D5.1.2: XLIFF DEEP WEB MT TRAINING EXPORTER

David Filip, Milan Karásek, Sean Mooney, David O'Carroll, Ray Kearney

Distribution: Public

MultilingualWeb-LT (LT-Web)
Language Technology in the Web

FP7-ICT-2011-7

Project no: 287815

Document Information

Deliverable number:	5.1.2
Deliverable title:	XLIFF Deep Web MT Training Exporter
Dissemination level:	PU
Contractual date of delivery:	30 September 2013
Actual date of delivery:	30 September 2013
Author(s):	Milan Karásek, David Filip, David O'Carroll, Ray Kearney
Participants:	Moravia, UL
Internal Reviewer:	Cocomore
Workpackage:	WP5
Task Responsible:	UL
Workpackage Leader:	DCU

Revision History

Revision	**Date**	**Author**	**Organization**	**Description**
1	30/09/2013	David Filip, Milan Karásek	Moravia, UL	Draft
2	20/10/2013	David O'Carroll	UL	Penultimate Draft
3	25/10/2013	David Filip, David O'Carroll	UL	Revised Version
4	4/12/2013	David Filip	UL	Final Version with feedback from internal review implemented

CONTENTS

1. Executive Summary

The goal of this deliverable is to provide a means how to use ITS 2.0 metadata available on bilingual content produced from ITS 2.0 decorated deep web content.

This goal has been achieved by the provided ITS 2.0 aware XLIFF Corpus Webservice.

This deliverable makes use of infrastructure and resources delivered under

D3.1.2 XLIFF Roundtripping plus XSLT for Hidden Web Formats and

D4.3 XLIFF Roundtripping Prototype based on M4Loc Work and Okapi Tools

The component itself was developed from scratch under the LT-Web funding and is ITS2 aware by design.

In the following enabling technologies are briefly described and an account of the corpus builder is given.

2. Introduction

ITS2 aware XLIFF is an ideal format for building MT training corpora in general and deep web training corpora in particular.

The same infrastructure can be used for building corpora based on shallow web or other publishing formats, however the underlying deep web format or CMS is likely to provide a fuller set of ITS2 metadata that will in turn allow for more fine-tuned approach to specialized corpora building based mainly on term, text analytics, domain, translate. Obviously, the data set being worked on can contain the full set of ITS 2.0 metadata and any number of other metadata based on XLIFF modules and extensions, some of the ITS2 categories are however particularly useful for profiling corpora. This is why the below described functionality works with translate, domain, term, and text analytics for now. Other sorts of filters can be creatively devised based on other sorts of metadata or their combinations.

3. Technology Overview

The ecosystem of tools that has been made ITS 2.0 metadata aware or developed as ITS 2.0 metadata aware under the deliverables D3.1.2 and D4.3 produces as its natural side effect a large amount of ITS 2.0 decorated aligned bitext.

Such data is ideally suited to be sliced and diced in order to produce finely tuned bilingual corpora. Such corpora in turn can be used by statistical machine translation systems (SMT) such as Moses as training corpora for their translation models.

3.1. XLIFF

XLIFF (XML Localisation Interchange File Format) is an XML-based format created to standardize the way localizable data are passed between tools during a localization process. [1][2]

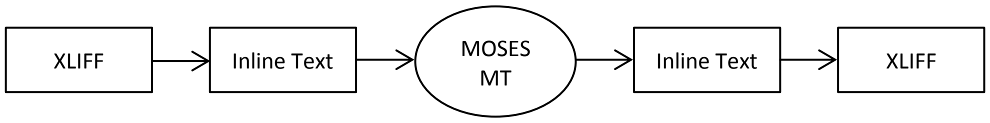
At the time the LT-Web project ends the latest official OASIS standard is XLIFF Version 1.2, nevertheless the implemented roundtripping prototype also supports the new Version 2.0 which is currently (fall 2013) under a Public Review.

