MMT: New Open Source MT for the Translation Industry

Nicola Bertoldi[1], **Roldano Cattoni**[1], **Mauro Cettolo**[1], **Amin Farajian**[1], **Marcello Federico**[1],

Davide Caroselli[2], **Luca Mastrostefano**[2], **Andrea Rossi**[2], **Marco Trombetti**[2]

Ulrich Germann[3], **David Madl**[2,3]

1) Fondazione Bruno Kessler, Trento, Italy
2) Translated srl, Rome, Italy
3) University of Edinburgh, United Kingdom

Abstract

MMT is a new open source machine translation software specifically addressing the needs of the translation industry. In this paper we describe its overall architecture and provide details about its major components. We report performance results on a multi-domain benchmark based on public data, on two translation directions, by comparing MMT against state-of-the-art commercial and research phrase-based and neural MT systems.

1 Introduction

MMT aims to consolidate the current state-of-the-art technology into a single easy-to-use product, evolving it and keeping it open to integrate the new opportunities in machine intelligence, such as deep learning. MMT was designed and developed to overcome four technology barriers that have so far hindered the wide adoption of machine translation software by end-users and language service providers: (1) long training time before a MT system is ready to use; (2) difficulty to simultaneously handle multiple domains; (3) poor scalability with data and users; (4) complex installation and set-up. As we will describe in the next section, MMT on the contrary is very fast to train, it instantly adapts to a specific translation domain, it is designed to scale well with data and users, and, finally, it is very easy to install and configure.

This paper describes the current advanced prototype of MMT, a statistical phrase-based machine translation system, which already covers all the above presented features and has being field-tested in real industrial settings. A comprehensive documentation of MMT, including installation manual, is available in the official website.[1]

We also report experiments conducted on a public multi-domain benchmark covering technical translations from English to German and English to French.

2 Main Features of MMT

2.1 MMT Can Ingest New Data Instantly

MMT uses high-performance embedded databases[2] to store parallel and monolingual language data and associated statistics. Instead of pre-computing feature function scores, these are computed on the fly, at translation time, from raw statistics. Thanks to its implementation with databases, MMT is a fully incremental MT system, that can ingest new parallel data while in use, very quickly and without any interruption nor re-training.

2.2 MMT Can Adapt Itself to the Task

Input to the system can be augmented with a snippet of surrounding text. This context information is leveraged by MMT to adapt the translation process to a specific domain. Adaptation is performed on the fly by biasing the data sampling process underlying the computation of the feature functions towards training data that is close to the provided context. (see Sec. 3.2.5 below).

2.3 MMT Scales Easily

MMT is designed as a distributed multi-node architecture, with cloud deployment in mind. There-

© 2017 The authors. This article is licensed under a Creative Commons 3.0 licence, no derivative works, attribution, CC-BY-ND.

[1] http://www.modernmt.eu
[2] RocksDB: https://github.com/facebook/rocksdb

fore it can scale dynamically in response to current demand, simply by adding or removing MMT nodes in the cluster. Single-host deployment for small use cases is also possible.

2.4 MMT Is Easy to Set Up

MMT is distributed as a ready to install package either through Docker, or directly from binary files.[3] In addition, instructions for installing MMT from source code are also available.

3 System Architecture

3.1 Distributed Infrastructure

MMT's distributed architecture is based on a Leader-Follower network where nodes form a cluster through the *Hazelcast* framework.[4]

Most inter-node communications are carried out through shared in-memory data structures, suitable for small data and thus employed for service communications and load balancing. When a node receives a translation request, it sends it to an Executor that transparently extracts jobs from its internal pool, chooses a worker node, and finally redirects the translation output to the original requesting node.

Bigger volumes of data are handled by the nodes in persistent messaging queues (*Kafka*[5]) and in an internal database (*Cassandra*[6]). The former is mostly used to distribute newly ingested resources, which any node may import during its life cycle; the latter to handle persistent application-internal data, like domains' and contributions' metadata.

Both the Followers and the Leader expose the same REST APIs; the Leader in addition hosts the messaging queue server and the internal database. To join the cluster, a worker node only needs to know the Leader IP.

