

# [DRAFT:] OpenMaTrEx: A free/open-source marker-driven example-based machine translation system

Sandipan Dandapat, Mikel L. Forcada\*, Declan Groves\*\*, Sergio Penkale, John Tinsley, and Andy Way

Centre for Next Generation Localisation, Dublin City University, Glasnevin, Dublin 9, Ireland

{sdandapat,mforcada,dgroves,spenkale,jtinsley,away}@computing.dcu.ie

*[Comments about this draft for the CNGL IP Committee appear in this typeface]  
[Paper to be submitted to IceTAL, August 16-18, 2010 Reykjavik, Iceland]*

**Abstract.** We describe OPENMATREX, a free/open-source example-based machine translation system based on the marker hypothesis, comprising a marker-driven chunker/chunk tagger, a collection of chunk aligners, and two engines: one based on a simple proof-of-concept monotonic EBMT recombinator or and a Moses-based decoder. OPENMATREX is a free/open-source release of the basic components of a baseline MATREX, the Dublin City University machine translation system. Experimental tests in which OPENMATREX is compared to Moses on a standard task are shown.

## 1 Introduction

We describe OPENMATREX, a free/open-source example-based machine translation (MT) system based on the marker hypothesis, comprising a marker-driven chunker, a collection of chunk aligners, and two engines: one based on a simple proof-of-concept monotonic recombinator or “decoder”<sup>1</sup> and a Moses-based decoder [1]. OPENMATREX is a free/open-source release of the basic components of a baseline MATREX, the Dublin City University machine translation system [2, 3], and it will be released under the GNU General Public Licence<sup>2</sup> (GPL), version 3. A great part of the code in OPENMATREX is written in Java, although there are many important tasks that are performed in a variety of scripting languages. A preliminary version *[version number, smaller than 1.0, will be given here]* may be downloaded from <http://www.openmatrex.org> *[not yet available at that*

---

\* Permanent address: Departament de Llenguatges i Sistemes Informàtics, Universitat d’Alacant, E-03071 Alacant, Spain

\*\* Permanent address: Traslán, Greystones, Co. Wicklow, Ireland

<sup>1</sup> Released in January 2010 under the name *Marclator*, <http://www.openmatrex.org/marclator/marclator.html>.

<sup>2</sup> <http://www.gnu.org/licenses/gpl.html>

*address: currently Marclator can be downloaded from there; the package is being finalized at the time of writing these lines].*

The architecture of OPENMATREX is the same as that of a baseline MATREX system; as MATREX, OPENMATREX can wrap around the Moses SMT decoder, using a hybrid translation table containing marker-based chunks as well as statistically extracted *phrase* pairs.

This paper is organized as follows. Section 2 describes the principles of training and translation in OPENMATREX and briefly reviews existing work performed with its components as part of MATREX; section 3 describes the components of OPENMATREX; section 4 outlines the installation process; section 5 explains how to run the available components. Experimental results comparing the performance of OPENMATREX to that of a standard statistical machine translation built using Moses. Sample experiments performed on standard tasks with OPENMATREX are described in section 7 and they are compared with a baseline Moses run. Future work is described in section 6 and some concluding remarks are made in section 8.

## 2 OpenMaTrEx basics

### 2.1 Training

During training:

1. Each example sentence in the sentence-aligned source text and its counterpart in the target training text are divided in subsentential segments using a marker-based *chunker*. For instance, the English sentence

*That delay was intended to allow time for the other institutions to consider our amendments and for the Commission to formulate amended proposals.*

would be chunked as

```
that delay was intended ||| to allow time ||| for the other
institutions ||| to consider ||| our amendments ||| and for
the commission ||| to formulate amended proposals.
```

Chunks may optionally be tagged according to the tag of the marker words used to delimit it (and tags may be used to guide the alignment process), hence the name *chunker*/chunk tagger (from now on, simply *chunker*).

