Technical description of the First MT pilot

Martin Popel, Eleftherios Avramidis, Luís Gomes, Iliana Simova,
Ondřej Dušek, Gorka Labaka, Gertjan van Noord

2015-02-24

Chapter 1

Introduction

This document briefly describes D2.3, which is the package with software and models for running MT Pilot 1. A full report about the MT Pilot 1 will be provided in D2.4. Deliverable D2.4 (due to be available at M17) aims at describing MT Pilot 1, that is the entry-level deep MT systems for all language pairs in the project, including their empirical evaluation, as introduced in D2.3.

The overall goal of the QTLeap project is to produce high-quality translation between English (EN) and another language (X in the following text) by using deep linguistic information. The MT approaches we are using are hybrid, combining statistical and rule-based processing. In line with the objectives of the usage sceanrio (WP3), the focus of direction X→EN is aimed at supporting cross-lingual information retrieval.

QTLeap MT Pilot 1 is based on several software components. Chapter 2 describes the deep-MT system TectoMT, which is used in ten Pilot 1 directions: English ↔ Basque, Czech, Dutch, Portuguese and Spanish. Chapter 3 describes the software used in English ↔ German Pilot 1. Finally, Chapter 4 describes the system used for English ↔ Bulgarian Pilot 1.

The D2.3 package contains three top-level directories:

- `models` – pretrained translation models for all language pairs,
- `src` – source codes for the three software components (described here in Chapters 2–4),
- `translations` – Batches 1 and 2 of the QTLeapCorpus v1.0 plus their translations by Pilot 1 for all language pairs.

The source codes are versioned in the QTLeap git repository.[1] The models and translations are archived in the QTLeap ownCloud data repository.[2]

[1] `https://redmine.ms.mff.cuni.cz/projects/qtleap/`
[2] `http://qtleap.eu/repository`

Chapter 2

TectoMT

TectoMT is a structural machine translation system with deep transfer, first introduced by Žabokrtský et al. [2008]. The transfer phase of the system is based on Maximum Entropy context-sensitive translation models [Mareček et al., 2010] and Hidden Tree Markov Models [Žabokrtský and Popel, 2009]. It is implemented within Treex – a modular framework for Natural Language Processing.[1]

For QTLeap, we have significantly improved the existing EN→CS TectoMT system and, more importantly, implemented from scratch the remaining 9 directions (CS→EN, EN↔EU, EN↔ES, EN↔NL and EN↔PT).

To run TectoMT Pilots 1, follow the general instructions in Section 2.1 and then the language-specific instructions in Sections 2.2–2.6.

2.1 General Treex instructions

You need to install the Treex framework following the instructions at `http://ufal.cz/treex/install.html`. Treex is implemented in Perl programming language under Linux. During the installation you will need to download several dependencies from CPAN.[2]

To replicate the Pilot 1 results in future, you should use the source codes from the D2.3 package (`D2_3/src/tectomt`) instead of using the newest Subversion/Git repository checkout. You should also copy the translation models from `D2_3/models/` to your local "share" directory `data/models/translation/`, so for example, for CS→EN you will do

```
export TMT_ROOT=~/.treex/
cp D2_3/models/cs/cs-en/20141209_lemma.maxent.gz \
   $TMT_ROOT/share/data/models/translation/cs2en/
```

By default, local "share" is located in `~/.treex/share` (create this directory

[1]http://ufal.cz/treex
[2]https://metacpan.org/

if needed) and so `$TMT_ROOT` should point to `~/.treex/`, but you can override it by setting the `resource_path` in `~/.treex/config.yaml`.

2.1.1 Installing English analysis tools

The scenario for English analysis used in Pilot 1 needs Morce tagger, NameTag named entity recognizer, MST parser and NADA coreference resolver. All other blocks in the English analysis scenario need no installation (they are pure-Perl modules).

Morce tagger

To install the Morce tagger, download the models from `http://ufallab.ms.mff.cuni.cz/tectomt/share/data/models/morce/en/` to your local share and then:

```
SVN_TRUNK=https://svn.ms.mff.cuni.cz/svn/tectomt_devel/trunk
# password is "public"
svn --username public export $SVN_TRUNK/libs/packaged /tmp/packaged
cd /tmp/packaged/Morce-English
perl Build.PL && ./Build && ./Build test
./Build install --prefix $HOME/perl5/lib/perl5
```

You can install the module into any path in your `$PERL5LIB` (instead of the suggested `~/perl5/lib/perl5`).