All nodes in a cluster must have the same initial configuration. It is recommended to perform the initial training on a single node and share the resulting models to the others manually. Once a node has received this initial configuration it can join the cluster at any time, and will automatically receive any new updates through the above mentioned messaging channels.

[3]Currently we distribute binaries for Ubuntu.
[4]https://hazelcast.com
[5]https://kafka.apache.org
[6]http://cassandra.apache.org/

3.2 MT Worker Nodes

The core architecture of each node is composed by several interacting modules.

3.2.1 Tag Management

XML tags occurring in the input text are removed and a map between the tags and their positions is stored. According to the output and the word alignment provided by the decoder, the XML tags are re-introduced after applying a few consistency checks and heuristics.

3.2.2 Numerical Expression Management

Numerical expressions, like numbers, currencies, dates, etc., are transformed into format- and position-dependent placeholders, and a map between the actual numerical values and their placeholders is stored. The placeholders found in the output of the decoder are finally transformed back into their actual numerical values using the word alignment and a few heuristics to resolve possible ambiguities.

3.2.3 Tokenization and De-tokenization

The tokenizer and de-tokenizer, based on third-party software credited in the official documentation, support 45 languages through a unique entry point.

3.2.4 Central Vocabulary

Internally, words are represented by integer IDs managed by a joint vocabulary for source and target language that allows incremental updates.

3.2.5 Context Analyzer

The Context Analyser (CA) is in charge of identifying training data that best matches the provided input context. To this purpose, parallel data is sharded into chunks according to the customer, subject area, genre, etc. In a very loose use of the term, we refer to these shards as "*domains*".

When queried, the Context Analyzer (CA) computes a ranked list of matching domains, with associated weights[7] that indicate how closely they match the input text. The CA is built on top of the *Apache Lucene*[8] framework, in particular *Lucene*'s *Inverted Index* data structure. The *Inverted Index* is complemented with a filesystem-based data structure, called *Corpora Storage*, where all the original indexed data are stored: one file corresponds to one

[7]The weights are computed by means of the *tf-idf* metrics and the *Cosine Similarity*.
[8]https://lucene.apache.org/core/

data shard and new content can be appended to the corresponding storage file. In this way, the *Corpora Storage* always maintains the most updated version of the data. When required, the *Inverted Index* is synchronized with the *Corpora Storage* by re-indexing the changed domains and adding the new ones.[9] This activity does not interfere with the look-up operations of the CA; *Lucene* allows concurrent reads and writes, always ensuring data availability and consistency.

3.2.6 Word Aligner

The Word Aligner (WA) performs many-to-many word-to-word alignment of sentence pairs.

The WA is built on top of *FastAlign* (Dyer et al., 2013); it computes two directional alignments, and symmetrizes them according to the *grow-diag-and-final* policy.[10] The WA is multi-threaded and permits persistent storage and re-loading of the alignment models after training. It is able to align individual new sentence pairs without re-training the models. The WA is trained on all parallel data available at training time, irrespective of their domain.

3.2.7 Decoder

The decoder developed in MMT is an enhanced version of the phrase-based decoder implemented in *Moses* (Koehn et al., 2007). Differently from *Moses*, MMT generates and scores translation hypotheses *according to the context of the input sentence*. In particular, the decoder queries its models with the domain weights computed by the CA from the input context.

3.2.8 Translation Model

The MMT Translation Model (TM) is an enhanced re-implementation of the suffix array-based phrase table by Germann (2015). Its original implementation creates a phrase table at run-time by sampling sentences from the pool of word-aligned parallel data with a uniform distribution, extracting phrase pairs from them, and computing their scores on the fly. The new version provides two enhancements. First, instead of a suffix array, it relies on a DB-backed prefix index of the data pool, thus allowing for fast updates (i.e., insertions and deletions of word-aligned parallel data). Second, it keeps track of the domains from which phrase pairs are extracted and performs *ranked sampling*: extracted phrases are ranked by their relevance (via the domain they were observed in). Translation scores are then obtained by going down the ranked list until a sufficient number of samples has been observed. Hence, by associating with all sentence pairs of each domain the corresponding weight, the TM selects and scores phrase pairs giving priority to the best-matching domain.