2. A complete Moses-Giza++<sup>3</sup> training run is performed up to step 5 (*phrase* extraction).
3. The subsentential segments of chunks are aligned using one of the *aligners* provided (some of which use the probabilities generated by Giza++).
4. Aligned chunk pairs from step 3 are *merged* with the *phrase* pairs generated by Moses in step 2. This process is described in greater detail in section 3.3.

<sup>3</sup> <http://www.fjoch.com/GIZA++.html>

## 2.2 Translation

Translation may be performed in two ways. One possibility is to use the monotone, *naïve* decoder that was released with Marclator (we will call this *Marclator mode*):

1. the new sentence is chunked using the source-language chunker;
2. translations for each of the source-language chunks are retrieved;
3. a translation for the sentence is built by the recombinator (“decoder”).

Marclator mode may have some uses as a complete EBMT engine, although such an engine is limited by the crude nature of the decoder provided.

However, the usual way to use it is to run the Moses decoder with the combined phrase table, as done in MATrEX (we will call this MATrEX *mode*). This is described further in section 3.3.

## 2.3 OpenMaTrEx: making MaTrEx research accessible

OPENMATrEX has been released as a free/open-source package so that MATrEX components which have successfully been used [4–7] may be used and combined with components from other free/open-source machine translation toolkits such as Cunei<sup>4</sup> [8], Apertium<sup>5</sup> [9], etc.<sup>6</sup> Indeed, in conjunction with components released in OPENMATrEX, researchers have previously:

- used alternative recombinators to generate translations [10, 11], sometimes using statistical models to rerank the results of recombination [12];
- used aligned, marker-based chunks in an alternative decoder which uses a memory-based classifier [13];
- combined the marker-based chunkers with rule-based components [14], or
- used the chunker to filter out the phrases generated by Moses, to discard those that do not have a linguistic motivation [15].

*[An edited version of this section should probably appear somewhere else in the paper, perhaps somewhere else in the paper and single out those papers reporting MATrEX experiments that can be reproduced using OPENMATrEX.]*

## 3 The components

*[This draft, as written, contains much detail about the actual Java classes that will be removed or abridged before submission.]*

---

<sup>4</sup> <http://www.cunei.org>

<sup>5</sup> <http://www.apertium.org>

<sup>6</sup> For a longer list of free/open-source MT software, visit <http://fosmt.info>.

### 3.1 Chunker

The main chunker/chunk tagger (class `chunkers/MarkerBasedChunker`, base class `chunkers/Chunker`) is based on Green’s *marker hypothesis* [16] which states that the syntactic structure of a language is marked at the surface level by a set of marker (closed-category) words or morphemes. In English, markers are predominantly right-facing and it is therefore a left-marking language (for instance, determiners or prepositions mark the start of noun phrases or prepositional phrases, respectively); there are also right-marking languages such as Japanese, with left-facing markers. The chunker in OPENMATREX deals with left-marking languages: a chunk starts where a marker word appears, and must contain at least one non-marker (content, open-category) word. In addition to marker words, punctuation is used to delimit the end of chunks.<sup>7</sup> The chunker includes a tokenizer and lowercaser.

*Marker files:* The system provides free/open-source marker files for Catalan, Czech, English, Portuguese, Spanish, French and Italian. In the near future, we plan to create marker files for new languages. [\[An edited version of this paper may contain a brief description of the format and contents of marker files.\]](#)

### 3.2 Chunk aligners

There are a number of different chunk aligners available in Marclator:

- The class `aligners/EditDistanceChunksAligner` aligns chunks using a regular Levenshtein edit distance with costs (base class `aligners/ChunkDistance` specified by `aligners/CombinedChunkDistance`, which combines at runtime the component costs listed by the user in the configuration file used to call it, such as `aligners/NumCharsChunkDistance` (based on the number of characters in the chunk), `aligners/NumWordsChunkDistance` (based on the number of words in the chunk), `aligners/TagBasedChunkDistance` (based on the tags assigned to each chunk by the chunker/tagger) `aligners/WordProbsChunkDistance` (see below), and `aligners/CognateChunkDistance` (which calculates cognate distance based on a combination of the Levenshtein distance, the lowest common rubsequence ratio and the Dice coefficient). A combination of the latter two is customarily used (for more details on alignment strategies, see [2]). Given that no substantial improvement was obtained by modifying these weights [2], the code uses equal weights for all component costs specified.
- `aligners/EditDistanceWJChunksAligner` is a version of `aligners/EditDistanceChunksAligner` but uses a modified edit distance with *jumps* or block movements [2].
- Other aligners such as `aligners/HmmChunksAligner`, `aligners/MostProbableChunkAligner`, `aligners/GreedyChunksAligner`, and `aligners/SillyChunksAligner` are also available for experimentation.

<sup>7</sup> There is also another chunker, `chunkers/SillyChunker` which just divides the sentence in three-word chunks, using no markers.

Since `aligners/WordProbsChunkDistance` uses word translation probabilities calculated using the Giza++ statistical aligner<sup>8</sup> and scripts available as part of the Moses MT engine,<sup>9</sup> these free/open-source packages need to be installed in advance (see section 4).

### 3.3 Translation table merging scripts

In order to run the system in MATREX *mode*, we provide a facility by which marker-based chunk pairs can be combined with phrase pairs from alternative resources. In this case, we use Moses phrases. Firstly, each chunk pair is assigned a word alignment based on the refined Giza++ alignments, for example

```
please show me ||| por favor muéstreme ||| 0-0 0-1 1-2 2-2
```

In cases where there is no word alignments for a particular chunk pair according to Giza++, we back-off to the set of alignment points as suggested by the chunk aligner.<sup>10</sup> Using these word alignments, we additionally extract a phrase orientation-based lexicalised reordering model *à la* Moses [17]. Finally, we may also limit the maximum length of chunks pairs that will be used. The resulting set of chunk pairs are in the same format as those phrase pairs extracted by Moses.

The next step is to combine the chunk pairs with Moses phrase pairs. In order to do this, the two sets of chunk/phrase pairs are merged into a single file. Moses training is then carried out from step 6 (*scoring*) which calculated the required scores for all feature functions, including the reordering model, based on the combined counts.

### 3.4 Monotone recombinator

OPENMATREX releases a very simple, monotone recombinator called the “naïve decoder” (class `decoders/NativeDecoder`, base class `decoders/Decoder`), which was released as part of the Marclator package and proceeds as follows:

- Each sentence is chunked using the marker-based chunker/tagger for the source language;
- the candidate translations for each chunk are retrieved, along with their weights;
- if no chunk translations are found, the decoder looks for translations for the words (as aligned by Giza++) and concatenates them in the same order;
- when no translation is found, the source word is left untranslated.

<sup>8</sup> <http://www.fjoch.com/GIZA++.html>

<sup>9</sup> <http://www.statmt.org/moses/>

<sup>10</sup> This is not very effective for marker-based chunks as there are no overlapping chunks for a particular sentence. However, this will be particularly useful in future releases which will accommodate the exploitation of different types of phrase pairs e.g. tree-based.

As said above, the released decoder has obvious limitations, and is likely to be of most use in the case of very related language pairs. We expect developers to contribute alternative decoders in the future.

### 3.5 Using Moses as a decoder

*[Moses is basically used as is, but this section will contain details, if any, on how the results of merging (section 3.3) are processed.]*

### 3.6 Evaluation scripts

*[The preliminary OPENMATREX may contain an easy interface to the NIST mteval package. This section will describe any particulars relating its use in OPENMATREX.]*

## 4 Installing OpenMaTrEx

*[This section still contains too much detail will be shortened to a couple of paragraphs and the reader will be referred to the INSTALL file in the OPENMATREX package]*

This is just an outline. For details on the installation of Marclator, please refer to the INSTALL file that will soon be added to the distribution.

