Moses version 2.1 Release Notes

Overview

The document is the release notes for the Moses SMT toolkit, version 2.1. It describes the changes to toolkit since version 1.0 from January 2013. The aim during this period has been to tackle more complicated issues to enable better expandability and reliability.

Specifically, the decoder has been refactored to create a more modular framework to enable easier incorporation of new feature functions into Moses. This has necessitate major changes in many other parts of the toolkit, including training and tuning.

As well as the refactored code, this release also incorporate a host of new features donated by other developers. Transliteration modules, better error handling, small and fast language models, and placeholders are just some of the new features that spring to mind.

We have also continue to expand the testing regime to maintain the reliability of the toolkit, while enable more developers to contribute to the project.

We distribute Moses as:

1. source code,
2. binaries for Windows (32 and 64 bit), Mac OSX (Mavericks), and various flavours of Linux (32 and 64 bit).
3. pre-installed in a Linux virtual machine, using the open source VirtualBox application.
4. Amazon cloud server image.

New Feature Function Framework

Feature functions are code that gives are score, or scores, to the partial or complete translation to indicate whether the function believe the translation is good or bad. Translation models and language models are examples of feature functions.

The framework for incorporating feature functions was changed to enable new feature functions to be added easily.

Before the refactoring, adding new feature function was ad-hoc, requiring the developer to:

1. Add a new section to the moses.ini file which hold information specific to the feature function. Add code in the **Parameter** class to read the new section.
2 Add code in **StaticData** class to instantiate and store the feature function, and delete it

on exit.

3. Add code to read the feature functions weights.
4. Create a class that implements the feature function.
5. Add code to the mert-moses.pl to tune the weights for the feature function.

This ad-hoc approach puts a burden on the developer and increases the likelihood of error if one of the above steps is not correctly implemented.

After refactoring:

1. No new section in moses.ini file is required.
2. The feature function framework takes care of managing feature functions instances. The developer just has to register the feature.
3. The framework reads the feature function weights.
4. Create a class that implements the feature function.
5. No changes to mert-moses.pl is required.

Also, before the refactoring, a core set of feature function were hard coded into the decoding algorithm: distortion, word penalty, and unknown word penalty. After refactoring, no feature functions are hardcoded.

As well as refactoring, the decoder source code was cleaned up and simplified where possible.

The documentation on the Moses website has been updated to reflect the refactored framework.

New Features

The following is a list of the major new features in the Moses toolkit since version 1.0, in roughly chronological order.

1. Transliteration Phrase-Table by Nadir Durrani
By default, Moses copies unknown words from the input to the output ad-verbatim. This is a good strategy when the languages are close, and their alphabets are the same. However, for wildly different language pairs, the unknown word must be transliterated to the target alphabet. This new feature integrated both the training and decoding into Moses.

2. **DALM integration by Jun-ya Norimatsu**. Smaller, faster language model[1].
3. **CoveredReferenceFeature by Ales Tamchyna**. Given a file with reference translation, reward phrases for matching words from the reference
    a. **Constrained Decoding by Hieu Hoang**. Restrict creation of hypothesis to known output.
4. **Picaro by Jason Riesa**. Tool for visualisation of word alignment.

[1] http://www.aclweb.org/anthology/D13-1023

5. **Neural LM by Lane Schwartz**. Integration of Ashish Vaswani's LM into Moses.
6. **Tokenization configuration files for Greek and Tamil by Dimitris Mavroeidis and Arththika Paramanathan**.
7. **DIMwid by Robin Kurtz**. Visualisation of SCFG decoding.
8. **Placeholder by Achim Ruopp and Hieu Hoang**. Better handling of names, numbers and dates.
9. **Backward LM by lane Schwartz**.
10. **Multimodel phrase-table by Rico Sennrich**. A phrase table that merges the results of multiple phrase-tables.
11. **Operation Sequence Model by Nadir Durrani**[2].
12. **Alternate weight setting by Philipp Koehn**. Run multiple language pairs systems in one decoder instance.
13. **Lattice and confusion network phrase-based decoding with any phrase-tables by Hieu Hoang.** Any phrase-table implementation can now be used to decode lattices and confusion networks. Previously, only Richard Zen's binary phrase-table could be used.
14. **Ondisk phrase-table for phrase-based model by Hieu Hoang.** The ondisk, load-on-demand phrase-table can be used for both phrase-based models and hierarchical models. Previously, the type of phrase-table implementation that could be used was dependent on the model.
15. **Clearer error messages by Hieu Hoang.** Abort errors have been converted to throw exceptions, with user-friendly messages where possible.

Operating-System Compatibility

We ensured Moses, IRSTLM, and MGIZA compiles on the following platforms:

i. Windows 8 (32-bit and 64-bit) with Cygwin 6.1
ii. Mac OSX 10.8 with MacPorts
iii. Ubuntu 12.04 and 12.10, 32 and 64-bit
iv. Debian 6.0, 32 and 64-bit
v. Fedora 17, 32 and 64-bit
vi. openSUSE 12.2, 32 and 64-bit

All the binary executables are made available for download for users who do not wish to compile their own version. The URL for downloads is

http://www.statmt.org/moses/RELEASE-2.1/binaries/

Issues:

1. IRSTLM was not linked statically. The 64-bit version fails to execute on Debian 6.0. All other platforms can run the downloaded executables without problem.
2. Mac OSX does not support static linking. Therefore, it is not known if the executables would work on any other platforms, other than the one on which it was tested.
3. mgiza compilation failed on Mac OSX with gcc v4.2. It could only be successfully compiled

[2] http://statmt.org/OSMOSES/osmoses.pdf

with gcc v4.5, available via MacPorts.

