LetsMT!: A Cloud-Based Platform for Do-It-Yourself Machine Translation

Andrejs Vasiļjevs
TILDE
Vienbas gatve 75a, Riga
LV-1004, LATVIA
andrejs@tilde.com

Raivis Skadiņš
TILDE
Vienbas gatve 75a, Riga
LV-1004, LATVIA
raivis.skadins@tilde.lv

Jörg Tiedemann
Uppsala University
Box 635, Uppsala
SE-75126, SWEDEN
jorg.tiedemann@lingfil.uu.se

Abstract

To facilitate the creation and usage of custom SMT systems we have created a cloud-based platform for do-it-yourself MT. The platform is developed in the EU collaboration project LetsMT!. This system demonstration paper presents the motivation in developing the LetsMT! platform, its main features, architecture, and an evaluation in a practical use case.

1 Introduction

Current mass-market and online MT systems are of a general nature and perform poorly for smaller languages and domain specific texts. The European Union ICT-PSP Programme project LetsMT! develops a user-driven MT "factory in the cloud" enabling web users to get customised MT that better fits their needs. Harnessing the huge potential of the web together with open statistical machine translation (SMT) technologies, LetsMT! has created an online collaborative platform for data sharing and MT building.

The goal of the LetsMT! project is to facilitate the use of open source SMT tools and to involve users in the collection of training data. The LetsMT! project extends the use of existing state-of-the-art SMT methods by providing them as cloud-based services. An easy-to-use web interface empowers users to participate in data collection and MT customisation to increase the quality, domain coverage, and usage of MT.

The LetsMT! project partners are companies TILDE (coordinator), Moravia, and SemLab, and the Universities of Edinburgh, Zagreb, Copenhagen, and Uppsala.

2 LetsMT! Key Features

The LetsMT! platform[1] (Vasiļjevs et al., 2011) gathers public and user-provided MT training data and enables generation of multiple MT systems by combining and prioritising this data. Users can upload their parallel corpora to an online repository and generate user-tailored SMT systems based on data selected by the user.

Authenticated users with appropriate permissions can also store private corpora that can be seen and used only by this user (or a designated user group). All data uploaded into the LetsMT! repository is kept in internal format, and only its metadata is provided to the user. Data cannot be downloaded or accessed for reading by any means. The uploaded data can only be used for SMT training. In such a way, we encourage institutions and individuals to contribute their data to be publicly used for SMT training, even if they are not willing to share the content of the data.

A user creates SMT system definition by specifying a few basic parameters like system name, source/target languages, domain, and choosing corpora (parallel for translation models or monolingual for language models) to use for the particular system. Tuning and evaluation data can be automatically extracted from the training corpora or specified by the user. The access level of the system can also be specified - whether it will be public or accessible only to the particular user or user group.

[1] http://letsmt.com

Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, pages 43–48,
Jeju, Republic of Korea, 8-14 July 2012.

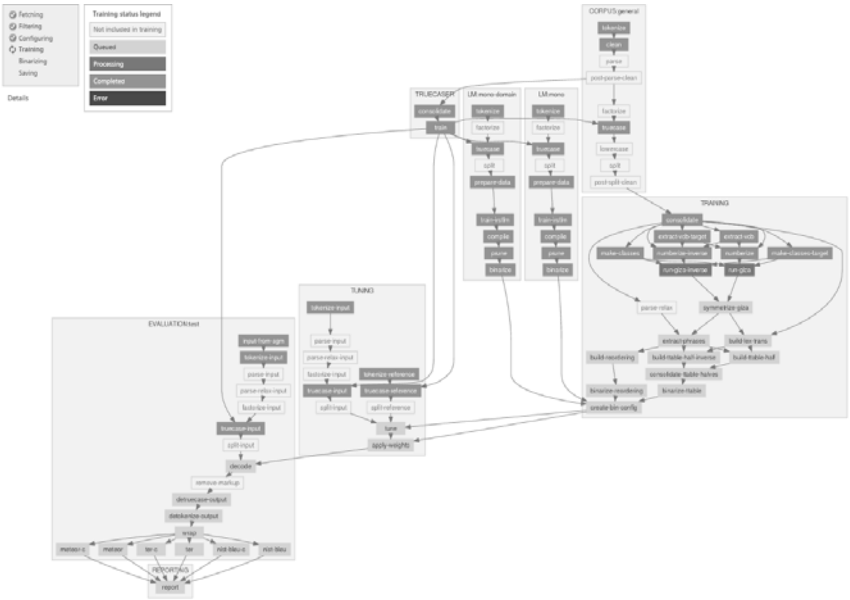

Figure 1. Training chart providing dynamic representation of training steps.