### 4.1 Required software

*Giza++:* First, the latest version of Giza++ should be installed (from <http://code.google.com/p/giza-pp/>). This also includes the installation of `snt2cooc.out` and `mkcls` which are required for Moses training.

*Moses:* Moses is at the core of the MATREX-mode decoder and is also used for learning a maximum likelihood bidirectional lexical translation table (estimating both target-to-source and source-to-target word translation probabilities) and to extract the translation (“phrase”) tables used by both in MATREX mode and in Marclator mode. To do that, it suffices to invoke Subversion as follows: `svn co https://mosesdecoder.svn.sourceforge.net/svnroot/mosesdecoder/trunk`. Once the GIZA++ and Moses scripts are installed, set the path of the `OPENMATREX_DIR` (home directory of the Marclator) and `MOSES_SCRIPTS_DIR` in the `OpenMaTrEx` shell.

*Moses scripts:* *[Perhaps we don't need this part]*

*Evaluation scripts:* *[Integrate if necessary.]* OPENMATREX uses the NIST `mteval` script to evaluate machine translation results. The latest version of `mteval` may be found at <ftp://jaguar.ncsl.nist.gov/mt/resources/> (currently `mteval-v13a-20091001.tar.gz`).

*Other software:* [\[Probably this is not needed.\]](#) Note that the current version of OPENMATREX includes source code for an old version of Kohsuke Kawaguchi's `args4j` library,<sup>11</sup> which is used to parse arguments when *Marcalator* components are invoked through the command line.<sup>12</sup> In future versions, we will remove the code, and just require it to be installed in advance.

## 4.2 Installing OpenMaTrEx itself

Inside the OpenMaTrEx package [\[Details about the name, the version number and the URL from where it can be downloaded would come here\]](#), a `build.xml` file is provided so that OPENMATREX may easily be built simply by invoking `ant` or an equivalent tool. The resulting `OpenMaTrEx.jar` contains all the relevant classes, some of which will be invoked using a shell, `OpenMaTrEx` (see below).

## 5 Running

[\[This section still contains too much detail will be shortened to a couple of paragraphs and the reader will be referred to the README file in the OPENMATREX package\]](#)

A shell (`OpenMaTrEx`) is used to hide a `make` call on a `Makefile` provided with the distribution; the makefile target is passed on to `make`. Targets are available to initialise the training set, to call the chunker/tagger, the aligner, and the decoder (not available yet). Some of these targets build an XML configuration file (always called `openmatrex.ini`) which is used to pass information to classes `main/Chunk`, `main/Align`, and `main/Decode` (not working yet).

What follows is a sketch of the procedure that describes how to run OPENMATREX. [\[Perhaps it does not make sense to show all of the individual targets, especially if they will be collected, as expected, into larger, easier to use "translate" and "train" targets.\]](#)

1. Create a working directory. In this directory, create a subdirectory `orig` to store your training and test data: `train.S` and `train.T`, `testset.S`, and `testset.T`, where *S* denotes the source language and *T* the target language. Each source file and its target counterpart must have the same number of lines (one sentence per line).
2. When these files are ready, type `OpenMaTrEx SL="S" TL="T"` to initialize the system.<sup>13</sup>
3. Run the source- and target-language marker-based chunker/tagger using the following commands respectively `OpenMaTrEx marker-based_chunking_source` and `OpenMaTrEx marker-based_chunking_target`. This will create a directory `chunked` which contains the marker-based chunked files `train.S` and `train.T`.

<sup>11</sup> <https://args4j.dev.java.net/>

<sup>12</sup> Not operational in the current version.

<sup>13</sup> One can also easily invoke the Moses filtering scripts to remove certain sentence pairs, etc.

