Project funded by the People Programme (Marie Curie Actions) of the European Union's Seventh Framework Programme FP7/2007-2013/ under REA grant agreement no. 317471.

Project reference: 317471

Project full title: EXPloiting Empirical appRoaches to Translation

D3.2: Multilingual Corpus

Authors: Eduard Barbu (Translated)

Document Number: EXPERT_D3.2_20160128

Distribution Level: Public

Contractual Date of Delivery: 30.09.15

Actual Date of Delivery: 28.01.16

Contributing to the Deliverable: WP3

WP Task Responsible: Translated

EC Project Officer: Thierry Colin

Table of Contents

Revision History

Revision	Date	Author	Organisation	Description
V 1.0	15.01.2016	Eduard Barbu	Translated	First Draft
V 2.0	25.01.2016	Hernani Costa	University of Malaga	Corrections and Suggestions
V 3.0	28.01.2016	Eduard Barbu	Translated	Integration of Corrections and Suggestions

1. Executive Summary

This deliverable describes the acquisition of a multilingual corpus by crawling the web and the Common Crawl repositories. It also describes a procedure for spotting false translations in Translation Memories. Figure 1.1 shows the interconnection of the components described in the rest of deliverable. The deliverable has two big sections corresponding to data acquisition and data cleaning.

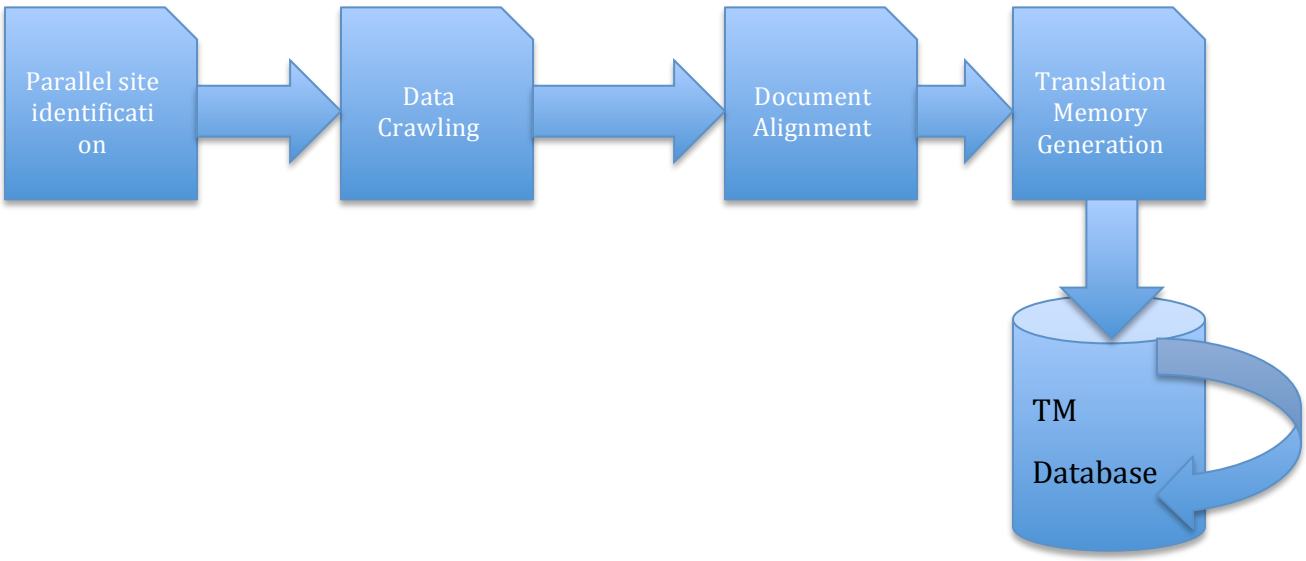

Figure 1.1. The interconnection of the components

The process of data collection comprises four stages pictured in figure 1.1. From left to right, the promising parallel sites are identified, then data is crawled using a CAE tool, afterwards the documents are aligned, the segments corresponding to the parallel documents are also aligned and the Translation Memory (TM) is generated. The Translation Memories are dropped in the TM Database, as can be seen in the last part of figure 1.1. The loop that starts and ends in TM Database represents the identification of the false translations, the subject of the second part of this deliverable.

2. Collecting the multilingual corpora

This effort of data collection is part of a larger project of collecting multilingual data that the author undertakes with Achim Ruopp from TAUS and Christian Buck from University of Edinburgh. Our purpose is to offer the Machine Translation Community a significant amount of multilingual data to that will help enhance the automatic translation quality.

For that purpose, we collected multilingual data from the Common Crawl repository and from the World Wide Web. The advantage of collecting multilingual data from the Common Crawl is that the web sites are already crawled and therefore the content of the web pages is easily accessible. Unfortunately, the way that Common Crawl collects data is not by crawling live sites. They use instead a list of web urls provided by the defunct searching engine Blekko[1] and download the pages corresponding to those links. This means that for some sites there is a huge gap between the content of the site in Common Crawl and the live content of the site.

In what follows we give a short presentation of Common Crawl. The presentation is meant to clarify the rest of the deliverable. For a detailed presentation the reader should consult the documentation on the Common Crawl site[2].

The Common Crawl corpus consists of petabytes of data collected from the web in the past 7 years. These data contain: the raw web page data, the metadata corresponding to the page and the extracted text of the pages. The data is collected and archived on the amazon servers. The data is collected in stages and spread over multiple archives. The format of the archived data changed during the years but it seems to be frozen now. Each crawl contains the following files:

- The WARC files store the raw crawls. The WARC (Web Archive) is a universal format for storing web archives.
- The WAT files store the metadata computed for the WARC files.
- The WET files contained the plain text in UTF-8 format extracted from the WARC files.

---

[1] https://en.wikipedia.org/wiki/Blekko

[2] https://commoncrawl.org

The data stored on the amazon servers can be either downloaded or processed online. For the second possibility an amazon account should be bought.

In addition to the documentation of the data, the Common Crawl site features tools for exploring the data[3], an index of the last crawled archives[4] and software for processing the archives written mostly in Java and Python.

