Vol.2015-CLE-16 No.4
2015/5/22

Laravel を用いた翻訳メモリエディタの開発

常盤祐司[†1]　出口大輔[†2]　宮崎誠[†3]　平岡斉士[†4]　喜多敏博[†5]　梶田将司[†6,†7]

Sakai, Moodle および Mahara は英語圏で開発されているため，日本語への翻訳が必要である．これらの翻訳においては様々な翻訳支援ツールが利用でき，多くのシステムでは事例ベースの翻訳とも言える翻訳メモリを使い，翻訳の一貫性を保っている．この翻訳メモリは翻訳を重ねるごとにその翻訳で生成された翻訳が追加され翻訳対象も増えるが，一方では「汚れ」と言われる不整合が生じる．また，複数のシステムの翻訳で生成された翻訳メモリを結合して共通翻訳メモリを作成する場合も，同一文章に対して複数の翻訳が設定されることがある．そのため，翻訳メモリの浄化が必要になるが，翻訳メモリは英語-日本語で一対となる翻訳セットが数万組に及ぶ．本稿ではこうした翻訳メモリの浄化に特化したアプリケーションを PHP 用フレームワーク Laravel にて開発して得られた知見について報告する．

A Development of Translation Memory Editor with Laravel

YUJI TOKIWA[†1]　DAISUKE DEGUCHI[†2]　MAKOTO MIYAZAKI[†3]
NAOSHI HIRAOKA[†4]　TOSHIHIRO KITA[†5]　SHOJI KAJITA[†6,†7]

A translation to Japanese is unavoidable for Sakai, Moodle, and Mahara since they are developed in English-speaking countries. In such a translation, many translation support systems are available and most systems ensure a consistency of translation to make use of a translation memory. Once a translation is completed, a translation memory obtains new translation units, however a translation memory will be “dirty” due to an inconsistency to existing translation units. In case that a common translation memory is generated to join plural translation memories for Sakai, Moodle, and Mahara, an inconsistency may arise to generate multiple translated targets to one original source. So far, several tens of thousands of translation units make it difficult to eliminate “dirty” translation from a newly generated translation memory. This paper reports the knowledge through a development and an application of translation memory editor with a Laravel framework.

1. はじめに

翻訳メモリを利用した英文マニュアルなどの翻訳は，1990 年代にはすでに企業内で組織的に行われていた．一方，教育用に利用される Sakai, Moodle, Mahara といったオープンソースソフトウェア(以下，OSS)の翻訳は，これまで特定のボランティアが行ってきていた．また，日本で開発された Eclipse ベースの OSS 翻訳支援ツール Benten が公開されたのは 2010 年である[1].

しかしながら海外では Web ベースの翻訳支援システムを提供する Transifex[2]および Crowdin LLC[3]が 2009 年に設立され，OSS 開発プロジェクトではそのサービスが無償で利用できることもあり，それらの提供するサービスが OSS コミュニティに浸透してきた. そこで著者らは Sakai, Moodle, Mahara を共通の翻訳メモリを使って翻訳し，ユーザがそれらのシステムを渡り歩いても違和感を持たない一貫性のある翻訳を目指したプロジェクトを 2013 年度から開始した[4].

そこで用いる共通翻訳メモリはそれぞれのシステムを翻訳支援システムで翻訳を行った際に生成される翻訳メモリを統合して作成した．しかしながら，同一の英語ソースに対して異なる日本語翻訳が散見され，それらを整理する必要が出てきた．また，統合した翻訳メモリはソースである英語および翻訳した日本語からなる翻訳ユニットが 33,000 組に及び何らかの支援システムが必要になった.

このプロジェクトは 5 大学による分散プロジェクトであるため Web ベースのシステムが必要となり，開発の簡便さから PHP で開発することとした．さらに翻訳メモリをデータベース化することから，ORM (Object relational mapping) が利用できるフレームワークとして昨今注目を浴びている Laravel を利用した. また，フレームワークの利用においては開発の容易さ，MVC (Model View Controller) デザインパターンによる整理された構造も期待した.

本稿では，開発した Web ベースの翻訳メモリエディタとシステム開発に用いた Laravel の概要および評価について報告する.

