Adaptive MT server

User manual and installation guide

(version 1.0)

(19 November 2014)

Adaptive MT server is a translation web service able to return a list of translation alternatives for a given input, automatically generated by an MT engine, and to adapt on-the-fly the MT engine models according to any provided feedback, consisting of a parallel sentence pair, i.e. a source text and its translation. Adaptive MT also computes the quality estimation of the automatic translations, and adapts these scores to the user feedback. Adaptive MT server also supports terminology, either pre-loaded or inserted on-the-fly.

Users of this toolkit might cite the official website of **Adaptive MT server**, containing this manual, source code, and examples is

www.mt4cat.org/software/adaptive-mt-server

Adaptive MT server has been partially funded by the EU project MateCat (ICT-2011.4.2-287688), and is distributed under the GNU Lesser General Public License (LGPL).

Index

1. Introducing adaptive MT server

1.1 Overview

Adaptive MT server is a translation web service, which returns, for a given input, a translation automatically generated by a Machine Translation (MT) engine. For each translation, adaptive MT server also returns additional information like an estimation of its quality, word and phrase alignment. MT server is **adaptive** because it adapts the MT and quality estimation (QE) models according to any user-provided feedback, consisting of parallel sentence pair, i.e. a source text and any user-approved translation.

Currently, the MT server is connected to the well-known state-of-the-art phrase-based SMT Moses Toolkit[1], supporting the online adaptation via cache-based models. In order to perform the online adaptation of the models, adaptive MT server relies on a word aligner. Currently, it supports both onlineMGIZA++[2] and Bitext-tokaligner[3] (Blain et al., 2012). The estimate of the translation quality is performed by means of AQET[4] (Turchi et al., 2014).

MT server was originally developed under the EC funded Matecat[5] Project (grant agreement n. 287688) to integrate Machine Translation and Translation Memory technologies into the developed Computer Assisted Translation (CAT) Tool, called MateCat Tool[6] (Federico et al., 2014).

---

[1] www.statmt.org/moses

[2] www.mt4cat.org/software/onlinemgiza

[3] github.com/fredblain/bitext-tokaligner

[4] www.mt4cat.org/software/aqet

[5] www.matecat.org

[6] ibidem, footnote 5.

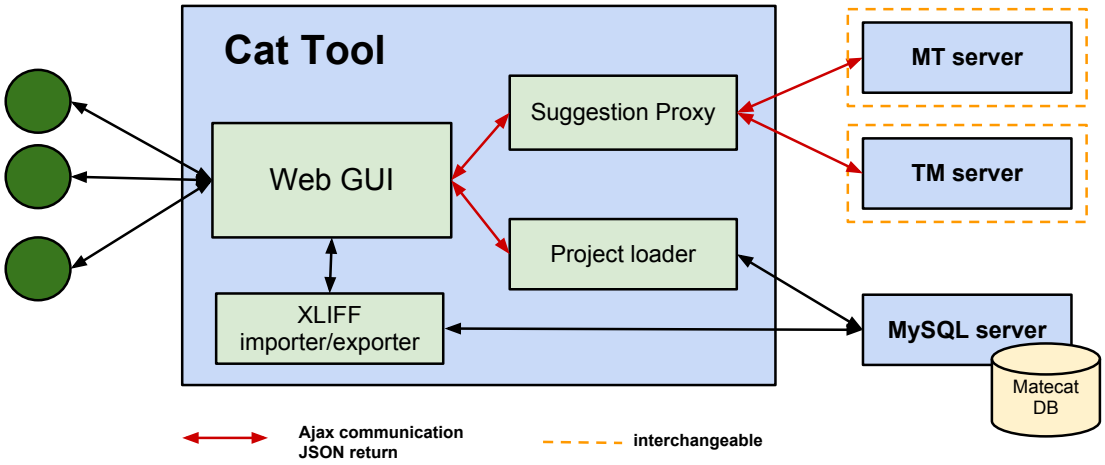

FIGURE 1: ARCHITECTURE OF MATECAT TOOL

In this scenario MT server provides enriched translation suggestions for a given input text which are shown, together with those provided by a Translation Memory server like MyMemory[7], to a professional translator with the final aim of improving her/his productivity, and hopefully, quality.

Moreover, like the post-edited sentences are sent to the TM server for incrementally populating the Translation Memory, they are also exploited for the online adaptation of the MT engine and the QE module.

Online adaptation mainly aims at modifying the MT and QE models embedding user corrections and preferences for improving translator's productivity and satisfaction.

