ACCURAT

Analysis and Evaluation of Comparable Corpora
for Under Resourced Areas of Machine Translation

www.accurat-project.eu

Project no. 248347

Deliverable D3.5
Tools for building comparable corpus from the Web

Version No. 3.0
29/06/2012

Document Information

Deliverable number:	D3.5
Deliverable title:	Tools for building comparable corpus from the Web
Due date of deliverable:	31/10/2011
Actual submission date of deliverable:	31/10/2011 (version 1.0) 31/03/2012 (version 2.0) 29/06/2012 (version 3.0)
Main Author(s):	Ahmet Aker, Evangelos Kanoulas, Judita Preiss, Monica Paramita, Rob Gaizauskas, Paul Clough, Emma Barker, Nikos Mastropavlos, Nikos Glaros, Radu Ion, Tiberiu Boroș, Mārcis Pinnis
Participants:	USFD, ILSP, RACAI, Tilde
Internal reviewer:	Tilde
Workpackage:	WP3
Workpackage title:	Methods and techniques for building a comparable corpus from the Web
Workpackage leader:	USFD
Dissemination Level:	**PU**: Public
Version:	V3.0
Keywords:	Retrieval tools, web crawler, focused crawler, news search, parallel and comparable corpora

History of Versions

Version	Date	Status	Name of the Author (Partner)	Contributions	Description/ Approval Level
V0.1	01/09/2011	Draft	USFD	Skeleton	Skeleton
V0.1	20/09/2011	Draft	ILSP, USFD	Contributions	Edited introduction, related work and the ILSP focused crawler
V0.1	21/09/2011	Draft	RACAI, USFD	Contributions	Edited RACAI crawler
V0.1	28/09/2011	Draft	USFD	Contributions	Early draft
V0.2	30/09/2011	Draft	USFD	Contributions, Drafting	For Internal Review
V0.3	10/10/2011	Draft	USFD	Response to Comments	For Internal Review
V0.4	27/10/2011	Draft	USFD	Grammar/style revisions	For Internal Review
V0.5	30/10/2011	Draft	ILSP	Response to Comments	For Internal Review
V1.0	31/10/2011	Final	Tilde	Final Review	Submitted to PO
V2.0	31/03/2012		Tilde, ILSP	Update after review	Submitted to PO
V2.1	09/05/2012		Tilde, USFD	USFD updates for final version	For Internal Review
V2.2	04/06/2012		Tilde, ILSP,	FMC and News	For Internal Review

			USFD	Downloader updates	
V2.3	25/06/2012		Tilde	Document prepared for release	For Internal Review
V3.0	28/06/2012	Final	Tilde		Version 3.0 released

EXECUTIVE SUMMARY

This document contains technical documentation of tools which have been used within the ACCURAT project to gather parallel and comparable corpora from the web. The tools include focused web crawlers, a Wikipedia corpus collector and news search tools.

These corpora collecting tools together with the toolkit for multi-level alignment and information extraction from comparable corpora (D2.6 deliverable) provide interested users with a complete suite of tools (ACCURAT toolkit) for acquiring general domain or domain-specific training data for their SMT or Example-based/Rule-based MT systems.

This deliverable contains step-by-step instructions explaining how to install and run the tools for comparable corpora acquisition. Significant effort has been made to ensure these instructions are understandable by users with average computer skills.

The ACCURAT Toolkit is stored at the ACCURAT repository and is freely available after completing the registration form (http://www.accurat-project.eu/index.php?p=toolkit).

Table of Contents

Abbreviations

Abbreviation	Term/definition
ACCURAT	Analysis and Evaluation of Comparable Corpora for Under Resourced Areas of Machine Translation
CLIR	Cross Lingual Information Retrieval
FMC	Focused Monolingual Crawler
MCC	Multilingual comparable corpora
MT	Machine Translation
RSS	Rich Site Summary
SMT	Statistical Machine Translation

1. Introduction

Important technological achievements in the Machine Translation (MT) field have significantly raised the output quality of contemporary MT systems. This has led to the systems being more and more widely adopted as core components in many translation solutions. As a result, a constantly increasing global demand for MT systems has been established. This in turn entails rising demand for parallel textual data[1] across all possible language pairs and domains, since the performance of most MT systems depends on large quantities of high quality parallel data. Traditionally, parallel corpora are the major source from which the required parallel language resources can be automatically obtained by using one of several well known alignment methodologies[2]. However, parallel corpora are hard to find, especially for less widely spoken languages and for narrow domains, as well as expensive to create.

The ACCURAT project is investigating for cost-effective and innovative ways to automatically produce parallel information for training MT systems and improving their overall performance. In particular, the project is investigating methods and techniques for extracting of parallel data from bilingual *comparable corpora*, i.e. bilingual corpora of source language-target language text pairs where the paired texts are not exact translations of each other but are related in some weaker way, e.g. by being on the same topic or about the same events.

Comparable corpora are potentially easier to build than parallel corpora for a large variety of languages and for many specific thematic areas. At the same time bilingual comparable corpora are likely to contain parallel information required for the training of MT systems. Although bilingual corpora can be comparable at a variety of levels and in various aspects[3], they are only able to improve MT system performance when they contain a good number of parallel textual segments. Therefore, ACCURAT focuses on gathering and processing bilingual comparable text corpora containing a significant amount of mappable textual data or, equivalently, on **bilingual "*mappable*" corpora**. The ACCURAT workflow (from the most abstract viewpoint) iterates through three major steps: (i) develop tools for comparable corpora acquisition and use those tools to collect such corpora, (ii) find efficient methods to extract parallel data from the collected comparable corpora and (iii) evaluate any possible gain in the performance of MT systems[4] originated from the extracted parallel data.

In respect to the first of the above three essential steps, ACCURAT has investigated efficient methods and developed tools for identifying and gathering large amounts of comparable textual data from the web[5], thus facilitating the building of comparable bi(multi)lingual corpora for under-resourced languages and narrow domains. Many different and apparently heterogeneous corpus collection techniques were explored, developed and tested extensively. This was necessary because it was not known at the outset how "mappable" the corpora that different techniques deliver would be. Moreover, the aim of collecting multilingual comparable corpora with significant mappable sentential or subsentential pieces of text is inherently hard to meet. The most successful of the corpus collection approaches have been identified, described and specified in D3.4 *Report on methods for collection of comparable corpora*. In the present deliverable (D3.5) prototype tools based on those approaches have been refined and documented to a level suitable for public release.

[1] That is, bilingual sentences, phrases, words, etc. that are translation equivalents of each other.

[2] D2.1 deliverable provides a survey of them.

[3] The concept of corpus comparability has been extensively specified and documented in the D1.1, D1.2 and D1.3 deliverables.

[4] Both Statistical MT and Rule-Based MT systems are considered here.

[5] While they may not be directly applicable, it is straightforward to adopt and apply our the methods for building comparable corpora from the web to digital archives or other off-line very large textual data collections.

In this deliverable, we first discuss related studies which focus on collecting comparable corpora (Section 2). Next, we describe the tools that the ACCURAT project has produced, including "how to install and use" instructions for public users (Section 3).

2. Related Work

Although the significance of comparable corpora has been acknowledged by the research community, as this is witnessed by the related literature, there are few reports on tools for collecting comparable corpora.

Indeed, several publications concerning the exploitation of comparable corpora can be found in the related literature. Tao and Zhai (2005) presented a method for examining frequency correlations of words in different languages in comparable corpora in order to find mappings between words and achieve cross-lingual information integration. Munteanu and Marcu (2006) attempted to extract parallel sub-sentential fragments from comparable bilingual corpora using a signal-processing approach for producing training data sets for MT systems. A framework for exploiting comparable and parallel corpora for generating named entity translation pairs was introduced by Hassan et al. (2007). Finally, efforts on taking advantage of comparable corpora for improving MT in less-resourced languages have recently started; an investigation on translation systems on eighteen European language pairs and preliminary SMT (Statistical Machine Translation) models with the purpose of discovering parallel parts within comparable corpora was presented by Eisele and Xu (2010), while Skadiņa et al. (2010) reported on research plans for analyzing and evaluating novel methods that could aid in compensating for the shortage of linguistic resources by using comparable corpora.

On the other hand, the number of available publications that address the issue of building such corpora is very limited. Early approaches were based on readily available resources. Sheridan and Ballerini (1996) introduced an approach for multilingual information retrieval, applying thesaurus-based query-expansion techniques over a collection of documents provided by the Swiss news agency. Braschler and Scäuble (1998) presented a corpus-based approach for building comparable corpora using the TREC CLIR data, while Talvensaari et al. (2007) presented a study which describes how a comparable corpus was built from articles by a Swedish news agency and a U.S. newspaper.

Initial work on acquiring comparable corpora from the web was reported by Utsuro et al. (2002). They collected articles in Japanese and English from News web sites and attempted to align them based on their publication dates. Talvensaari et al. (2008) used a focused crawling system to produce comparable corpora in the genomics domain in English, Spanish and German. Even though our work follows the same methodological approach, two critical differences are: (i) less-resourced languages are targeted, which significantly increases the challenge of this task and (ii) a number of different topical domains are crawled extensively to produce the final results. Ion et al. (2010) presented a customizable application that could be used for building comparable corpora from Wikipedia and the web by merging and organizing different web crawlers. An overview on different methodologies used to collect small-scale corpora in nine language pairs and various comparability levels was reported by Skadiņa et al. (2010) and the collected corpora were investigated in order to define criteria and metrics of comparability.

3. Description of Tools

3.1. A Workflow Based Corpora Crawler

3.1.1. Overview and Purpose

Multilingual comparable corpora (MCC) have been around for a while in the context of Machine Translation (MT) research, as an alternative to parallel corpora which were (and still are for certain pairs of languages and domains) hard to find. By comparison with parallel corpora which contain pairs of equivalent translation units of text (sentences or paragraphs), MCC exist with different degrees of comparability: weakly comparable corpora, strongly comparable corpora, quasi-comparable corpora, very-non-comparable corpora, etc. (Skadina et al. 2010). A general definition of MCC that we find useful is given by (Munteanu and Marcu, 2006). They say that a (bilingual) comparable corpus is a set of paired documents that, while not parallel in the strict sense, are related and convey overlapping information. The measure of this overlapping should give the degree of the comparability between the two documents in a pair (for instance, a real number ranging between 0 and 1 with 0 indicating complete divergence of topic and 1 indicating parallelism: one document is the translation of the other).

We provide a tool for automatic extraction of a parallel or strongly comparable corpus. The logic of the application is controlled from within the FLOW EDITOR which enables the user to easily create and manage workflows thus having a global view of the extraction process. Note that this is not intended as a standalone crawler but more as a development system for data extraction applications.

Command line tools are applications designed to be user operated via a text-only computer interface. There are numerous console-based applications or scripts written in interpreters such as PHP or Perl that provide useful processing resources for text corpora. When trying to combine tools such as these you must create a master application design to provide input/output management for each unit.

Using the workflow approach we are trying to provide the means for interaction between such text-based tools and other applications. There are two ways to accomplish this:

1. When using console application we give a simple regular expression driven mechanism for input/output control. When the flow is executed, the output of each unit is processed and an input for the next unit is generated. If Input/Output regular expressions are defined, these are applied to I/O data.
2. When the first method is unusable, the user can create plugins in order to implement the needed functionality. Plugins are .NET assemblies which implement the *ProcessingBlock* or *DecisionBlock* interfaces.

3.1.2. Software Dependencies and System Requirements

The crawler is written in C# and it requires .NET Framework version 4.0. In order to be able to collect documents from the Internet, an active Internet connection is also required.

3.1.3. Installation

Other than the .NET Framework installation, this application does not require further installation.

3.1.4. Execution Instructions

The use of a workflow gives the means for high scalability and integration of modules written in different programming languages or interpreters. This system gives the advantage of organizing the logic of the application around processing units and decision blocks. The user can alter the global application behavior by adding new blocks or modifying the way the I/O data is being handled. Another advantage is that the independent processing modules are unloaded when no data is available in order to preserve computational resources.

We start by creating a basic crawling workflow. We refer to the units that do the actual work as **active blocks** as opposed to the start and end nodes that are just **visual markers**.