2. 翻訳メモリエディタ概要と要件

2.1 翻訳プロセス概要

図 1 に著者らが考案した翻訳プロセスを示す．詳細な説明は参考文献[4]に記載しているので，ここでは翻訳支援システムとして Transifex を用いた翻訳の概要を述べる.

†1 法政大学　情報メディア教育研究センター
Research Center for Computing and Multimedia Studies, Hosei University
†2 名古屋大学　情報連携統括本部
Information and Communications Headquarters, Nagoya University
†3 畿央大学　教育学習基盤センター
Center for Teaching , Learning and Technology, Kio University
†4 熊本大学　社会文化科学研究科
Graduate School of Social and Cultural Science, Kumamoto University
†5 熊本大学　e ラーニング推進機構
Institute for e-Learning Development, Kumamoto University
†6 京都大学　情報環境機構 IT 企画室
IT Planning Office, Institute for Information Management and Communication, Kyoto University
†7 京都大学　学術情報メディアセンター
Academic Center for Computing and Media Studies, Kyoto University

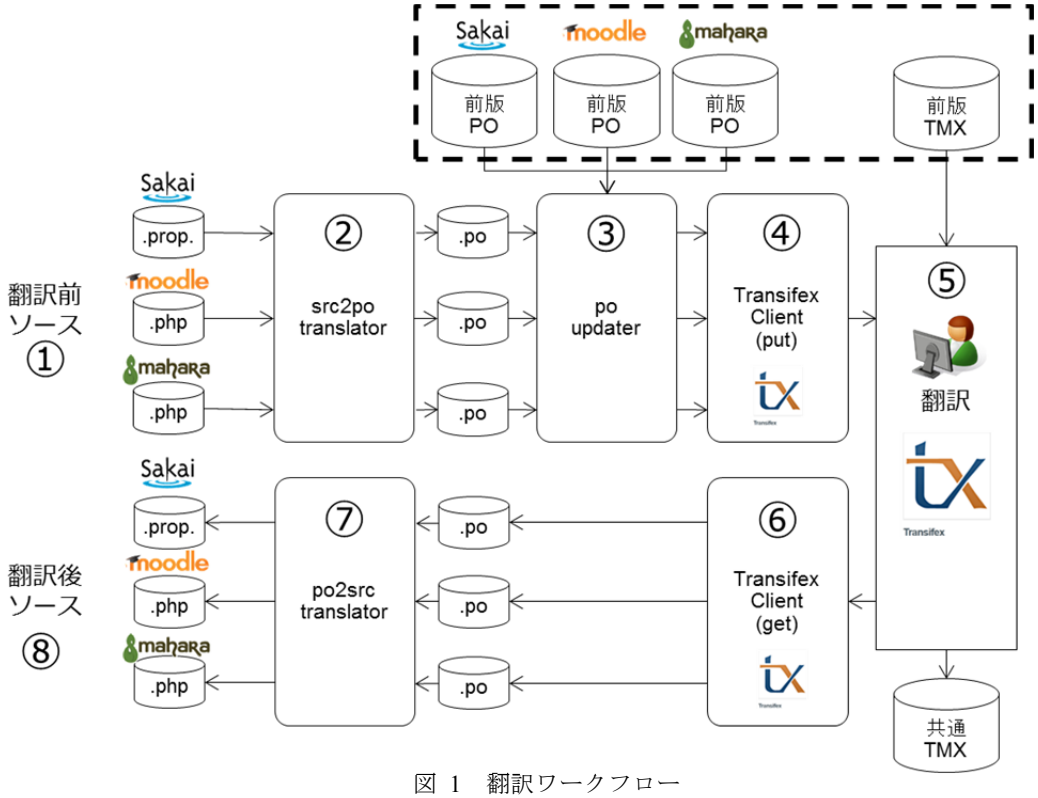

図 1　翻訳ワークフロー

Figure 1　Translation workflow

まずそれぞれのシステムのソースを準備する(①). 次にそれらのソースから翻訳情報が含まれるファイルを gettext 形式の PO ファイルに変更し(②), 一世代前のバージョン翻訳時に生成された前版 PO ファイルを参照して文脈依存の翻訳を適用し(③), それらを PC 上で稼働するツールである Transifex Client にて翻訳支援システムにアップロードする(④). Web ベースの翻訳支援システムでは，前のバージョン翻訳時に生成された前版翻訳メモリ(図中，前版 TMX)を適用して翻訳を行う. 翻訳と同時に翻訳メモリが更新され，新たな共通翻訳メモリ(図中，共通 TMX)が作成される(⑤). 翻訳が完了した後，前述の Transifex Client にて PO ファイルをダウンロードし(⑥), それらの PO ファイルをそれぞれのシステムのソース形式に変換する(⑦). こうして翻訳された翻訳後ソースが作成される(⑧).

将来，新たなバージョンの翻訳を行う場合，図 1 ⑤にて生成される共通翻訳メモリおよび⑥にて生成される PO ファイルを，図 1 の破線で囲んだ枠内にある前版 TMX ファイルおよび前版 PO ファイルとして再利用する．ただし，今回利用した翻訳メモリ(図中，前版 TMX）は図 1 のワークフローの初回実行であり，共通翻訳メモリ(図中，共通 TMX)が存在していなかったため，Sakai, Moodle, Mahara のそれぞれのシステムの翻訳から得られた翻訳メモリを結合して作成した．当初，整理すべき対象としては翻訳メモリのみを考えていたが，プロジェクト途上にて PO ファイルによる文脈を考慮した翻訳も有効であることが明らかになったため，PO ファイルについても浄化の対象とした.