The next subsection presents an attempt to estimate the amount of parallel data in Common Crawl. Then, we make an evaluation of three Crawl Align and Export (CAE) tools that given a multi-lingual site, find the corresponding parallel pages. Finally, we present the methodology for building parallel corpora from web and Common Crawl and the results to date.

I. Estimation of the quantity of data in Common Crawl

We should first answer the question: What is the quantity of parallel data in Common Crawl? To answer this question we have made a rough estimation of the amount of parallel data available using sampling. We have drawn a sample of pages from the Common Crawl index and then checked if the pages in the sample have a translation in any language. Before computing the size of a relevant sample let us introduce some statistical measures related to the task:

- **The population size**. The population size comprises all the web pages in the Common Crawl. The estimation of the population size is not trivial for two reasons. In the first place, Common Crawl is updated every few months; therefore the estimation becomes obsolete very fast. In the second place, there is a significant amount of duplicate pages in the Common Crawl repository.

- **Confidence Level** -It reflects how certain we are that our sample accurately reflects the population, within its margin of error.

[3] http://blog.commoncrawl.org/2013/03/url-search-tool/

[4] http://blog.commoncrawl.org/2013/01/common-crawl-url-index/

- **Margin of error** - It reflects how close the answer our sample gives is to the true value in the population. Given that the proportion of the pages that are translations in the Common Crawl or on the web is small we would accept a margin of error of maximum 2 percents.

When the evaluation has been performed we have made a rough estimation of the number of pages in the Common Crawl based on the reports people made on the Common Crawl forum. Our estimation has been close to 20 billions of pages.

Christian Buck from University of Edinburgh has made a better estimation of the population size. He has created a system for indexing the Common Crawl pages starting with the archives in 2013, the first ones that have the format discussed before, and ending with the last but one archive in 2015. The results are given in the following three graphs[5]. In the first graph we plot the number of web pages in each crawl archive with blue. With red is plotted the number of unique pages in the crawl archives. The trend is already obvious: the older crawls have a higher degree of duplication than the newer ones. To describe the trend more formally, in the second graph we show the relative percentage of unique urls in a single crawl. This percentage seems to converge to the 0.9 in more recent crawls. The third graph plots the cumulative number of web pages in each crawl (blue color), the cumulative number of local unique (red color) and the cumulative number of global unique (yellow color). The real number of unique pages in Common Crawl is close to 5 billions. Surprisingly, our estimation is close to the cumulative number of unique local pages. This number is now 27 billions but 7 months ago when we did the estimation was lower (around 23 billions).

[5] Printed with permission.

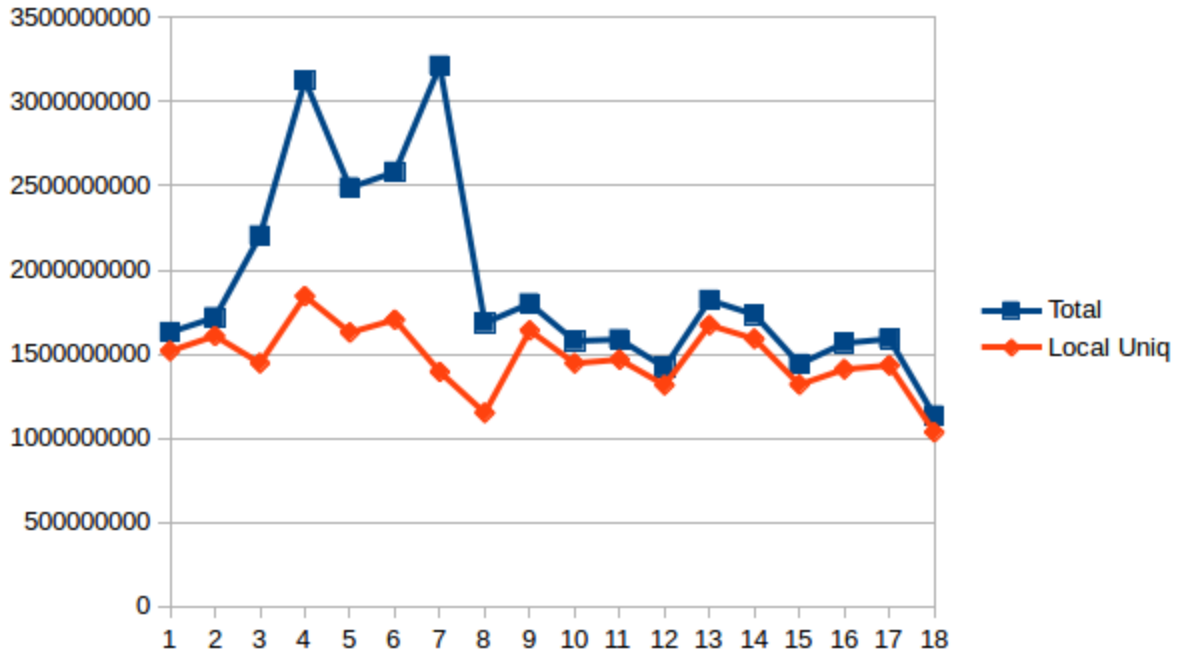

Figure 2.1 Number of web pages and unique web pages in each Common Crawl archive.

