Technical description of MT Pilot2

2015-10-05

Chapter 1

Introduction

This document briefly describes D2.7, which is the package with software and models for running MT Pilot 2. A full report about the MT Pilot 2 will be provided in D2.8. Deliverable D2.8 aims at describing MT Pilot 2 (second deep MT pilot, enhanced with lexical 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 2 is based on several software components. Chapter 2 describes the deep-MT system TectoMT, which is used in ten Pilot 2 directions: English ↔ Basque, Czech, Dutch, Portuguese and Spanish. Chapter 3 describes the software used in English ↔ German Pilot 2. Finally, Chapter 4 describes the system used for English ↔ Bulgarian Pilot 2.

The D2.7 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, 2 and 3 and the News test set of the QTLeap Corpus plus their translations by Pilots 0, 1 and 2 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 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 2, 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 2 results in future, you should use the source codes from the D2.7 package (`D2_7/src/tectomt`, `D2_7/src/qtleap`) or you can run `git clone https://github.com/ufal/treex; git checkout QTLeap_Pilot2`. The pretrained translation models will be downloaded automatically when first used, but you can also copy the translation models from `D2_7/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_7/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 2 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 2

The relevant makefiles for running and training (on new parallel data) Pilots 2 are in `D2_7/src/qtleap/translate/`. To replicate Pilot 2 results with the pre-trained models for a given direction (e.g. `CS`→`EN` and `EN`→`CS`) and a given test dataset (QTLeap Corpus Batch1), run:

```
cd cs-en/batch1q && make translate eval
cd en-cs/batch1q && 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_7/src/qtleap/cuni_train/conf/cs_en.conf` and run

```
make transl_models TRANSL_PAIR=cs_en
```

Both training and running the Pilots 2 expect 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 2, 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.

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/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 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/binary/versions/`

Chapter 3

System combination: EN↔DE system

The German Pilot2 is the hybrid system documented in Deliverable D2.7. 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. Pilot2 includes a word-sense disambiguation component that is also documented below. In the end of this section, we describe the configuration of the WSD-Moses engine that is part of this hybrid architecture.

3.1 Selection Mechanism

The QTLeap redmine code git contains all the models and source codes and jar files (in `Pilot2/de/en-de/selection_mechanism/models/de/` and `selection_mechanism/qualitative/`, respectively) needed to replicate the selection mechanism of the EN→DE Pilot2.

3.1.1 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 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 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.

The system receives input from Moses and Lucy that have to be installed separately.

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 `/cfg/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 Pilot2, we have created a bundled script with the ability to perform many actions with direct interaction with the translation systems. The script `wsd_xmlrpcserver.py` controls the respective processes and provides the Pilot2 web service. Input parameters are host, port, WSD-Moses URL, Lucy URL, LucyMoses-URL, WSD-server URL, source language, target language, classifier filename, and name of the config file. An example call for the German Pilot2 is provided below:

```
/selection_mechanism/qualitative/src/app/hybrid/wsd_xmlrpcserver.py
blade-2.dfki.uni-sb.de
31240
http://blade-1.dfki.uni-sb.de:9400
http://msv-3251.sb.dfki.de:8080/AutoTranslateRS/V1.2/mtrans/exec
http://lns-87009.dfki.uni-sb.de:9200
http://blade-1.dfki.uni-sb.de:32008
en
de
/selection_mechanism/models/models/en-de/0.model.dump --reverse
/selection_mechanism/qualitative/config/autoranking/annotation.cfg
/selection_mechanism/qualitative/config/autoranking/annotation.en-de.cfg
```

3.2 Word Sense Disambiguation System

Pilot2 uses a REST server implementing the Word Sense Disambiguation system by Weißenborn et al. [2015] that assigns BabelNet-senses to nouns and has recently shown improvements over state-of-the-art results on several corpora. The software to run the

REST server is provided in the QTLeap git repository in `Pilot2/de/en-de/wsd/mood/`. As the trained model is about 15G in size, it can be provided on request. Babelnet needs to be obtained separately. The WSD server takes as starting parameters the host and port. An example call is provided below:

```
cd /wsd/mood/smt_lib
bash run_rest.sh  blade-1.dfki.uni-sb.de 32008
```

3.3 WSD-Moses Engine

For Pilot 2, the simple phrase-based SMT of Pilot 1 is replaced by word-sense-disambiguated phrase-based SMT. It is a phrase-based system with two decoding paths, one basic and one alternative. In the basic path, all nouns of the source language (English) have been annotated by a Word Sense Disambiguation system [Weißenborn et al., 2015] that assigns BabelNet-senses. For use in QTLeap, we have developed and installed a REST-server version of the WSD system for online use (see above).

The Moses ini-files can be found in the code git at `Pilot2/de/en-de/wsd-moses/`. The WSD-preprocessed training corpora are found on the QTLeap OwnCloud (`Pilot2/systems/en-de/WSD-Moses/Data/`).

Chapter 4

Deep Factored MT: EN↔BG system

4.1 BG→EN system

The Bulgarian to English Pilot2 system performs several NLP tasks, implemented as a Bulgarian pipeline, and then it calls the Moses system with the appropriate model.

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

```
java -cp .:btbpipeline.jar:lib/*
         org.bultreebank.btbpipe.main.PipeRunner
         --input [input file or folder]
         --modules [comma-separated list of modules]
         --output [output file or folder]
         --out-format naf
         --in-format txt
```

The availabe modules are:

- preprocess (tokenization + pos tagging + some additional preprocessing; OBLIGATORY)
- parse (syntactic parsing, using the Mate parser)
- coref (coreference resolution and named entity recognition)
- wsd (word sense disambiguation, using the UKB tool)
- factorsTrain (module for generating factors for running the Moses system in training mode)
- factorsTest (module for generating factors for running the Moses system in testing mode)

Currently the system supports only txt input and naf output formats.
Command for producing the necessary output for Moses training:

```
java -cp .:btbpipeline.jar:lib/*
         org.bultreebank.btbpipe.main.PipeRunner
         --input [input file or folder]
         --modules preprocess,wsd,factorsTrain
         --output [output file or folder]
         --out-format naf
         --in-format txt
```

Command for producing the necessary output for Moses testing:

```
java -cp .:btbpipeline.jar:lib/*
         org.bultreebank.btbpipe.main.PipeRunner
         --input [input file or folder]
         --modules preprocess,wsd,factorsTest
         --output [output file or folder]
         --out-format naf
         --in-format txt
```

IMPORTANT:

- The folders ./resources and ./lib must exist at the same level as the jar file in order for it to run.
- The file config.properties (also at the same level as the jar) 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, the folder "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 Dictionary file ("dictionary")
- the POS mapping file ("posMapping")

4.2 EN→BG system

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

```
java -jar IxaPipeWrapperNaf.jar
          --input [input file or folder]
          --output [output file or folder]
```

```
--preserve-lines
[--factors]
[--trainMode]
```

Including the "factors" parameter in the command configures the pipeline to output factors for the Moses system. By default, it will output NAF documents. Including the "trainMode" parameter in the command forces the pipeline to output factored format suitable for the training of the Moses system. By default, it is run the test mode. If "–factors" is not included, then "–trainMode" is irrelevant.

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 dictionary file ("dictionary")
- the POS mapping file ("posMapping")

Bibliography

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`.

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.

Dirk Weißenborn, Leonhard Hennig, Feiyu Xu, and Hans Uszkoreit. Multi-objective optimization for the joint disambiguation of nouns and named entities. In *53nd Annual Meeting of the Association for Computational Linguistics, July*. ACL, 2015.

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`.