The TM scores are the forward and backward probabilities at lexical and phrase level; the phrase-level probabilities are weighted according to the domain weights.

3.2.9 Lexicalized Reordering Model

The same incremental DB-based implementation of the TM is also exploited by the Lexicalized Reordering Model. Similarly, its scores are computed on the fly exploiting the counts extracted from the sampled sentences and the corresponding word alignments, and some global counts stored in the DB. The scores are the forward and backward probabilities for monotone, swap, and discontinuous orientations.

3.2.10 Language Model

The MMT LM linearly combines a static background LM with a context-adaptive LM.

The static LM, implemented with the KenLM toolkit (Heafield et al., 2013), features 5-grams, interpolation of lower-order models, and the Kneser-Ney smoothing technique. It is trained on all monolingual target text regardless the domain information, and does not change over time.

The context-adaptable LM is an *internal mixture* LM (Federico and Bertoldi, 2001) using domain-specific counts extracted from the corresponding data shards and the weights of the CA.[11] The LM features 5-gram statistics, interpolation of lower-order models, and Linear Witten-Bell smoothing. Noteworthy, n-gram probabilities are not pre-estimated in the training phase, but computed on the fly, by exploiting domain-specific n-gram and global statistics, which are stored in a key-value DB.

3.2.11 Manager

The Manager controls the communication between all components to satisfy the translation and updating requests.

[9] For performance reasons, synchronization is subject to a time-out.

[10] http://www.statmt.org/moses.

[11] For efficiency, only the LMs actually activated by the CA are included in the mixture.

3.3 Functionalities

From a functional perspective four phases can be identified, namely training, tuning, updating and translation.

3.3.1 Training

The training phase sets up MMT starting from a collection of bilingual and (possibly) monolingual corpora, which can be domain-specific or not-specialized. In particular, the DBs required by CA, LM and TM, are created, which respectively exploit only the source side, only the target side, or both sides of the training data. Texts are pre-processed by the corresponding modules.

3.3.2 Tuning

MMT implements a standard Minimum Error Rate Training procedure (Och, 2003) to optimize the decoder feature weights.

3.3.3 Updating

Once a system is trained, new bilingual data can be added to it,[12] either to an existing domain or establishing a new one. This operation is performed by updating the corresponding DBs of the CA, the TM and the LM. Such updates do not interfere with the translation process.

3.3.4 Translation

In a standard scenario, MMT translates one document as follows; it (i) processes and sends to the CA the whole document, considered as context for all its sentences, and gets the domain weights, (ii) pre-processes and sends all sentences to the available decoders, independently and in parallel, and gets their translations, and (iii) post-processes and returns all translations by re-creating the original document layout. More generally, however, MMT is able to translate any single sentence provided with some context, even made of a single word.

3.4 APIs

MMT system exposes APIs for its integration in third-party software. Plug-ins are under advanced construction to permit the integration of MMT in various commercial CAT tools.

[12] For instance, new data can be a translation memory of a new customers, or the post-edits of professional translators.

4 Evaluation

4.1 Points of Comparison

Although the main scope of the paper is the description of the components and features of the MMT system, an experimental comparison is proposed against a few popular MT engines. In particular, two phrase-based MT systems, Moses and the Google's web translation service, and two neural MT systems.

4.1.1 Moses

A Moses (Koehn et al., 2007) engine was trained on the concatenation of all the available training corpora. Word alignment models were trained with FastAlign (Dyer et al., 2013) and a 5-gram language model was estimated by means of the KenLM toolkit (Heafield et al., 2013). Feature weights were tuned with batch MIRA (Cherry and Foster, 2012) to maximize BLEU on the pooled dev sets. No adaptation was performed.

4.1.2 GT

The Google web translation service (GT), one of the most used engines by the translation industry, was accessed through its public API[13] at the beginning of March 2017.