There are two types of active blocks: **decision blocks** and **processing units**. Every block has an external application (script, compiled program, etc.)/plugin associated with it that takes the data from the preceding block, processes it and passes the result to the next block in the chain. By clicking on the active blocks, the user can edit the following parameters:

1. *Name*: represents the label on the block that will be displayed on the screen.
2. *Execution parameters*:

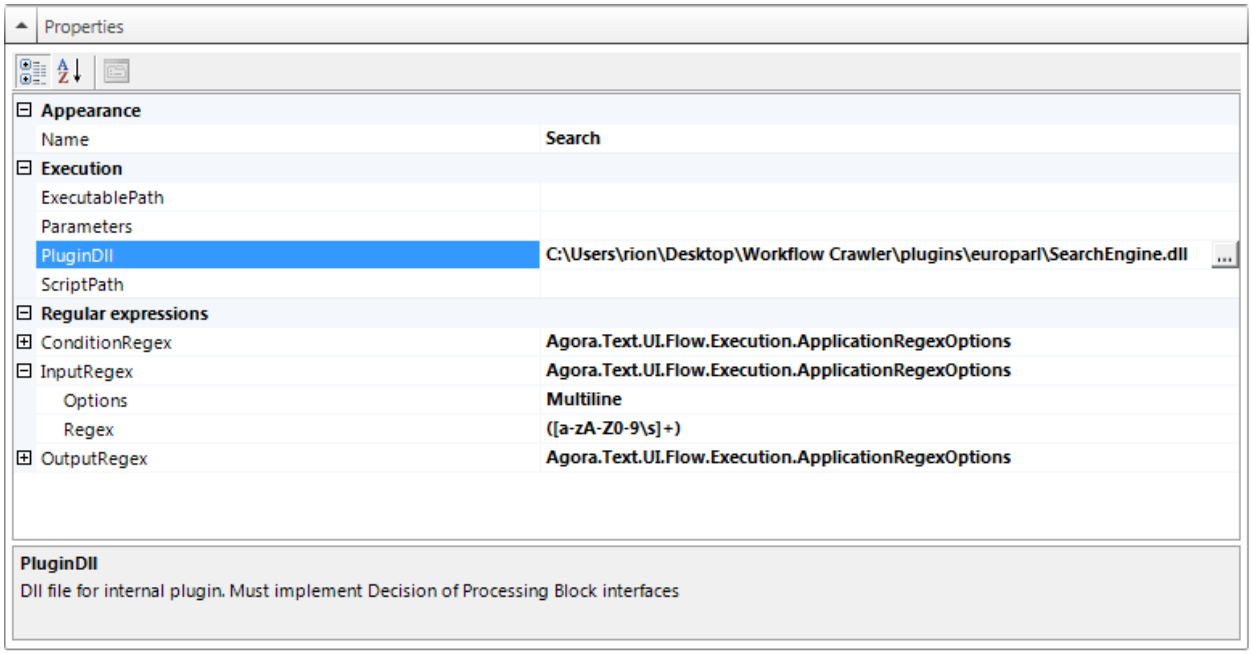

Figure 1: Block property editor

   2.1. ***Executable path***: path to the application that will be executed when the block is invoked. It can be a standalone application or an interpreter for the script.
   2.2. ***Command line parameters***: will be passed to the application. We use special keywords like "$script" for the scrip filename or "$input" for the input produced by the parent node;
   2.3. ***PlugInDLL***: the full path to a C# plugin DLL which implements the *ProcessingBlock* or *DecisionBlock* interfaces (included in the distribution of this crawler);
   2.4. ***Script path***: should be used only with interpreted languages and will be passed as a command line argument.

3. Regular expressions (applied only in case of external applications invoked by the respective block):

   3.1. ***InputRegex***: This will be applied to the text input before it is passed to the external application (script, compiled application, etc.); it must have capturing parentheses because only the captured text is actually passed on;
   3.2. ***OuputRegex***: Used to preprocess the output of the external application before it is passed to the next block
   3.3. ***ConditionRegex***: Used only on decision blocks. Produces "true" if the output matches the regular expression and "false" otherwise.

After the workflow is defined, the user may save it and execute it from the toolbar above the workflow creation area. Depending on the selected plugins, a dialog will open that will ask for information such as the destination folder.

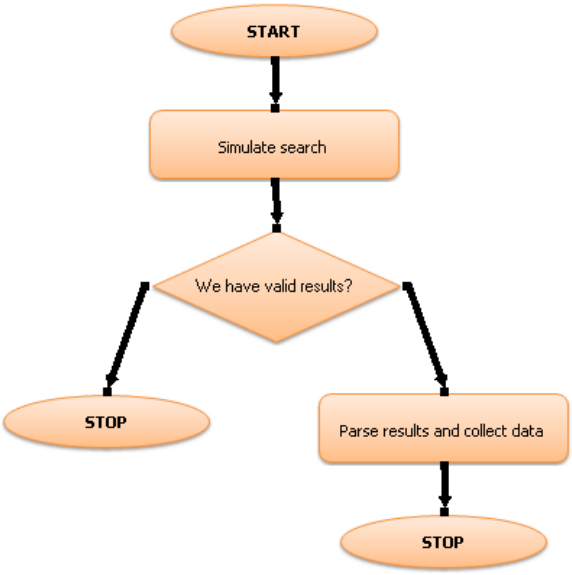

Figure 2: The typical crawling workflow (if we are sure that all results are valid, we may do without the decision block)

The workflow-based crawler is shipped with plugins and workflows (saved in XML files) for crawling Wikipedia the Europarl sites.

3.1.5. Input and Output Date Format

The I/O data formats are not fixed. These are application/plugin dependent but, usually, the "crawl" block will produce HTML documents and the "results processing" block will extract the text out of these documents. For an example, please load and run the "*europarl.xml*" crawling workflow shipped with the crawler.

3.1.6. Creating Blocks

Blocks are .NET assemblies that contain a "*MainClass*" (name is important) and implement one of the two interfaces in "*agora.dll*".

There are two types of blocks that can be written: processing blocks and decision blocks. All blocks share memory, but this will be explained later.

Processing blocks are the actual workers. They must implement the "*ProcessingBlock"* interface and write the logic for the "*ProcessData"* method. The output is of type "*object"* and can be anything from *NULL* (presuming the current block uses the shared memory to communicate with the next block) to lists of objects (eg. "*List<string>"*). In one of the crawlers provided as an example the "*SearchBlock"* returns a list of pages that need to be downloaded as a "*List<string>"* object. The "*ResultsProcessing"* block takes this list and starts downloading the pages in the given folder. The folder location is transmitted through the shared memory. To be more precise we will explain the following lines of the code:

```
1 #region ProcessingBlock Members
2 object ProcessingBlock.ProcessData(object data,
Agora.Builder.System.BaseApplication MyApplication) {
3 string[] urls = (string[])data;
4 String BasePath = MyApplication.Memory[0].ToString();
5 return null;
6 }
7 #endregion
```

Parameters passed to the "*ProcessData"* method (line 2) are:

1. The results from the previous processing block (object data)
2. The context of the application (BaseApplication MyApplication) which is used to access the shared memory (MyApplication.Memory[]).

The list of URLs is obtained by a simple cast of the data object (line 3). There is no restriction regarding the output of one block other than the fact that it must be compatible with the next block. In this case the two blocks are specially designed to work together so the "*ResultProcessing"* block knows exactly what type of data is being handed.

The location where this block should store the results (*BasePath*) is given trough the shared memory system (line 4). Each location in the shared memory is associated with only one plugin. The access can be done as follows:

```
MyApplication.GetMyMemory() – read only
```

or

```
MyApplication.Memory[pluginId] – read/write
```

A plugin can get its pluginId from the application context (MyApplication.CurrentPluginID).

Decision blocks allow shaping the flow of data. They implement the "*DecisionBlock"* interface and must write logic for the following two methods:

```
bool                DecisionBlock.EvaluateCondition(object                data,
Agora.Builder.System.BaseApplication
MyApplication)
And
object                    DecisionBlock.GetData(object                    data,
Agora.Builder.System.BaseApplication
MyApplication)
```

The parameters passed on to these methods have the same meaning and content as explained for the "*ProcessingBlocks"*.

The "*EvaluateCondition"* method returns "*false*" or "*true*" and gives the direction of the data flow (left - *false* or right - *true*). The "*GetData"* method is identical to "*ProcessData"* (Processing blocks). The simplest implementation is a pass through: return data.

3.1.7. Contact

For further information and technical support installing and/or running this tool, please email to Tiberiu Boroș: tibi@racai.ro.

3.2. ILSP Focused Monolingual Crawler (FMC)

3.2.1. Overview and Purpose

The ILSP **Focused Monolingual Crawler** (FMC) tool is used to collect narrow domain bi(multi)lingual comparable corpora from the web. It does so by making a separate crawl for each language specified and by retrieving each time only web pages that are relevant to a pre-defined narrow domain or topic. The comparability of the bi(multi)lingual documents retrieved is achieved by ensuring that, for each language specified, the FMC is made to return web documents that are all close to the same topic.

Given a language pair(or a set of languages) and a topic, the user has to first create a list of **topic-specific** single- or multi-word **terms** as well as a simple **list of URLs** being considered highly relevant to the topic in question. These data (input to the FMC) have to be prepared for each language.

The list of (generally) multi-word expressions related to a specific topic can be created either manually, possibly with the aid of some available online resources (e.g. Eurovoc[6]) or can be automatically extracted from small topic-specific corpora using tf-idf and term extraction algorithms.

The list of topic-related URLs, that the FMC treats as seed URLs, can be constructed semi-automatically (using directories from known search engines, e.g. Yahoo, Google, dmoz), or automatically. One possible way to automate the construction of the URLs list is to use BootCaT's (Baroni et al. 2004) tuple generation algorithm, as follows: first generate a number of n-topic-terms combinations, and then Google search them and keep the top 5 or 10 URL results from each search as candidates for the final seed URL list.

Once topic definition terms and seed URLs list have been generated, the user may then optionally choose to configure the crawler engine. That is, the user has the option to adjust various crawler settings prior to crawling start. For example, the user can set file types to download (e.g. PDF, doc, and xls), domain filtering options (using regular expressions), self-terminating conditions (size or time limits), crawling politeness parameters, etc.

Having concluded with the above steps, the user should next run FMC once per each language.

In short, the main steps the crawler executes are illustrated in **Figure 3**. More details on the underlying FMC workflow are given in the D3.4 deliverable.

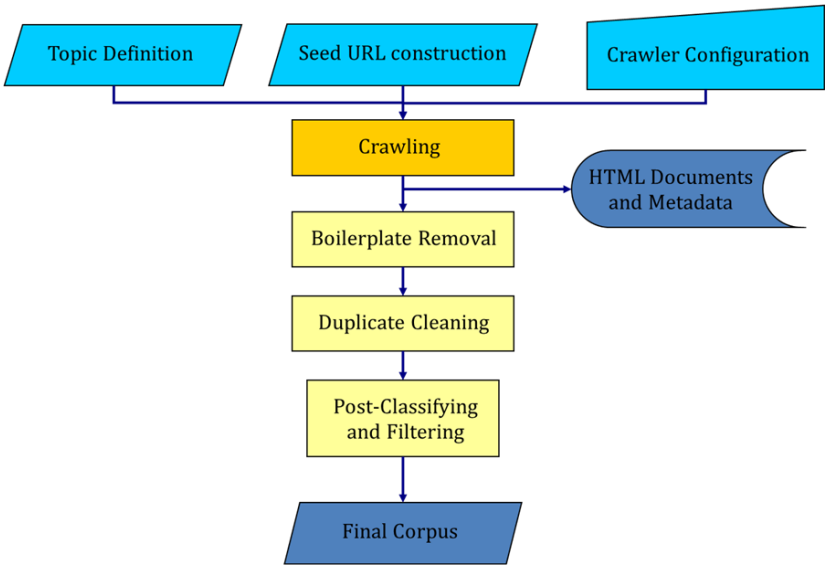

Figure 3 FMC crawler workflow.

3.2.2. Software Dependencies and System Requirements

Since the FMC is using the Bixo[7] open-source web mining toolkit, the software dependencies are shared between the Bixo and FMC projects. All software dependencies for the FMC are bundled with the main executable of the crawler. The final tool is a Java compiled executable that does not require any specific platform to operate. Any system that runs a Java Runtime Environment 1.6 or above can handle the FMC executable.

