MOSES
statistical machine translation system

Moses
Overview
Manual
Online Demos
FAQ
Mailing Lists
Get Involved
Recent Changes

Getting Started
Source Installation
Baseline System
Packages
Releases
Sample Data
Links to Corpora

Tutorials
Phrase-Based Tutorial
Syntax Tutorial
Factored Tutorial
Optimizing Moses
Experiment.Perl

Training
Overview
Prepare training data
Factored Training
1 Prepare data
2 Run GIZA
3 Align words
4 Lexical translation
5 Extract phrases
6 Score phrases
7 Reordering model
8 Generation model
9 Configuration file
Language Models
Tuning
Training Reference
Decoder Reference

User Documentation
Advanced Models
Efficient Phrase and Rule Storage
Search (Decoding)
Unknown Words
Hybrid Translation
Moses as a Service
Incremental Training
Domain Adaptation
Constrained Decoding
Cache-based Models
Sparse features
Support Tools
External Tools
Web Translation
Pipeline Creation Language
Obsolete Features

Moses » WebTranslation

Search »

Translating Web pages with Moses

(Code and documentation written by Herve Saint-Amand.)

We describe a small set of publicly available Perl scripts that provide the mechanisms to translate a Web page by retrieving it, extracting all sentences it contains, stripping them of any font style markup, translating them using the Moses system, re-inserting them in the document while preserving the layout of the page, and presenting the result to the user, providing a seamless translation system comparable to those offered by Google, BabelFish and others.

Introduction

Purpose of this program

Moses is a cutting-edge machine translation program that reflects the latest developments in the area of statistical machine translation research. It can be trained to translate between any two languages, and yields high quality results. However, the Moses program taken alone can only translate plain text, i.e., text stripped of any formatting or style information (as in `.txt` files). Also, it only deals with a single sentence at a time.

A program that can translate Web pages is a very useful tool. However, Web pages contain a lot of formatting information, indicating the color, font and style of each piece of text, along with its position in the global layout of the page. Most Web pages also contain more than one sentence or independent segment of text. For these reasons a Web page cannot be fed directly to Moses in the hope of obtaining a translated copy.

The scripts described in this document implement a Web page translation system that, at its core, relies on Moses for the actual translation task. The scripts' job, given a Web page to translate, is to locate and extract all strings of text in the page, split paragraphs into individual sentences, remove and remember any style information associated with the text, send the normalized, plain-text string to Moses for translation, re-apply the style onto the translated text and re-insert the sentence at its place in the original page structure, (hopefully) resulting in a translation of the original.

A word of warning

These scripts are a proof-of-concept type of demonstration, and should not be taken for more than that. They most probably still contain bugs, and possibly even security holes. They are not appropriate for production environments.

Intended audience and system requirements

This document is meant for testers and system administrators who wish to install and use the scripts, and/or to understand how they work.

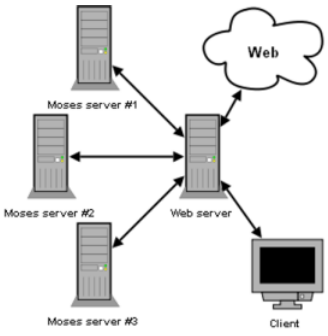
Before starting, the reader should ideally possess basic knowledge of:

- UNIX-type command-line environments
- TCP/IP networking (know what a hostname and a port are)
- how to publish a Web page using a CGI script on an Apache server
- how to configure and operate the Moses decoder

and have the following resources available:

- an Apache (or similar) Web server
- the possibility of running CPU- and memory-intensive programs, either on the Web server itself (not recommended), or on one or several other machines that can be reached from the Web server
- Moses installed on those machines

Overview of the architecture



The following is a quick overview of how the whole system works. An attempt at illustrating the architecture is in the figure above. File names refer to files available from an Git repository, as explained in the download section.

1. The Moses system is installed and configured on one or several computers that we designate as **Moses servers**.
2. On each Moses server, a daemon process, implemented by `daemon.pl`, accepts network connections on a given port and copies everything it gets from those connections straight to Moses, sending back to the client what Moses printed back. This basically *plugs* Moses directly onto the network.
3. Another computer, which we designate as the **web server**, runs Apache (or similar) Web server software.
4. Through that server, the CGI scripts discussed in this document (`index.cgi`, `translate.cgi` and supporting files) are served to the client, providing the user interface to the system. It is a simple matter to configure `translate.cgi` so that it knows where the Moses servers are located.
5. A client requests `index.cgi` via the Web server. A form containing a textbox is served back, where the user can enter a URL to translate.

6. That form is submitted to `translate.cgi`, which does the bulk of the job. It fetches the page from the Web, extracts translatable plain text strings from it, sends those to the Moses servers for translation, inserts the translations back into the document, and serves the document back to the client. It adjusts links so that if any one is clicked in the translated document, a translated version will be fetched rather than the document itself.

The script containing all the interesting code, `translate.cgi`, is heavily commented, and programmers might be interested in reading it.

Mailing list

Should you encounter problems you can't solve during the installation and operation of this program, you can write to the moses support mailing list at `moses-support@mit.edu`. Should you not encounter problems, the author (whose email is found in the source file headers) would be astonished to hear about it.

Detailed setup instructions

Obtaining a copy of the scripts

The scripts are stored in the `contrib/web` directory in the Moses distribution.

Setting up the Web server

The extracted source code is ready to be run, there is no installation procedure that compiles or copies files. The program is entirely contained within the directory that was downloaded from Github. It now needs to be placed on a Web server, in a properly configured location such that the CGI scripts (the two files bearing the `.cgi` extension) are executed when requested from a browser.