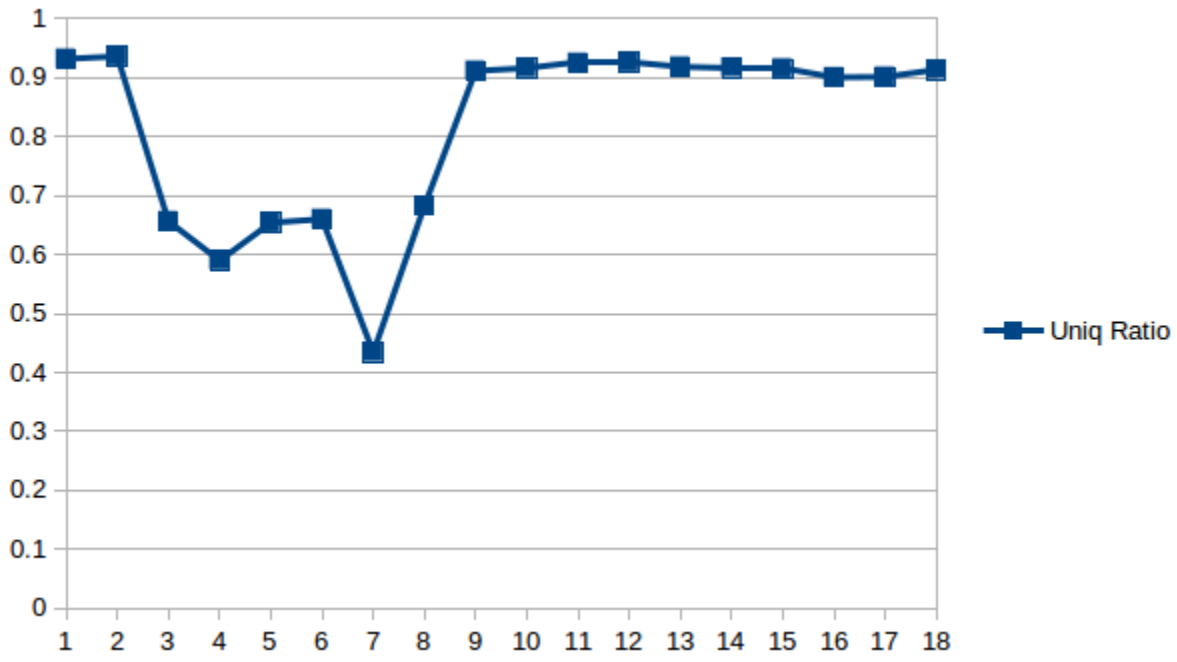

Figure 2.2 Relative percentage of unique urls.

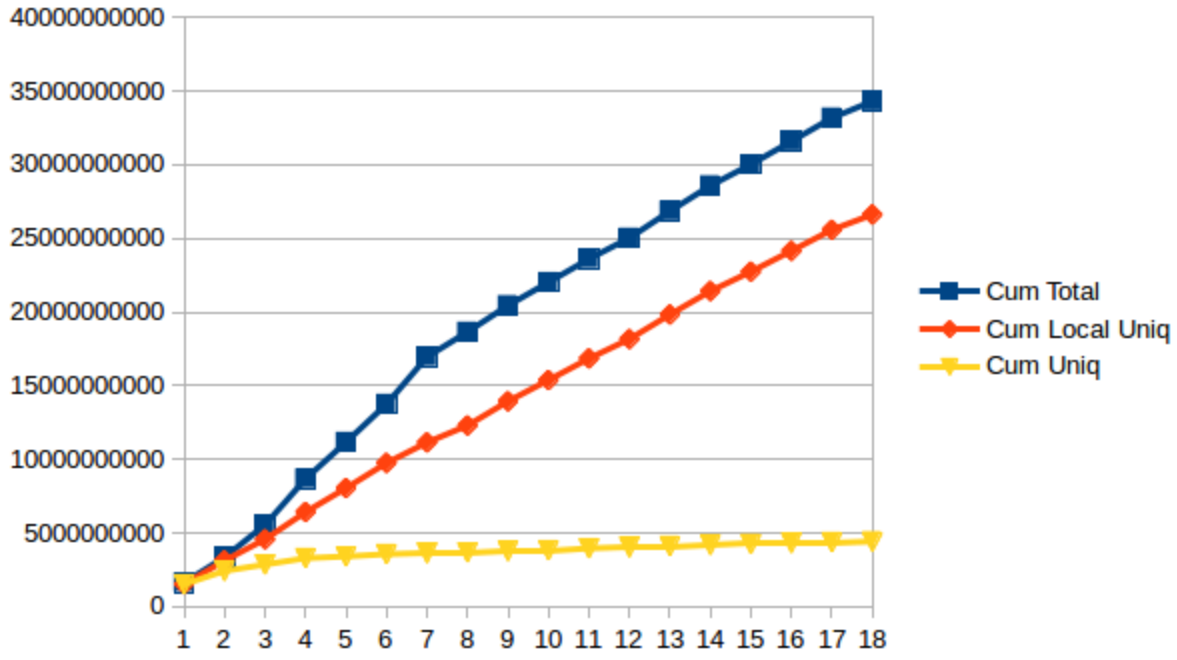

Figure 2.3 Number of web pages and unique web pages in each Common Crawl archive.

To draw a statistically significant sample at a confidence level of 80 % and a margin of error of 2% we should annotate 1024 pages or more. Because the annotation is a tedious task we have only annotated a sample of 350 pages. For this reason and because of the difficulties related to a correct estimation of the population size the results we have got should be taken "cum grano salis".

An annotation consists of the following attributes:

- url: is the url to be annotated;
- lang_id: if the URL contains a language identifier annotate "yes", otherwise "no". The purpose of this annotation is to compute the percent of urls that can be guessed by simple heuristics;
- long_text: if the text in the page is long then annotate "yes". The purpose of this annotation is to identify the percentage of long texts that have translations;
- Content translation: annotate "yes" if the content is translated and annotate "no" if what is translated is boilerplate;

- Interface translation: annotate “yes” if the elements of the interface are translated;
- Translated URL: if the target URL exists, this URL is copied;
- Estimation Translated Words: we give an estimation of the number of words translated in case the translation URL exists.

The estimation is that 6.59 percent of the pages in the sample are translated and each translated page has an average number of 107 words. With 5 billions unique pages in the Common Crawl, we should have approximately 32 billion parallel words.

II. An Evaluation of three CAE tools

In this chapter we present the evaluation of three pieces of software that identify parallel pages in sites that contain multilingual content, align the parallel documents at sentence level and export the results in a TMX format. From now on we will call these pieces of software CAE (Crawl Align and Export) tools. The three CAE tools we are going to evaluate are: the ILSP crawler (Papavassiliou and Prokopidis, 2013), Bitextor (Esplà-Gomis, 2009) and Taus Data Spider (Ruopp and Xia, 2008).