However, the FMC facilitates the Hadoop[8] clustering libraries in order to be able to run in distributed computing environments. These libraries contain a number of UNIX operations that a Windows system will not support by default.

[6] the EU's multilingual thesaurus, http://eurovoc.europa.eu/

[7] http://bixo.101tec.com/

[8] http://hadoop.apache.org/

In order to provide support for such operations on a Windows system, Cygwin[9] has to be installed. The default installation procedure is enough for providing the required support.

Finally, after Cygwin is installed, the installation path must be added to the PATH environment variable of the Windows system (on a default installation this path will be "*C:\Cygwin\bin*").

To add "*C:\Cygwin\bin*" to the PATH environment variable, simply run the following command from a Windows console:

```
set PATH = %PATH%;C:\Cygwin\bin
```

Similarly, there are no specific hardware requirements beyond those needed for running the JRE. We have tested the tool on both simple workstation and server environments with the following setups:

Table 1 FMC system requirements

	Windows Workstation	**Linux Server**
Operating system	Microsoft Windows XP(sp3)	Fedora 13.0
Memory	4GB	8GB
Hard disk space	2 x 250GB	RAID 5 setup (~1TB)
CPU	INTEL Core 2 @3GHz	INTEL Xeon @ 3GHz (quad core)

Proper cluster setup and testing could not be completed due to lack of adequate hardware resources. However, a small testing Hadoop[10] cluster (up to 3 workstations) was configured and tested with the FMC executing successfully.

Due to the nature of the tool, although it will execute successfully on almost any setup, it is important to have a dedicated machine in order to achieve reasonable performance. On a common crawl run (usually assigning 16-32 processing threads on each machine) it will take approximately 24 hours to fetch an average of 20k html pages. These will be stored in the file system and will require (along with their exported metadata files) about 10 gigabytes of hard disk storage. Therefore, the actual storage requirements depend on the crawl job specifications and schedule.

3.2.3. Installation

There is no real installation procedure for the FMC to work. One should only verify that the FMC executable is located somewhere on the system where the user has "execute" rights. From there on, the user will select the location where the crawling database will be hosted and therefore the executable should be able to access and write in that location (since this is a Java application, all rights are inherited from the user running it, therefore if the user has read/write rights on the specific location, so does the FMC).

3.2.4. Execution Instructions

Currently, there are four modes for running the FMC, which can be displayed by simply executing the JAR file with no arguments:

```
java –jar ilsp-fmc-bixo.jar
```

This will return the following output:

```
usage: ilsp-fmc-bixo.jar
-a,--agentname        user agent name
-c,--crawlduration    target crawl duration in minutes
```

[9] http://www.cygwin.com/

[10] http://hadoop.apache.org/

```
-cfg,--config          XML file with configuration for the crawler.
-d,--domain            domain to crawl (e.g. cnn.com) or path to file
                       with domains to crawl. Use for crawling ONLY
                       inside specific domain(s)
-dbg,--debug           debug logging
-f,--force             Force to start new crawl. Caution: This will
                       remove any previous crawl data (if they exist).
-h,--help              Help
-k,--keepboiler        Annotate boilerplate content in parsed text
-l,--loggingAppender   set logging appender (console, DRFA)
-lang,--language       Target language. If more than one, separate with
                       ';', i.e. en;el
-len,--length          Minimum number of tokens per text block
-n,--numloops          number of fetch/update loops
-o,--outputdir         output directory
-of,--outputfile       output list file
-op,--operation        operation to conduct:
                       crawlandexport|crawl|export|config
-t,--threads           maximum number of fetcher threads to use
-tc,--topic            Topic definition
-te,--textexport       Export plain text files
-u,--urls              file with list of urls to crawl
```

Using a particular operation argument will trigger the appropriate mode.

3.2.4.1. Crawling Mode

The first mode is the main crawling phase and can be activated by issuing the following command:

```
java –jar ilsp-fmc-bixo.jar –op crawl
```

The arguments used in this mode are explained in detail in the table that follows (Table 2).

Table 2 Command line input arguments when FMC runs in crawl mode.

Argument	Description
-a, --agent-name	The agent name is used to provide some identification information to the web sites that will be visited by the crawler. This is a mandatory argument. Valid values can be a name or title of an organization, an email address etc. i.e. "*-a ilsp*", "*-a nmastr@ilsp.gr*"
-c, --crawlduration	The time in minutes the crawler should run. This is an alternative terminating condition to the number of loops argument (*--numloops*) ("*-c*" or "*-n*" must be used). It should be noted that the crawler will not terminate at exactly the specified time; rather it will wait for the current crawling loop to end and will not initiate a new one. e.g. "*-c 3000*"
-cfg, --config	Optional argument for providing a custom configuration UTF-8 encoded file (in XML format) for the crawler. The default configuration file can be supplied by the crawler's config mode which will be presented later. e.g. "*-cfg /mypath/config.xml*"
-d, --domain	Optional argument for crawling only a specific web domain. This is an alternative to supplying a seed URL list (*--urls*). Valid values are the web

Argument	Description
	domain without the prefixes (cnn.com) or a file with one entry per line. e.g. "*-d cnn.com*" (will only crawl inside the cnn.com web domain) e.g. "*-d /mypath/domains.txt*" (will only crawl inside web domains mentioned in domains.txt)
-dbg, --debug	Optional boolean argument to enable debug mode. If enabled, crawler will output a lot more info while running, should only be used for debugging purposes. This argument does not take any values ("*true*" if used, "*false*" otherwise).
-f, --force	Each crawl job creates a file and folder structure within a specified folder. If this folder already exists when a crawl starts, this optional boolean argument can be used to reset the file system. This should be used with caution, its effect is irreversible. This argument does not take any values ("*true*" if used, "*false*" otherwise). Attention: If a crawl job terminated due to an error or any other unplanned way, the file structure will be incomplete and therefore resuming will be impossible. In this case, the –f argument must be used to reset the file system. Alternatively, the two last created folders in the crawl storing file system have to be manually deleted (folders are named sequentially so that it is easy to identify which ones to remove).
-h, --help	This argument has exactly the same effect as running the crawl mode with no arguments (*java -jar ilsp-fmc-bixo.jar crawl*).
-k, --keepboiler	Optional boolean argument to conserve content that was marked as boilerplate[11] in the fetched web pages. If enabled, the exported XML files will contain text marked as boilerplate. This argument does not take any values ("*true*" if used, "*false*" otherwise).
-lang, --language	Mandatory argument for specifying the target language(s) of the fetched web pages. Valid values are the ISO-639-1 codes separated by '*;*' when more than one. e.g. "*-lang el*", "*-lang en;lt;lv*"
-n, --numloops	Number of loops before terminating the crawler. A loop is defined as the process of choosing a batch of URLs from the available list, fetching, parsing, extracting links, classifying and storing in the file system. This is an alternative terminating condition to the crawl duration argument (*--crawlduration*)("*-c*" or "*-n*" must be used). Valid values are simple integer numbers. e.g. "*-n 10*", "*-n 5300*"
-o, --outputdir	Mandatory argument for specifying the exact location where all crawl-related files will be stored. If the specified folder does not exist, it will be created. If it exists, the crawler will attempt to resume crawling unless the "*-f*" argument is used in which case, the folder will be deleted and recreated. Valid values are strings specifying the desired location. e.g. "*-o /var/crawloutput/crawldb*", "*-o C:\crawloutput\crawldb*"

[11] HTML page content with little or no relevance to the web page's topic (i.e. advertisements, disclaimers, navigation links)

Argument	Description
-t, --threads	Optional argument for specifying the number of processing threads to be used for fetching web pages. This should be specified according to hardware specifications. Default value is 10 threads. e.g. "*-t 32*"
-tc, --topic	Optional argument for specifying the location and name of the topic definition file to be used by the classifier. If not supplied, the crawler will run in "*un-focused*" mode, simply downloading everything it finds. Valid values are strings specifying the topic definition file. e.g. "*-tc /var/crawlinput/topicdef_EN.txt*", "*-tc C:\crawlinput\topicdef_RO.txt*"
-u, --urls	Location and name of the seed URL file. This is an alternative to the domain argument (*-d*)("*-d*" or "*-u*" must be used). In contrast with supplying domains, URLs in the seed URL file will be used only as starting points and pose no crawling restrictions. Valid values are strings specifying the seed URL file. e.g. "*-tc /var/crawlinput/seed_HR.txt*", "*-tc C:\crawlinput\seed_EL.txt*"

A few running examples:

- Run a 10-minute crawl as ilsp, with 32 threads, preserving boilerplate information and fetch Greek web pages using the seed.txt and topic.txt as seed URL and topic definition files. Output must be stored in the crawldb folder.

```
java –jar ilsp-fmc-bixo.jar –op crawl –a ilsp –c 10 –k –lang el –t 32
–o /var/crawlruns/RenEner/EL/crawldb
–tc /var/crawlruns/RenEner/EL/topic.txt
–u /var/crawlruns/RenEner/EL/seed.txt
```

- Run a two-day crawl as ilsp, with 5 threads, fetching English web pages from the cnn.com web domain with no specific topic and saving results in crawldb.

```
java –jar ilsp-fmc-bixo.jar –op crawl –a ilsp –c 2880 –lang en –t 5
–d cnn.com
–o /var/crawlruns/cnn/EN/crawldb
```

- Run a 10-loop crawl as user@email.com, with 1 thread, fetching Latvian and Lithuanian web pages using seed.txt and topic.txt as seed and topic definitions and using a custom configuration file.

```
java –jar ilsp-fmc-bixo.jar –op crawl –a user@email.com –n 10 –lang lt;lv –
t 32
–cfg /var/crawlruns/LT-LV/myconfig.xml
–o /var/crawlruns/LT-LV/crawldb
–tc /var/crawlruns/LT-LV/topic.txt
–u /var/crawlruns/LT-LV/seed.txt
```

It should be stressed that by using the –d option without specifying a term definition file makes FMC to behave as a different tool that let us easily collect parallel data from a given multilingual parallel site or sites (provided that language selection on these sites is done by means of specific links and not through user interaction).

3.2.4.2. Export Mode

The second mode for running FMC is the export mode. This mode is used for exporting the fetched web pages and their metadata and is used by issuing:

```
java –jar ilsp-fmc-bixo.jar –op export
```

The following table details the arguments used in this mode.

Table 3 Command line input arguments when FMC runs in export mode.

Argument	Description
-h, --help	This argument has exactly the same effect as executing the jar with no arguments (*java -jar ilsp-fmc-bixo.jar*).
-o, --outputdir	Mandatory argument for specifying the location where the crawler saved files during the crawl phase. Should be the same as the value given to the "*-o*" argument when running in crawl mode. e.g. "*-o /var/crawloutput/crawldb*", "*-o C:\crawloutput\crawldb*"
-lang, --language	Mandatory argument for specifying the target language of the fetched web pages. This will be used to annotate each paragraph of each exported file as being in the target language or not. Valid values are ISO-639-1 language codes. e.g. "*-lang el*"
-len, --length	Optional argument for specifying a minimum number of tokens per text block. Any text block with fewer tokens than the specified value, will be annotated with a "*crawlinfo ooi-length tag*" in the final XML file. Valid values are integer numbers. e.g. "*-len 10*"
-tc, --topic	Optional argument for specifying the name and path of the topic definition file that was used in the crawl mode. If supplied, each text block in the final XML file will be annotated with a "*topic*" attribute.. Valid values are strings with the full path and name of the topic definition file. e.g. "*-tc /var/crawlinput/topic.txt*", "*-tc C:\crawlinput\topic.txt*"
-of,--outputfile	Optional argument for specifying the name and path of a file to write a simple file listing of the exported xml files. Valid values are strings with the full path and name of a file to write. e.g. *"-of /var/crawloutput/exports.txt",* *"-of C:\crawloutput\exports.txt"*
-te,--textexport	Optional argument, which when present, causes the collected web documents to be exported in text format (UTF-8 encoding) in addition to HTML format. If not present, only HTML docs are exported.