When the system is specified, the user can begin training it. Progress of the training can be monitored on the dynamic training chart (Figure 1). It provides a detailed visualisation of the training process showing (i) steps queued for execution of a particular training task, (ii) current execution status of active training steps, and (iii) steps where any errors have occurred. The training chart remains available after the training to facilitate analysis of the performed trainings. The last step of the training task is automatic evaluation using BLEU, NIST, TER, and METEOR scores.

A successfully trained SMT system can be started and used for translation in several ways:

- on the translation webpage of LetsMT! for testing and short translations;
- using LetsMT! plug-ins in computer-assisted translation (CAT) tools for professional translation;
- integrating the LetsMT! widget for web-site translation;
- using LetsMT! plug-ins for IE and FireFox to integrate translation into the browsers;
- using LetsMT! API for MT integration into different applications.

LetsMT! allows for several system instances to run simultaneously to speed up translation and balance the workload from numerous translation requests.

LetsMT! user authentication and authorisation mechanisms control access rights to private training data, trained models and SMT systems, permissions to initiate and manage training tasks, run trained systems, and access LetsMT! services through external APIs.

The LetsMT! platform is populated with initial SMT training data collected and prepared by the project partners. It currently contains more than 730 million parallel sentences in almost 50 languages. In the first 4 months since launching the invitation only beta version of the platform, 82 SMT systems have been successfully trained.

3 SMT Training and Decoding Facilities

The SMT training and decoding facilities of LetsMT! are based on the open source toolkit Moses. One of the important achievements of the project is the adaptation of the Moses toolkit to fit into the rapid training, updating, and interactive access environment of the LetsMT! platform.

The Moses SMT toolkit (Koehn et al., 2007) provides a complete statistical translation system distributed under the LGPL license. Moses includes all of the components needed to pre-process data and to train language and translation models. Moses is widely used in the research community and has also reached the commercial sector. While the use of the software is not closely monitored, Moses is known to be in commercial use by companies such as Systran (Dugast et al., 2009), Asia Online, Autodesk (Plitt and Masselot, 2010), Matrixware[2], Adobe, Pangeanic, Logrus[3], and Applied Language Solutions (Way et al., 2011).

The SMT training pipeline implemented in Moses involves a number of steps that each require a separate program to run. In the framework of

[2] Machine Translation at Matrixware: http://ir-facility.net/downloads/mxw_factsheet_smt_200910.pdf

[3] TDA Members doing business with Moses: http://www.tausdata.org/blog/2010/10/doing-business-with-moses-open-source-translation/

LetsMT!, this process is streamlined and made automatically configurable given a set of user-specified variables (training corpora, language model data, tuning sets). SMT training is automated using the Moses experiment management system (Koehn, 2010). Other improvements of Moses, implemented by the University of Edinburgh as part of LetsMT! project, are:

- the incremental training of SMT models (Levenberg et al., 2010);
- randomised language models (Levenberg et al., 2009);
- a server mode version of the Moses decoder and multithreaded decoding;
- multiple translation models;
- distributed language models (Brants et al., 2007).

Many improvements in the Moses experiment management system were implemented to speed up SMT system training and to use the full potential of the HPC cluster. We revised and improved Moses training routines (i) by finding tasks that are executed sequentially but can be executed in parallel and (ii) by splitting big training tasks into smaller ones and executing them in parallel.

4 Multitier Architecture

The LetsMT! system has a multitier architecture (Figure 2). It has (i) an interface layer implementing the user interface and APIs with external systems, (ii) an application logic layer for the system logic, (iii) a data storage layer consisting of file and database storage, and (iv) a high performance computing (HPC) cluster. The LetsMT! system performs various time and resource consuming tasks; these tasks are defined by the application logic and data storage and are sent to the HPC cluster for execution.

The Interface layer provides interfaces between the LetsMT! system and external users. The system has both human and machine users. Human users can access the system through web browsers by using the LetsMT! web page interface. External systems such as Computer Aided Translation (CAT) tools and web browser plug-ins can access the LetsMT! system through a public API. The public API is available through both REST/JSON and SOAP protocol web services. An HTTPS protocol is used to ensure secure user authentication and secure data transfer.