The CAE tools perform the following operations:

- **Page fetching**. The remote web pages are downloaded, the character set of the document is identified and the relevant metadata is extracted. Moreover the links in the page content are also fetched. A good crawler can be restricted to crawl a certain web domain or subdomain.
- **Data cleaning**. The boilerplate is removed or annotated and the elements identified as advertisements are removed. Sometimes it is fine to keep the boilerplate, but in other situations it is better to remove it.
- **Language Detection**. The language of the web document is detected. More sophisticated systems of language detection assign language codes to each part of the pages that have content in more than one language.
- **Duplicate removal**. The duplicate pages are identified and removed. The most used technique for duplicate identification is a simple md5 check.
- **Export**. Some crawlers (e.g. ILSP crawler) export the crawl in a standard xml format.

- **Document Alignment**. The parallel documents are identified and aligned. This is the key component of a CAE tool. The heuristics the CAE tool use are heuristics based on the URL, heuristics based on the document structure or heuristics based on the text content.
- **Sentence Alignment.** The documents are divided in sentences and the sentences are aligned with a classical sentence aligner like: Gargantua (Braune and Fraser, 2010), Hungalign (Varga et al., 2005) or Maligna (Jassem and Lipski, 2008).
- **TMX exporter**. The result of the alignment is exported as TMX files. These files can be easily imported in Translation Memories Database.

Now we describe the heuristics used by the three CAE tools to align parallel documents. The ILSP Focused Crawler is built on top of Bixo, an open source web mining toolkit. It uses heuristics based on document structure and document content. The ILSP crawler assumes that the parallel documents should be on the same level in the web site tree. This assumption makes the computation efficient for large web sites. It computes four features fed to a SVM classifier whose task is to decide if the documents are parallel. The first feature is the structure similarity of documents. The second feature is the ratio of the two candidate documents in terms of paragraphs. If this ratio is under a threshold it might be that the documents are parallel. The third feature is the ratio of the candidate documents text. The fourth feature is the Jaccard similarity coefficient of the image list on the pages of the documents.

Bitextor combines the structure similarity with the Church Gale score of the text length and with a bilingual dictionary to decide if the documents are parallel. Bitextor is the only CAE tool that uses a bilingual dictionary to find parallel documents.

Taus Data Spider is an in-house tool developed and maintained by Achim Ruopp for crawling the web, which uses some very basic strategies for mapping parallel documents. It computes potential parallel documents based on the URLs of the documents.

III. The data for evaluation

We have selected a set of multilingual sites that have web pages in English, Italian, German and French. The selected web sites are diverse in terms of number of pages hosted, the length of documents and organisation. The evaluation is always a tricky business because there are many

parameters that can be tuned for CAE tools. We have tried more than one run for each site to find the best parameters.

Bitextor and Taus Crawler only allow bilingual crawls. However, for a proper evaluation we need to know the number of monolingual documents on the site. Fortunately, ILSP crawler has a modality for monolingual crawling. In the table 2.1 we show the results of the crawl for ILSP crawler. Identical tables have been produced for Bitextor and Taus Spider[6].

Site	Language Pair	N. Parallel Documents	N. En. Documents	N. Dest. Documents	N. pag. Google	Time Crawl
http://www.nke.at/en/	en - it	14	18	9	2840	1 hour
http://europa.eu/	en -it	3358	3976	2300	110000	50 cycles
http://www.terex.com/en/	en - it	13	344	7	10100	1 hour
http://www.iisd.org/sd/default.aspx	en- fr	12	327	14	18000	1 hour
http://www.medicusmundi.ch/en	en-fr	36	390	97	14500	1 hour
http://www.hettahuskies.com/	en-de	33	247	27	160	1 hour
http://www.medicusmundi.ch/en	en-de	76	389	186	13800	1 hour
http://www.tekstwerk.com/	en-de	4	3	3	35	1 hour

Table 2.1. Results for the bilingual and monolingual crawl for the selected sites.

There are three sites that contain parallel pages in English and Italian, two sites that contain parallel pages in English and French and three sites that contain parallel pages in English and German. The column *N. Parallel Documents* lists the number of parallel documents identified by the ILSP crawler. The column "*N. En. Documents"* lists the number of

[6] Christian Buck and Achim Ruopp have produced the tables. They are not part of this deliverable.

English Documents crawled in monolingual setting. The column "*N. Dest. Documents"* gives the number of documents crawled in monolingual setting for the destination language. The column "*N. pag. Google"* presents an estimation of the number of pages indexed by Google for the respective site. Finally, the last columns give the time we have instructed ILSP crawler to crawl the site. Please notice that for a large site like Europa.eu we instructed the ILSP crawler to crawl for 50 cycles. The crawl lasted more than 24 hours. Sometimes the number of destination documents crawled in the monolingual setting is less that the number of parallel documents found. In theory this is impossible, but in practice this happens due to a bug of the ILSP crawler or a problem with the network when the crawl was performed.

IV. The results of the evaluation

We have drawn a statistical significant sample of pages from the crawled sites and manually annotated this sample. To compute the size of the sample we have estimated the population size for a language pair as the union of all web pages found in the bilingual and monolingual settings. The size of the sample for each language pair has been computed for 90 % confidence interval and 5% margin of error. The Bilingual Layer consists of all the pages crawled in the Bilingual Setting. The Monolingual Layer consists of all pages crawled in the Monolingual Setting. We have performed a stratified sampling from the Bilingual Layer and the Monolingual Layer. The size of the layers and the size of the samples are:

- For English-Italian language pair the size of the population is 8787, therefore the size of the sample should be 265. The size of the Bilingual Layer is 7852 and the size of the Monolingual Layer is 935. We sample from the Bilingual Layer 236 pages and from the Monolingual Layer 29 pages according to the proportion of the layers.
- For English-French language pair the size of the population is 2528, therefore the size of the sample should be 246. The size of the Bilingual Layer is 1740 and the size of the Monolingual Layer is 788. We sample from the Bilingual Layer 170 pages and from the Monolingual Layer 76 pages according to the proportion of the layers.
- For English-German language pair the size of the population is 1292, therefore the size of the sample should be 226. The size of the Bilingual Layer is 490 and the size of the

Monolingual Layer is 802. We sample from the Bilingual Layer 86 pages and from the Monolingual Layer 140 pages according to the proportion of the layers.