3.2.4.3. Crawl and Export Mode

Alternatively, the **crawlandexport** mode combines the functionality of the crawl and export modes by automatically issuing the exporter after the crawl is terminated. To use, simply issue the command:

```
java –jar ilsp-fmc-bixo.jar –op crawlandexport
```

and use the arguments as described in the crawl and export mode. In both export and crawlandexport modes, the exported file collection is stored in a subfolder named "xml" inside the path designated by the "-o, --outputdir" argument. An example of using the crawlandexport mode:

```
java –jar ilsp-fmc-bixo.jar –op crawlandexport
–a ilsp
–c 10
–k
–lang el
–t 32
–o /var/crawlruns/RenEner/EL/crawldb
–tc /var/crawlruns/RenEner/EL/topic.txt
–te
–len 10
–u /var/crawlruns/RenEner/EL/seed.txt
```

3.2.4.4. Config Mode

Finally, the crawler operates in config mode, in which a built-in utility is being invoked for supplying the user with the crawler's default configuration file. With this file, one can easily create a custom configuration which one can then use in crawl or crawlandexport modes (as an argument to the "*-cfg*" parameter). To use this utility, simply issue the following command:

```
java –jar ilsp-fmc-bixo.jar -op config -of <mypath>
```

where "<*mypath*>" is the full path and name of the file in which the default configuration parameters will be saved for the user to modify them as necessary. For instance, if we wish to get the configuration file in "*C:\temp\tempconfig.xml*", then we must execute the command:

```
java –jar ilsp-fmc-bixo.jar –op config -of C:\temp\tempconfig.xml
```

3.2.5. Input and Output Data Format

As mentioned above, the input tothe FMC consists of the topic definition file, seed URL list and optionally an XML configuration file. The topic definition is a simple text file using UTF-8 encoding containing one triplet per line. Each triplet corresponds to a term, its weight and its domain or subdomain.

```
<weight>: <term>=<domain>
```

For example, in the Renewable Energy domain, an excerpt of the topic definition file is shown below:

```
100: renewable energy=RenewableEN
100: natural resources=RenewableEN
100: natural processes=RenewableEN
100: biogas=RenewableEN
100: renewable power generators=RenewableEN
...
```

It is important to note that each term must contain only alphanumeric characters for the parser to be able to handle it correctly. The seed URL file is again a simple text file UTF-8 encoded containing one URL per line. A sample seed URL file for the Renewable Energy domain is the following:

```
http://alternative-energy-resources.net/
http://ebrdrenewables.com/sites/renew/biofuels.aspx
http://en.wikipedia.org/wiki/Portal:Renewable_energy/Selected_article
http://en.wikipedia.org/wiki/Renewable_energy
http://green.wikia.com/wiki/Renewable_Energy
```

There is no limit to the number of terms or URLs that these files can contain, however, a very large number of terms in the topic definition file will have an impact on the classifier's performance.

The configuration file must be based on the default XML provided by FMC's config mode. It provides mainly technical configuration options for the crawling mode of FMC and specifically issues that have to do with crawling politeness (i.e. time intervals between revisiting a web page from the same web domain, number of simultaneous threads running on a specific web domain), buffer sizes, and timeouts as well as some options for the text classification. The default XML used by FMC is illustrated below:

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
	<agent>
		<email>yourmail@mail.com</email>
		<web_address>www.youraddress.com</web_address>
	</agent>
	<classifier>
		<min_content_terms>
			<value>5</value>
			<description>Minimum number of terms that must exist in clean content of each web page in order to be stored.</description>
		</min_content_terms>
		<min_unique_content_terms>
			<value>3</value>
<description>Minimum unique terms that must exist in clean content</description>
		</min_unique_content_terms>
		<max_depth>
			<value>2</value>
<description>Maximum depth to crawl before abandoning a specific path. Depth is increased every time a link is extracted from a non-relevant web page.</description>
		</max_depth>
	</classifier>
	<fetcher>
		<fetch_buffer_size>
			<description>Maximum number of urls to fetch per run</description>
			<value>256</value>
		</fetch_buffer_size>
		<socket_timeout>
			<value>10000</value>
			<description>Socket timeout in milliseconds(per URL)</description>
		</socket_timeout>
		<connection_timeout>
			<value>10000</value>
			<description>Connection timeout in milliseconds(per URL)</description>
		</connection_timeout>
		<max_retry_count>
			<value>2</value>
```

```
<description>Maximum number of attempts to fetch a Web page before giving up</description>
            </max_retry_count>
            <min_response_rate>
                  <value>0</value>
<description>Minimum bytes-per-seconds for fetching a web page</description>
            </min_response_rate>
            <valid_mime_types>
                  <mime_type value="text/html" />
                  <mime_type value="text/plain" />
                  <mime_type value="application/xhtml+xml" />
                  <description>Accepted mime types</description>
            </valid_mime_types>
            <crawl_delay>
                  <value>1500</value>
                  <description>Delay in milliseconds between requests</description>
            </crawl_delay>
            <max_content_size>
                  <value>531072</value>
<description>Maximum content size (bytes) for downloading a web page</description>
            </max_content_size>
            <max_requests_per_run>
                  <value>50</value>
<description>Maximum fetch set size per run (Sets are made by URLs from the same host)</description>
            </max_requests_per_run>
            <max_requests_per_host_per_run>
                  <value>50</value>
                  <description>Maximum URLs from a specific host per run</description>
            </max_requests_per_host_per_run>
            <max_connections_per_host>
                  <value>32</value>
                  <description>Maximum number of fetching threads for each host</description>
            </max_connections_per_host>
            <max_fetched_per_host>
                  <value>1000</value>
                  <description>Maximum web pages to fetch per host</description>
            </max_fetched_per_host>
            <max_redirects>
                  <value>5</value>
                  <descriptions>Maximum number of redirects</descriptions>
            </max_redirects>
            <request_timeout>
                  <value>600000</value>
```

```
<description>Maximum time to wait for Fetcher to get all URLs in a run</description>
            </request_timeout>
      </fetcher>
</configuration>
```

The FMC output consists of the collected web documents in both HTML and text format (UTF-8 encoding) as well as their metadata. The metadata are stored in XML files using a cesDOC format that can be validated against the available XCES[12] standard schemas. A sample XML file is provided below:

```
<?xml version='1.0' encoding='UTF-8'?>
<cesDoc        version="0.4"        xmlns="http://www.xces.org/schema/2003" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <cesHeader version="0.4">
            <fileDesc>
                  <titleStmt>
                        <title>Japan quake among worst ever recorded - Chicago Breaking News</title>
                        <respStmt>
                              <resp>
                                    <type>Crawling and normalization</type>
                                    <name>ILSP</name>
                              </resp>
                        </respStmt>
                  </titleStmt>
                  <publicationStmt>
                        <distributor>ACCURAT project</distributor>
                        <eAddress            type="web">http://www.accurat-project.eu/</eAddress>
                        <availability>Under review</availability>
                        <pubDate>2012</pubDate>
                  </publicationStmt>
                  <sourceDesc>
                        <biblStruct>
                              <monogr>
                                    <title>Japan quake among worst ever recorded - Chicago Breaking News</title>
                                    <author></author>
                                    <imprint>
                                          <format>text/html</format>
                                          <publisher></publisher>
                                          <pubDate></pubDate>

      <eAddress>http://articles.chicagobreakingnews.com/2011-03-11/news/28682007_1_japan-quake-tsunami-indonesia</eAddress>
                                    </imprint>
                              </monogr>
```

[12] http://www.xces.org/

```
                        </biblStruct>
                  </sourceDesc>
            </fileDesc>
            <profileDesc>
                  <langUsage>
                        <language iso639="en"/>
                  </langUsage>
                  <textClass>
                        <keywords>
                              <keyTerm>news</keyTerm>
                        </keywords>
                        <domain></domain>
                        <subdomain>DisastersEN</subdomain>
                        <subject/>
                  </textClass>
                  <annotations>
                        <annotation>/D:/PROGRAMS/eclipse/workspace/ilsp-
fmc-bixo/crawlresults/test/xml/1.html</annotation>
                  </annotations>
            </profileDesc>
      </cesHeader>
      <text>
            <body>
                  <p id="p1" crawlinfo="boilerplate">You Are Here: Home >
Collections</p>
                  <p id="p2" crawlinfo="ooi-length" type="title">Japan
quake among worst ever recorded</p>
                  <p id="p3" crawlinfo="ooi-length">By Caroline Kyungae
Smith | Tribune reporter</p>
                  <p id="p4" crawlinfo="ooi-length">March 11, 2011</p>
                  <p id="p5" topic=" quake; earthquake; japan;
magnitude">The quake that hit Japan was a magnitude 8.9, the biggest
earthquake to hit the country since officials began keeping records in the
late 1800s, and one of the biggest ever recorded in the world, according to
the U.S. Geological Survey.</p>
                  <p id="p6" topic=" depth; quake; Tokyo">The quake struck
at a depth of six miles, about 80 miles off the eastern coast, the agency
said. The area is 240 miles northeast of Tokyo.</p>
                  <p id="p7" topic=" tsunami; japan; tsunami">A tsunami
warning was extended to a number of Pacific, Southeast Asian and Latin
American nations, including Japan, Russia, Indonesia, New Zealand and
Chile. In the Philippines, authorities ordered an evacuation of coastal
communities.</p>
                  <p id="p8" topic=" quake; earthquake">“Japan’s earthquake
will be considered a great quake,” said Dale Grant, a geophysicist with the
Geological Survey in Golden, Colo.</p>
                  <p id="p9" topic=" quake">Damage from such a quake can
span hundreds to thousands of miles.</p>
                  <p id="p10" crawlinfo="ooi-lang">A few days earlier,
Japan was hit with a 7.2 earthquake. “A 7.2 quake has 80 or 90 times less
energy than an 8.9 quake,” Grant said.</p>
```

```
                    <p     id="p11"     topic="     aftershocks;     earthquake;
aftershocks">As of 3 a.m. Chicago time, there were at least 12 aftershocks
following the earthquake, with the greatest aftershocks measuring 7.1 and
6.8, Grant said.</p>
                    <p  id="p12"  crawlinfo="ooi-length">“This  is  what  we’d
expect from an 8.9 earthquake.”</p>
                    <p id="p13" topic=" tsunami; quake; tsunami">The greater
concern  is  the  tsunami  triggered  by  the  quake,  he  said.  “Tsunamis  can
travel up to 450 miles per hour,” he said.</p>
                    <p id="p14">“Warnings have been issued for the Hawaiian
Islands,” he said. “We’ll probably see an impact.”</p>
                    <p  id="p15"  topic="  earthquake">The  biggest  earthquakes
in  recent  history  occurred  last  year  in  Chile  at  8.8  and  in  2004  in
Indonesia at 9.1, Grant said.</p>
                    <p                                                  id="p16"
crawlinfo="boilerplate">cxsmith@tribune.com</p>
                    <p id="p17" crawlinfo="boilerplate">Advertisement</p>
                    <p id="p18" crawlinfo="boilerplate">RELATED ARTICLES</p>
                    <p             id="p19"             crawlinfo="boilerplate"
type="listitem">Chicagoans struggle for news from Japan</p>
                    <p id="p20" crawlinfo="boilerplate" type="listitem">March
11, 2011</p>
                    <p             id="p21"             crawlinfo="boilerplate"
type="listitem">Japanese immigrants pray for quake victims</p>
                    <p id="p22" crawlinfo="boilerplate" type="listitem">March
13, 2011</p>
                    <p             id="p23"             crawlinfo="boilerplate"
type="listitem">Vacation turns to evacuation in Hawaii</p>
                    <p id="p24" crawlinfo="boilerplate" type="listitem">March
11, 2011</p>
                    <p id="p25" crawlinfo="boilerplate">Terms of Service</p>
                    <p id="p26" crawlinfo="boilerplate">Privacy Policy</p>
                    <p id="p27" crawlinfo="boilerplate">Index by Date</p>
                    <p id="p28" crawlinfo="boilerplate">Index by Keyword</p>
                    <p  id="p29"  crawlinfo="boilerplate">A  Tribune  Newspaper
website</p>
            </body>
      </text>
</cesDoc>
```

Documentation for the elements in the XCES standard can be obtained through the XCES web site. A few notes on the additional elements used by FMC:

- The ***profileDesc*** group is used for providing information regarding the language of the text found in this web page in ISO-639-1 format, the keywords of the web page, the domain or subdomain it belongs to and the location of the file.
- The ***fileDesc*** group is used for storing the title of the web page, its original format and its URL.
- Paragraph elements (<p>) may contain one or more of the following attributes:
  - ***crawlinfo*** with possible values: “*boilerplate*”, meaning that the paragraph has been considered boilerplate; “*ooi-length*”, denoting that this paragraph is so short that either it is not useful, or it can confuse the language identifier; and “*ooi-lang*”, denoting that the paragraph is not in the targeted language.
  - ***type*** with possible values: “*title*”, “*heading*” and “*listitem*”.

- ***topic*** with a string value storing all terms from the topic definition detected in this paragraph

3.2.6. FMC Complementary tools

This subsection briefly documents two additional tools that have been found useful in the FMC-based process of building bilingual domain-specific comparable corpora from the web. Namely, these tools are (i) an FMC system packed for bilingual focused crawling and (ii) an FMC output filter that helps cleaning up the bilingual narrow domain corpora, collected by FMC, from document pairs that appear to be very weakly comparable.

3.2.6.1. WFMC – an FMC system packed for bilingual focused crawling

The WFMC tool is in fact a wrapper of FMC that can be used for collecting bilingual web documents from given bi(multi)lingual web sites. It configures FMC to crawl only within user-specified web domain(s) and launches two FMC crawls, one for each language of a user-defined language pair. It returns two separate sets of documents, one set per language. The comparability degree of the returned corpora strongly depends on the comparability/parallelism of the the given web domain(s).

Both WFMC and FMC tools have common software dependencies, system requirements, installation instructions, as well as input and output data format. WFMC execution instructions are simpler. The syntax for running WFMC is the following:

```
java -jar ilsp-fmc-bilingual.jar -a agentname -c crawlduration -d domain -
tc1 topic1 -tc2 topic2 -o outputdir -t threads -lang1 language1 -lang2
language2 -te –k
```

where WFMC command arguments are described in the following table (Table 4).

Table 4 Command line input arguments for running WFMC tool.

Argument	Description
-a, --agent-name	The agent name is used to provide some identification information to the web sites that will be visited by the crawler. This is a mandatory argument. Valid values can be a name or title of an organization, an email address etc. i.e. "*-a ilsp*", "*-a nmastr@ilsp.gr*"
-c, --crawlduration	The time in minutes the crawler should run for each language. e.g. "*-c 2880*"
-d, --domain	Full path of a txt file containing bilingual web domain(s) (one domain per textline), such as *www.cnn.com*
-tc1, --topic1	Full path of topic definition file for the first language (more info in subsection 3.2.5 above). e.g. "*-tc1 /var/crawlinput/topicdef_EN.txt*"
-tc2, --topic2	Full path of topic definition file for the second language (more info in subsection 3.2.5 above). e.g. "*-tc2 /var/crawlinput/topicdef_EL.txt*"
-o, --outputdir	Full path of the folder to write results (further down to this folder two separate subfolders will be created, one for each language). Valid values are strings specifying the desired location. e.g. "*-o /var/crawloutput/crawldb*", "*-o C:\crawloutput\crawldb*"
-t, --threads	Optional argument for specifying the number of processing threads to be used for fetching web pages. This should be specified according to hardware specifications. Default value is 10 threads.

Argument	Description
	e.g. "*-t 32*"
-lang1, --language1	Language identifier (mandatory argument) for specifying the first target language of the web pages to be fetched. Valid values are the ISO-639-1 codes. e.g. "*-lang lv*"
-lang2, --language2	Language identifier (mandatory argument) for specifying the second target language of the web pages to be fetched. Valid values are the ISO-639-1 codes. e.g. "*-lang ro*"
-te,--textexport	Optional boolean argument that does not take any values ("true" if present, "false" otherwise). When argument is present, causes the collected web documents to be exported in text format (UTF-8 encoding) in addition to HTML format. If not present, only HTML docs are exported.
-k, --keepboiler	Optional boolean argument to save boilerplate information (found in the fetched web pages) to the XML metadata output files. If enabled, the exported XML files will contain text marked as boilerplate. This argument does not take any values ("*true*" if used, "*false*" otherwise).

A sample usage of WFMC follows:

```
java -jar ilsp-fmc-bilingual.jar -a ilsp
-c 1
-d data/domains.txt
-tc1 data/topic1.txt
-tc2 data/topic2.txt
-o crawlresults/lv-el_bilingualcrawling
-t 4
-lang1 lv
-lang2 el
-te
-k
```

The above sample command calls FMC twice for bilingual crawling as follows:

- use ilsp as agent name
- crawl 1 min for Latvian docs and 1 min for Greek docs
- confine crawl in web domains contained in data/domains.txt file
- get topic specific terms for Latvian language from data/topic1.txt
- get topic specific terms for Greek language from data/topic2.txt
- output collected docs to crawlresults/lv-el_bilingualcrawling folder (lv docs will be saved to crawlresults/lv-el_bilingualcrawling/lv and el docs to crawlresults/lv-el_bilingualcrawling/el)
- use 4 threads for each crawl
- use Latvian as first language
- use Greek as second language
- return collected docs in txt format (UTF-8 encoding) in addition to HTML format

- preserve boilerplate information found in the fetched web pages by saving it to the XML metadata output files.

3.2.6.2. PreDAF – an FMC output filter

In cases where the domain specific comparable corpora harvested from the web are large enough, there is a special need for introducing a fast and computationally cheap way to filter out (preferably) all least comparable bilingual document pairs. In this manner the comparability degree of the corpora collected by FMC could be increased, the search scope of the corpora comparability evaluation tools (WP1) and of the multi-level alignment tools (WP2) would be reduced, hence significant (corpora) processing time savings would be achieved.

For this purpose, the Pre-Document-Alignment Filter (PreDAF) has been developed in the form of a post-crawling step. The filter processes bilingual comparable corpora (actually their metadata) collected (and generated) by FMC or WFMC and runs independently of those two tools. The output of the filter is a list of document pairs that are likely to be comparable based on term-overlap criteria. The filter computes a simple comparability score for every document pair and returns only those document pairs that have received a score higher than a given threshold. The comparability score of a bilingual document pair currently depends on the number of common terms the two documents share. This comparability evaluation approach is lightweight, fast and insensitive to the direction of language pair (same results for both e.g. en-el and el-en language pairs), but it does not go through other important corpora comparability aspects. Therefore, it should rather be used mostly when either the source or the target language corpus returned by the focused crawlers is large enough (e.g. includes around a hundred of thousands of web documents or more).

For enabling PreDAF's comparability score computation the user has to provide parallel term lists. That is, topic definition files that have been used for crawling web documents for given source and target languages must be made "parallel" or "aligned" before being fed to the pre-alignment filter. "Alignment" of the topic definition files is done by inserting an ID code after every text line of those files. Terms sharing the same ID code are considered parallel. As described in subsection 3.2.5 above, the format of every text line of a topic definition file is as follows:

```
<Term Score>: <Word or Multi Word Expression>=<Subtopic>
```

The previous text line format should be changed to[13]:

```
<Term Score>: <Word or Multi Word Expression>=<Subtopic> tab <ID>
```

For example, two topic definition files that have been used for collecting English and Greek web documents on renewable energy topic after they have been "aligned" for being used by the pre-alignment filter should look like this:

Excerpt of the aligned English topic definition file

```
100: renewable energy=RE      1
100: natural resources=RE     2
100: natural processes=RE     3
100: biogas=RE    4
100: renewable power generators=RE  5
100: wind turbines=RE   6
100: wind speed=RE      7
....
```

Excerpt of the aligned Greek topic definition file

```
100: ανανεώσιμες πηγές ενέργειας=RE 1
100: φυσικές πηγές=RE   2
```

[13] Note the tab character between <Subtopic> and <ID> fields.

```
100: φυσικές διεργασίες=RE    3
100: βιοαέριο=RE  4
100: γεννήτριες ηλεκτροπαραγωγής=RE 5
100: ανεμογεννήτρια=RE  6
100: ταχύτητα ανέμου=RE 7
.....
```

Again, lines of the two "aligned" topic definition files having the same ID indicate that the underlying EN and EL terms are parallel, e.g. "biogas" is parallel to "βιοαέριο", because they share the same ID (= 4).

PreDAF tool is a standalone Java executable, which can be executed by any system that runs a Java Runtime Environment 1.6 or above.There are no specific hardware requirements beyond those needed for running the JRE. No installation is needed. The tool must run after FMC or WFMC has successfully exited a given extract mode, since it relies on data files that have been created by those two crawlers. Additional information on other input and output data format is given by the Table 5 that follows.

The syntax for running PreDAF is the following:

```
java -jar ilsp-fmc-prealigner.jar -i1 indir1 -i2 indir2 -l logfile -t1
topic1 -t2 topic2 -o outfile -t threshold
```

PreDAF command arguments are described in the following table (Table 5).

Table 5 Command line input arguments for running PreDAF tool.

Argument	Description
-i1, --indir1	Full path of folder containing collected web documents for first language, *e.g. -i1 /home/ACCURAT/FMC2/CrawledData/WE/EN/xml/*
-i2, --indir2	Full path of folder containing collected web documents for second language, *e.g. -i1 /home/ACCURAT/FMC2/CrawledData/WE/EL/xml/*
-l, --logfile	Name of the file to store logs.
-t1, --topic1	Full path of topic definition file for the first language; the topic definition file must have been filled in with IDs (see above for explanations).
-t2, --topic2	Full path of topic definition file for the second language; the topic definition file must have been filled in with IDs (see above for explanations).
-o, --outfile	Full path or name of the file to write results. The file lists pairs of bilingual documents that have comparability score above threshold. Every text line has the following format (L1; 1st language, L2: 2nd language): */L1_folder/L1_filename TAB /L2_folder/L2_filename TAB Comparability Score* A sample output file should look like this: */en/en_1.txt TAB /el/el_1.txt TAB 8* */en/en_2.txt TAB /el/el_3.txt TAB 6* */en/en_2.txt TAB /el/el_9.txt TAB 5*

Argument	Description
-t, --threshold	An integer value specifying the documents comparability threshold. This currently is indicated by the minimum number of common terms two documents must share in order to be considered comparable and enlisted to the output file (document pairs list). Typical values are 3 to 5.

Note that the –h switch alone displays usage instructions on screen.

3.2.7. Contact

For further information and technical support installing and/or running this tool, please email to Nikos Glaros: nglaros@ilsp.gr.

3.3. Wikipedia Retrieval Tool

3.3.1. Overview and Purpose

Wikipedia contains a large number of documents from various topics and languages. When these articles describe the same topic, they are connected to each other by Wikipedia *interlanguage links*, enabling us to extract a corpus which is already aligned in document level. Even though these documents contain the same topic, their ranges of comparability varies widely: some articles might be translation of each other; however, some of them might be written independently and do not contain any shared information. Inclusion of these documents in the corpora might reduce the MT performance. Therefore, a retrieval tool is needed to identify and gather the comparable documents only.

Different to a crawling tool, this retrieval tool makes use of available Wikipedia dump data (available for download in http://dumps.wikimedia.org) which contains extensive information of Wikipedia documents, including interlanguage links between bilingual articles. A subset of Wikipedia dump data which was downloaded in March 2010 has been included in this tool for testing reasons. Should different data be needed for further evaluation, the process of downloading them is described in section 3.3.4.

This tool aims to identify and retrieve comparable documents by specifically looking for pairs which contain similar sentences (sentences with information overlap, such as links – also referred to as anchor texts – and words). Language independent feature is used in this method, therefore this tool can be applied to retrieve Wikipedia documents for any language pairs. The full description of this retrieval methods is described in D3.4.

This tool will produce corpora containing comparable documents and their alignments at document level.

3.3.2. Software Dependencies and System Requirements

This tool is developed in Perl and can be run in Windows and Linux platforms. This tool required the following:

1. Perl v5.10 or above; and
2. 1+ GB RAM.

3.3.3. Installation

To start using this tool, simply copy and extract "*WikipediaRetrieval.rar*" and read the "*readme.txt*" to get further information. The description of execution instructions is described in the next section.

3.3.4. Execution Instructions

As described in the overview, this tool retrieves comparable documents from Wikipedia by analysing comparable segments in the documents. Therefore, this tool needs Wikipedia documents of the source and target languages along with the alignment file as initial corpus. To extract these documents from Wikipedia, users need to follow these steps:

1. First, Wikipedia regularly releases database dump containing article contents of each available language, which is made available in http://dumps.wikimedia.org/[lang]wiki, e.g. http://dumps.wikimedia.org/enwiki/ for English Wikipedia. Language should be specified by using ISO-639-1 language codes. At the time when this document is written, these files are named as "*[lang]wiki-[date]-pages-articles.xml.bz2*", e.g. "*enwiki-20110901-pages-articles.xml.bz2*" for English documents. This XML file contains all contents of Wikipedia documents in that particular language. Users will need to download two files: one of the source language and one of the target language, later referred to as "*[pageOfSourceLang]*" and "*[pageOfTargetLang]*".
2. After these files are downloaded and extracted, users need to run this command to save each document in each language into a separate file[14] to enable further processing, and extract alignment information between documents in both languages:

```
perl WikipediaExtractor.pl --source [sourceLang] --target [targetLang] --sourcePage [pageOfSourceLang] --targetPage [pageOfTargetLang] --output [outputFolder]
```

   This script requires several parameters:
   a. ***[sourceLang]*** and ***[targetLang]*** represent any ACCURAT languages which are represented in ISO-639-1 language codes, such as "*lv*" (Latvian), "*lt*" (Lithuanian), "*en*" (English), etc.
   b. ***[pageOfSourceLang]*** and ***[pageOfTargetLang]*** represent the Wikipedia database dumps decribed previously.
   c. ***[outputPath]*** represents an output folder in which the extracted documents and the alignment will be stored.

   Please note that this process in general is very time consuming especially for English and German as they contain a large number of documents.
3. This process will have the initial corpus and alignment file as outputs and store them in the specified "*[outputFolder]*". For example, when using "*C:/Corpora/*" as an output folder when extract LV-EN documents, this initial corpus will be stored in "*C:/Corpora/lv/*" for the LV documents and "*C:/Corpora/en/*" for the EN documents; alignment file will be stored in "*C:/Corpora/lv_en.txt/*".

After these files are made available, users can start the retrieval process by running this command:

```
perl WikipediaRetrieval.pl --source [sourceLang] --target [targetLang] --alignment [alignmentFile] -sourceFolder [folderPathForSourceDocs] --targetFolder [folderPathForTargetDocs] --output [outputFolder]
```

This script requires several parameters:

1. ***[sourceLang]*** and ***[targetLang]*** represent the source and target language of documents which are represented in the ISO-639-1 language codes, such as "*lv*" (Latvian), "*lt*" (Lithuanian), "*en*" (English), etc.
2. ***[alignmentFile]*** represents the file containing the alignment between the documents in both languages.
3. ***[folderPathForSourceDocs]*** and ***[folderPathForTargetDocs]*** represent the absolute path of the folders containing the Wikipedia documents for both languages.
4. ***[outputFolder]*** represents the path of the folder in which the corpora will be stored.

[14] We disregard Wikipedia documents which do not contain main topics, such as User pages, Discussion pages, Category pages, or Redirection pages.

This retrieval method contains five main processes as shown in Figure 4.
The description of each process is described below:

1. ExtractBilingualLexicon.pl:
   This script builds bilingual lexicons by extracting document titles which are connected by *interlanguage links* by Wikipedia. Therefore, this retrieval tool does not need any linguistic resources to perform translation.
2. FilterWikipedia.pl:
   This script filters out unnecessary information in Wikipedia documents, such as footnotes, table formatting, images, etc.
3. SentenceSplitter.pl:
   This script aims to split documents into sentences, enabling further process to find information overlap in sentence level.
4. ReplaceAnchorsUsingBilingualLexicon.pl:
   This script replaces all links (anchor texts) in the source documents with its corresponding text in the target language if they are available in the bilingual lexicon
5. ComparabilityMeasurerOnSentenceLevel.pl:
   Last, this script measures the comparability of the documents on the sentence level on all the available document pairs in the specified language pair by measuring word overlap.

A more detailed explanation of this process is described in D3.4.

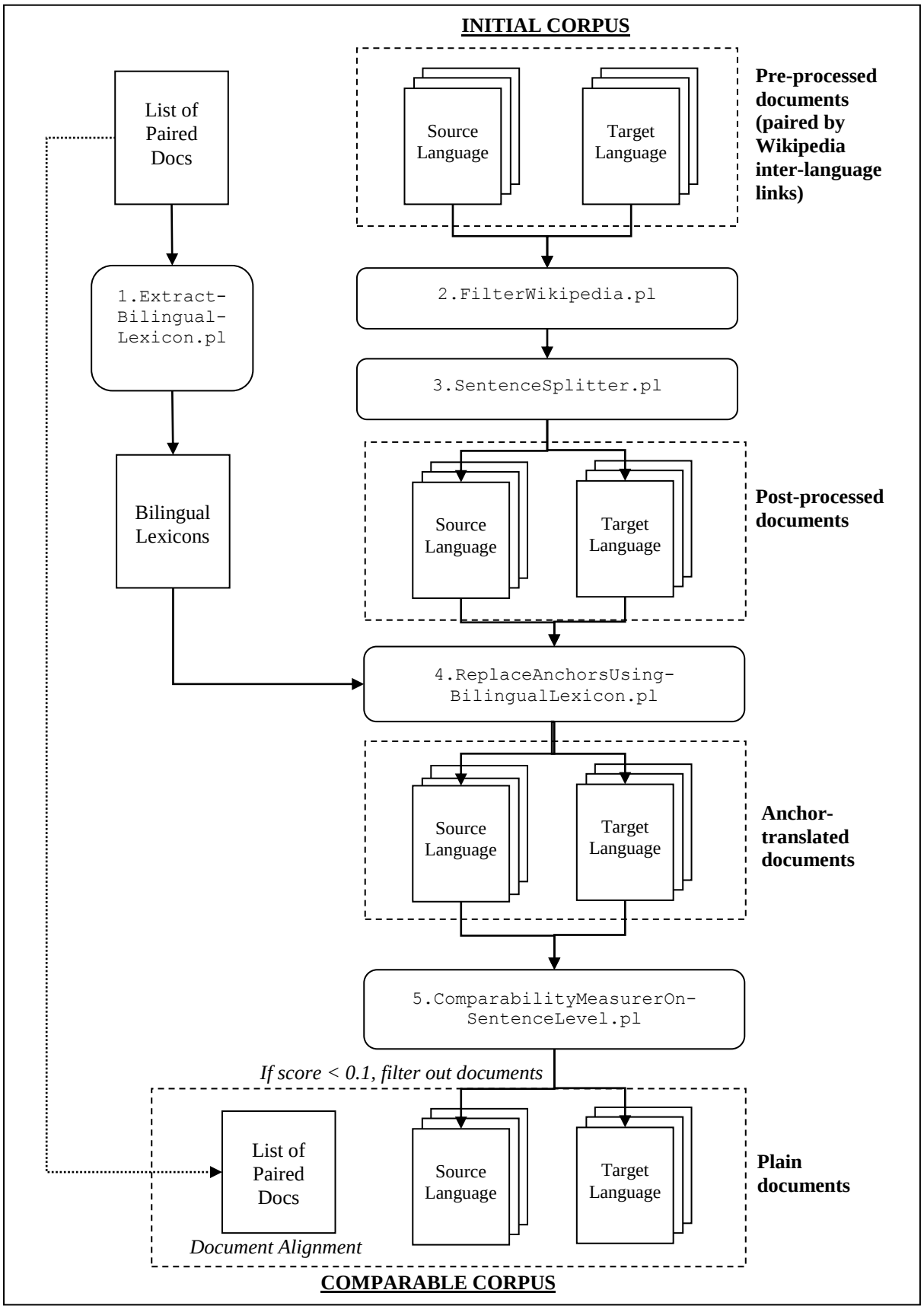

Figure 4. Data Flow Diagram of Wikipedia Retrieval

3.3.5. Input and Output Data Format

This section will describe the format of input and output data for this tool. To run the "*Wikipedia Retrieval*" tool, users need to have the following files as input:

1. Alignment file for each language pair, which shows the following information: *[sourceDoc ID]*, *[sourceDoc Title]*, *[targetDoc ID]* and *[targetDoc Title]*. An example of this file is shown in Figure 5. This file can be extracted from Wikipedia by using the process described in the previous section.

```
1151884     Informationsökonomie    10    Information_economics
2398435     Strahlenphysik    13664288    Radiophysics
837854      Jugoslawiendeutsche     6026882     Germans_of_Yugoslavia
.....
```

Figure 5. Example Alignment File of DE-EN language pair

2. Wikipedia articles in the Wiki markup[15] for each language pair. As shown in Figure 6, the file contains not only the text, but also all the links contained in the article, shown in the *[[anchor]]* text. These files are also the results from WikipediaExtractor tool described in the previous section.

```
'''Europe''' () is, by convention, one of the world's seven [[continent]]s.
Comprising the westernmost [[peninsula]] of [[Eurasia]], Europe is
generally divided from [[Asia]] to its east by the [[water divide]] of the
[[Ural Mountains]], the [[Ural (river)|Ural River]], the [[Caspian Sea]],
the [[Caucasus Mountains]] (or the [[Kuma-Manych Depression]]), and the
[[Black Sea]] to the southeast.  Europe is bordered by the [[Arctic Ocean]]
and other bodies of water to the north, the [[Atlantic Ocean]] to the west,
the [[Mediterranean Sea]] to the south, and the [[Black Sea]] and connected
waterways to the southeast. Yet the borders for Europe—a concept dating
back to [[classical antiquity]]—are somewhat arbitrary, as the term
''continent'' can refer to a [[human geography|cultural and political]]
distinction or a [[physical geography|physiographic]] one.
...
```

Figure 6. Example of English Wiki Article

This retrieval tool will produce several outputs during the process:

1. *"[sourceLang]_[targetLang]_bilingualLexicon.txt"*: this file contains bilingual lexicon previously extracted from document titles in the alignment file.

```
16. septembris    September 16
Lejassaksija      Lower Saxony
Bluā grāfi  Counts of Blois
Mursija     Murcia
Gņevkovo    Gniewkowo
Jamalas Ņencu autonomais apvidus    Yamalo-Nenets Autonomous Okrug
Aiči prefektūra   Aichi Prefecture
...
```

Figure 7. Example of Latvian-English Bilingual Lexicon

2. Filtered documents: two folders named "*[sourceLang]-filtered*" and "*[targetLang]-filtered*" contain Wikipedia documents which have been filtered. In this phase, all unnecessary

[15] http://en.wikipedia.org/wiki/Help:Wiki_markup

formatting and unused sections are filtered out. Examples of files for each filtering phase are explained in more detail in D3.4.

3. Line separated documents: two folders named "*[sourceLang]-filtered-lineSeparated*" and "*[targetLang]-filtered-lineSeparated*" contain Wikipedia documents which have been split into sentences.
4. Anchors replaced documents: two folders named "*[sourceLang]-filtered-lineSeparated-anchors-replaced*" and "*[targetLang]-filtered-lineSeparated-anchors-replaced*" contain Wikipedia documents which have had all the links (anchors) translated.
5. Comparability scores, named "*[sourceLang]-[targetLang]-scores.txt*" contains the comparability scores for all the document pairs. An example output is shown in Figure 8. It contains information regarding numbers of sentences in each document, numbers of valid sentences (based on filtering described in D3.4) and numbers of paired sentences. Three different comparability scores are shown in the document:

   5.1. maximal comparability score of a sentence pair in that document
   5.2. average score of the paired sentences
   5.3. average score of the document.

   The first line of the comparability score document contains tab-separated column titles. In order to present a sample document within the documentation, the heading line contains an abbreviated form of titles, where:

   - "*Src_ID*" represents "*Source_ID*"
   - "*Trg_ID*" represents "*Target_ID*"
   - "*Src_Sn*" represents "*Source_NoOfSentence*"
   - "*Trg_Sn*" represents "*Target_NoOfSentence*"
   - "*VSrcSn*" represents "*ValidSourceSentence*"
   - "*VTrgSn*" represents "*ValidTargetSentence*"
   - "*PrdSn*" represents "*NoOfPairedSentence*"
   - "*MaxParSn*" represents "*MaxScoreOfParSentences*"
   - "*AvgAllPrdSn*" represents "*AvgScoreForAllPairedSentences*"
   - "*AvgDc*" represents "*AvgScoreForDocs*"

Src_ID	Trg_ID	Src_Sn	Trg_Sn	VSrcSn	VTrgSn	PrdSn	MaxParSn	AvgAllPrdSn	AvgDc
178527	127731	40	15	20	11	1	0.11	0.11	0.01
14363	127002	69	9	30	8	2	0.18	0.14	0.036
7538	70521	23	24	12	18	0	0	0	0
23302	27399	30	41	20	30	0	0	0	0
65537	63067	8	8	2	2	1	0.36	0.36	0.18
63162	165479	63	6	50	3	3	0.28	0.18	0.18
...									

Figure 8. Comparability Scores Output

Please note that all five previous outputs are produced by processes involved in the tool as shown in the data flow diagram. They are needed to identify and retrieve comparable documents, however, once the corpora are built, all these documents will not be needed for further processing and may be deleted from the system unless are needed for other purposes.

6. The main output from this tool is the comparable corpus, which is stored in the output path previously specified by users. This path will include a subfolder named "*[sourceLang]-[targetLang]*", e.g. "*lv-en*" for Latvian-English corpora. This folder will contain the following files:

   6.1. Alignment file, listing the absolute path of source document and target document in the corpus, as shown in Figure 9.

