LetsMT!

Platform for Online Sharing of Training Data and Building User Tailored MT
www.letsmt.eu/

Project no. 250456

Deliverable D3.1
Adapdation of Moses SMT Toolkit

Version No. v1.0

June 30, 2011

Document Information

Deliverable number:	D3.1
Deliverable title:	Adapdation of Moses SMT Toolkit
Due date of deliverable according to DoW:	30/06/2011
Actual submission date of deliverable:	30/06/2011
Main Author(s):	Oliver Wilson
Participants:	Edinburgh
Reviewer	Tilde
Workpackage:	WP3
Workpackage title:	SMT training facilities and SMT web service
Workpackage leader:	Edinburgh
Dissemination Level:	PU
Version:	v1.0
Keywords:	data formats

Version History

Version	Date	Status	Name of Author (Partner)	Contributions	Description/ Approval Level
v0.1	June 21, 2011	initial draft	Edinburgh		ready for comments from partners
v1.0	June 30, 2011	final version	Edinburgh		include feedback from TILDE

EXECUTIVE SUMMARY

Existing state-of-the-art statistical MT methods (Moses toolkit) has been used as a platform to train MT systems available through LetsMT!. Although Moses toolkit is a mature and appropriate solution for SMT training and the requirements of the LetsMT! system; it needed some improvements to be used in a rapid training, updating and interactive access environment. Detailed specification of necessary improvements in Moses toolkit have been prepared and the required improvements implemented.

Contents

1 Introduction **4**

2 Implemented Improvements **4**
2.1 Multithreaded Decoding 4
2.2 Multithreading Support in RandLM 4
2.3 Moses Server 5
2.4 Multiple Translation Models 5
2.5 Incremental Training 5
2.6 Distributed Language Models 6

3 Summary **6**

List of Figures

1 Introduction

This deliverable describes the improvements implemented in the Moses SMT toolkit for the LetsMT! project, and where the code for the implementation can be found.

For features which reside completely or in part in the Moses source code, specific files and or directories in the code are given. The Moses source code can be downloaded from:

`http://mosesdecoder.svn.sourceforge.net/viewvc/mosesdecoder/trunk/?view=tar`

or alternatively viewed online at:

`http://mosesdecoder.svn.sourceforge.net/viewvc/mosesdecoder/trunk/`

Where new features have been implemented in separate packages, links to the source code for those packages are given.

The documentation for all features implemented can be found online at:

`http://www.statmt.org/moses/index.php?n=Main.HomePage`

Links to the relevant section of the documentation are given for each feature. A printable PDF version of the documentation can be found at:

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

2 Implemented Improvements

2.1 Multithreaded Decoding

With multicore CPUs now very much the norm, applications need to support multithreading in order to take full advantage of the performance they offer.

Moses now supports translating many sentences in parallel. This results in far better use of hardware resources, and far greater translation throughput.

The code implementing this feature can be found in *moses/src/ThreadPool.cpp* and *moses-cmd/src/Main.cpp* and it is documented here:

`http://www.statmt.org/moses/?n=Moses.AdvancedFeatures#ntoc19`

2.2 Multithreading Support in RandLM

RandLM is a Language Model implementation that allows large models to be compressed into a smaller size, at the cost of a small loss in translation quality. It has been upgraded to allow it to be accessed in parallel by multithreaded Moses.

The code for this is integrated into the latest release of RandLM and can be viewed online at:

`http://randlm.cvs.sourceforge.net/viewvc/randlm/main/randlm/`

or downloaded in full from:

`http://randlm.cvs.sourceforge.net/viewvc/randlm/main/randlm/?view=tar`

Enabling this feature is documented in the *src/README* file of the RandLM source code.

2.3 Moses Server

The data structures used by Moses are typically very large, and thus the startup time can sometimes be quite long. In the interactive environment of the LetsMT! platform, a long delay for the user while the translation system starts up is unacceptable.

To address this, a server mode version of Moses was implemented. This allows a Moses instance to be started and remain in the background until it is needed. When text needs to be translated, it is handed to the already running Moses instance via XML-RPC.

The documentation for this feature can be found here:

`http://www.statmt.org/moses/?n=Moses.AdvancedFeatures#ntoc20`

and the code can be found in the *server/* directory of the Moses source code.

2.4 Multiple Translation Models

In the academic environment Moses was initially developed in, it was only necessary to translate between a single pair of languages. In the LetsMT! platform however, there will be many users wishing to translate between many language pairs.

To address this, Moses was altered to support multiple translation models in a single running instance. Thus supporting the many language pairs of LetsMT! users, without having to have multiple decoders running at the same time.

The documentation for this feature can be found here:

`http://www.statmt.org/moses/?n=Moses.AdvancedFeatures#ntoc21`

and the main classes implementing it can be found in *moses/src/TranslationSystem.cpp* in the Moses source.

2.5 Incremental Training

Translation models for Moses are typically batch trained. This process can take some time, and if a user then wishes to take advantage of any new training data they may have, the process must be started from scratch.

Incremental training allows the user to integrate new training data as they acquire it, without having to go through the time consuming process of retraining from scratch. New data can even be given to a running Moses instance via XML-RPC, thus avoiding any long restart time or interuption to service.

The code implementing the incremental word alignments can be found at:

`http://code.google.com/p/inc-giza-pp/source/browse/#svn%2Ftrunk%253Fstate%253Dclosed`

and the classes which integrate this into Moses in *moses/src/PhraseDictionaryDynSuffixArray.cpp* and *moses/src/DynSuffixArray.cpp* in the Moses source. Documentation can be found at

`http://www.statmt.org/mosesdev/?n=Moses.AdvancedFeatures#ntoc27`

This work has been based on a paper by Abby Levenberg, Chris Callison-Burch, and Miles Osborne - Stream-based Translation Models for Statistical Machine Translation which can be downloaded from here:

`http://homepages.inf.ed.ac.uk/miles/papers/naacl10b.pdf`

2.6 Distributed Language Models

Research has shown that significant improvements in translation quality can be achieved using very large Language Models built from very large monolingual corpora. Even with the compression offered by Language Model implementations such as RandLM, such models can become too large to host on a single machine.

Distributing the Language Model across many machines overcomes this limitation. It also allows many Moses decoder instances to access the model simultaneously, resulting in much better resource utilisation.

The code implementing the beta version of this can be found at:

`http://homepages.inf.ed.ac.uk/owilson/DMapLM.tar.gz`

with classes integrating it in to Moses in *moses/src/LanguageModelDMapLM.cpp* of the Moses source. The documentation can be found here:

`http://www.statmt.org/moses/?n=Moses.AdvancedFeatures#ntoc31`

This work is motivated by a paper from Google by Thorsten Brants et al - Large Language Models in Machine Translation, which is available here:

`acl.ldc.upenn.edu/D/D07/D07-1090.pdf`

3 Summary

The demands placed on Moses by the LetsMT! platform were challenging, but have none the less been met. The changes made to Moses in this deliverable have not only enabled the goals of the LetsMT! platform to be realised, but have been of great benefit to the wider academic and industrial Machine Translation community.