Native Italian, French and German language speakers annotate the English-Italian, English-French, and English-German samples. The task of the annotator is to find the correct target URL for the source URL. The target URL can be the target URL found by one of the CAE tool or a different one. The attributes of the annotations are the following:

- Source URL – it is an English URL extracted from Bilingual Layer or an English URL /Destination Language URL extracted from the Monolingual Layer.
- Target-URL- it is a destination language URL proposed by one of the CAE tools.
- Human-URL -it is the URL found by the annotator.
- Full Translation-"yes" if the Human-URL or the Target-URL contains a full translation of the source.
- Partial Translation- "yes" if the Human-URL or the Target-URL contains a partial translation of the source.
- Comments- in this section the annotators add comments related to the annotation process. The most frequent comment state that in a particular annotation instance only the boilerplate has been translated.

Based on the annotator's output we compute the following measures:

- NRC (Number Retrieved Crawler)-the number of source-target URL pairs retrieved by the CAE tool.
- NRH (Number Retrieved by Human)-the number of source-target URL pairs found by the human evaluators.
- NCC (Number Correct CAE)-the number of source-target URL pairs correctly retrieved by the CAE tool.
- The Precision of the CAE tool is computed as the ratio between NCC and NRC
- The Recall of the CAE tool is computed as the ratio between NCC and NRH

The results for the three CAE tools are given in table 2.2:

Language Pair	Crawler	NRC	NRH	NCC	P	R	F1
en - it	ILSP	96	216	86	0.89	0.39	0.55
en - it	Bitextor	6	216	2	0.33	0.009	0.018
en - it	Taus-Spider	3	216	3	1	0.013	0.027
en - fr	ILSP	9	131	1	-	-	-
en - fr	Bitextor	90	131	1			
en -fr	Taus-Spider	8	131	0			
en -de	ILSP	27	83	27	1	0.32	0.49
en -de	Bitextor	28	83	28	1	0.33	0.5
en -de	Taus-Spider	4	83	4	1	0.04	0.09

Table 2.2 Evaluation results for the three language pairs.

The results are dependent on the CAE tool and the particularities of the sites crawled. However, a trend in the data is obvious: the recall is low, ranging from a little more than 0% to a maximum of 40%. Taus-Spider has good precision, but its only strategy for finding parallel pages is not sufficient for a large crawling effort. The results obtained by ILSP crawler and Bitextor are comparable for English-French and English-German. For English-Italian the superior results of ILSP Crawler are due in part to the fact that the default configuration of HTTrack, the crawler used by Bitextor, does not properly download part of the **europa.eu** web site. In the case of English-French sites, however, we could not compute the precision or recall, because the CAE tools retrieved, with one exception, only false parallel documents. The reason for this failure is that the French content is mostly made up of similar web pages representing products. A cursory look can mislead even the human evaluators, because in many cases the same image belongs to multiple pages. What makes these pages different are the attributes or the values of a number of attributes (e.g. length and width) of the products.

Despite the failure in the French case, we believe that Bitextor and the ILSP crawler can be used to find parallel documents in the majority of web sites. For example, for both English-Italian language pair and English-German language pair the precision is between 0.9 and 1.

V. Building the parallel corpus

The effort for building the parallel corpus has three components:

- The first strategy is to mine the Common Crawl data using Bitextor. This is the strategy pursued by the other members of the team.
- The second strategy is to find promising sites in Common Crawl and then mine the web site live version using the ILSP crawler.
- The third strategy is to find parallel pages in Common Crawl and then align them using an aligner.

The author of this deliverable has pursued the second and the third strategies. In what follows we will discuss these two strategies in turn.

The idea behind the second strategy is that the parallel pages can be found applying simple heuristics based on the way the URL is written. The idea is not new and was used in the past to mine Common Crawl Data (Skandis et al., 2014). The novelty is the extension of the heuristics and the using of an index of Common crawl for easily accessing the pages. The heuristics for identifying parallel pages are the following ones:

- **Heuristics 1.** The final part of an URL path contains a language code like in in the example bellow. The final file of the source URL is mi0064_en.htm. If we substitute the language code with the Italian code we obtain mi0064_it.htm. The rest of the path remains unchanged.
    - *Source URL:* http://europa.eu/legislation_summaries/internal_market/single_market_for_goods/motor_vehicles/motor_vehicles_technical_harmonisation/mi0064_en.htm
    - *Destination URL:* http://europa.eu/legislation_summaries/internal_market/single_market_for_goods/motor_vehicles/motor_vehicles_technical_harmonisation/mi0064_it.htm
- **Heuristics 2.** The path contains a language code change. There are two cases here: either the path without the final file contains a language code change or the path and final file contain a language code change like in the example bellow.
    - *Source URL:* http://www.rhi.at/internet_en/corporate_news_query_2011_en.html
    - *Destination URL:* http://www.rhi.at/internet_it/corporate_news_query_2011_it.html
- **Heuristics 3.** The source English language is not marked with a language code in the URL, but the destination language is added to the destination URL as in the example:
    - *Source URL:* http://www.airmalta.com/flypass
    - *Destination URL:* http://www.airmalta.com/flypass-it-IT
- **Heuristics 4.** The source and destination pages contain the language codes in the form of parameters as in the example bellow:
    - *Source URL:* http://chess.com/chess_start.html?lang=en

    - *Destination URL:* http://chess.com/chess_start.html?lang=it
- **Heuristics 5.** Multiple language codes can be present as shown bellow:
    - *Source URL:* http://3dmconcept.com/3dmfinal_english/en_modelisation3d_services.html
    - *Destination URL:* http://3dmconcept.com/3dmfinal_italiano/it_modelisation3d_services.html

The mapping of the parallel documents is performed in two stages. First, all potential candidates for the language pairs English-Italian, English-French and English-German are extracted and then the 5 heuristics for mapping are applied. The extraction phase is run on all web pages indexed for the Common Crawl archives starting with 2013 and ending with the last but one archive in 2015. We then count the number of potential parallel documents per site. If this number is more than a threshold (fixed at 10 in our experiment) then we add the site to the list of the sites to be crawled.

