AIR-based light clients for supporting Moses engine training

Jeffrey Rueppel
Adobe Systems Inc.
345 Park Avenue
San Jose, CA USA
jrueppel@adobe.com

Li Jiang
Adobe Systems Inc.
345 Park Avenue
San Jose, CA USA
ljiang@adobe.com

Gong Yu
Adobe Systems Inc.
345 Park Avenue
San Jose, CA USA
ygong@adobe.com

Ray Flournoy
Adobe Systems Inc.
345 Park Avenue
San Jose, CA USA
flournoy@adobe.com

Abstract

The Moses open source machine translation system is a powerful research tool. However, for doing machine translation at a production level Moses remains a work in progress. Adobe Systems Inc. has been working on completing a set of lightweight AIR[1] based tools to address these limitations. We have created a set of tools to improve corpus production, to speed up automatic training, and to carry out automatic scoring of Moses driven MT projects. In the future these three tools will be the building blocks for producing a fully integrated and automatic Moses based machine translation system.

1 Introduction

Building and training machine translation systems for research has become easier as the Moses machine translation system has matured. At Adobe we have been investigating what the limitations of Moses are and working to develop tooling which will make it possible to use Moses in a production environment. The hurdles to be addressed in order to use Moses for production ton an industry level scale can in general be reduced to those answering those questions which address the stability and efficiency of Moses. I.E. The technological level of Moses, and the quality of the machine translation that can be produced is high enough now to meet industry needs, but integrating Moses into an existing linguistic production processes is the crux of the problem. To address these needs we have developed a trio of lightweight client tools. First, we built a Corpus Cleaning tool, which processes .tmx[2] files into a Moses-ready format. Second, we built a training harness to simplify and speed up engine training. Third and finally we've put together a simple scoring harness, which can call automatic scoring routines to evaluate the quality of the engine being used. These tools have been quickly produced using Adobe Air technology for ease of installation and the ability to run anywhere. When taken all together, these 3 tools become the building blocks of a larger system. We will be using them to put in place a fully integrated and automated engine training and deployment schema for doing machine translation with Moses.

The Adobe Moses tooling set (Corpus Tool, Training Harness, Testing Tool) will install with minimal system requirements onto MacOS and Linux. It can be run in a client mode from the local machine, and will allow the user to connect directly to a previously installed Moses setup to answer translation requests.

[1] Adobe AIR, is a cross- platform runtime environment developed by Adobe Systems for building Rich Internet Applications (RIA) using Adobe Flash, Adobe Flex, HTML, and AJAX that can be run as desktop applications.

[2] TMX (Translation Memory Exchange) is an open XML standard for the exchange of translation memory data created by computer-aided-translation and localization tools. The tool could be easily configured to handle different input file types.

2 Corpus Cleaning Tool

Moses and other MT systems require aligned bi-bilingual corpus for training. MT engine developers frequently have access to legacy translations from previous localization cycles. These resources can be used to train MT engines if properly filtered before hand. To make legacy translation memory useful for MT training we have built a lightweight, and configurable tool for handling corpus cleaning. In this way we can quickly remove noise and less useful data from a linguistic resource to achieve a higher quality engine after training. A user should be familiar enough with Moses to have it already installed, but beyond that the tool is ready on installation to run. In its current form the Corpus Cleaning Tool takes an industry standard translation memory exchange file .tmx as input and produces a pair of flat, Moses ready, aligned, monolingual files as output. (In the future other file types, .xliff for example, will also be available to the user).

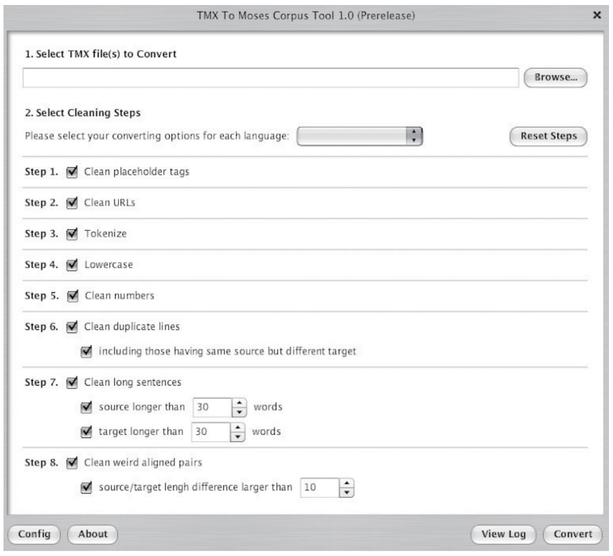
2.1 Current corpus cleaning options

Not all legacy corpuses are the same and there are many ways a corpus developer might wish to clean them in preparation for MT training. While we have established a set of options and ordered them to be effective, there remain many cleaning options that might be useful against different corpus. It might also improve the quality of the corpus to reorder the sequence of cleaning options. With this in mind we built the corpus tool in a modular fashion. Users may select which options they want to invoke. Users may reorder the execution of the selected options however they like. Finally we are including provisions by which users may define their own cleaning options using regular expressions. Current options on the Corpus Cleaning Tool include:

1. Placeholder handling
2. URL cleaning
3. Tokenization
4. Casing
5. Cleaning Numbers
6. Cleaning Duplicate Entries
7. Cleaning Long Entries
8. Cleaning Misaligned Translation Units.

The current options are a mixture of those standard to the normal requirements of preparing data for MT engine training and of those specific to cleaning the legacy data files we had to work with. For example, options such as “Placeholder Cleaning” are specific to working with data derived from software localization while the “Cleaning Misaligned Units” could be expected of any data set where data formatting has become corrupted.

In addition to the options listed below we have provided access points, which allow the user to install or develop their own language specific parsers. By default, the Corpus Cleaning Tool uses a Moses provided standard Western European language parser but it can also be retargeted to use Simplified Chinese, Japanese or other double byte language parsers should they be required. The corpus tool greatly reduces corpus cleaning production time allowing a single engineer to multiply their effectiveness by 5x-10x. In addition, the tool allows a non-technical user to establish default “batch style” processing routines for corpus cleaning and to avoid the labor of composing and executing parsing commands by hand.



1. Select TMX file(s) to Convert

2. Select Cleaning Steps

Please select your converting options for each language:

Step 1. ✓ Clean placeholder tags

Step 2. ✓ Clean URLs

Step 3. ✓ Tokenize

Step 4. ✓ Lowercase

Step 5. ✓ Clean numbers

Step 6. ✓ Clean duplicate lines
✓ including those having same source but different target

Step 7. ✓ Clean long sentences
✓ source longer than 30 words
✓ target longer than 30 words

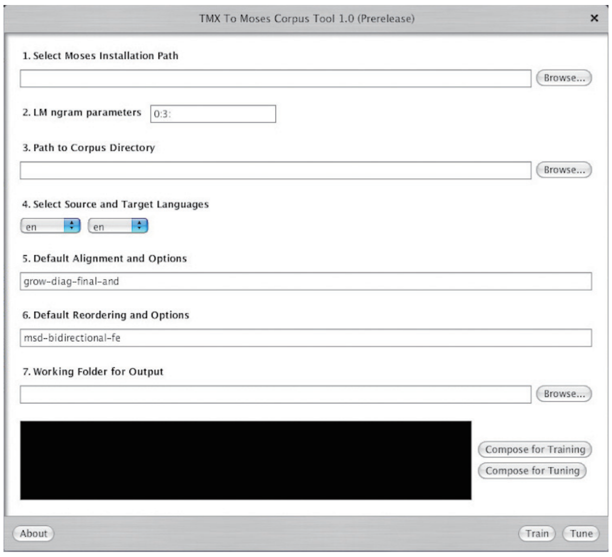
Step 8. ✓ Clean weird aligned pairs
✓ source/target lengh difference larger than 10

Figure 1. Moses Corpus Cleaning Tool UI

3 Training Harness Tool

The Adobe Training Harness is designed to simplify and automate the training of Moses based MT engines. Moses, as it is currently conceived, requires the user to compose and execute a long string of commands which contain the data of file paths to executables and corpus, as well as all the options flags required for the configuration. This is an acceptable solution to providing inputs to a Moses based engine while doing individual training rounds for very specific tasks. It does however have considerable drawbacks when you begin to assess the use of Moses in a production environment.

The Training Harness Tool simplifies the running of Moses by taking all the information from that single composed string and breaks it down into its component parts. All of the elements of the string can be defined and redefined within the GUI of the tool. The time required to change an element is reduced and the chance of introducing errors drops as well. In addition, the ability to run subsequent training runs with small modifications to basic elements becomes a task that can be executable programmatically. So that, given a particular data set and a Moses engine install, the user is able to script the tool to train a series of engines while using different reordering and alignment options. It is possible as well to tune this series of engines based on those settings or others.