1.2 Features

The main features of the adaptive MT server are briefly listed

[7] mymemory.translated.net

- It provides a Google-compliant REST API for translation, and MT engine adaptation.
- It provides a REST API for adaptation of MT engine adaptation and QE module.
- It provides a REST API for resetting the MT engine models and hence restoring the original behaviour.
- It serves multiple requests asynchronously.
- It provides additional information about the translations, like an estimate of the quality, word and phrase alignment between input text and translation, and source and target character-based tokenization.
- It copes with formatting tags possibly occurring in the input.
- It inputs and outputs case-sensitive and un-tokenized text.
- It exploits case-sensitive and tokenized text for the internal computation, by performing standard pre- and post-processing to normalize input and output.
- It is based on Moses toolkit, one of the best performing phrase-based statistical machine translation engine. Nevertheless, MT server is implemented in such a way that replacing Moses with other decoder is rather easy.
- It relies on either onlineMgiza++ or a pivot-based word aligner to perform the word-alignment as an intermediate step for the adaptation of the Moses models.
- It relies on AQET (Adaptive Quality Estimation Tool) to compute the estimate of the translation quality.
- It adapts Moses and AQET models according to the feedback provided by the user, consisting of parallel sentence pair, i.e. a source text and its translation.

1.3 Architecture and workflow

A high-level description of the adaptive MT server architecture is depicted in Figure 2. The manager controls the communication flow from/to the external users and the internal modules: it receives requests from external users and performs the required actions communicating with the proper modules. Three types of user requests are currently supported: **translate**, **update**, and **reset**.

Translate: Each translation request is handled as follows. The adaptive MT server

i) pre-processes the text of the query by tokenizing the text and handling the formatting tags;

ii) sends the text to and receives the translation from the MT engine;

iii) queries the quality estimation module the confidence score;

iv) post-process the translation by re-introducing the formatting tags and de-tokenizing the text;

v) returns the final translation to the correct user.

Update: Each adaptation request is handled as follows. The adaptive MT server

i) pre-processes both source and post-edit texts by tokenizing them;

ii) sends them to the updater to get the features needed for the MT adaptation;

iii) send these features to the MT engine for the adaptation of its models;

iv) sends the post-edit to the quality estimation module for the adaptation of its models;

v) returns a success message to the correct users, without waiting that adaptation processes end.

Reset: Each resetting request is handled as follows. The adaptive MT server

i) resets MT models deleting all changes occurred thanks to user feedback, and restoring the original behaviour of the MT engine;

ii) returns a success message to the correct users, without waiting that resetting process ends.

Note that AQET is not affected by the reset request.

All these actions are performed asynchronously. Adaptive MT server takes trace of the user URL, which the request comes from, and of the segment ID, which it is processing.