The sites are crawled with the ILSP crawler, making sure that the crawler stays inside the site and that the downloaded and mapped documents can have any length.

The ILSP crawler produces a series of intermediate files that are no longer needed once the crawl ends its work. A script extracts the relevant files from the crawl and arranges them in an easy-to-read structure.

The results of the crawl are a series of XML and TMX files. Each XML file represents the normalisation of a corresponding HTML downloaded document. The normalisation consists in the encoding of the text in UTF-8 format, the annotation of boilerplate and the assignation of XML tags to the content. The metadata in the XML file consists in the link to the original HTML document and a language code representing the language of the document. Each TMX file represents the alignment with Maligna sentence aligner of two parallel documents. The TMX files can be easily imported in a translation memory database.

The statistics for the documents crawled is kept in a database table with the following fields:

1. *site*: the URL of the crawled site.
2. *sitedir*: the directory where the TMX files for the crawled site are stored.

3. *ncrawleddocuments*: the number of documents crawled for the site.
4. *nparalleldocuments*: the number of parallel documents identified by the CAE tool.
5. *nparallelsegments*: the number of parallel segments found by the sentence aligner.
6. *nwordssource* : the number of words in the source side of parallel documents.
7. *nwordstarget*: the number of words in the target side of parallel documents.
8. *slanguage*: the two letter code of the source language.
9. *dlanguage*: the two letter code of the target language.
10. *crawler:* the name of the crawler used to crawl the site.

This database table can be used to store the results of crawling with any CAE tool not only with ILSP crawler. In addition to this database table we will release the URL of the parallel documents identified.

So far we have crawled 23.094 sites for roughly 6 hours each corresponding to 19.300 crawl events. Please notice that more than one crawl event can correspond to each site (e.g. when the site has content in more than 2 languages). Unfortunately, for roughly half of the sites (12.195) the ILSP crawler did not find parallel documents. The main reason for this failure is that the strategies used for finding parallel documents implemented in the crawler and the strategies implemented in our rules to map potential parallel documents are different. This observation validates the third strategy for mapping the parallel documents.

The sites in the crawler list that produced parallel documents gave us roughly 383 millions English words corresponding to roughly 386 million words in Italian, French and German. The complete results of the crawl are given in Table 3.1:

Source language	Target language	Parallel documents	Source words	Target words
en	it	214.037	80.237.106	82.001.321
en	fr	401.957	156.820.061	171.929.022
en	de	422.949	145.984.997	132.723.983

Table 3.1 Size of the parallel data collected.

Even if the web crawling is a time-wise expensive activity, this approach produced a fair amount of data and it will be continued. Since we have stored the crawled data, we are allowed to attempt the pairing of parallel documents using different strategies. Thus, we hope to increase the number of parallel documents found and therefore the number parallel words crawled.

The second strategy is to use the result of document mapping by the 5 heuristics presented above and a sentence aligner to extract the parallel corpus. At the time of writing of this deliverable this is work in progress.

3. Cleaning Translation Memories and Multilingual Corpora

Most of the work reported in this section has been reported in (Barbu, 2015)[7].

Translation memories store parallel segments that are translations. The most used file format to store translation memories is called TMX (Translation Memories eXchange). The translation memories can be compiled either by the translators themselves or by aligning parallel corpora. Of course, the translation memories compiled by translators are generally trustworthy and have fewer errors than the translation memories compiled by aligning parallel corpora. In this section we propose a framework to identify faulty translations in translation memories and parallel corpora obtained by aligning web documents.

[7] The author want to thank Luca Mastrostefano from Translated for help in extracting and preparing the English Italian data.

The framework starts with an empirical inspection of a sample of aligned segments. This is necessary because the errors one can find depend on the type of the resource (e.g. aligned corpora, translation memory obtained by collaborative effort) and on the idiosyncrasies of the alignment process. Nevertheless in all cases the task of identifying faulty translations can be thought as a classification process. Ideally the classifier should return "yes" if the segments are translations and "no" if the segments are not translations.

The features the classifier bases the decision depend on the type of error one identifies upon the empirical inspection. The next two sections describe two experiment for cleaning MyMemory translation memory and Europarl (Koehn, 2005) parallel corpus.

I. Cleaning Translation Memories

For various reasons, some of which will be explored in what follows, translation memories contain bisegments where the source segment is not a true translation of the target segment. Our purpose is to devise an automatic procedure to identify with a good precision such segments.

One should notice that the task of identification of this type of segments is different from the customary task of finding mistakes in the segments of translation memory. For examples there are software tools like Apsic X-Bench[8] that incorporate basic methods of data cleaning of translation memories. Apsic X-Bench implements a series of syntactic checks for the segments. It checks for example if the opened tag is closed, if a word is repeated or if a word is misspelled. It also integrates terminological dictionaries and verifies if the terms are translated accurately. The main assumptions behind these validations is that the translation memories bi-segments contain accidental errors (e.g. tags not closed) or that the translators sometimes use inaccurate terms that can be spotted with a bilingual terminology. These assumptions might hold for the translation memories produced by professional translators but not for collaborative translation memories and translation memories derived from parallel corpora. Our task concerns the content of translation not to the form; it is therefore semantic and not syntactic.

[8] http://www.xbench.net

A task similar to spotting false translations is Quality Estimation in Machine Translation. Quality Estimation can also be modelled as a classification task where the goal is to distinguish between accurate and inaccurate translations (Li and Khudanpur, 2009). The difference is that the sentences whose quality should be estimated are produced by Machine Translations systems and not by humans. Therefore the features that help to discriminate between good and bad translations in this approach are different from those in ours.

MyMemory[9] (Trombetti, 2009) is the largest translation memory in the world. It contains more than 1 billion bi-segments in approximately 6000 language pairs. MyMemory is built using three methods. The first method is to aggregate the memories contributed by translators. Hence because of privacy reasons there are public translation memories when the translators agreed to make public the contributions and private ones. The second method is to use translation memories extracted from corpora, glossaries or data mined from the web. The current distribution of the automatically acquired translation memories is given in figure 3.1.

