Technical description of MT Pilot3

2016-09-01

Chapter 1

Introduction

This document briefly describes D2.10, which is the package with software and models for running MT Pilot 3. A full report about the MT Pilot 3 will be provided in D2.11, which aims at describing MT Pilot 3 (third MT pilot, enhanced with deep semantics), including its empirical evaluation.

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 scenario (WP3), the focus of direction X→EN is aimed at supporting cross-lingual information retrieval. The direction EN→X (also called outbound direction) is aimed for presenting the translated answers to the users, so here the focus is not only adequacy, but also fluency.

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

The D2.10 package contains three top-level directories:

- `models` – pretrained translation models for all language pairs,
- `src` – source codes for the four software components: TectoMT, Qualitative, NeuralMonkey and DeepFactoredMoses (described here in Chapters 2, 3.2, 3 and 4),
- `translations` – Batches 1, 2, 3, 4 and the News test set of the QTLeap Corpus plus their translations by Pilots 0, 1, 2, 3 for all language pairs

The source codes were taken from QTLeap git version control systems.[1] The translations are also versioned in a git repository.[2] The models are archived in the QTLeap ownCloud data repository.[3]

[1] `https://redmine.ms.mff.cuni.cz/projects/qtleap/` and `https://github.com/ufal/treex`

[2] `https://redmine.ms.mff.cuni.cz/projects/qtleap-corpus`

[3] `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 context-sensitive Maximum Entropy translation models [Mareček et al., 2010] or alternatively Vowpal Wabbit translation models 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 3, 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 3 results in future, you should use the source codes from the D2.10 package (`D2_10/src/tectomt`, `D2_10/src/qtleap`) or you can run `git clone https://github.com/ufal/treex; git checkout QTLeap_Pilot3`. The pretrained translation models will be downloaded automatically when first used, but you can also copy the translation models from `D2_10/models/` to your local "share" directory `data/models/translation/`. So for example, for EN→NL you will do

```
export TMT_ROOT=~/.treex/
cp -R D2_10/models/en-nl/ $TMT_ROOT/share/data/models/translation/en2nl
```

By default, local "share" is located in `~/.treex/share` (create this directory if needed) and so `$TMT_ROOT` should point to `~/.treex/`, but you can override it by setting the `resource_path` in `~/.treex/config.yaml`.

[1] http://ufal.cz/treex

[2] https://metacpan.org/

2.1.1 Installing English analysis tools

The scenario for English analysis used in Pilot 3 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 3

The relevant makefiles for running and training (on new parallel data) Pilots 3 are in `D2_10/src/qtleap/translate/`. To replicate Pilot 3 results with the pre-trained models for a given direction (e.g. `CS`→`EN` and `EN`→`CS`) and a given test dataset (QTLeap Corpus Batch4), run:

```
cd cs-en/batch4q && make translate eval
cd en-cs/batch4a && make translate eval
```

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 `D2_10/src/qtleap/cuni_train/conf/cs_en.conf` and run

```
make transl_models TRANSL_PAIR=cs_en
```

Both training and running the Pilots 3 expect a parallelization via an SGE cluster by default. Use LRC=0 for running in one thread locally.

2.2 EN↔CS system

The scenario for Czech analysis used in Pilot 3, needs MorphoDiTa tagger, NameTag named entity recognizer and a 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.

Vowpal Wabbit needs to be installed following the instructions at `https://github.com/JohnLangford/vowpal_wabbit` and `vw` executable must be in `PATH`.

2.3 EN↔EU system

The Basque scenarios makes use of the analysis tools developed by UPV/EHU. The PoS tagger can be downloaded from `http://ixa2.si.ehu.es/ixa-pipes/eu/ixa-pipe-pos-eu.tar.gz` and it needs to be compiled and installed into the installed_tools/eustagger/ subdirectory of your local "share" directory. The dependency parser doesn't need compilation and, after downloading it from `http://ixa2.si.ehu.es/ixa-pipes/eu/ixa-pipe-dep-eu.tar.gz`, has to be unpacked into the installed_tools/ixa-pipe/EU subdirectory of your local "share" directory.

2.4 EN↔ES system

The Spanish analysis scenario make use of IXA-pipes (`http://ixa2.si.ehu.es/ixa-pipes/`) tools developed by UPV/EHU. To install the tools, download them from `http://ixa2.si.ehu.eus/glabaka/QTLeap/es-tools.tgz` and unpack them to the installed_tools/ixa-pipe 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. If Alpino is not present, it will be downloaded from the RUG website[3] and unpacked directly by Treex/TectoMT upon first use. Note that Alpino packages are intended for Linux x86-64 systems only.