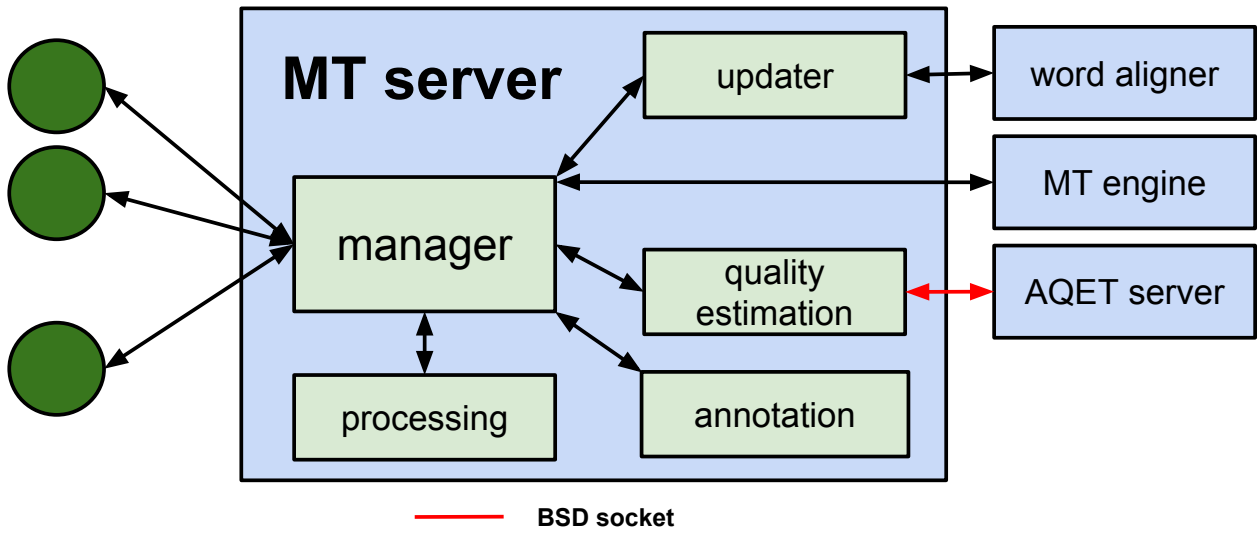

FIGURE 2: ARCHITECTURE OF THE ADAPTIVE MT SERVER

2. Using adaptive MT server

Adaptive MT server exposes its services by means of REST APIs. The services consist in (i) translating a text, (ii) updating the MT engine models, and optionally AQET, according to any feedback, consisting of parallel sentence pair, i.e. a source text and its translation, and (iii) resetting the adaptive MT engine models to remove any previous feedback information. Note that AQET cannot be reset to the original conditions.

2.1 Translation API

The API for querying translations is Google-compliant has the following syntax:

http://my.mtserver.url:8080/translate

Mandatory attributes:

q text to translate

source source language (ISO 639-1)

target target language (ISO 639-1)

key password for identification (currently unused)

Optional attributes:

segid a numerical identifier of the text to translate; if not specified, the value is set to “0000”

2.2 Updating API

The API for updating MT engine and optionally AQET has the following syntax:

http://my.mtserver.url:8080/update

Mandatory attributes:

segment	source sentence of the feedback
translation	target sentence of the feedback
source	source language (ISO 639-1)
target	target language (ISO 639-1)
key	password for identification (currently unused)

Optional attributes:

segid	a numerical identifier of the sentence pair; if not specified, the value is set to “0000”

If AQET is enabled, in order to have a correct adaptation of its models, the attribute segid must coincide with that of a previous translation query.

2.3 Resetting API

The API for reset adaptive models of the MT engine has the following syntax:

http://my.mtserver.url:8080/reset

Mandatory attributes:

source	source language (ISO 639-1)
target	target language (ISO 639-1)
key	password for identification (currently unused)

3. Installing adaptive MT server

This section describes how to install and configure the adaptive MT server, and how to optionally enable the functionality for the online adaptation and for the adaptive quality estimation.

3.1 Preliminary notes

The following installation and configuration instructions assume that:

- MT server will be installed under **/software/mtserver-adaptive** as **root**.
- Moses toolkit is already installed under **/software/moses** on the same machine. Moses includes the facility for the cache-based online adaptation of the translation and language models. Currently, this feature is available in branch "dynamic-models" in the official Github repository.
- onlineMgiza++ toolkit is already installed under **/software/onlinemgizapp** on the same machine. onlineMgiza++, an enhanced version of Mgiza++, includes the facility of alignment any new sentence pair on-the-fly. Currently, onlineMgiza++ is available at www.mt4cat.org/software/onlinemgiza
- If the pivot-based word aligner is used instead of onlineMGIZA++, the software is already installed under **/software/pivot-aligner**. Software and user manual are available at github.com/fredblain/bitext-tokaligner
- AQET toolkit is already installed under **/software/aqet**. Software and manuals of AQET are available at www.mt4cat.org/software/aqet
- AQET server is running and listening on url **aqet.server.url** and on port **9998**
- Moses models are available in this directory **/models/moses-adaptive_models**. Please look below how to get and install them.
- Models to support the online adaptation are in this directory **/models/updater_models**. Please look below how to get and install them.

3.2 Requirements

Adaptive MT server has been successfully tested on the following operating systems:

- Linux Ubuntu 14.04 LTS distribution
- Mac OSx 10.6.8 (Snow Leopard)

Nevertheless, many other Linux distributions and more recent Mac OSx versions are probably compatible.

Adaptive MT server requires Python (2.7 or higher), Perl (5.10 or higher), and Bash command language interpreter (3.2 or higher). It also requires the Python packages **python-cherrypy3**, and optionally **curl**, which is just used for testing purpose, and **wget**, which is used to get the source code and the sample models.

Adaptive MT server relies on Moses toolkit, supporting the online adaptation via cache-based models.

In order to enable the online adaptation, one of the following two word-alignment softwares is required:

- onlineMgiza++, an enhanced version of the word alignment toolkit Mgiza++, which permits to align any new sentence pairs using pre-trained alignment models
- pivot-based word aligner, which combines the Moses word-alignment from the source and the MT output and the TER-based alignment between the MT output and the post-edit.

In order to enable the quality estimation, AQET is required as well.