```
C:\Corpora\lv\52836_lv.txt          C:\Corpora\en\697540_en.txt
C:\Corpora\lv\19028_lv.txt          C:\Corpora\en\59727_en.txt
C:\Corpora\lv\45155_lv.txt          C:\Corpora\en\4410977_en.txt
...  ...
```

Figure 9. Example of LV-EN Alignment File

6.2. Plain text of comparable Wikipedia documents. These files will be included in two folders named "*[sourceLang]*" and "*[targetLang]*", e.g. for LV-EN, these folders will be named "*lv*" and "*en*".

3.3.6. Contact

For further information and technical support installing and/or running this tool, please contact Monica Paramita: m.paramita@shef.ac.uk.

3.4. News Information Downloader using RSS feeds

3.4.1. Overview and Purpose

Reports in different languages about the same event can be regarded as comparable because, while not direct translations of each other, they are likely to say some of the same things and hence are likely to share some textual units, which are translations of each other. The aim of this tool is to collect news articles from the web. To do this it uses monolingual RSS feeds, which are XML structured documents. It parses each of the XML documents and records the news published within each RSS feed document.

Please note that the output of this tool as well as the output of the tools described in sections 3.5 and 3.6 are collected in pools to produce the comparable corpora. More precisely, for each language a pool is generated that contains all the output of all these tools for that particular language. The entries in each pool are compared to the others to perform alignment between the news articles. For instance, we compare the articles from the English pool with the ones in the German pool to produce a English-German comparable corpora. The alignment is performed using the News Article Alignment and Content Downloading Tool described in section 3.7.

3.4.2. Software Dependencies and System Requirements

The tool requires Java 1.5 or above and 1 GB RAM. It also requires an Internet Connection.

3.4.3. Installation

The tool does not require any specific installation. It can be run using the feedDownloader.jar file.

3.4.4. Execution Instructions

To run the tool simply use this command (works both in Unix and Windows):

```
java -jar feedDownloader.jar pathToInputFile pathToSaveOutput languageCode
```

The command requires the following parameters:

- ***pathToInputFile*** is the full path path to a file that contains all the RSS feed URLs.
- ***pathToSaveOutput*** is the full path to a file where the tool needs to write the outputs (e.g. "*C:\\feedDownloader\\results.txt*")
- ***languageCode*** is the language code in ISO-639-1 standard (e.g. "*de*" for German)

3.4.5. Input and Output Data Format

The input to the tool is a file that contains in each entry a RSS feed URL as shown below:

```
http://www.kasjauns.lv/objs/xml/news.xml?news_lang_id=1
http://feeds.tvnet.lv/tvnet/izklaide/latest?fmt=xml
```

```
http://feeds.feedburner.com/BootlvPortativieDatori?fmt=xml
```

The output is another file (encoded in UTF-8) that contains on each line information about a news event (each field is tab separated):

```
url
language Code
title
date of publication
```

An example output entry is:

```
http://feedproxy.google.com/~r/BootlvPortativieDatori/~3/3jwRpPoC0NE/    lv
      Motorola izlaidusi Lapdock 100 viedtālruņu "dokstaciju"     Wed,    12
Oct 2011 08:07:18 +0000
```

3.4.6. Contact

For further information and technical support for installing and/or running this tool, please contact Ahmet Aker: a.aker@dcs.shef.ac.uk.

3.5. News Information Downloader using Google News Search

3.5.1. Overview and Purpose

Similar to the *News Information Downloader using RSS feeds* tool this tool downloads monolingual news articles from the Web. It uses Google News search[16] to obtain the news articles. The details how it searches in the Google News is described in Deliverable 3.4.

3.5.2. Software Dependencies and System Requirements

The tool requires Java 1.5 or above and 1 GB RAM. It also requires an Internet Connection.

3.5.3. Installation

The tool does not require any specific installation. It can be run using the googleNewsDownloader.jar file.

3.5.4. Execution Instructions

To run the tool simply use this command (works both in Unix and Windows):

```
java –jar googleNewsDownloader.jar pathConfigFile pathToSaveOutput
```

The command requires the following parameters:

- ***pathToConfigFile*** is the path to a file that contains all the requried information:
  - language code (e.g. "*de*"), in ISO-639-1 standard
  - Google News Search registration key
  - User's web site address

  Each field must be supplied on a separate line in the order shown above. You can get the Google News Search registration id from *http://code.google.com/apis/loader/signup.html*. The web site address is the one the user enters when registering for the Google News Search Key. Please note that the user must create this configuration file himself. It is not provided with this tool.

  Example of such a config file:

```
de
```

[16] http://news.google.co.uk/

```
googleId111111
www.dummy.com
```

- pathToSaveOutput is the full path to a file where the tool will write its outputs (e.g. "*C:\\googleNewsDownloader\\results.txt*")

3.5.5. Input and Output Data Format

The input to the tool is a configuration file as described in the preceeding section. The output is another file that contains on each line the following data about a news event (each field is tab separated):

- url
- language Code
- title
- date of publication

An example output entry is:

```
http://feedproxy.google.com/~r/BootlvPortativieDatori/~3/3jwRpPoC0NE/    lv
      Motorola izlaidusi Lapdock 100 viedtālruņu "dokstaciju"     Wed, 12
Oct 2011 08:07:18 +0000
```

3.5.6. Contact

For further information and technical support for installing and/or running this tool, please contact Ahmet Aker: a.aker@dcs.shef.ac.uk.

3.6. News Text Crawler and RSS Feed gatherer

3.6.1. Overview and Purpose

A crawler suitable for extracting texts for parallel phrase extraction. Given a list of URLs, the tool observes any restrictions imposed on automatic programs visiting the webpages specified in the domain's robots.txt files. The tool's focus is news texts, with the assumption that the same news stories are likely to be covered in similar ways in multiple languages. Some online newspapers do not allow a full automatic crawl of their webpages (by specifying this in their "*robots.txt*" file), instead they provide an RSS feed. Therefore our "crawler" consists of two parts:

- *URL crawl.*
- *RSS feed gathering.*

Given files containing URLs to crawl, the crawler (*retrieve_crawled.pl*) enforces "*robots.txt"* compatibility, and tags downloaded files with the time-stamp of the download. To prevent

duplicates (should the crawl be restarted), md5sums of the new pages are compared against existing downloaded pages -- in case of duplication, the more recent download is discarded. Therefore, the crawler can be restarted to allow re-crawls of news sites, or could even be run in a continuous loop through the use of a wrapper script (such a script is not included and would need to be created by the user). Given files containing lists of RSS feeds, the RSS feed retrieval tool (*retrieve_rss_feeds.pl*) downloads the most recent RSS stories from each feed. As with the URL crawler, the user can implement either of two options for repeated downloading:

- set up the program to repeat with a time-based job scheduler (e.g., every 10 minutes). For example, under Unix, this would be a repeating *cron* job.
- a wrapper script to repeat the program's execution; a loop returning to the start of the RSS links list, possibly with an enforced time delay, a maximum number of repeats or set to repeat infinitely.

3.6.2. Software Dependencies and System Requirements

The tool consists of multiple Perl programs and as such is not dependent on any particular platform. Should speed be an issue, parallelization could be achieved simply by a division of the input URL / RSS lists. The amount of data requested will affect the storage requirements. The Perl packages required for correct operation of the tool include:

- *LWP::RobotUA*
- *HTML::LinkExtractor*
- *XML::RSS::Parser::Lite (and thus XML::Parser and XML::Parser::Lite)*
- *LWP::UserAgent*
- *LWP::UserAgent*
- *Digest::MD5*
- *File::Copy*

3.6.3. Installation

Perl is not a compiled program, however, the Perl packages described above need to be installed for the tool to run. For example, on a Unix machine running Fedora 15 these could be installed through the Comprehensive Perl Archive Network (CPAN – directions for installation of *cpan* are available from http://www.cpan.org if this is not already a part of the user's system) by:

- *cpan[1]> install LWP::RobotUA*
- *cpan[2]> install HTML::LinkExtractor*
- *cpan[3]> install XML::Parser*
- *cpan[4]> install XML::Parser::Lite*
- *cpan[5]> install XML::RSS::Parser::Lite*
- *cpan[6]> install Digest::MD5*
- *cpan[7]> install File::Copy*

The News text crawler and RSS feed gatherer is located in the "*D3_5_Section_3_6_rssNewsCrawlerUSFD.zip" archive, which* contains the following files:

- *Useful.pm*
- *Crawler.pm*
- *retrieve_crawled.pl*
- *rss_parse.pl*
- *retrieve_rss_feeds.pl*

During execution, programs should be given overriding arguments for "*<path to data directory>*". In the absence of an overriding value, the tools assume that "*HOME/projects/accurat/data*" is a viable path on the system.

3.6.4. Execution Instructions

The programs are all invoked through a command line.

Crawling Listed URLs

```
perl retrieve_crawled.pl <path to data directory>
```

This program assumes the existence of an "*url_source*" directory within the "*<path to data directory>*", and will create a "*url_downloads"* directory (if this is does not exist already) in the same location. The "*url_downloads*" directory will contain the tool's output.

The "*url_source*" directory should contain files containing URLs (divided per language), for example:

```
[accurat url_source]$ ls
url_Cro.txt url_Deu.txt url_Lva.txt url_Svn.txt
```

```
url_Cze.txt url_Grc.txt url_Rom.txt url_UK.txt
```

The filename is expected to start with "*url_*", be followed by a country identifier (which needs to match one of the abbreviations above), and have a text file extension. The URLs are listed in a one per line form as follows:

```
http://www.pravednes.cz/
http://www.eurozpravy.cz
http://www.novinky.cz/deniky
...
```

Gathering Listed RSS Feeds

```
perl retrieve_rss_feeds.pl <path to data directory>
```

This program assumes the existence of an "*rss_source*" directory within the "*<path to data directory>*", and will create an "*rss_downloads*" directory (if this does not already exist). The "*rss_downloads*" directory will contain the output.

The "*rss_source*" directory should contain files containing URLs (divided per country), for example:

```
[accurat rss_source]$ ls
rss_Cro.txt rss_Deu.txt rss_Lva.txt rss_Svn.txt
rss_Cze.txt rss_Grc.txt rss_Rom.txt rss_UK.txt
```

The filename is expected to start with "*rss_*", be followed by a country identifier (which needs to match one of the abbreviations above), and have a text file extension. The RSS feeds are listed in a one per line form as follows:

```
http://www.rssmonitor.cz
http://seznam.sblog.cz/rss.xml
http://rss.eurozpravy.cz/
http://www.scienceweek.cz/rss.php
...
```

3.6.5. Input and Output Data Format

3.6.5.1. Crawling Listed URLs

The crawler, "***retrieve_crawled.pl***", is invoked with a path to a data directory containing a subdirectory "*url_source*" with a number of "*url_<country>*" files. Each "*url_<country>*" file consists of one link per line, links relating to starting points for the crawler. For example:

```
http://www.pravednes.cz/
http://www.eurozpravy.cz
http://www.novinky.cz/deniky
...
```

While crawling, the program produces output in a subdirectory of the data directory named "*url_downloads*", creating a directory per "*<country>*".