For instance, if you are on a shared Web server (e.g., a server provided by your university) and your user directory contains a directory named `public_html`, placing the `moses-web` directory inside `public_html` should make it available via the Web, at an address similar to `http://www.dept.uni/~you/moses-web/`.

Troubleshooting

- **404 Not Found** Perhaps the source code folder is not in the right location? Double-check the directory names. See if the home folder (parent of `moses-web` itself) is reachable. Ask your administrator.
- **403 Forbidden**, or you see the Perl source code of the script in your browser} The server is not configured to execute CGI scripts in this directory. Move `moses-web` to the `cgi-bin` subdirectory of your Web home, if it exists. Create a `.htacess` file in which you enable the `ExecCGI` option (see the Apache documentation).
- **Internal server error** Perhaps the scripts do not have the right permissions to be executed. Go in `moses-web` and type the command `chmod 755 *cgi`.

The scripts are properly installed once you can point your browser at the correct URL and you see the textbox in which you should enter the URL, and the 'Translate' button. Pressing the button won't work yet, however, as the Moses servers need to be installed and configured first.

Setting up the Moses servers

You now need to install Moses and the `daemon.pl` script on at least one machine.

Choosing machines for the Moses servers

Running Moses is a slow and expensive process, at least when compared to the world of Web servers where everything needs to be lightweight, fast and responsive. The machine selected for running the translator should have a recent, fast processor, and as many GBs of memory as possible (see the Moses documentation for more details).

Technically, the translations could be computed on the same machine that runs the Web server. However, the loads that Moses places on a system would risk seriously impacting the performance of the Web server. For that reason, we advise not running Moses on the same computer as the Web server, especially not if the server is a shared server, where several users host their files (such as Web servers typically provided by universities). In case of doubt we recommend you ask your local administrator.

For the sake of responsiveness, you may choose to run Moses on several machines at once. The burden of translation will then be split equally among all the hosts, thus more or less diving the total translation time by the number of hosts used. If you have several powerful computers at your disposal, simply repeat the installation instructions that follow on each of the machines independently.

The main translation script, which runs on the Web server, will want to connect to the Moses servers via TCP/IP sockets. For this reason, the Moses servers must be reachable from the Web server, either directly or via SSH tunnels of other proxy mechanisms. Ultimately the translation script on the Web server must have a hostname/port address it can connect to for each Moses server.

Installing the scripts

Install Moses

For each Moses server, you will need to install and configure Moses for the language pair that you wish to use. If your Moses servers are all identical in terms of hardware, OS and available libraries, installing and training Moses on one machine and then copying the files over to the other ones should work, but your mileage may vary.

Install daemon.pl

Once Moses is working, check out, on each Moses server, another copy of the `moses-web` source directory by following again the instructions in the download section. Open `bin/daemon.pl`, and edit the `$MOSES` and `$MOSES_INI` paths to point to the location of your `moses` binary and your `moses.ini` configuration file.

Choose a port number

Now you must choose a port number for the daemon process to listen on. Pick any number between 1,024 and 49,151, ideally not a standard port for common programs and protocols to prevent interference with other programs (i.e., pick a port not mentioned in

your `/etc/services` file).

Start the daemon

To activate a Moses server, simply type, in a shell running on that server:

```
./daemon.pl <hostname> <port>
```

where `<hostname>` is the name of the host you're typing this on (found by issuing the `hostname` command), and `<port>` is the port you selected. It may be misleading that despite its name, this program does not fork a background process, it is the background process itself. To truly launch the process in the background so that it continues running after the shell is closed, this command might be more useful:

```
nohup ./daemon.pl <hostname> <port> &
```

The `bin/start-daemon-cluster.pl` script distributed with this program provides an automation mechanism that worked well in the original setup on the University of Saarland network. It was used to start and stop the Moses servers all at once, also setting up SSH tunneling on startup. Because it is very simple and trimmed to the requirements of that particular installation, we do not explain its use further here, but the reader might find inspiration in reading it.

Test the Moses servers

The daemon should now be listening on the port you chose. When it receives a connection, it will read the input from that connection one line at a time, passing each line in turn to Moses for translation, and printing back the translation followed by a newline.

If you have the NetCat tool installed, you can test whether it worked by going to a shell on the Web server and typing `echo "Hello world" | nc <hostname> <port>`, replacing `Hello world` by a phrase in your source language if it is not English, and `<hostname>` and `<port>` by the values pointing to the Moses server you just set up. A translation should be printed back.

Configure the tokenizer

The `translate.cgi` script uses external tokenizer and detokenizer scripts. These scripts adapt their regular expressions depending on the language parsed, and so tokenizing is improved if the correct language is selected. This is done by opening `translate.cgi` with your favourite text editor, and setting `$INPUT_LANG` and `$OUTPUT_LANG` to the appropriate language codes. Currently the existing language codes are the file extensions found in the `bin/nonbreaking_prefixes` directory. If yours are not there, simply use `en` -- end-of-sentence detection may then be suboptimal, and translation quality may be impacted, but the system will otherwise still function.

Configure the Web server to connect to the Moses servers

The last remaining step is to tell the frontend Web server where to find the backend Moses servers. Still in `translate.cgi`, set the `@MOSES_ADDRESSES` array to the list of `hostname:port` strings identifying the Moses servers.

Here is a sample valid configuration for three Moses servers named `server01`, `server02` and `server03`, each with the daemon listening on port 7070:

```
my @MOSES_ADDRESSES = ("server01:7070", "server02:7070", "server03:7070");
```

Stopping the daemons once done

The daemon processes continuously keep a copy of Moses running, so they consume memory even when idle. For this reason, we recommend that you stop them once they are not needed anymore, for instance by issuing this command on each Moses server: `killall daemon.pl`