Instead of Morce tagger, you can use a faster MorphoDiTa tagger (substitute `W2A::EN::TagMorce` with `W2A::EN::TagMorphoDiTa`, the differences in the resulting BLEU scores are small).

NameTag NER

To get NameTag NER, simply install the `Ufal::NameTag` module from CPAN using `cpanm Ufal::NameTag`. Just make sure you have a C++11 compiler (`g++` 4.7 or newer).

MST parser

The MST parser jar file and model should be automatically downloaded when you run it for the first time. You just need `java` installed.

NADA coreference resolver

To install the NADA coreference resolver:

```
svn --username public export \
    $SVN_TRUNK/install/tool_installation/NADA /tmp/NADA
cd /tmp/NADA && perl Makefile.PL && make && make install
```

Instead of installing NADA, you can delete `A2T::EN::MarkReferentialIt` from the scenario, the differences in BLEU on QTLeapCorpus translation are negligible.

2.1.2 Installing English synthesis tools

The English synthesis pipeline requires Flect and MorphoDiTa morphological generators.

Flect (`https://github.com/UFAL-DSG/flect`) is automatically installed from GitHub upon first use, but it requires Python 2.7 (`https://www.python.org/download/releases/2.7/`) and Scikit-Learn (`http://scikit-learn.org/`) to be installed. In Ubuntu 14.04, you can run just the following command to install the dependencies:

```
sudo apt-get install python2.7-sklearn
```

MorphoDiTa (`http://ufal.mff.cuni.cz/morphodita`) can be installed from CPAN using `cpanm Ufal::MorphoDiTa`. It requires C++11 to compile. You may optionally remove it from the synthesis scenario, which yields just slightly worse BLEU scores.

2.1.3 Running and training Pilots 1

The relevant makefiles for running and training (on new parallel data) Pilots 1 are in `D2_3/src/tectomt/devel/qtleap/`. To replicate Pilot 1 results with the pre-trained models for a given direction (e.g. CS→EN and EN→CS) and a given test dataset (QTLeapCorpus Batch1, resp. Batch2), run:

```
make translate eval TRANSL_PAIR=cs_en TEST_DATASET=Batch1q
make translate eval TRANSL_PAIR=en_cs TEST_DATASET=Batch1a
```

Run `make help` to see a brief introduction to the capabilities of the Makefile (e.g. comparing different runs of the same system).

To re-train the translation models (and completely re-analyze training parallel data), e.g. for CS→EN, remove the pre-trained translation models from your "share", edit the `TRAIN_DATA` variable in `devel/qtleap/conf/cs_en.conf` and run

```
make transl_models TRANSL_PAIR=cs_en
```

Both training and running the Pilots 1 expects a parallelization via an SGE cluster by default. Use LRC=0 for running in one thread locally. See QTLM tool (described in Section 2.6), which can be used as an alternative to the makefiles.

2.2 EN↔CS system

The scenario for Czech analysis used in Pilot 1, needs MorphoDiTa tagger, NameTag named entity recognizer and an Czech-adapted version of MST parser.

Both MorphoDiTa and NameTag can be installed from CPAN using `cpanm Ufal::MorphoDiTa` and `cpanm Ufal::NameTag`. The Czech models will be downloaded automatically when you run it for the first time.

The jar file and model needed by `W2A::CS::ParseMSTAdapted` will be downloaded automatically as well.

2.3 EN↔EU system

The Basque scenarios make use of analysis tools developed by UPV/EHU. To install such us tools download them from `http://ixa2.si.ehu.eus/glabaka/QTLeap/tools/eu/` and unpack them to the `installed_tools/parser/` subdirectory of your local "share" directory.

2.4 EN↔ES system

The Spanish analysis scenario make use of IXA-pipes (`ixa2.si.ehu.es/ixa-pipes/`) tools developed by UPV/EHU. To install such us tools download them from `http://ixa2.si.ehu.eus/glabaka/QTLeap/tools/eu/` and unpack them to the `installed_tools/parser/` subdirectory of your local "share" directory.