4.1.3 Neural MT Systems

We developed two neural MT systems using an in-house branch (Farajian et al., 2017) of the Nematus toolkit[14] implementing the encoder-decoder-attention model architecture by (Bahdanau et al., 2014). This first system is a *generic NMT* (gNMT) system trained on all the pooled training data. Then, following common practice (Luong and Manning, 2015), *adapted NMT* (aNMT) systems were trained for each domain by tuning the generic NMT system to the training data of each domain.

4.2 Experiments

We present experiments carried out on two translation tasks involving a collection of eight domain-specific corpora and two translation direction, English-French and English-German. When comparing the four types of MT systems, we consider translation quality (BLEU), training time, tuning time, and translation speed (seconds per sentence).

[13] https://www.googleapis.com/language/translate/v2

[14] https://github.com/rsennrich/nematus

		English-French			English-German		
		segments	source	target	segments	source	target
train	dom	1,332,972	17,581,131	19,297,282	1,004,214	15,772,744	14,427,002
	gen	4,255,604	92,363,974	101,236,914	4,165,505	104,489,832	98,381,272
dev	dom	3,527	46,640	52,484	3,073	37,023	35,187
test	dom	6,962	93,243	98,312	6,011	72,995	5,856
	out	4,503	104,831	111,050	5,168	111,331	106,443

Table 1: Statistics of training, dev and test sets for the English-French and English-German tasks: number of segments, source and target words. Figures refer to texts processed with the MMT modules.

	English-French					English-German				
	MMT	Moses	gNMT	aNMT	GT	MMT	Moses	gNMT	aNMT	GT
dom	62.48	61.78	49.23	63.00	43.62	48.27	48.51	37.41	48.95	31.37
out	30.11	28.93	33.28	–	36.47	19.08	16.84	22.82	–	27.13
training	1h	10h	100h	100h	–	1h	10h	100h	100h	–
tuning	1h	10h	–	10h	–	1h	10h	–	10h	–
translation	1s	1s	1s	1s	0.1s	1s	1s	1s	1s	0.1s

Table 2: Quality and speed performance of MMT and few competitor systems: BLEU scores on *dom* and *out* test sets for both English-French and English-German; overall time (order of magnitude in hours) to complete training and tuning ; the average time (order of magnitude in seconds) to translate one sentence.

4.2.1 Data

We consider eight publicly available parallel corpora as representatives of specific domains (*dom*): European Central Bank, Gnome, JRC-Acquis, KDE4, OpenOffice, PHP, Ubuntu, and UN documents.[15] To increase the training data, two additional generic corpora (*gen*) were added to the pool, namely CommonCrawl[16] and Europarl,[17] which are not considered for the evaluation.

Each domain-specific corpus was randomly partitioned into training, development and test portions. Additional test data from WMT[18] was prepared, in order to test the systems on out-of-domain data (*out*). Duplicate sentence pairs were removed from all dev and test sets. Statistics about training, dev and test sets are reported in Table 1.

4.2.2 Performance

Table 2 reports the translation quality performance (BLEU score), the overall computational cost for the compared systems to complete training and tuning, and the average time to translate one sentence in isolation. Time measures have to be taken with grain of salt because experiments were not run under very comparable conditions. For instance, neural MT systems were run on PCs equipped with GPU cards, while MMT and Moses were run only on multi-core CPUs. Hence, the order of magnitude, which are definitely reliable, is reported.

4.2.3 Discussion

In the following, we try to point out strengths and drawbacks of MMT against the other competitors.

MMT vs Moses MMT and Moses perform similarly as expected in terms of translation quality, because both share the same phrase-base MT paradigm. MMT performs better than Moses in the out-of-domain condition thanks to its adaptability feature (+1.18 and +2.24 gains). While translation speed is comparable, training and tuning of MMT is one order of magnitude faster.

MMT vs gNMT The BLEU scores on the out-of-domain condition (out) confirms that NMT has a better generalization capability than MMT (-3.17 and -3.74 losses), while MMT performs largely better when translating domain specific data (+13.25 and +10.86 gains). The training time is largely in favour of MMT, hundreds of hours for gNMT versus few hours for MMT. Translation speeds are actually comparable.