2.2 PO ファイルと TMX ファイル

PO ファイルおよび翻訳メモリファイル(以下，TMX ファイル)について，その構造と事例を図 2[5]および図 3[6]に示す. いずれの構造も英語ソースと日本語翻訳からなる一組の翻訳ユニットが基本となり，Sakai, Moodle, Mahara のそれらを結合した PO ファイルおよび TMX ファイルは数万組に及ぶ翻訳ユニットとヘッダー情報から構成される.

PO ファイルおよび TMX ファイルはテキスト形式のファイルであるため，キーあるいはタグをフィールドとしてテーブルを定義することによって，リレーショナルデータベースを用いて管理することができる.

キー	概要
#.	OSS におけるソースの変数名
#:	ソース内での出現位置
msgctxt	ソースと変数
msgid	ソース文字列
msgstr	翻訳後文字列
1 翻訳ユニット事例	
#. java.acccess #: access/access-impl/impl/src/bundle/access.properties:1 msgctxt "access/access-impl/impl/src/bundle/ ↵ access.properties: java.acccess" msgid "Access:" msgstr "アクセス:"	

図 2 PO ファイル 構造と事例

Figure 2 PO file description with an example

タグ	概要
<tu>	Translation Unit, 翻訳ユニットを形成する.
<tuv>	Translation Unit Variant, 指定された言語における文字列が含まれる. xml:lang にてロケールを指定する.
<seg>	ソース文字列または翻訳文字列を指定する.

1 翻訳ユニット事例

```
<tu>
  <tuv xml:lang="en">
      <seg>Access</seg>
  </tuv>
  <tuv xml:lang="ja">
      <seg>アクセス</seg>
  </tuv>
</tu>
```

図 3 TMX ファイル 構造と事例

Figure 3 Translation memory file description with an example

2.3 エディタ要件

PO ファイルおよび TMX ファイルを編集する Web ベースの翻訳メモリエディタの主な要件を記述する.

- アジャイル開発とし, 利用時に新たに必要となった機能を随時加えながらシステム開発を行う.
- PO ファイルおよび TMX ファイルから生成する，それぞれ PO テーブルおよび TMX テーブルは同じデータベースにて管理する.
- 英語ソースおよび日本語翻訳からなる一組の翻訳ユニットには生成元システムである Sakai, Moodle, Mahara を属性として追加する.
- 英語ソースを任意の文字列で検索し, 検索語を含むに英語ソースと日本語翻訳を並べて表示する.
- 日本語翻訳を任意の文字列にて like および not like にて検索し，英語ソースの検索と AND 条件で論理演算する.
- Transifex のそれのように, 編集する場合には元となる日本語翻訳をコピーして修正できる.
- PO テーブルおよび TMX テーブルを編集する場合，元となるデータを直接更新せず, 別フィールドに編集結果を記載する.