2.5 EN↔NL system

The Dutch analysis and synthesis scenarios combine Treex with Alpino, a parser and generator developed by RUG. To install Alpino, download the latest binary package from `http://www.let.rug.nl/~vannoord/alp/Alpino/binary/versions/` and unpack it to the `installed_tools/parser/` subdirectory of your local "share" directory. Note that the binary package includes all sources - it is for convencience that the binaries are included in the package as well.

2.6 EN↔PT system

The EN-PT translation system is based on the TectoMT/Treex framework. All Treex Blocks used in the English-Portuguese and Portuguese-English translation scenarios have been committed into the main SVN repository. Some blocks require a working network connection and an access key.

Thus, the scenarios for EN-PT and PT-EN translation should include the following lines at the beginning:

```
Util::SetGlobal lxsuite_host=194.117.45.198 lxsuite_port=10000
Util::SetGlobal lxsuite_key=nlx.qtleap.13417612987549387402
```

The FCUL team has developed qtlm (QTLeap Manager), a tool intended to ease the development workflow on a single machine, as the original infrastructure is tailored to be deployed on a cluster. To use this tool please download and extract `qtlm_r173.tgz` from the shared repository and refer to `doc/ReadMe.html`.

Chapter 3

System combination: EN↔DE system

The goal of the provided software is to receive translated input from Moses and Lucy and perform sentence-level system combination, by choosing the best output for each sentence, given several deep fluency and adequacy criteria. This way, the transfer-based system output is used only if it is better than the statistical baseline. Preliminary tests for German suggest that this is the case for 30%-40% of the sentences.

Package D2.3 contains all the models (in `D2_3/models/de/`) and source codes and jar files (in `D2_3/src/qualitative/`) needed to replicate the EN↔DE Pilot 1.

3.1 Installation

The software is implemented in Python 2.7 and has been developed and tested for operation in a Linux operating system. The code has to be downloaded and the Python path needs to be set to the `/src` directory, where all python scripts, modules and packages reside. Much of the functionality is provided by several publically available Python extensions, which need to be installed prior to any execution. The vast majority of these are provided by the Python `pip` package management system, so it is enough to run the respective `pip install` commands for the packages detailed in `INSTALL.txt`, These installations can easily take place on the user's home folder, without requiring root access (e.g. in experiment server environments).

The toolkit interacts with several java applications whose 'jar' and class files have to be placed in the directory `/lib`. An installation script that automatically downloads all required java dependencies is provided. Additionally, one needs to execute externally the LM server by Nitin Madnani. The Language

Model scores are provided by using LM server, which wraps around SRILM Stolcke [2002] and needs to be compiled and executed separately. In our imminent plans is to remove this dependency and include KenLM Heafield [2011]

The system receives input from Moses and Lucy, so the systems have to be installed separately. The installation of these systems will not be covered here. For Moses, the translation model produced as a baseline is re-used.

3.2 Resources and Configuration

The quality features for the given sentences are generated through a pipeline of NLP analysis tools. Many of these tools require specific resources to be acquired prior to the execution of the program. In particular, for the out-of-the-box installation and pre-trained models, one needs a language model, a PCFG grammar and a truecaser model for the source and target languages.

All file locations and several other parameters (e.g. the translation language direction) can be specified in one or more complementary configuration files. Sample configuration files are provided in `/cfg/autoranking` and can be modified accordingly to fit the user's specific installation. The configuration files may also include references to many language-specific resources; the program will only use the ones which are relevant to the chosen languages.

The reason for allowing many configuration files is that one may want to split the configuration parameters depending on the environment, i.e. some settings may be generic and applicable to all computational servers, whereas some others may change from machine to machine.

3.3 Execution

The current version of the system requires the user to proceed with each step separately. Later versions will provide a bundled script with the ability to perform many actions with direct interaction with the translation systems. The steps currently required are:

1. Obtain the outputs from Lucy and Moses
2. Use `/src/support/preprocessing/jcml/manytxt2jcml.py` to wrap all outputs in one XML file
3. Use `/src/app/hybrid/annotate_batch.py` to generate features on the system outputs from the XML file
4. Use `/src/app/hybrid/test.py` to load the relevant model, perform selection and automatic scoring on the XML file

Chapter 4

Deep Factored MT: EN⟶BG system

The backbone of this machine translation system is a factored model (Koehn and Hoang [2007]) built with the Moses open source toolkit, which allows for additional linguistic information to be incorporated into the translation process. The system uses a language model created with the help of the language modeling toolkit SRILM, Stolcke [2002].