The application logic layer contains a set of modules responsible for the main functionality and logic of the system. It receives queries and commands from the interface layer and prepares answers or performs tasks using data storage and the HPC cluster. This layer contains several modules such as the Resource Repository Adapter, the User Manager, the SMT Training Manager, etc. The interface layer accesses the application logic layer through the REST/JSON and SOAP protocol web services. The same protocols are used for communication between modules in the application logic layer.

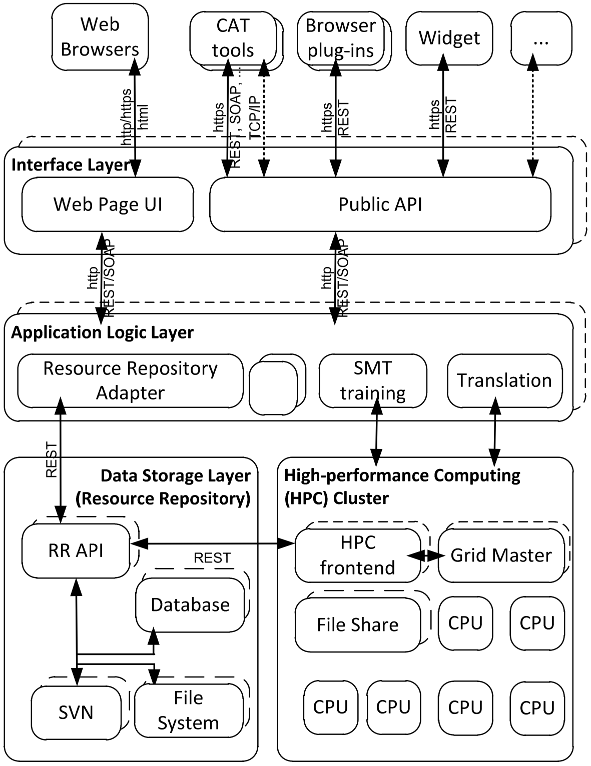

Figure 2. The LetsMT! system architecture

The data is stored in one central Resource Repository (RR). As training data may change (for example, grow), the RR is based on a version-controlled file system (currently we use SVN as the backend system). A key-value store is used to keep metadata and statistics about training data and trained SMT systems. Modules from the application logic layer and HPC cluster access RR through a REST-based web service interface.

A High Performance Computing Cluster is used to execute many different computationally heavy data processing tasks – SMT training and running, corpora processing and converting, etc. Modules from the application logic and data storage layers create jobs and send them to the HPC cluster for execution. The HPC cluster is responsible for accepting, scheduling, dispatching, and managing remote and distributed execution of large numbers of standalone, parallel, or interactive jobs. It also manages and schedules the allocation of distributed resources such as processors, memory, and disk space. The LetsMT! HPC cluster is based on the Oracle Grid Engine (SGE).

The hardware infrastructure of the LetsMT! platform is heterogeneous. The majority of services run on Linux platforms (Moses, RR, data processing tools, etc.). The Web server and application logic services run on a Microsoft Windows platform.

The system hardware architecture is designed to be highly scalable. The LetsMT! platform contains several machines with both continuous and on-demand availability:

- Continuous availability machines are used to run the core frontend and backend services and the HPC grid master to guarantee stable system functioning;
- On-demand availability machines are used (i) to scale up the system by adding more computing power to training, translation, and data import services (HPC cluster nodes) and (ii) to increase performance of frontend and backend server instances.

To ensure scalability of the system, the whole LetsMT! system including the HPC cluster is hosted by Amazon Web Services infrastructure, which provides easy access to on-demand computing and storage resources.

5 Data Storage and Processing Facilities

As a data sharing and MT platform, the LetsMT! system has to store and process large amounts of SMT training data (parallel and monolingual corpora) as well as trained models of SMT systems. The Resource Repository (RR) software is fully integrated into the LetsMT! Platform and provides the following major components:

- Scalable data storage based on version-controlled file systems;
- A flexible key-value store for metadata;
- Access-control mechanisms defining three levels of permission (private data, public data, shared data);
- Data import modules that include tools for data validation, conversion and automatic sentence alignment for a variety of popular document formats.