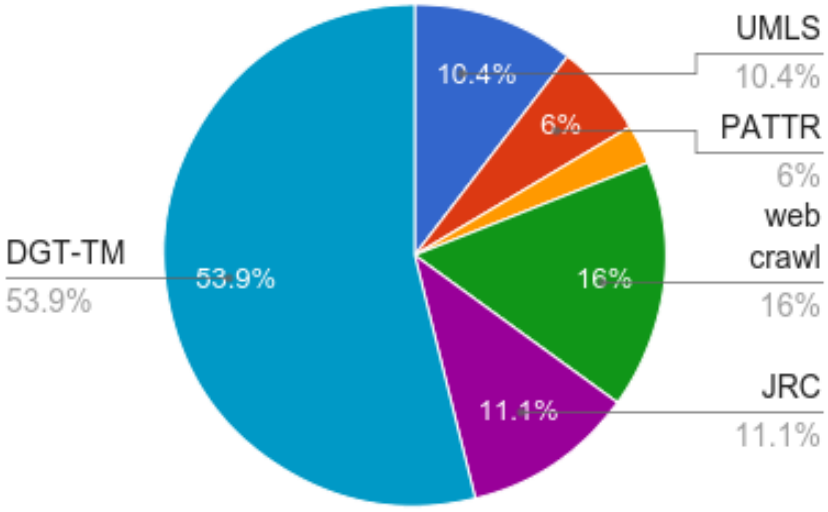

Figure 3.1 Distribution of automatically acquired memories in MyMemory.

Approximately 50% of the distribution is occupied by the DGT-TM (Steinberger et al., 2013), a translation memory built for 24 EU languages from aligned parallel corpora. The glossaries are represented by the Unified Medical Language System (UMLS) (Humphreys and Lindberg,

[9] https://mymemory.translated.net/

1993), a terminology released by the National Library of Medicine.

The third method is to allow anonymous contributors to add source segments and their translations through a web interface.

The quality of the translations using the first method is high and the errors are relatively few.

However the second method and especially the third one produce a significant number of erroneous translations. The automatically aligned parallel corpora have alignment errors and the collaborative translation memories are spammed or have low quality contributions.

In the next section we look better at the data using statistical measures to evaluate data quality and the distribution of faulty segments.

II. Training and Test Set

In this section we describe the process of obtaining the training and test data and make list of characteristic errors found in the data. We also explore statistically the data and show that the data coming from the collaborative interface is different from the data obtained from MateCat.

The positive training examples are segments where the target segment correctly translates the source segment. The negative training examples are translation memory segments that are not true translations. Before explaining how we collected the examples it is useful to understand what kind of errors the translation memories part of MyMemory contain. The errors have been identified by manually examining a sample of bi-segments. They can be roughly classified in the four types:

- **Random Text**. The Random Text errors are cases when one or both segments is/are a random text. They occur when a malevolent contributor uses the platform to copy and paste random texts from the web.
- **Chat**. This type of errors verifies when the translation memory contributors exchange messages instead of providing translations. For example the English text “How are you?” translates in Italian as “Come stai?”. Instead of providing the translation the contributor answers “Bene” (“Fine”).
- **Language Error**. This kind of errors occurs when the languages of the source or target segments are mistaken. The contributors accidentally interchange the languages of source

and target segments. For example they know that they should translate from English to Italian but mistakenly label the Italian segment with the English language code and the English segment with the Italian language code. We would like to recover from this error and pass to the classifier the correct source and target segments. There are also cases when a different language code is assigned to the source or target segment. This happens when the parallel corpora contain segments in multiple languages (e.g. the English part of the corpus contains segments in French) and the aligner does not check the language code of the aligned segments.

- **Partial Translations**. This error verifies when the contributors translate only a part of the source segment. For example, the English source segment "Early 1980s. Muirfield C.C." is translated in Italian partially: "Primi anni 1980" ("Early 1980s").

The errors **Random Text** and **Chat** take place in the collaborative strategy of enriching MyMemory. The **Language Error** and **Partial Translations** are pervasive errors.

It is relatively easy to find positive examples because the high majority of bisegments are correct. Finding good negative examples is not so easy as it requires reading a lot of translation segments. Inspecting small samples of bi-segments corresponding to the three methods, we noticed that the highest percentage of errors come from the collaborative web interface. To verify that this is indeed the case we make use of an insight first time articulated by Church and Gale (Gale and Church, 1993). The idea is that in a parallel corpus the corresponding segments have roughly the same length. To quantify the difference between the length of the source and destination segments we use a modified Church-Gale length difference (Tiedemann, 2011) presented in equation 3.1:

$$CG = \frac{ls - ld}{\sqrt{3.4(ls + ld)}}$$

Equation 3.1. Church Gale length difference.

In figures 3.2 and 3.3 we plot the distribution of the relative frequency of Church Gale scores for two sets of bi-segments with source segments in English and target segments in Italian. The first set, from now on called the Matecat Set, is a set of segments extracted from the output of

Matecat[10]: a free web based CAT tool. The bi-segments of this set are produced by professional translators and have few errors. The other bi-segment set, from now on called the Collaborative Set, is a set of collaborative bi-segments obtained using a user interface that does not require authentication.

If it is true that these two sets come from different distributions then the plots should be different. This is indeed the case. The plot for the Matecat Set is a little bit skewed to the right but close to a normal plot. In figure 3.2 we plot the Church Gale score obtained for the bi-segments of the Matecat set and in figure 3.3 we add a normal curve over the histogram to better visualize the difference from the Gaussian curve. For the Matecat set the Church Gale score varies in the interval −4.18 ...4.26.