In order to perform factored translation with the Moses open source toolkit, the input English or Bulgarian text needs to be preprocessed to encode various linguistic information as factors. The following lines describe the two pipelines used to prepare data so that it is a suitable input for factored translation. The operations which are performed include tokenization, sentence splitting, lemmatization, part-of-speech tagging, dependency parsing, and factor generation.

Package D2.3 contains all the models (in `D2_3/models/bg/`) and source codes and jar files (in `D2_3/src/DeepFactoredMT/`) needed to replicate the EN↔BG Pilot 1.

4.1 Prerequisites

The two pipelines for producing factored input for the Moses factored model are distributed as self-contained jar files for ease of use, and no installation is required. The only requirement which should be satisfied to run the jar files is Java 1.7. They have been tested under Linux (RedHat and LinuxMint), and Windows (Windows 7).

The projects used to produce the jar files have a similar general structure, with all required libraries located in folder `lib/` and all required resources located in folder `resources/` of the respective projects. In case the pipeline jar files and resources are not in the same location, the paths to the various language-specific resources (sub-folders in folder resources) needs to be specified in the provided configuration files.

The Moses open source toolkit and the SRILM language modelling toolkit need to be installed in addition.

4.2 Execution

1. Run the Bulgarian pipeline

   - display usage information
     `java -jar smt-btbpipe-1.0.0.jar --help`
   - run the pipeline on some plain text input, and extract factors 0 (word form), 2 (part of speech), and 8 (variable type of MRS elementary predicate). The number of lines in the input will correspond to the number of lines in the output if the option `--preserve-lines` is used. This is important in case of parallel data. If this option is omitted, the result will be a one sentence per line factored output.
     ```
     java -jar smt-btbpipe-1.0.0.jar --input plain-text-input.txt
     --output factored-output.txt --factors 0,2,8 --preserve-lines
     ```

2. Run the English pipeline

   - display usage information
     `java -jar smt-ixapipe-1.0.0.jar --help`
   - run the pipeline on some plain text input, and extract factors 0 (word form), 2 (part of speech), and 8 (variable type of MRS elementary predicate). The number of lines in the input will correspond to the number of lines in the output if the option `--preserve-lines` is used. This is important in case of parallel data. If this option is omitted, the result will be a one sentence per line factored output.
     ```
     java -jar smt-ixapipe-1.0.0.jar --input plain-text-input.txt
     --output factored-output.txt --factors 0,2,8 --preserve-lines
     ```

3. Translate the factored data with the provided Moses factored model

   - Convert the input to lowercase with the Moses script `lowercase.perl`
   - Filter the provided model for this test set with the Moses script `filter-model-given-input.pl`
   - Translate the test set with the filtered model
     ```
     mosesdecoder/bin/moses -f filtered-model/moses.ini
     < factored-input > translation-output
     ```
   - Recase and detokenize the translation output

Bibliography

Kenneth Heafield. KenLM: faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, United Kingdom, 2011. Association for Computational Linguistics.

Philipp Koehn and Hieu Hoang. Factored translation models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 868–876. Association for Computational Linguistics, 2007.

David Mareček, Martin Popel, and Zdeněk Žabokrtský. Maximum Entropy Translation Model in Dependency-Based MT Framework. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and Metrics-MATR*, pages 201–206, Uppsala, Sweden, July 2010. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W10-1730`.

Andreas Stolcke. SRILM – an extensible language modeling toolkit. In *Proceedings of International Conference on Spoken Language Processing*, pages 901–904, Denver, CO, USA, 2002. International Speech Communication Association.

Z. Žabokrtský, J. Ptáček, and P. Pajas. TectoMT: highly modular MT system with tectogrammatics used as transfer layer. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 167–170. Association for Computational Linguistics, 2008.

Zdeněk Žabokrtský and Martin Popel. Hidden Markov Tree Model in Dependency-based Machine Translation. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, ACLShort '09, pages 145–148, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics. URL `http://dl.acm.org/citation.cfm?id=1667583.1667628`.