3.2. Moses MT

Moses is a statistical machine translation (SMT) system that allows for automatic training of translation models for any language pair. A collection of previously translated texts (bi-lingual parallel corpus) is used for training of the translation model. Once the trained model is created, an efficient search algorithm quickly finds the highest probability translation among the exponential number of choices. [3][4]

3.3. M4Loc

The goal of the M4Loc project is to provide tools to translate localization-specific formats with Moses and to integrate Moses in localization workflows. The M4Loc is dealing with main problem during MT processing in Moses: in-line tags. As Moses MT can work only with a plain text at the input, it is not possible to preserve inline tags in localisation formats, moreover at the proper place. The M4Loc ensure that with a set of tools, including Okapi Framework enabling us to convert all localisation formats (like XLIFF) into the “InlineText” file format which is understandable to Moses MT and (after translation) can be converted back into original format. [5][6]



3.4. Okapi Framework

The Okapi Framework is a set of interface specifications, format definitions, components and applications that provides an environment to build interoperable tools for the different steps of the translation and localization process.

The goal of the Okapi Framework is to allow tools developers and localizers to build new localization processes or enhance existing ones to best meet their needs, while preserving a level of compatibility and interoperability. It also provides them with a way to share (and re-use) components across different solutions. The project uses and promotes open standards, where they exist. For the aspects where open standards are not defined yet, the framework offers its own. The ultimate goal is to adopt the industry standards when they are defined and useable. [7]

3.5. SOLAS

SOLAS (Service Oriented Localisation Architecture Solution) is a component-based localisation platform that seeks to address the emerging challenges of user driven localisation. This innovative platform empowers the content owner and the community that seeks the content in a specific language, in particular the small-to-medium sized enterprises and NGOs (not-for-profits and charities). [8]

The distributed nature of SOLAS allows for cross-organizational localisation. Tasks can be automated by components on behalf of the user, while also allowing them to integrate fully featured applications as components using a set of RESTfull interfaces.

Data is shared between the components using XLIFF (the XML Localization Interchange File Format) that acts as the single uniform message format throughout any SOLAS orchestrated roundtrip.

For details of the SOLAS platform see deliverable **D3.1.2: XLIFF Roundtripping plus XSLT for Hidden Web Formats**

3.6. BaseX

Since SOLAS is using as its sole message format the XLIFF, which is an XML vocabulary, its datastore is best implemented based on an XML database. UL has chosen BaseX http://basex.org/ as the XML database technology underlying its datastores.

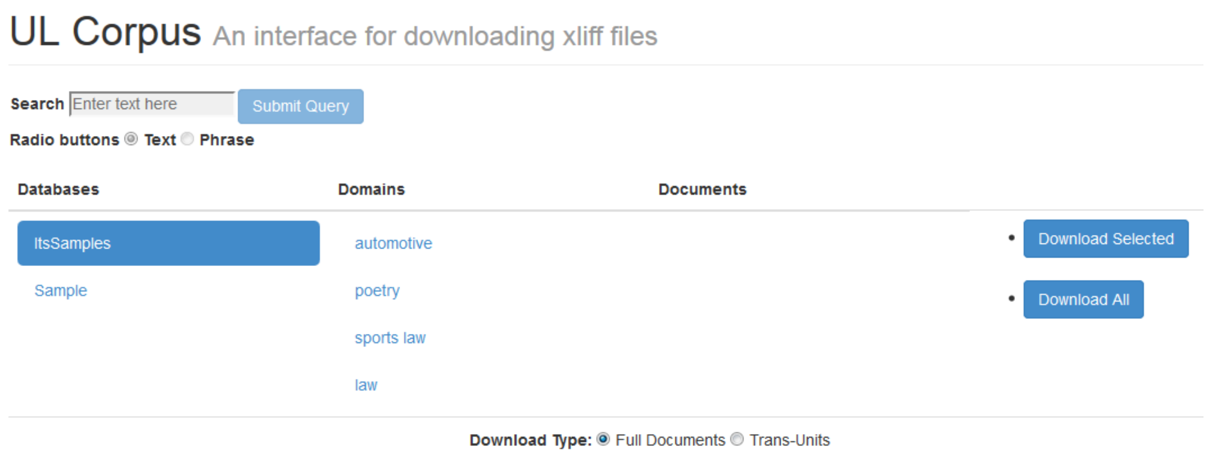
XLIFF files handled by LocConnect and a few other SOLAS components are stored in a single BaseX datastore that is therefore suitable for XPath and other XML native querying.