2.4 実装環境

表 1 に実装環境を示す．開発環境では，あらかじめデータベースに PO テーブルと TMX テーブルを設定し，PO ファイルおよび TMX ファイルから別途開発したプログラムによりデータをインポートし，その環境にて Laravel を用いて翻訳メモリエディタを実装した. Laravel を採用した背景は，データベースの利用が必須であり ORM が適用できること，かつ多くの変更が予想されるユーザインターフェース開発にて HTML テンプレートが使えること, さらに最新のフレームワークであるため多くの機能を有していることである.

表 1 実装環境

Table 1 The development environment

システム	開発環境	公開環境
OS	Windows 7 Professional SP1, 64 ビット	CentOS 6.2
Apache	2.4.10	2.4.7
PHP	5.5.15	5.5.6
MySQL	5.0.11	5.0.11
Laravel	4.2.9	4.2.8

公開環境の設定では，開発環境で編集したテーブルを公開用データベースにインポートし，開発されたプログラムを Apache の DocumentRoot にコピーした.

PO テーブルおよび TMX テーブルの構造については付録に記載する.

3. 実装

図 4 に実装したシステムの Web インターフェースフローを示す.

図中，最上段の図が初期画面となるので，はじめに画面構成の概要を示す. TMXadmin というタイトル下に, PO と TMX の選択情況が表示される．図 4 では翻訳メモリ tmx が選択されている．その下には，英語ソースに対する検索語入力域，演算子，日本語翻訳に対する検索語入力域が順に設置される．図では英語ソースに対する検索語として “item(s)”が入力され, 日本語翻訳に like 演算子でマッチする検索語として任意の文字列を表す “%”が入力されている. その下には，対象とするシステムとして，Sakai, Moodle, Mahara の選択オプションがある．その右には，検索された英語ソースを含む翻訳ユニットのうち，検索された日本語翻訳を含む翻訳ユニットの割合が示され，さらにその右にはページャが設置されている. 以上で検索条件が設定され, 画面下には検索された翻訳ユニットが，1 行に 1 翻訳ユニットにて表示される．フィールドは左から，データベースのテーブルにて付与されている id，その翻訳ユニットの出典元システム oss，英語ソース source，日本語翻訳 target と続く．verify は編集操作を表し，翻訳ユニットを削除する場合には delete，修正する場合には modify を選択する. その右には, 修正した英語ソースの入力域 source if modify, さらにその右には修正した日本語ソースの入力域 target if modify がある．日本語ソースの入力域の下にあるチェックボックスは，日本語ソース target を修正した日本語ソース入力域 target if modify にコピーする場合にチェックする.

ここまで Web 画面の概要を述べたので, 以下に操作の概要を述べる.