4. First, run MOSES until step 5 (*phrase* extraction): `OpenMaTrEx moses_training_steps FIRST_STEP=1 LAST_STEP=5` The bidirectional lexical translation tables are created in the `moses_training/model` directory named `lex.0-0.e2f` and `lex.0-0.f2e`
5. [\[Update the following info:\]](#) Align the marker-based chunked files using the marker-based aligner. Currently we have three different entry points for three different aligners (i.e. `ebmt_alignments`, `ebmt_alignments_with_ID`, `ebmt_alignments_with_context`). Any of these aligners can be used to perform the desired task using a suitable command, such as `OpenMaTrEx ebmt_alignments`. This creates the `ebmt_alignments` directory with the chunk aligned file.
6. [\[Check this step:\]](#) (Only in MATrEX mode) Merge the chunk pairs produced in step 5 and the *phrase* pairs produced in step 4 into a single *phrase* pair file to be used by Moses: `OpenMaTrEx merge_ebmt_with_moses`.
7. (Only in MATrEX mode). Run Moses decoder steps from 6 to 9 by invoking... [\[To be finished.\]](#)
8. (Only in Marclator mode). Run the *Naïve decoder* using `OpenMaTrEx ebmt_decode` [\[This has to be completed.\]](#)
9. [\[A note about how to run the evaluation scripts could be included here.\]](#)

Test files will be provided in the `examples` directory of the OpenMaTrEx package. [\[Either that, or information about where to download them from, in case they cannot be distributed under the GPL.\]](#)

## 6 Future work

[\[This section will contain information on improvements that will take OPENMATrEX from this version to version 1.0, and could also give some information on expected additions or plugins.\]](#)

## 7 Some experimental results

[\[The experiments have not been performed yet. The plan is to use a standard testset from a past evaluation campaign in which the baseline OPENMATrEX released clearly improves over a “vanilla” Moses run on a standard task, and give enough details about it so that it can be easily reproduced by the readers. Candidate tasks are those in IWSLT2006 where MATrEX was first tested as such \[2\], or perhaps recent WMT10 tasks, to see how it works.\]](#)

## 8 Concluding remarks

We have presented OPENMATrEX a free/open-source example-based machine translation system based on the marker hypothesis which includes a marker-driven chunker/tagger (with marker word files for a few languages), a collection



of chunk aligners, a simple monotonic recombinator or “decoder”, and a wrapper around Moses so that it can be used as a decoder for a merged translation table containing Moses phrases and marker-based chunk pairs. OPENMATREX releases the basic components of a baseline version of MATREX, the Dublin City University machine translation system, under a free/open-source license (the GNU General Public License, version 3) so that they are easily accessible to researchers and developers of machine translation systems. The main modules of the system are written in Java, but scripting languages are used extensively too. The first stable release of OPENMATREX, version [\[give version number here\]](#) is available from <http://www.OpenMaTrEx.org> [\[details will be given in the final version of the paper\]](#).

*Acknowledgements:* The original MATREX code which is the base of OPENMATREX was developed among others by Steve Armstrong, Yvette Graham, Nano Gough, Declan Groves, Hany Hassan, Yanjun Ma, Nicolas Stroppa, John Tinsley, Andy Way, Bart Mellebeek. We specially thank Yvette Graham and Yanjun Ma for her advice about MATREX. Mikel L. Forcada’s sabbatical stay at Dublin City University is supported by Science Foundation Ireland (SFI) through ETS Walton Award 07/W.1/I1802 and by the Universitat d’Alacant (Spain). Andy Way acknowledges support from SFI through grants 05/IN/1732 and 06/RF/CMS064.