To install Alpino manually, download the latest binary package from `http://www.let.rug.nl/~vannoord/alp/Alpino/versions/binary/` 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 convenience that the binaries are included in the package as well.

2.6 EN↔PT system

The Portuguese analysis and synthesis scenarios require remote access to the LX-Suite XML-RPC service, which is configured by a file named `.lxsuite2` (note the dot at the beginning) which should exist in the user's home directory.

The configuration file has a simple key-value format, as shown below:

```
host=194.117.45.198
port=10010
key=nlx.qtleap.13417612987549387402
```

The key presented above grants access to the service, although limited to 12 simultaneous connections (globally).

[3]`http://www.let.rug.nl/~vannoord/alp/Alpino/versions/binary/`

Chapter 3

Qualitative: EN↔DE system

The German Pilot3 is the hybrid system called "Qualitative" documented in Deliverable D2.11. The central component is the sentence selection mechanism that receives input from the different SMT and RbMT components and performs sentence-level system combination by choosing the best output for each sentence, given several deep fluency and adequacy criteria. Below, we document its installation and execution.

3.1 Selection Mechanism

3.1.1 Installation

Download software: Download the latest version from GitHub. The checkout command switches to the exact version used for the delivery of Pilot 3, while newer versions may still be available in the future.

```
cd ~
git clone https://github.com/lefterav/qualitative.git
cd qualitative
git checkout QTLeap_Pilot3
```

Installation of necessary system software: The software is implemented in Python 2.7 and has been developed and tested for operation in a Debian/Ubuntu Linux operating system. The following commands install the necessary system software. Administration (root) access is required for this.

```
#install java 8
sudo apt-add-repository ppa:openjdk-r/ppa
sudo apt-get update
sudo apt-get install openjdk-8-jdk

#install requirements for compiling python libraries
sudo apt-get install python-dev g++ build-essential
sudo apt-get install python-pip libblas-dev liblapack-dev gfortran
sudo pip install --upgrade pip
```

Python virtual environment: A virtual environment encloses all necessary Python configuration, so that you don't risk the Python installation of your system:

```
cd ~
venv qualitative_venv
cd qualitative_venv
echo "export PYTHONPATH:\$PYTHONPATH:$HOME/qualitative/src" >> bin/activate
source bin/activate
```

Installation of Python libraries: Much of the functionality is provided by several publicly 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:

```
pip install setuptools
pip install nltk==2.0.5
easy_install -U distribute
# enter the directory where qualitative has been installed
cd qualitative
pip install -r requirements.txt
pip install https://github.com/kpu/kenlm/archive/master.zip
```

Installation of libraries and resources: 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, the required resources for running the models are provided.

```
cd ~/qualitative
cd lib/
bash install.sh
cd ../res/
bash install sh
```

After this, you should modify the files `config/autoranking/features.cfg` so that it points correctly to the locations of the downloaded resource files.
The system receives input from Moses (Pilot 0), Lucy and NeuralMonkey that have to be installed separately. If the server mode of these systems is installed, Qualitative can communicate and fetch translations from these servers. The location and the parameters of these servers should be provided in `config/autoranking/mt.cfg` following the given examples.

3.1.2 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 `config/autoranking` and can be modified accordingly to fit 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.

3.1.3 Execution

For Pilot3, we have created a bundled script with the ability to perform many actions with direct interaction with the translation systems. The script `app/hybrid/translate_selection.py` communicates with the provided engines, fetches the translations and performs the selection. The input can be passed with a text file and the output can be stored in another text file. Verbose input with all inherent annotation is produced in a XML file. The exact syntax is explained with the parameter `--help`. A sample commandline is given below:

```
python src/app/hybrid/translate_selection.py \
--config config/autoranking/features.cfg config/autoranking/mt.cfg \
--engines Moses Lucy NeuralMonkey \
--ranking_model 0.model.dump \
--text_output batch2a.de \
--input batch2a_en_v1.1.txt
```

3.2 Neural MT System

We included a neural component called NeuralMonkey in the Pilot 3 systems. The implementation of the neural machine translation (NMT) system follows the encoder-decoder design with the attention mechanism [Cho et al., 2014, Bahdanau et al., 2014]. We also incorporated the Byte-pair encoding preprocessing to overcome the large open-vocabulary problem [Sennrich et al., 2015].

To run the translation in server mode, run:

```
cd /neuralmonkey
bin/neuralmonkey-server <MODEL_INI> [--host <HOST> --port <PORT>]
```

where the `MODEL_INI` argument is the path to the configuration file. By default, the server runs on localhost. Set the host to 0.0.0.0 to make the server visible from the network.