初期画面で検索条件を設定した後，図中①で示す”OK”をクリックする．図 4 では検索条件にヒットした sakai お

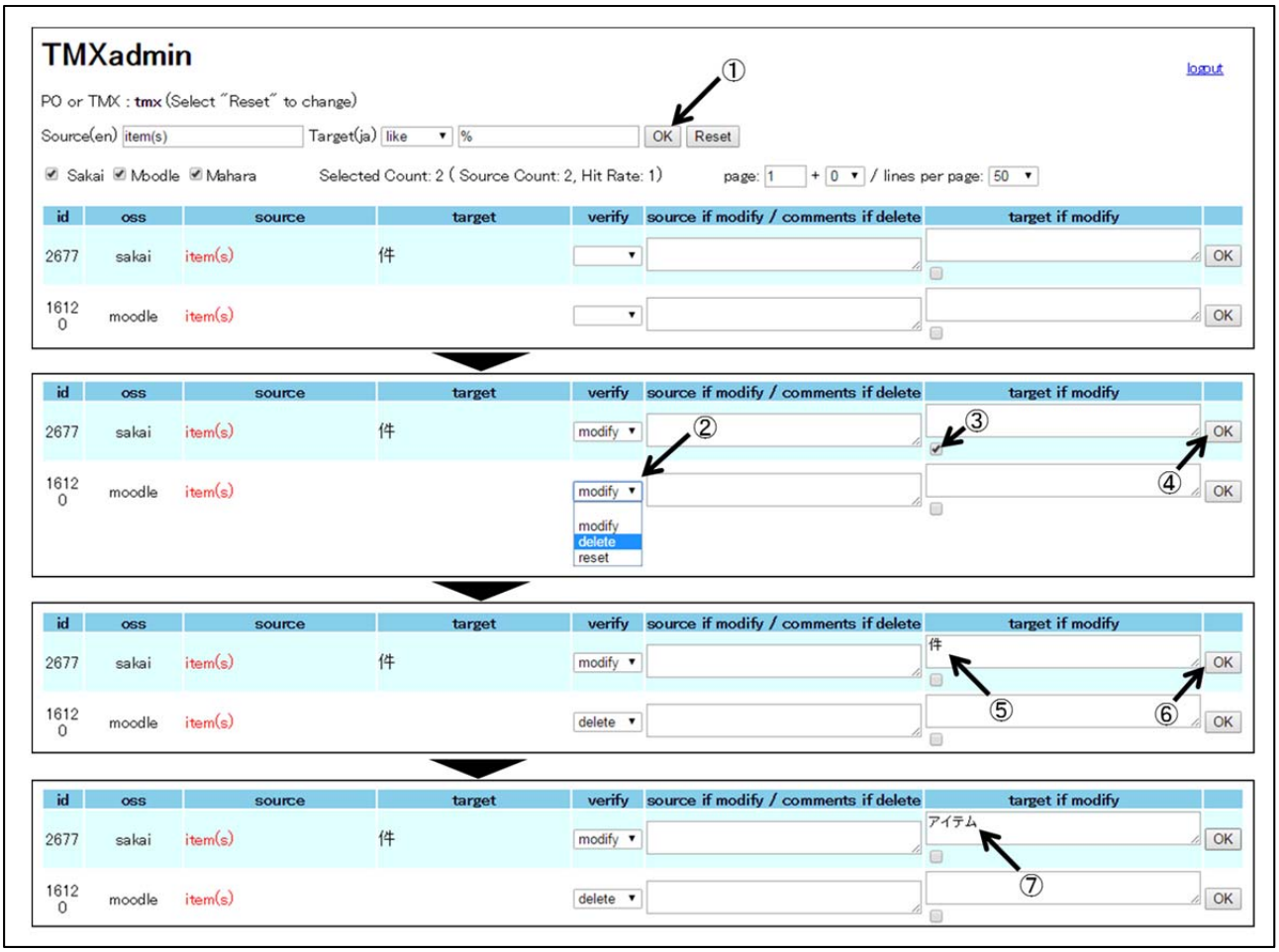

図 4 システム利用事例

Figure 4 A typical use case of the translation memory editor

よび moodle で生成された翻訳ユニットが 1 行目および 2 行目に表示されている．次に sakai で生成された 1 行目の日本語翻訳の修正と，重複して定義されている moodle で生成された 2 行目のレコードを削除する．それぞれのレコードの verify 欄にて処理方法を設定し(②)，1 行目では Sakai の翻訳をコピーするためにチェックボックスにチェックを入れ(③)，最後に”OK”ボタンをクリックする(④)．表示された画面にて，target 欄にコピーされた翻訳(⑤)を”アイテム”に変更して，”OK”をクリックする(⑥)．

表示された最下段の画面では，1 行目のレコードでは item(s)の翻訳が“件”から“アイテム”(⑦)に変更され，2 行目のレコードでは item(s)のレコードを削除するための”delete”が設定されている．

なお，編集を完了した後，修正版 PO ファイルおよび修正版 TMX ファイルをデータベースから書き出すために，別途プログラムを作成した．

4. 翻訳メモリエディタ評価

開発した翻訳メモリエディタを使用して PO ファイルと TMX ファイルの整理を行った．なお事前準備として，Sakai, Moodle, Mahara の翻訳にて生成されたそれぞれの TMX ファイルを結合した TMX ファイルを用いて，同一英語ソースに対して異なる日本語翻訳のリストを作成した．