```
[accurat url_downloads]$ ls
cro cze deu grc lva rom svn uk
```

The HTML source of each crawled page is stored within the "*<country>*" directory. As much as possible, the original name of the web page is used to create the name of the file, subject to:

- A maximum length limit
- Replacing of special characters (such as spaces etc.)
- The addition of the timestamp of the download

For example, the URL http://www.twitter.com downloaded on the 19 October 2011 at 8am would yield the following filename:

```
httpXYYtwitterZcom-20111019080000
```

The program also produces output to *STDERR* to allow monitoring of its progress, reporting on the current web page sought, the status of retrieval, whether it was saved and how many links are remaining to be searched.

3.6.5.2. Gathering Listed RSS feeds

The RSS gatherer, "***retrieve_rss_feeds.pl***", is invoked with a path to a data directory containing a subdirectory "*rss_source*" with a number of "*rss_<country>*" files. Each "*rss_<country>*" file consists of one link per line, links relating to an RSS feed. For example:

```
http://www.rssmonitor.cz
http://seznam.sblog.cz/rss.xml
http://rss.eurozpravy.cz/
http://www.scienceweek.cz/rss.php
...
```

While retrieving RSS feeds, the program produces output in a subdirectory of the data directory named "*rss_downloads*", creating a directory per "*<country>*".

```
[accurat rss_downloads]$ ls
Cro Cze Deu Grc Lva Rom Svn UK
```

The Resource Description Framework (RDF) encoded document of the retrieved RSS feeds is stored within the "*<country>*" directory. As much as possible, the original name of the web page is used to create the name of the file, subject to:

- A maximum length limit
- Replacing of special characters (such as spaces etc)
- The addition of the timestamp of the download

For example, repeated downloads of the RSS feeds from http://www.scienceweek.cz/rss/ create the following files in the "*rss_downloads*" subdirectory:

```
...
httpXYYwwwZscienceweekZczYrssY-20111011135319.rdf
httpXYYwwwZscienceweekZczYrssY-20111011142237.rdf
httpXYYwwwZscienceweekZczYrssY-20111011145150.rdf
httpXYYwwwZscienceweekZczYrssY-20111011152603.rdf
...
```

The program also produces output to *STDERR* to allow monitoring of its progress, reporting on the current RSS feed being sought, whether it was found to be well formed, and how many items were on the page if it was well formed.

3.6.6. Contact

For further information and technical support installing and/or running this tool, please contact Judita Preiss: j.preiss@sheffield.ac.uk.

3.7. News Article Alignment and Downloading Tool

3.7.1. Overview and Purpose

The purpose of this tool is to (1) align or pair news articles written in different languages and (2) to download the content of the paired News URLs. This uses as input the output produced by the tools

described in Sections 3.4, 3.5 and 3.6. For a pair of given News URLs it streams the HTML codes, extracts the text from them and saves the extracted text in separate files. The files are encoded in UTF-8.

In step (1) it uses the titles of the news articles and the date information to produce alignment between the articles written in source and target langauges. The source language could be English and the target language German. In this case it aligns English news articles with German ones to produce comparable corpora. The details of the pairing process is described in Deliverable 3.4.

To perform step (2) the tool uses an HTML parser[17] to construct a parsing tree from the HTML document following the Document Object Model (DOM)[18]. Within this parsing tree the tool checks only the BODY and the TITLE tags of the document. It ignores the SCRIPT, TABLE and the FORM tags within the BODY tag as these are likely not to contain relevant text. In addition to this, it ignores parts of the BODY that contains enumeration of information such as menu items, copy right information, privacy notices and navigation hyperlinks. Furthermore, short texts are ignored as well as they are likely to contain advertisements. A text is considered short if it has less than 5 words. This number was optimized after a couple of experiments. The text identified by the tool as pure is then prepared for saving. The tool replaces any ASCII coding within the text and adds a dot on the end of each paragraph if it is not ended with a punctuation

3.7.2. Software Dependencies and System Requirements

The tool requires Java 1.5 or above and 1 GB RAM. It also requires Internet Connection.

3.7.3. Installation

The tool does not require any installation. It can be run using the newsContentDownloader.jar file (see section 3.7.4).

3.7.4. Execution Instructions

To run the tool simply use this command (works both in Unix and Windows):

```
java –jar NewsContentDownloader.jar [pathToSourceNewsFile]
[pathToTargetNewsFile] [pathToOutputFolder] [sourceLangCode]
[targetLangCode] [threshold] [translationOption]
[pathToTargetTitlesTranslatedIntoSourceLang]
```

The command requires the following parameters:

- ***pathToSourceNewsFile*** is the path to a file that contains all the information collected through the tools described in Sections 3.4, 3.5 and 3.6. The News articles are in the source language (e.g. in English).
- ***pathToTargetNewsFile*** is the path to a file that contains all the information collected through the tools described in Sections 3.4, 3.5 and 3.6. The News articles are in the target language (e.g. in German).
- ***pathToOutputFolder*** is the full path to a folder where the tool needs to write the output
- ***sourceLangCode*** is the source language code according to the ISO-639-1 standard (e.g. „en" for English)
- ***targetLangCode*** is the target language code according to the the ISO-639-1 standard (e.g. „de" for German)
- ***threshold*** is the similarity value between two titles and ranges between 0 to 1. If 0 is given than every title in the the source language will be paired with every title in the target language. In case of 1 only exact matches of source language title and translation of target language titles are paired.

[17] http://htmlparser.sourceforge.net/

[18] http://www.w3.org/DOM/

- ***translationOption*** - two values: „*DICT*" or „*EXIST*".
  - In case of „*DICT*" the tool will perform a translation based on dictionaries. It will translate the target language article titles into the source language. To do this it requires target-to-source dictionaries. These dictionaries must be stored in the „dict" subdirectory of „*NewsContentDownloader*" and must have the file name according to „*targetLangCode*" + „_" + „*sourceLangCode*" + „*.txt*", e.g. „*de_en.txt*". The format of the dictionaries is the same as in the DicMetric tool described in Deliverable 2.6.
  - In case of „*EXIST*" the user can providea translation file that contains translations of the target language article titles. To do this an additional argument has to be provided to the tool: ***pathToTargetTitlesTranslatedIntoSourceLang.***
- ***pathToTargetTitlesTranslatedIntoSourceLang*** is the path to a file that contains all target language titles translated into the source language. Note that for this purpose translation tools described in Deliverable 2.6 can be used. The file contains in each line a title translation. Each line must correspond to a line in the ***pathToTargetNewsFile.*** This parameter is not needed if the „DICT" translation option is used.

3.7.5. Input and Output Data Format

The inputs to the tool are listed in Section 3.7.4.

The output is a collection of files containing the textual content of the paired News URLs. The files are encoded in UTF-8. The tool generates under the given folder (***pathToOutputFolder***) a sub-folder. This sub-folder is named *"[source language code]-[target language code]"* (e.g. "*en-de*"). Within this sub-folder the tool creates two further sub-folders, one for each language (e.g. "*en*" and "*de*"). The source files are saved into the source folder and the target files into the target folder. It also creates a file that gives alignment or pairing information. This file ("alignment.txt" is saved e.g. in the "*en-de*" folder. The structure of this file is as following:

```
F:/output/en-de/en/html-1.txt \t F:/output/en-de/de/html-1.txt \t score
F:/output/en-de/en/html-2.txt \t F:/output/en-de/de/html-2.txt \t score
…
```

if "*F:/output*" was the path where to save. The last value in each line (score) specifies the score of the alignment and can vary betwee 0 and 3.

3.7.6. Contact

For further information and technical support installing and/or running this tool, please contact Ahmet Aker: a.aker@dcs.shef.ac.uk.

4. Conclusion

In this document we provide technical documentation of tools which have been used within the ACCURAT project to collect parallel and comparable corpora from the web.

We described tools including focused web crawlers, a Wikipedia corpus collector and news search and crawl tools. For the news search and crawl tools we also provide an alignment and HTML text content downloading tool that aligns news articles written in different languages and downloads the textual content from the HTML presentation of the aligned articles to compose news comparable corpora.

In combination with the tools described in D2.6 the user is now fully equipped to obtain additional training data for SMT or Example-based/Rule-based MT. This documentation contains step-by-step instructions explaining how to install and run the tools. Significant effort has been made to ensure these instructions are understable by users with average computer skills.

5. References

5.1. A Workflow-based Corpora Crawler

Radu Ion, Dan Tufiş, Tiberiu Boroş, Alexandru Ceauşu, and Dan Ştefănescu. *On-Line Compilation of Comparable Corpora and their Evaluation*. In Marko Tadić, Mila Dimitrova-Vulchanova, and Svetla Koeva (eds.), Proceedings of The 7th International Conference Formal Approaches to South Slavic and Balkan Languages (FASSBL-7), pp. 29—34, Croatian Language Technologies Society – Faculty of Humanities and Social Sciences, Zagreb, Croatia, October 2010. ISBN: 978-953-55375-2-6.

Skadina, I., Vasiljevs, A., Skadinš, R., Gaizauskas, R., Tufis, D. Gornostay, T.: Analysis and Evaluation of Comparable Corpora for Under Resourced Areas of Machine Translation. In Proceedings of the 3rd Workshop on Building and Using Comparable Corpora, LREC 2010, Malta, pp. 6-14.

Munteanu, D. S., and Marcu, D. (2006). Extracting Parallel Sub-Sentential Fragments from Non-Parallel Corpora. In Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 81–88, Sydney, July 2006. ©2006 Association for Computational Linguistics

5.2. ILSP FMC tool references

M. Braschler and P. Scäuble. 1998. Multilingual information retrieval based on document alignment techniques. In ECDL ’98: Proceedings of the Second European Conference on Research and Advanced Technology for Digital Libraries, London: Springer-Verlag. 183–197.

A. Eisele and J. Xu. May 2010. Improving Machine Translation Performance Using Comparable Corpora. Proceedings of the 3rd Workshop on Building and Using Comparable Corpora, European Language Resources Association (ELRA), La Valletta, Malta. 35-41.

A. Hassan, H. Fahmy and H. Hassan. 2007. Improving named entity translation by exploiting comparable and parallel corpora. In Proceedings of the 2007 Conference on Recent Advances in Natural Language Processing (RANLP), AMML Workshop.

R. Ion, D. Tufiş, T. Boroş, A. Ceauşu, D. Ştefănescu. October 2010. On-Line Compilation of Comparable Corpora and their Evaluation. Proceedings of the 7th International Conference Formal Approaches to South Slavic and Balkan Languages (FASSBL7), Croatian Language Technologies Society – Faculty of Humanities and Social Sciences, University of Zagreb, Dubrovnik, Croatia. 29-34.

D. S. Munteanu. December 2006. Exploiting Comparable Corpora. PhD Thesis, University of Southern California. ©2007 ProQuest Information and Learning Company.

P. Sheridan and J. P. Ballerini. 1996. Experiments in multilingual information retrieval using the SPIDER system. In SIGIR ’96: Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval, New York: ACM Press. 58–65.

I. Skadiņa, A. Aker, V. Giouli, D. Tufis, R. Gaizauskas, M. Mieriņa, N. Mastropavlos. October 2010. A Collection of Comparable Corpora for Under-resourced Languages. Proceedings of the Fourth International Conference Baltic HLT 2010, IOS Press, Frontiers in Artificial Intelligence and Applications, Vol. 219, Riga, Latvia. 161-168.

I. Skadiņa, A. Vasiljevs, R. Skadinš, R. Gaizauskas, D. Tufis, T. Gornostay. 2010. Analysis and Evaluation of Comparable Corpora for Under Resourced Areas of Machine Translation. Proceedings of the 3rd Workshop on Building and Using Comparable Corpora. LREC, Malta. 6-14.

T. Talvensaari, J. Laurikkala, K. Järvelin, M. Juhola and H. Keskustalo. 2007. Creating and exploiting a comparable corpus in cross-language information retrieval. ACM Trans. Inf. Syst., 25(1), 4.

T. Talvensaari, A. Pirkola, K. Järvelin, M. Juhola and J. Laurikkala. October 2008. Focused Web crawling in the acquisition of comparable corpora. Information Retrieval, 11, 5, 427-445.

T. Tao and C. Zhai. 2005. Mining Comparable Bilingual Text Corpora for Cross-Language Information Integration. In Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'05). 691–696.

T. Utsuro, T. Horiuchi, Y. Chiba and T. Hamamoto. 2002. Semi-automatic compilation of bilingual lexicon entries from cross-lingually relevant news articles on WWW news sites. In AMTA '02: Proceedings of the 5th Conference of the Association for Machine Translation in the Americas on Machine Translation: From Research to Real Users, London: Springer-Verlag. 165–176.