[15] UN corpus is used only for English-French. All corpora are available in http://opus.lingfil.uu.se

[16] http://www.statmt.org/wmt15/translation-task.html

[17] http://www.statmt.org/europarl/

[18] *newtest2014* and *newsdiscuss2015* for English-French, and *newstest2015* and *newtest2016* for English-German.

MMT vs aNMT After adaptation of gNMT to each specific domain, aNMT systems perform on pair with MMT on the in domain condition (dom). It is worth noticing, that under this condition distinct domain-specific NMT systems have to be tuned and translation should be run in a supervised way, by dispatching each test to the appropriate system. As a difference, MMT requires one system and does not require any domain labels at test time. The extra time needed to tune the aNMT systems on each domain is tens of hours.

MMT vs GT The comparison of MMT against Google Translate, show that the latter performs significantly better on the out of domain test (-6.35 and -8.05 losses), very likely due to the much larger training data available to the commercial system. On the contrary MMT perform largely better than GT on the in domain condition (+18.86 and +16.09 gains). With respect to translation speed, GT is significantly faster than MMT.

5 Conclusion

MMT aims to develop an innovative solution for the translation industry, by providing both better MT quality for post-editing as well as a better integration of MT with commercial CAT tools. MMT actually targets two use cases: (i) the enterprise use case, in which a language service provider or localisation department of a large company installs MMT to manage its translation workflow, and (ii) the translator use case, in which single translators install the MMT plugin in their favorite CAT tool and use MMT as their preferred source of suggestions/matches for their daily workflow.

For both scenarios MMT can provide machine translation technology that instantly adapts to the document to be translated and that quickly learns from the users' data – e.g. translation memories– and their post-editing work.

In this paper, we have presented an advanced phrase-based MT prototype of MMT, which shows competitive performance against similar approaches. In order to improve the generalization capability of MMT in operating conditions with a severe domain mismatch between testing and training data, work is in progress to integrate also neural MT in the final MMT release, which is planned for the end of 2017.

Acknowledgements

This work has been supported by the EC-funded project ModernMT (grant no. 645487).

References

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. 2014. "Neural machine translation by jointly learning to align and translate." *arXiv preprint arXiv:1409.0473*.

Cherry, Colin and George Foster. 2012. "Batch tuning strategies for statistical machine translation." *Proc. NAACL-HLT*, 427–436. Montreal, Canada.

Dyer, Chris, Victor Chahuneau, and Noah A. Smith. 2013. "A simple, fast, and effective reparameterization of IBM Model 2." *Proc. of NAACL*, 644–648. Atlanta, GA, USA.

Farajian, M. Amin, Marco Turchi, Matteo Negri, Nicola Bertoldi, and Marcello Federico. 2017. "Neural vs. Phrase-Based Machine Translation in a Multi-Domain Scenario." *Proc. of EACL*. Valencia, Spain.

Federico, Marcello and Nicola Bertoldi. 2001. "Broadcast news lm adaptation using contemporary texts." *Proc. of Eurospeech*, 239–242.

Germann, Ulrich. 2015. "Sampling phrase tables for the Moses statistical machine translation system." *The Prague Bulletin of Mathematical Linguistics*, 104:39–50.

Heafield, Kenneth, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013. "Scalable modified kneser-ney language model estimation." *Proc. of ACL (Volume 2: Short Papers)*, 690–696. Sofia, Bulgaria.

Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. "Moses: Open source toolkit for statistical machine translation." *Proc. of ACL (Interactive Poster and Demonstration Sessions)*, 177–180. Prague, Czech Republic.

Luong, Minh-Thang and Christopher D Manning. 2015. "Stanford Neural Machine Translation Systems for Spoken Language Domains." *Proc. of IWSLT*, 76–79. Da Nang, Vietnam.

Och, Franz Josef. 2003. "Minimum error rate training in statistical machine translation." *Proc. of ACL (Volume 1)*, 160–167. Sapporo, Japan.