3.3 Install the required packages

If needed, install (as root) the following Linux packages:

python-cherrypy3, **curl**, **wget**

by typing:

```
sudo apt-get install python-cherrypy3 curl wget
```

3.4 Download and install the code

Download and un-archive the source code in your directory of choice by typing:

```
cd /software
wget http://www.mt4cat.org/file-cabinet/mtserver-adaptive.zip
unzip mtserver.zip
```

The above will create the directory **/software/mtserver-adaptive** and will download all code in it. All next steps assume we are in this directory.

```
cd /software/mtserver-adaptive
```

Please, be sure that your shell is bash; instructions are not tested for other shell types.

3.5 Test Moses

To test whether **moses** is properly working, open a shell, and execute:

```
/software/moses/bin/moses -f /models/moses-adaptive_models/moses-adaptive.ini –t
```

Given a source text like

the Contracting Parties .

the response is (according to the sample models):

le |0-0| Parti contraenti |1-2| . |3-3|

Moses models can be modified using an input annotation like:

<dlt type="cbtm" id="CBTM0" cbtm="Parties|||parti"/>
<dlt type="cblm" id="CBLM0" cblm="parti"/>

Moses replies with an empty line. Note that the ids (CBTM0 and CBPT0) must be coherent with those specified in the configuration file.

The quotation mark must be the ASCII character 42; other characters are not correctly interpreted.

Now, re-translating the same input as before

the Contracting Parties .

the response is changed according to the original sample models and the proposed modifications:

le |0-0| parti |2-2| contraenti |1-1| . |3-3|

3.6 Configure MT server

Change directory into **SERVER**

```
cd SERVER
```

Make a copy of the template configuration file (**template_server-adaptive.config**), by typing:

```
cp template_server-adaptive.config server-adaptive.config
```

Edit the actual configuration file to setup parameters specific to MT server.

MTSERVER_ROOT=/software/mtserver-adaptive

MTSERVER_URL=0.0.0.0

MTSERVER_PORT=8080

MTSERVER_SRCLNG=en

MTSERVER_TGTLNG=it

MTSERVER_URL must be set to the URL(s) from where MT server accepts queries. **0.0.0.0** means from anywhere.

MTSERVER_PORT is the port to connect to the MT server.

An MT server can serve queries only if they fit the defined source and target languages, MTSERVER_SRCLNG and MTSERVER_TGTLNG, respectively.

Make the system aware of the defined parameters, by typing:

```
source server.config
```

Note that the command "source" is local to the shell where the command is run.

3.7 Configure Moses

The MT server configuration file includes the parameters to set up Moses.

Edit the actual configuration file to setup parameters related to **Moses** according to your needs. In particular, you may be interested in

MOSES_ROOT=/software/moses

MOSES_THREADS=2

MOSES_CONFIG/models/moses-adaptive_models /moses-adaptive.ini

MOSES_LOG=/tmp/log

MOSES_CONFIG is a Moses configuration file suitable for the online adaptation.

Make the system aware of the defined parameters, by typing:

```
source server-adaptive.config
```

3.8 Configure the updater module

The MT server configuration file includes the parameters to enable and set up the updater module.

Edit the actual configuration file to setup parameters related to the feature extraction module according to your needs. In particular, you may be interested in

UPDATER_CONFIG=/models/updater_models/updater.ini

UPDATER_CONFIG is the configuration file, which define all parameters for the word-alignment and the feature extraction.

Make the system aware of the defined parameters, by typing:

```
source server-adaptive.config
```

3.9 Configure AQET

The MT server configuration file includes the parameters to enable and set up AQET.

Edit the actual configuration file to setup parameters related to AQET according to your needs. In particular, you may be interested in

QEMODULE_CLIENT=/software/aqet/bin/onlineqeclient

QEMODULE_SERVER_URL=127.0.0.1

QEMODULE_SERVER_PORT=9998

QEMODULE_CLIENT is the client of the AQET toolkit, communicating with the AQET server.

QEMODULE_SERVER_URL and ***QEMODULE_SERVER_PORT*** are the URL and the port on which AQET server is listening.

Make the system aware of the defined parameters, by typing:

```
source server-adaptive.config
```

3.10 Start MT server

Start the MT server as follows

```
start-mtserver-adaptive.sh
```

Some debugging info and a message will be output in the shell

[date:time] ENGINE Serving on my.mtserver.url:8080

[date:time] ENGINE Bus STARTED

which confirms that **MT server** is listening for queries on the specified port.

3.11 Test MT server

To test whether the MT server works properly, receiving input and returning translation, make a request following the MT server API.