For running the translation in batch mode, run:

```
cd /neuralmonkey
bin/neuralmonkey-run <MODEL_INI> <DATA_INI>
```

where `MODEL_INI` is the model configuration file (as above), and the `DATA_INI` is a configruation file containing information about the datasets you need to translate.

The software is publicly available on `http://github.com/ufal/neuralmonkey`. The documentation can be found on `http://neural-monkey.readthedocs.io`.

Chapter 4

DeepFactoredMoses: EN↔BG system

4.1 BG→EN system

To run the pipeline from the jar file, use the following command:

```
java -cp .:btbpipe-bg.jar:lib/* org.bultreebank.btbpipe.main.PipeRunner
          --config-file [path to config file]
          --in-format [txt, xml1 or xml2]
          --inputFile [path to input file]
```

or

```
java -cp .:btbpipe-bg.jar:lib/* org.bultreebank.btbpipe.main.PipeRunner
          --config-file [path to config file]
          --in-format [txt, xml1 or xml2]
          --inputString [string to be processed]
```

Depending on the input option given, the pipeline will perform different processing steps.

- the ”txt” work mode calls the Mate processors (tokenization, POS tagger, parser)
- the ”xml1” work mode calls a module for the preparation of factors for a factor-based Moses model
- the ”xml2” work mode calls a module which transfers information from the pipeline for the source language, applies constraints and modifies the result from the factor-based Moses

IMPORTANT:

- The folders ./resources and ./lib must exist on the same level as the jar file in order for it to run.

- The file config.properties must list the locations of the relevant resources. In the case of the UKB tool, the path provided should point to the folder where all UKB folders are located. Namely ./ukb_wsd should contain the following subfolders: bin, input, lkb_sources, naf_ukb, ukb. Consult the NAF UKB installation documentation for further details (`https://github.com/asoroa/naf_ukb/blob/master/INSTALL`; note that while all the rest of the folders are described in the documentation, the folder ./ukb_wsd/input must be created manually by the user). The resources whose absolute paths must be given in the configuration file are:
- the knowledge base or relations (”knowledgeBase”; the best available graph is distributed in the folder /pilot3/en/ukb_pilot3_bg/)
- the dictionary file (”dictionary”; available in /pilot3/en/ukb_pilot3_bg/)
- the POS mapping file (”posMapping”; available in /pilot3/en/ukb_pilot3_bg/)

4.2 EN→BG system

To run the pipeline from the jar file, use the following command:

```
java -cp .:btbpipe-en.jar:lib/* org.bultreebank.btbpipe.main.PipeRunner
          --config-file [path to config file]
          --in-format [txt, xml1 or xml2]
          --inputFile [path to input file]
```

or

```
java -cp .:btbpipe-en.jar:lib/* org.bultreebank.btbpipe.main.PipeRunner
          --config-file [path to config file]
          --in-format [txt, xml1 or xml2]
          --inputString [string to be processed]
```

Depending on the input option given, the pipeline will perform different processing steps.

- the ”txt” work mode calls the CoreNLP processors (tokenization, POS tagger, parser)
- the ”xml1” work mode calls a module for the preparation of factors for a factor-based Moses model
- the ”xml2” work mode calls a module which transfers information from the pipeline for the source language, applies constraints and modifies the result from the factor-based Moses

IMPORTANT:

- The folders ./resources and ./lib must exist on the same level as the jar file in order for it to run.

- The file config.properties must list the locations of the relevant resources. In the case of the UKB tool, the path provided should point to the folder where all UKB folders are located. Namely ./ukb_wsd should contain the following subfolders: bin, input, lkb_sources, naf_ukb, ukb. Consult the NAF UKB installation documentation for further details (`https://github.com/asoroa/naf_ukb/blob/master/INSTALL`; note that while all the rest of the folders are described in the documentation, the folder ./ukb_wsd/input must be created manually by the user). The resources whose absolute paths must be given in the configuration file are:

- the knowledge base or relations (”knowledgeBase”; the best available graph is distributed in the folder /pilot3/en/ukb_pilot3_en/)

- the dictionary file (”dictionary”; available in /pilot3/en/ukb_pilot3_en/)

- the POS mapping file (”posMapping”; available in /pilot3/en/ukb_pilot3_en/)

Bibliography

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014. URL `http://arxiv.org/abs/1409.0473`.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014. URL `http://arxiv.org/abs/1406.1078`.

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 MetricsMATR*, pages 201–206, Uppsala, Sweden, July 2010. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W10-1730`.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *CoRR*, abs/1508.07909, 2015. URL `http://arxiv.org/abs/1508.07909`.

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`.