TMX To Moses Corpus Tool 1.0 (Prerelease)

1. Select Moses Installation Path

2. LM ngram parameters: 0:3:

3. Path to Corpus Directory

4. Select Source and Target Languages: en, en

5. Default Alignment and Options: grow-diag-final-and

6. Default Reordering and Options: msd-bidirectional-fe

7. Working Folder for Output

Figure 2. Moses Training Tool UI

3.1 Current training tool options

The training tool's major use case is to make feeding commands to a Moses engine both a repeatable and a scriptable action. To this end it is constructed to mimic the standard commands that a Moses user would use to call a Moses engine for a training run. It allows the user to preset data for:

1. The Moses engine installation
2. A language model
3. Training scripts
4. Corpus to be used for training
5. Target and source languages
6. To set alignment and reordering options.

It allows the user to pre-visualize the command being sent to the Moses engine. Commands can be easily inspected in order to make changes and the results stored programmatically for repeated use. Finally the Training Tool can be run in tuning mode to execute tuning on the newly produced engine automatically.

4 Testing Harness Tool

The Adobe testing harness is a clean and simple tool that allows the user to automatically test the output quality of an MT engine. It allows the user to quickly set paths to reference, source, and target files. It also allows the user to determine which testing method should be used. The user may target the Testing tool to use any particular testing regime they would like. For example, the user may choose between using Bleu or Meteor, etc. for scoring. Results from the testing are displayed in the GUI window of the tool and can be outputted to log files for future automated comparison.

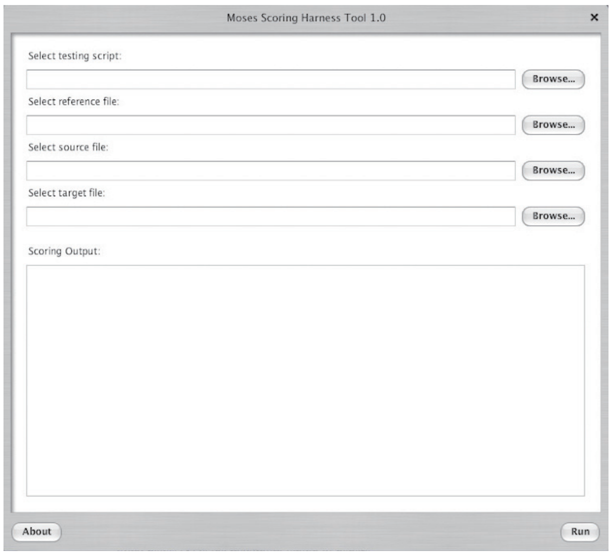


Figure 3. Moses Testing Tool UI

5. Next Steps To Integrating Moses Tooling For Automated Training

The individual pieces of Adobe's Moses development work were built as stand alone solutions to solve unique problems that we faced in adapting Moses to a production environment. This though is only the beginning of meeting those needs. Beyond being able to function independently, each piece is also a building block towards creating a fully integrated solution to provide machine translation as a service on demand. We mentioned, concerning all three pieces of the tooling set, our intention to make them scriptable in nature. This would allow them to be run in a headless and GUI free state, remotely, and in sequence with each other. While each individual part of the tooling set greatly speeds up what was before a manual operation, when each part can be run in an integrated system then we've closed the loop on providing linguistic translation services on demand via Moses.

For the next steps our goal is to provide a complete real world solution for doing machine translation with Moses. We plan to integrate the corpus tool with a remote repository of corpus resources. These resources will be, as for most translation systems static and will have been previously cleaned during set up. In time, as we move away from the launching of the system, new data will come into the database with regular frequency. The integrated corpus tool will be notified that new data for a specific language pair and functional domain is available for cleaning. The Corpus Cleaning Tool then kicks off automatically using preset values and cleans the incoming data.

When the new data has been cleaned, its reintegrated to the existing stock of clean data and in turn sends notice to the Corpus Training Tool to begin a new training run using both legacy data and the newly included content. The Training Tool can be pre-scripted to run against a series of default configurations and to produce a series of engines for each set. New engines are trained and tuned and they too are stockpiled on a remote MT server. Consequently, as new engines are trained, the remote Testing Tool is called to action and begins to run testing operations on each new engine generating scores for them.

The Testing Tool can score new engines against each other and also score those new engines against previously deployed engines. Now, if a new engine surpasses in quality a previously existing engine, system users can be notified and given the option of promoting the engine above the existing one.

By integrating all three tools, with the addition of a data repository on one end, and a consolidated linguistic service layer on the other end, we can now automatically complete the round trip. New data can come into the system. New engines will be automatically trained and scored. Finally new engines can be pushed automatically the production Moses server to provide translation services as an end product.

Acknowledgments

We would like to acknowledge the ongoing support we've had both within Adobe Systems and from external organizations, such as TAUS, in enabling us to make Machine Translation more accessible to industry, researchers, and consumers.

References

Moses Statistical MT system: http://www.statmt.org/moses/