その結果として，PO ファイルおよび TMX ファイルに含まれる翻訳ユニット数，修正した翻訳ユニット組数，削除した翻訳ユニット組数を表 2 に示す．削除したユニット数にて PO ファイルが 0 で TMX ファイルが 1,532 である理由は以下の通りである．

- PO ファイルでは文脈依存の情報を有するため重複していても，適用個所が異なるので問題は生じない．
- TMX ファイルでは文脈情報を有さないため，基本的には英語ソースに対して唯一の日本語翻訳が望ましい．そのため，同じ英語ソースに対して異なる日本語翻訳がある場合には，それらを削除する．

表 2　PO ファイルおよび TMX ファイル編集結果
Table 2 Summary of editing PO file and TMX file

	PO ファイル	TMX ファイル
翻訳ユニット数	41,096	33,554
修正したユニット数	219	143
削除したユニット数	0	1,532

次に開発した翻訳メモリエディタの評価について述べる．本システムは評価基準を設定して開発されたわけではないため，定量的な評価結果ではなく，得られた知見について示す．

- 数万組といったデータを処理する場合にはプログラムあるいはツールによる処理が不可欠である．そのためには，データのデータベース化と容易にプロトタイピングができる環境が必要である．
- 修正および削除といった編集処理は元となったテーブルには直接適用せず，別フィールドに処理区分，修正後の内容を記載しているため，修正前後の比較が翻訳ユニットごとにでき，かつ検索することができる．
- テーブルを参照するだけであれば MySQL では標準的に使われる phpmyadmin で様々な検索条件で検索ができるが，英語ソースに対する日本語翻訳を繰り返し確認するような場合には，操作が簡略化される専用のアプリケーションが望ましい．

このように専用のプログラムを開発することによって得られるメリットは多い．しかしながら，プログラム開発にはそれなりの時間を要する．本プロジェクトでは Laravel フレームワークを用いて開発を行ったので，そこで得られた知見を次章で示す．

5. Laravel フレームワーク評価

5.1 Laravel フレームワーク

Laravel は PHP をベースとしたフレームワークで，2011 年にリリースされた．2005 年にリリースされた CakePHP などに比べ，新しいフレームワークである．そのため Ruby on Rails, CakePHP などのフレームで取り入れられている主な機能が利用できる．また他のフレームワークと同様ネット上で多くの情報を入手できる[7]．執筆時点で Laravel は github[8]あるいは Google Trend[9]にて最も注目されている PHP フレームワークと言われている．

本開発では，次の機能を用いることによって，単に PHP で開発することに対して，フレームワークを利用するメリットを確認できた．

- アプリケーションを Model, View, Controller に分割する MVC 構造のサポート
  データベースを定義する PHP ファイルは models ディレクトリ，ユーザインターフェースを定義する PHP ファイルは views ディレクトリ，処理を定義する PHP ファイルは controllers ディレクトリに配置される．従って，ひとつの PHP ファイルにデータベース制御，各種処理, HTML 書き出しといったすべてのコードを記述してしまうことはなく，読みやすいプログラムとなる．
- データベースのテーブルをクラスとして参照できる ORM
  TMX テーブルの検索では次の事例のように，TMX テーブルをマッピングした Tmx クラスに続けて検索のためのメソッドを記述するので，コードが簡略化される．下記事例では TMX テーブルに対して，検索条件で指定した select 文が実行され，その結果が配列 sql に格納される．

```
$data['sql']= Tmx::whereRaw(検索条件) -> get();
```

- View における HTML 表示ルーチンを条件に応じて制御できる機能

```
<?php
if ($table === 'po'){
echo '<th width="8%">module</th>';
}
?>
```

通常の PHP では上記のような制御構文を使うが，Laravel では@で識別される Laravel 特有の制御構文を利用できる．

```
@if ($table === 'po')
<th width="8%">module</th>
@endif
```

5.2 研究における Laravel の意義

大学における Web アプリケーションの研究開発において Laravel を利用する意義を下記に示す．

- オープンソース(MIT)であり，マルチプラットフォームであるため，大学の研究基盤および複数大学からなるコミュニティの研究基盤として適合する．
- MVC 構造，ORM, HTML 条件制御などの概念は CakePHP, Ruby on Rails, Python の Django などのフレームワークで採用されており，それらの機能を体験できる．また Laravel の利用で得られた概念を他のフレームワーク利用時に応用できる．
- Laravel は Restful インターフェースの作成，メール，キューサービスなどのサービス，phpunit といったユニットテスト， マイグレーションの rollback など，開発したアプリケーションの機能向上に役立つ豊富な機能を有している．