4. Corpus Web Service

The corpus web service can be used to build a corpus of XLIFF files. It provides an API written in Python for retrieving and storing XLIFF files. The files are stored in a BaseX database. BaseX is an XML database that can be queried using XPath expressions. Using these XPath expressions it is possible to query the DB using some sophisticated queries. For example, it is possible to pull all the trans-units from any of the files that are in a certain domain. Using this API it should be possible to build relevant training data for MT engines.

A thin web interface was written using Google's Dart language which can be used to store files in a corpus or retrieve files from a corpus. It calls into the Python API and serves as an example of how to call the web service. Using the web service it is possible to generate a file which contains the XLIFF of all documents in the corpus with specific domains.

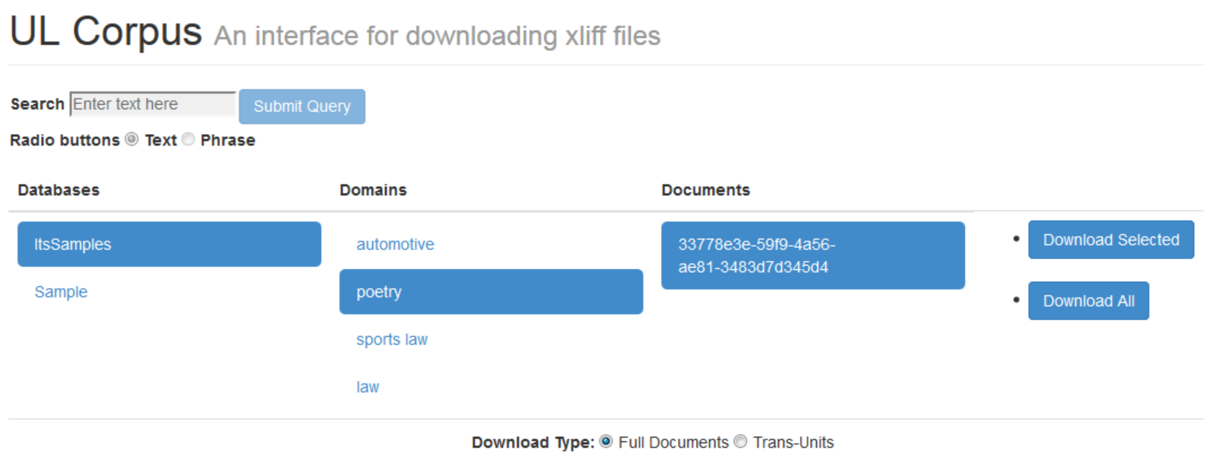
Below is a screenshot of the corpus web service web interface. It shows a list of available databases. It also shows domains available for the selected database.



Download Type: [✓ Full Documents] Trans-Units

Corpus web UI with a selected database

The next image is the same interface with a domain and document selected. The buttons on the right allow the user to download just the selected files or all the files with the selected domain. The radio buttons at the bottom change the download behaviour. If the Full Documents button is selected then the entire document will be present in the downloaded file. If the Trans-Units button is selected only the trans units from the selected documents will be downloaded.



Download Type: [✓ Full Documents] Trans-Units

Corpus web interface with documents selected.

When downloading XLIFF files from the corpus UI they are wrapped in some simple custom XML. Below is the format of the downloaded XML (Note: the actual XLIFF file has been left out for simplicity).

```
<dbs>
     <db id="ItsSamples">
          <docs>
               <doc id="33778e3e-59f9-4a56-ae81-3483d7d345d4">
                    <xliff>...</xliff>
               </doc>
               <doc id=”231fddf3-5d39-vs34-124f-123f35fd365g”>
                    <xliff>...</xliff>
               </doc>
          </docs>
     </db>
</dbs>
```

The corpus web interface can be accessed at http://demo.solas.uni.me/corpus-ui/

4.1. ITS 2.0 Support

The Coprus Webservice currently allows for subsetting of XLIFF files, groups and units by the following ITS 2.0 data categories

- Domain – i.e. content belonging to particular domains
- Term – content containing particular terms
- Text Analytics - content containing particular term candidates provide by a text analytics service

Support for other categories such as Provenance or Localization Quality issue and rating can be easily added.