Either, a browser can be used and the following

```
http://my.mtserver.url:8080/translate
        ?q="the Contracting Parties"
        &source=en&target=it&key=anythingisfine
```

or the **curl** command as follows:

```
curl    'http://my.mtserver.url:8080/translate
        ?q=the%20Contracting%20Parties
        &source=en&target=it&key=anythingisfine'
```

or:

```
curl    --data “q=the Contracting Parties”
         –data “source=en“
        –data “target=it“
        --data “key=anythingisfine”
        http://my.mtserver.url:8080/translate
```

The response in JSON format should be similar to

```
{ "data":
 { "translations”:
  [ { "translatedText": ”le Parti contraenti.”,
      "phraseAlignment": [[[0], [0]], [[1, 2], [1, 2]], [[3], [3]]],
```

"sentence_confidence": "88.6097", ...

Spaces and tabs depend on the method used.

To test whether the MT server works properly, receiving modification request and modifying its behaviour accordingly, first make an update request following the MT server API.

Either, a browser can be used and the following

```
http://my.mtserver.url:8080/update
        ?segment="the Contracting Parties"
        &translation="le parti contraenti."
        &source=en&target=it&key=anythingisfine
```

or the **curl** command as follows:

```
curl 'http:// my.mtserver.url:8080/update
        ?segment=the%20Contracting%20Parties
        &translation=le%20parti%20contraenti.
        &source=en&target=it&key=anythingisfine'
```

or:

```
curl    --data “segment=the Contracting Parties.”
        --data “translation=le parti contraenti.”
        –data “source=en“
        –data “target=it“
        --data “key= anythingisfine”
        http:// my.mtserver.url:8080/translate
```

The response in JSON format should be similar to

```
{"data”:
 {"code": "0”,
  "string": "OK", ...}}
```


Now, re-submitting the same translation requests as before, the response is changed according to the original sample models and the proposed modifications:


```
{ "data":
 { "translations”:
  [ { "translatedText": ”le parti contraenti.”,
     "phraseAlignment": [[[0], [0]], [[2], [1]], [[1], [2]], [[3], [3]]],
     "sentence_confidence": "92.0256", ...
```

4. Installing sample models for Moses

4.1 Download and install models

Download the archive with the sample models to use with Moses from:

```
https://www.mt4cat.org/file-cabinet/moses-adaptive_sample-models.zip
```

Un-archive the archive in the model directory by typing:

```
cd /models
unzip moses-adaptive_sample-models.zip
cd moses-adaptive_models/
```

The above will create the directory **/models/moses-adaptive_models** and will put all required files in it. All next steps assume you are in this directory.

4.2 Configure models

In order to actualize the configuration file to the current path, execute the following command, where the parameter refers to the current directory:

```
cat template_moses-adaptive.ini

        | actualize.sh /models/moses-adaptive_models

        > moses-adaptive.ini
```

5. Installing sample models for updater

5.1 Download and install models

Download the archive with the sample models to use with the upadter from:

```
https://www.mt4cat.org/file-cabinet/updater_sample-models.zip
```

Un-archive the archive in the model directory by typing:

```
cd /models
unzip updater_sample-models.zip
cd updater_models
```

The above will create the directory **/models/updater_models** and will download all required files in it. All next steps assume we are in this directory.

5.2 Configure models

In order to actualize the configuration files, execute the following command, where the parameter refers to current directory, and source and target languages (ISO-639 format):

```
cat template_updater.ini
        | ./actualize.sh /models/updater_models en it
        > updater.ini
cat template_SRC2TRG_gizacfg.online
        | ./actualize.sh /models/updater_models en it
        > en-it_gizacfg.online
cat template_TRG2SRC_gizacfg.online
        | ./actualize.sh /models/updater_models it en
```

- it-en_gizacfg.online

6. References

[Blain et al., 2012]: Blain, Frederic, Holger Schwenk, and Jean Senellart. "Incremental Adaptation Using Translation Information and Post-Editing Analysis", Proceedings of the International Workshop on Spoken Language Translation (IWSLT), Hong-Kong, China, pp. 234-241, 2012.

[Federico et al., 2014]: Federico, Marcello, Nicola Bertoldi, Mauro Cettolo, Matteo Negri, Marco Turchi, Marco Trombetti, Alessandro Cattelan, Antonio Farina, Domenico Lupinetti, Andrea Martines, Al-berto Massidda, Holger Schwenk, Loïc Barrault, Frederic Blain, Philipp Koehn, Christian Buck, and Ulrich Germann. "THE MATECAT TOOL". Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: System Demonstrations, Dublin, Ireland, pp. 129–132, 2014.

[Turchi et al., 2014]: Turchi, Marco, Antonios Anastasopoulos, José G. C. de Souza, and Matteo Negri. "Adaptive Quality Estimation for Machine Translation". Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Baltimore, Maryland, pp. 710–720, 2014.