6. おわりに

翻訳支援システムによるOSSの翻訳で利用するPOファイルおよびTMXファイルは数万組の翻訳ユニットから構成される．これらのファイルは翻訳のたびに新たな翻訳ユニットが追加されるが同時に不整合が生じていく．この不整合の浄化のための編集を行うには，対象とするデータ数からツールあるいはプログラムを用いる必要があるが，プログラム開発に要する工数はできるかぎり少ない方が望ましい. そこで従来ではPHPなどによりこうしたアプリケーションを開発することが一般的だったが，執筆時点で最も注目を浴びているLaravelフレームワークを使ってWebベースの翻訳メモリエディタの開発を行った．その結果として必要な機能条件を実現でき，POファイルおよびTMXファイルの整理を行うことができた．

今後ビッグデータなどを取り扱う場合には何らかのWebアプリケーションを開発することが求められるであろうが，PHPによるプログラミングに習熟されている場合にはLaravelはひとつの選択肢であろう．本稿ではLaravelを用いて，翻訳メモリエディタを題材としてWebアプリケーションを開発したが，今後同様のアプリケーション開発をされる際に参考になれば幸いである．

謝辞

本研究はJSPS科研費 25280127の助成を受けたものです．

参考文献

1) http://sourceforge.jp/projects/benten/
2) https://www.transifex.com/
3) https://crowdin.com/
4) 常盤祐司・出口大輔・宮崎誠・平岡斉士・喜多敏博・梶田将司：教育用オープンソースソフトウェア群のローカライゼーションと共通翻訳メモリの開発　ー一貫性のある用語による教育支援システムを目指してー, 情報処理学会デジタルプラクティス, pp 79-88, Vol.6 No.2 (Apr. 2015)
5) GNU gettext utilities, https://www.gnu.org/software/gettext/manual/gettext.html
6) TMX 1.4b Specification: http://www.gala-global.org/oscarStandards/tmx/
7) Laravel - The PHP framework for web artisans, .http://laravel.com/
8) PHP Framework (online), available from <https://github.com/search?l=PHP&o=desc&q=php+framework&s=stars&type=Repositories&utf8=%E2%9C%93> (accessed 2015-04-20)
9) Google トレンド(online), available from <http://www.google.com/trends/explore?hl=ja#q=%2Fm%2F0cdvjh%2C%20CakePHP%2C%20symfony%2C%20codeigniter%2C%20laravel&cmpt=q&tz=> (accessed 2015-04-20)

付録

付録 A.1 POテーブル構造

veryfyにはmodify, deleteといた編集方法が設定される．msgid_modおよびmsgstr_modはそれぞれmsgidおよびmsgstrの編集後の文字列が設定される．msgid_modにはverifyがdeleteの場合には，その理由が記録される．

フィールド	型	対応する POファイル項目
id	int(10) unsigned	
oss	varchar(255)	
string	varchar(255)	#.
reference	varchar(255)	#:
msgctxt	varchar(4095)	msgctxt
msgid	varchar(4095)	msgid
msgstr	varchar(4095)	msgstr
verify	varchar(255)	
msgid_mod	varchar(4095)	
msgstr_mod	varchar(4095)	
created_at	timestamp	
updated_at	timestamp	

付録 A.2 TMXテーブル構造

veryfyにはmodify, deleteといた編集方法が設定される．en_modおよびja_modはそれぞれenおよびjaの編集後の文字列が設定される．

en_modにはverifyがdeleteの場合には，その理由が記録される．

フィールド	型	対応する TMXファイル項目
id	int(10) unsigned	
oss	varchar(255)	
en	varchar(4095)	<tuv xml:lang="en">
ja	varchar(4095)	<tuv xml:lang="ja">
verify	varchar(255)	
en_mod	varchar(4095)	
ja_mod	varchar(4095)	
created_at	timestamp	
updated_at	timestamp	