REFERENCES

1. XLIFF definition at Wikipedia,
   http://en.wikipedia.org/wiki/XLIFF

2. OASIS XML Localisation Interchange File Format (XLIFF) TC homepage
   https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xliff

3. Moses MT at Wikipedia
   http://en.wikipedia.org/wiki/Moses_(machine_translation)

4. Moses MT project homepage
   http://www.statmt.org/moses/

5. M4Loc project homepage
   https://code.google.com/p/m4loc/

6. Okapi Framework
   http://okapi.sourceforge.net/index.html

7. Service-Oriented Localisation Architecture Solution (SOLAS)
   http://www.localisation.ie/solas/

APPENDIX

List of methods of the webservice

Method: showDatabases()

Route(Get): /dbs/

Description: Returns all database names in the system

Output:

```
<dbs>
    <db id='  database ID ' />
</dbs>
```

Method: showDocumentIds(dbName)

Route(Get): /dbs/<dbName>/

Description: Returns all document id's in a specific database

Output:

```
<dbs>
        <db id=' database ID ' >
                <docs>
                        <doc id=' document ID '>
                        </doc>
                </docs>
        </db>
</dbs>
```

Method: showDocumentsInDatabase(dbName)

Route(Get): /dbs/<dbName>/docs/

Description: Returns all documents in a specific database

Output:

```
<dbs>
        <db id=' database ID ' >
                <docs>
                        <doc id=' document ID '>
                                {Full xliff document}
                        </doc>
                </docs>
        </db>
</dbs>
```

Method: createDoc(dbName)

Route(Post): /dbs/<dbName>/docs/

Description: Sends document to the database

Output: *None – Post function*

Method: retrieveElement(dbName, idStr)

Route(Get): /dbs/<dbName>/docs/<idStr>/

Description: Returns specified document from specified database

Output:

```
<dbs>
    <db id=' database ID ' >
        <docs>
            <doc id=' document ID '>
                {Full xliff document}
            </doc>
        </docs>
    </db>
</dbs>
```

Method: retrieveDomainsInDocument(dbName,idStr)

Route(Get): /dbs/<dbName>/docs/<idStr>/domains/

Description: Returns all domains in a specified document

Output:

```
<dbs>
    <db id=' database ID ' >
        <docs>
            <doc id=' document ID '>
                <domains>
                    <domain id = 'domain ID' />
                </domains>
            </doc>
        </docs>
    </db>
</dbs>
```

Method: retrieveDomains(dbName)

Route(Get): /dbs/<dbName>/domains/

Description: Returns all domains in a database

Output:

```
<dbs>
        <db id=' database ID ' >
                <domains>
                        <domain id = 'domain ID' />
                </domains>
        </db>
</dbs>
```

Method: retrieveDocsByDomain(dbName, domain)

Route(Get): /dbs/<dbName>/domains/<domain>/

Description: Returns document id's with a specified domain

Output:

```
<dbs>
        <db id=' database ID ' >
                <docs>
                        <doc id=' document ID '>
                        </doc>
                </docs>
        </db>
</dbs>
```

Method: retrieveDocumentsInDomains(dbName, domain)

Route(Get) /dbs/<dbName>/domains/<domain>/docs/

Description: Returns all documents with a specified domain

Output:

```
<dbs>
        <db id=' database ID ' >
                <docs>
                        <doc id=' document ID '>
                                {Full xliff document}
                        </doc>
                </docs>
        </db>
</dbs>
```

Method: retrieveTransUnitsDataBase(dbName, domain)

Route(Get): /dbs/<dbName>/domains/<domain>/trans-units/

Description: Returns transunits which contain specified domains from a specified database

Output:

```
<dbs>
    <db id=' database ID ' >
        <docs>
            <doc id=' document ID '>
                <xliff>
                    <file>
                        <header/>
                        <body>
                            <trans-unit id='trans-unit
ID'>
                                {xliff document
segment}
                            </trans-unit>
                        </body>
                    </file>
                </xliff>
            </doc>
        </docs>
    </db>
</dbs>
```

Method: updateElement(dbName, idStr)

Route(Delete): /dbs/<dbName>/docs/<idStr>/

Description: Delete a specified document

Output: *None – Delete function*

Method: dropDatabase(dbName)

Route(Delete): /dbs/<dbName>/

Description: Drops the specified database

Output: *None – Delete function*