## References

1. Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open source toolkit for statistical machine translation. *Annual Meeting of the Association for Computational Linguistics (ACL), demonstration session, Prague, Czech Republic, June 2007*, 2007.
2. N. Stroppa and A. Way. MaTrEx: DCU machine translation system for IWSLT 2006. In *Proceedings of the International Workshop on Spoken Language Translation*, pages 31–36, 2006.
3. N. Stroppa, D. Groves, A. Way, and K. Sarasola. Example-based machine translation of the Basque language. In *Proceedings of AMTA 2006*, pages 232–241, 2006.
4. D. Groves and A. Way. Hybrid example-based SMT: the best of both worlds? *Building and Using Parallel Texts: Data-Driven Machine Translation and Beyond*, 100:183, 2005.
5. Stephen Armstrong, Marian Flanagan, Yvette Graham, Declan Groves, Bart Mellebeek, Sara Morrissey, Nicolas Stroppa, and Andy Way. Matrex: Machine translation using examples. In *TC-STAR Open- Lab Workshop on Speech Translation*, Trento, Italy, 2006.
6. H. Hassan, Y. Ma, A. Way, and I. Dublin. MaTrEx: the DCU machine translation system for IWSLT 2007. In *Proceedings of the International Workshop on Spoken Language Translation*, pages 69–75. Citeseer, 2007.

7. J. Tinsley, Y. Ma, S. Ozdowska, and A. Way. MaTrEx: the DCU MT system for WMT 2008. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 171–174. Association for Computational Linguistics, 2008.
8. Aaron B. Phillips and Ralf D. Brown. Cunei machine translation platform: System description. In Mikel L. Forcada and Andy Way, editors, *Proceedings of the 3rd International Workshop on Example-Based Machine Translation*, pages 29–36, November 2009.
9. Francis M. Tyers, Mikel L. Forcada, and Gema Ramírez-Sánchez. The Apertium machine translation platform: Five years on. In F. Sánchez-Martínez J.A. Pérez-Ortiz and F.M. Tyers, editors, *Proceedings of the First International Workshop on Free/Open-Source Rule-Based Machine Translation*, pages 3–10, November 2009.
10. N. Gough and A. Way. Robust large-scale EBMT with marker-based segmentation. In *Proceedings of the Tenth Conference on Theoretical and Methodological Issues in Machine Translation (TMI-04)*, pages 95–104, Baltimore, MD, 2004.
11. A. Way and N. Gough. Comparing example-based and statistical machine translation. *Natural Language Engineering*, 11(03):295–309, 2005.
12. D. Groves and A. Way. Hybridity in MT: Experiments on the Europarl corpus. In *Proceedings of the 11th Annual Conference of the European Association for Machine Translation (EAMT-2006)*, pages 115–124. Citeseer, 2006.
13. A. van den Bosch, N. Stroppa, and A. Way. A memory-based classification approach to marker-based EBMT. In *Proceedings of the METIS-II Workshop on New Approaches to Machine Translation*, pages 63–72, Leuven, Belgium, 2007.
14. Felipe Sánchez-Martínez, Mikel L. Forcada, and Andy Way. Hybrid rule-based – example-based MT: Feeding Apertium with sub-sentential translation units. In Mikel L. Forcada and Andy Way, editors, *Proceedings of the 3rd Workshop on Example-Based Machine Translation*, pages 11–18, Dublin, Ireland, November 2009.
15. Felipe Sánchez-Martínez and Andy Way. Marker-based filtering of bilingual phrase pairs for SMT. In *Proceedings of EAMT-09, the 13th Annual Meeting of the European Association for Machine Translation*, pages 144–151, Barcelona, Spain, 2009.
16. T. Green. The necessity of syntax markers. two experiments with artificial languages. *Journal of Verbal Learning and Behavior*, 18:481–496, 1979.
17. Philipp Koehn, Amittai Axelrod, Alexandra Birch Mayne, Chris Callison-Burch, Miles Osborne, and David Talbot. Edinburgh system description for the 2005 IWSLT speech translation evaluation. In *Proceedings of the International Workshop on Spoken Language Translation*, Pittsburgh, PA, 2005.