The general architecture of the Resource Repository is illustrated in Figure 3. It is implemented in terms of a modular package that can easily be installed in a distributed environment. RR services are provided via Web API's and secure HTTP requests. Data storage can be distributed over several servers as is illustrated in Figure 3. Storage servers communicate with the central database server that manages all metadata records attached to resources in the RR. Data resources are organised in slots that correspond to file systems with user-specific branches. Currently, the RR package implements two storage backends: a plain file system and a version-controlled file system based on subversion (SVN). The latter is the default mode, which has several advantages over non-revisioned data storage. Revision control systems are designed to handle dynamically growing collections of mainly textual data in a multi-user environment. Furthermore, they keep track of modifications and file histories to make it possible to backtrack to prior revisions. This can be a strong advantage, especially in cases of shared data access. Another interesting feature is the possibility to create cheap copies of entire branches that can be used to enable data modifications by other users without compromising data integrity for others. Finally, SVN also naturally stores data in a compressed format, which is useful for large-scale document collections. In general, the RR implementation is modular, other storage backends may be added later, and each individual slot can use its own backend type.

Another important feature of the RR is the support of a flexible database for metadata. We decided to integrate a modern key-value store into the platform in order to allow a maximum of flexibility. In contrast to traditional relational databases, key-value stores allow the storage of arbitrary data sets based on pairs of keys and values without being restricted to a pre-defined schema or a fixed data model. Our implementation relies on TokyoCabinet[4], a modern implementation of schema-less databases that supports all of our

[4] https://fallabs/tokyocabinet

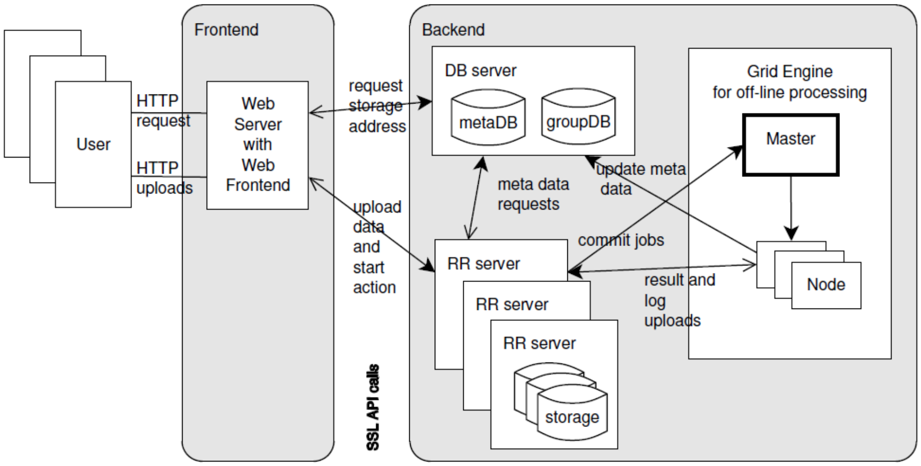

Figure 3. Resource repository overview

requirements in terms of flexibility and efficiency. In particular, we use the table mode of TokyoCabinet that supports storage of arbitrary data records connected to a single key in the database. We use resource URL's in our repository to define unique keys in the database, and data records attached to these keys may include any number of key-value pairs. In this way, we can add any kind of information to each addressable resource in the RR. The software also supports keys with unordered lists of values, which is useful for metadata such as languages (in a data collection) and for many other purposes. Moreover, TokyoCabinet provides powerful query language and software bindings for the most common programming languages. It can be run in client-server mode, which ensures robustness in a multi-user environment and natively supports data replication. Using TokyoCabinet as our backend, we implemented a key-value store for metadata in the RR that can easily be extended and queried from the frontend of the LetsMT! Platform via dedicated web-service calls.

Yet another important feature of the RR is the collection of import modules that take care of validation and conversion of user-provided SMT training material. Our main goal was to make the creation of appropriate data resources as painless as possible. Therefore, we included support for the most common data formats to be imported into LetsMT!. Pre-aligned parallel data can be uploaded in TMX, XLIFF, and Moses formats. Monolingual data can be provided in plain text, PDF, and MS Word formats. We also support the upload of compressed archives in zip and tar format. In the future, other formats can easily be integrated in our modular implementation.

Validation of such a variety of formats is important. Therefore among others, we included XML/DTD validation, text encoding detection software, and language identification tools with pre-trained models for over 60 languages.

Furthermore, our system also includes tools for automatic sentence alignment. Import processes automatically align translated documents with each other using standard length-based sentence alignment methods (Gale and Church, 1993; Varga et al., 2005).

Finally, we also integrated a general batch-queuing system (SGE) to run off-line processes such as import jobs. In this way, we further increase the scalability of the system by taking the load off repository servers. Data uploads automatically trigger appropriate import jobs that will be queued on the grid engine using a dedicated job web-service API.