Also, we ran a phrase-based training one language language pair end-to-end on Debian 32 and 64-bit. We make available the Virtual Machines for users to download and use for their own experiments.

The Moses 2.1 release has also been incorporated by Amittai Axelrod into a virtual SMT research & development platform for use on Amazon's commercial cluster, the Elastic Compute Cloud (EC2). This Amazon Machine Image (AMI id: ami-feec8dc) comes pre-installed with popular research tools such as Moses version 2.1, Joshua v5.0, and Cdec. Initial registration is here:
https://portal.aws.amazon.com/gp/aws/application/edit/index.html?productCode=38C7C397
Virtual machines may be launched directly from here:
https://console.aws.amazon.com/ec2/v2/home?region=us-west-2#LaunchInstanceWizard:ami=ami-feec8dce

Performance

The Moses regression tests were ran to ensure that the output remained constant despite the major refactoring changes. 22 new regression tests were created to improve the test coverage.

End-to-end testing of the Moses pipeline was run for 9 European language pairs, trained on the Europarl corpus, to ensure that translation quality had been maintained. The tests were initiated for Moses v 0.91 and have been expanded to test new features in version 2.1. There seems to be no consistent degradation in the translation quality, besides random variation. The results are shown below.

			RELEASE 0.91	RELEASE 1.0	RELEASE-2.1
En-es	1	pb	24.70	24.81	24.61
	2	hiero	24.18	24.20	23.74
	3	(1) + placeholders			24.80
	4	(1) + CreateOnDiskPt			24.67
Es-en	1	pb	22.87	23.01	22.97
	2	hiero	22.32	22.37	22.20
	3	(1) + placeholders			22.78
	4	(1) + CreateOnDiskPt			23.06
En-cs	1	pb	10.98	11.04	11.16
	2	hiero	10.92	10.93	10.90
	3	(1) + placeholders			10.5
	4	(1) + CreateOnDiskPt			11.12
	5	(2) + Ken's incre algo			10.76
Cs-en	1	pb	16.00	15.72	15.81
	2	hiero	15.89	15.68	15.66
	3	(1) + placeholders			15.65
	4	(1) + CreateOnDiskPt			15.80
	5	(2) + Ken's incre algo			15.43

En-de	1	pb	11.72	11.87	11.71
	2	hiero	11.59	11.62	11.45
	3	(1) + CreateOnDiskPt			11.64
	4	(1) + POS de	11.55	11.67	11.75
	5	(4) + POS en	11.64	11.75	11.69
De-en	1	pb	15.71	15.75	15.80
	2	hiero	15.74	15.53	15.56
	3	(1) + POS de	15.71	15.84	15.70
	4	(3) + POS en	15.84	15.93	15.94
	5	(1) + CreateOnDiskPt			15.70
	6	(3) + CreateOnDiskPt			15.76
	7	(4) + CreateOnDiskPt			15.79
En-fr	1	pb truecase	24.43	24.38	24.45
	2	pb recase	22.95	22.94	22.67
	3	(2) + hiero	22.35	22.28	22.11
	4	(2) + POS en	23.34	23.05	23.06
	5	(2) + kbmira			22.81
	6	(2) + pro			22.52
	7	(5) + mira			21.59
	8	(2) + CreateOnDiskPt			22.89
	9	(2) + CompactPt			22.67
Fr-en	1	pb truecase	24.09	24.06	23.87
	2	pb recased	22.40	22.41	22.43
	3	(2) + hiero	18.19	18.05	17.78
	4	(2) + en POS	22.51	22.55	22.47
	5	(2) + kbmira			22.46
	6	(2) + pro			22.44
	7	(2) + CreateOnDiskPt			22.50
	8	(2) + CompactPt			22.58

The speed of the decoder was also measure before and after refactoring. The refactored decoder is approximately 30% to 50% slower, when translating 50 sentences, with a cs-en Europarl (filtered) model. This concurs with large scale experiments we have run at the University of Edinburgh.

	v 1.0	v 2.1	
Phrase-Based			
Binary	20.132	29.334	-46%
Compact	19.985	26.422	-32%
On-Disk		26.831	
Hierarchical			
On-Disk	39.433	53.943	-37%

The slowdown may be due to the less efficient translation rule caching mechanism, or the exception error handling, used in version 2.1.

End-to-End Testing and Pre-Made Models

A number of full scale experiments are run. This is the final test to ensure that the Moses pipeline can run from beginning to end, uninterrupted, with ‘real-world’ datasets. The translation quality, as measured by BLEU, is also noted, to ensure that there is no decrease in performance due to any interaction between components in the pipeline.

The end-to-end tests produces a large number of tuned models. The models, as well as all configuration and data files, are made available for download. This is useful as a template for users setting up their own experimental environment, or for those who just want the models without running the experiments.

The URL for downloading the pre-made model is
http://www.statmt.org/moses/RELEASE-2.1/models/