extracted from MyMemory:
   Training/Fastalign/ full-English-Italian-Features.csv
2. Two *English-Italian* Fastalign models (forward model and backward model) trained on a 100 millions parallel corpus. You should download them from here:
   https://drive.google.com/open?id=0ByeL7E9Xcb3weWNrOTZsTGNqaEk.
   Un-archive the file and copy the models inside the folder *AlignmentModels*.
3. An *English-Italian* test file obtained from automatically aligning a web site containing English and Italian parallel documents. The file contains positive and negative segments.

a. The file to be classified: *Resources/Examples/Fastalign/about-small-en-it.txt*
b. The file annotated with correct labels for evaluation: *Resources/Examples/Fastalign/about-small-en-it-annotated.txt*

Training:

The configuration parameters for training are in "*Parameters/Fastalign/p-Training-XXX.txt*" files. You should copy and edit the corresponding file to fit your purposes.

```
python generateFeaturesAndClassify.py --features --config
Parameters/Fastalign/p-Training-Italian.txt
```

For this tutorial you do not need to train: we did the training for you and obtained the model presented in the previous section.

Classification:

English-Italian example.

1. Copy the English Italian test file inside the "TestFiles" directory taking care that the TestFiles directory is empty.
   a. cp Resources/Examples/Fastalign/about-small-en-it.txt TestFiles/
2. Classification using the default algorithm "SVM with linear kernel"
   a. `python generateFeaturesAndClassify.py --classify --config Parameters/Fastalign/p-Batch-Italian.txt`
3. Classification using the algorithm "Logistic regression" with the default class 0 and the threshold 0.7 (To see what this means read the configuration file)
   a. `python generateFeaturesAndClassify.py --classify --config Parameters/Fastalign/p-Batch-Italian.txt --mlalgorithm LogisticRegression`

Evaluation:

To see how well the algorithms performed look inside the directory:
“Resources/Examples/Fastalign/Evaluation”.
To perform the evaluation and know about each file returned by the evaluation script read the Evaluation tutorial.