6 Evaluation for Usage in Localisation

One of the usage scenarios particularly targeted by the project is application in the localisation and translation industry. Localisation companies usually have collected significant amounts of parallel data in the form of translation memories. They are interested in using this data to create customised MT engines that can increase productivity of translators. Productivity is usually measured as an average number of words translated per hour. For this use case, LetsMT! has developed plug-ins for integration into CAT tools. In addition to translation candidates from translation memories, translators receive translation suggestions provided by the selected MT engine running on LetsMT!.

As part of the system evaluation, project partner Moravia used the LetsMT! platform to train and evaluate SMT systems for Polish and Czech. An English-Czech engine was trained on 0.9M parallel sentences coming from Moravia translation memories in the IT and tech domain part of the Czech National Corpus. The resulting system increased translator productivity by 25.1%. An English-Polish system was trained on 1.5M parallel sentences from Moravia production data in the IT domain. Using this system, translator productivity increased by 28.5%.

For evaluation of English-Latvian translation, TILDE created a MT system using a significantly larger corpus of 5.37M parallel sentence pairs, including 1.29M pairs in the IT domain. Additional tweaking was made by manually adding a factored model over disambiguated morphological tags. The resulting system increased translator productivity by 32.9% (Skadiņš et al., 2011).

7 Conclusions

The results described in this paper show that the LetsMT! project is on track to fulfill its goal to democratise the creation and usage of custom SMT systems. LetsMT! demonstrates that the open source SMT toolkit Moses is reaching maturity to serve as a base for large scale and heavy use production purposes. The architecture of the platform and Resource Repository enables scalability of the system and very large amounts of data to be handled in a variety of formats. Evaluation shows a strong increase in translation productivity by using LetsMT! systems in IT localisation.

Acknowledgments

The research within the LetsMT! project has received funding from the ICT Policy Support Programme (ICT PSP), Theme 5 – Multilingual web, grant agreement 250456.

References

L. Dugast, J. Senellart, P. Koehn. 2009. Selective addition of corpus-extracted phrasal lexical rules to a rule-based machine translation system. Proceedings of MT Summit XII.

T. Brants, A.C. Popat, P. Xu, F.J Och, J. Dean. 2007. Large Language Models in Machine Translation. Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), 858-867. Prague, Czech Republic

W. A. Gale, K. W. Church. 1993. A Program for Aligning Sentences in Bilingual Corpora. Computational Linguistics 19 (1): 75–102

P. Koehn, M. Federico, B. Cowan, R. Zens, C. Duer, O. Bojar, A. Constantin, E. Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. Proceedings of the ACL 2007 Demo and Poster Sessions, 177-180. Prague.

P. Koehn. 2010. An experimental management system. The Prague Bulletin of Mathematical Linguistics, 94.

A. Levenberg, M. Osborne. 2009. Stream-based Randomised Language Models for SMT. Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing.

A. Levenberg, C. Callison-Burch, M. Osborne. 2010. Stream-based Translation Models for Statistical Machine Translation. Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (HLT '10)

M. Plitt, F. Masselot. 2010. A Productivity Test of Statistical Machine Translation Post-Editing in a Typical Localisation Context. The Prague Bulletin of Mathematical Linguistics, 93(January 2010): –16

R. Skadiņš, M. Puriņš, I. Skadiņa, A. Vasiļjevs. 2011. Evaluation of SMT in localization to under-resourced inflected language. Proceedings of the 15th International Conference of the European Association for Machine Translation EAMT 2011, 35-40. Leuven, Belgium

A. Vasiļjevs, R. Skadiņš, I. Skadiņa. 2011. Towards Application of User-Tailored Machine Translation in Localization. Proceedings of the Third Joint EM+/CNGL Workshop “Bringing MT to the User: Research Meets Translators” JEC 2011, 23-31. Luxembourg

D. Varga, L. Németh, P. Halácsy, A. Kornai, V. Trón, V. Nagy. 2005. Parallel corpora for medium density languages. Recent Advances in Natural Language Processing IV Selected papers from RANLP05, 590-596

A. Way, K. Holden, L. Ball, G. Wheeldon. 2011. SmartMATE: online self-serve access to state-of-the-art SMT. Proceedings of the Third Joint EM+/CNGL Workshop “Bringing MT to the User: Research Meets Translators” (JEC ’11), 43-52. Luxembourg