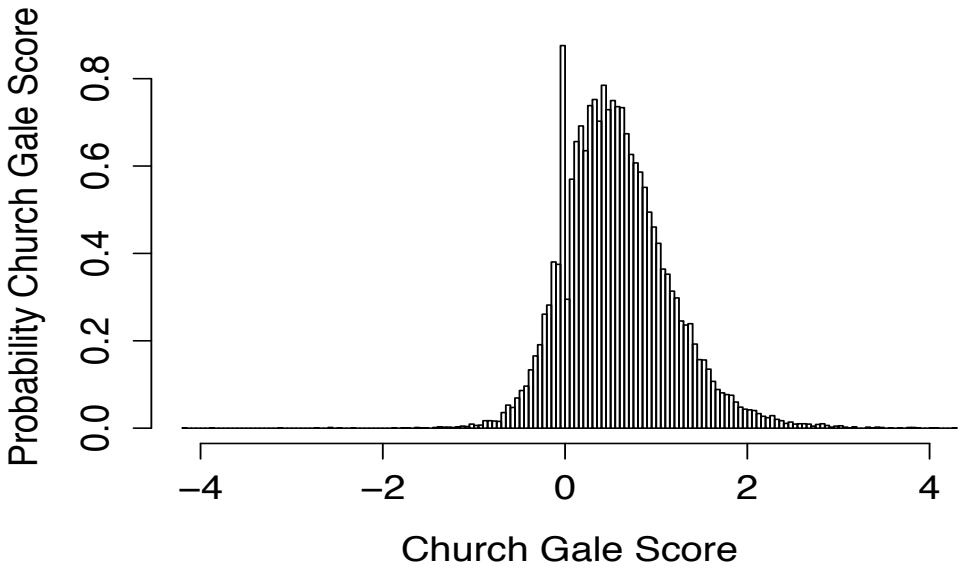

Figure 3.2. Distribution of Church Gale score in the MateCat set.

[10] https://www.matecat.com

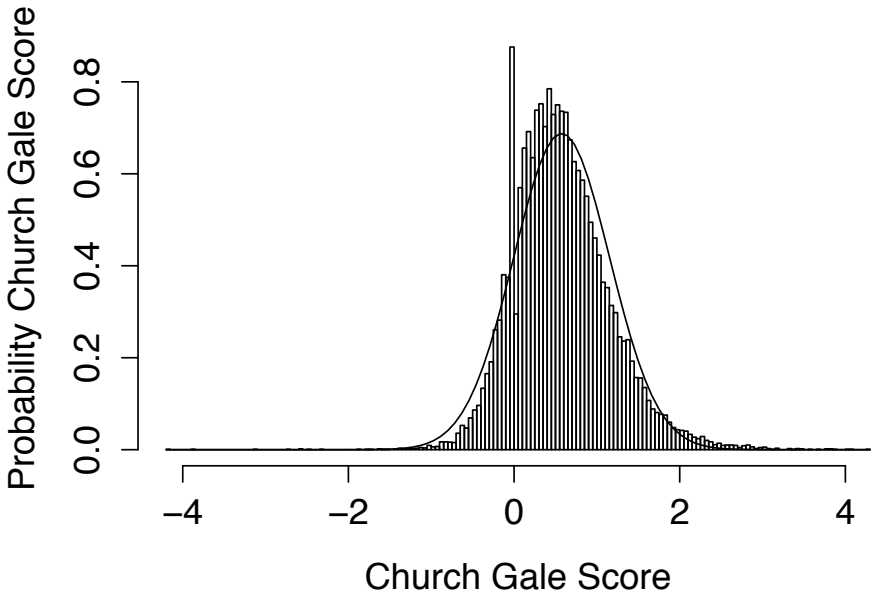

Figure 3.3 Distribution of Church Gale score in the MateCat set relative to the normal distribution.

The plot for the Collaborative Set has the distribution of scores concentrated in the centre as can be seen in figure 3.4. Like before, in figure 3.5 we add a normal curve to the 4th histogram. The relative frequency of the scores away from the centre is much lower than the scores in the centre. Therefore to get a better view of the distribution the y-axis is reduced to the interval 0...0.1. For the Collaborative set the Church Gale score varies in the interval −131.51 ...60.15.

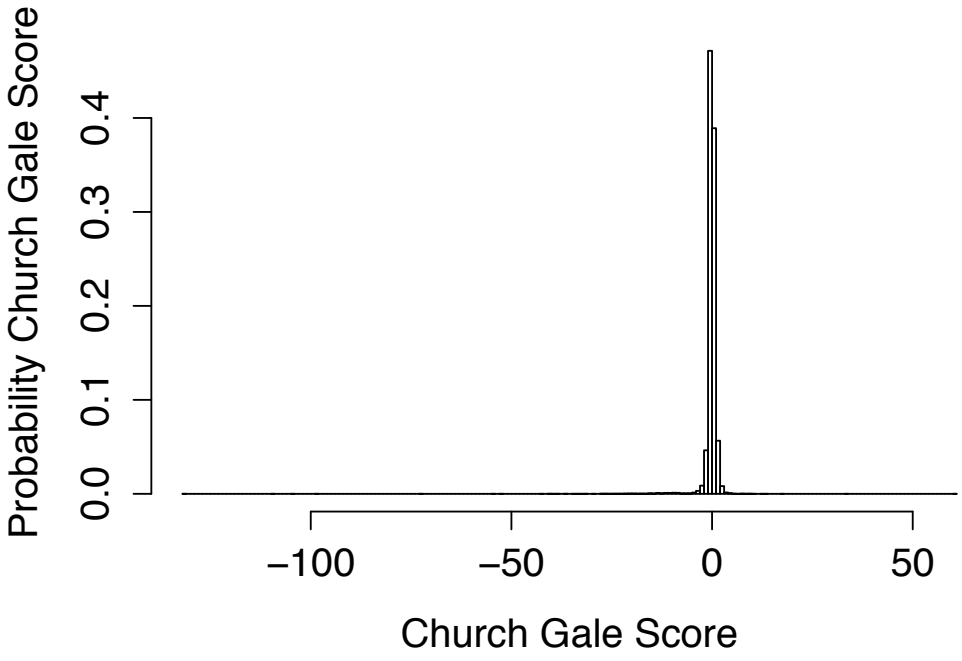

Figure 3.4 Distribution of Church Gale score in the Collaborative set.

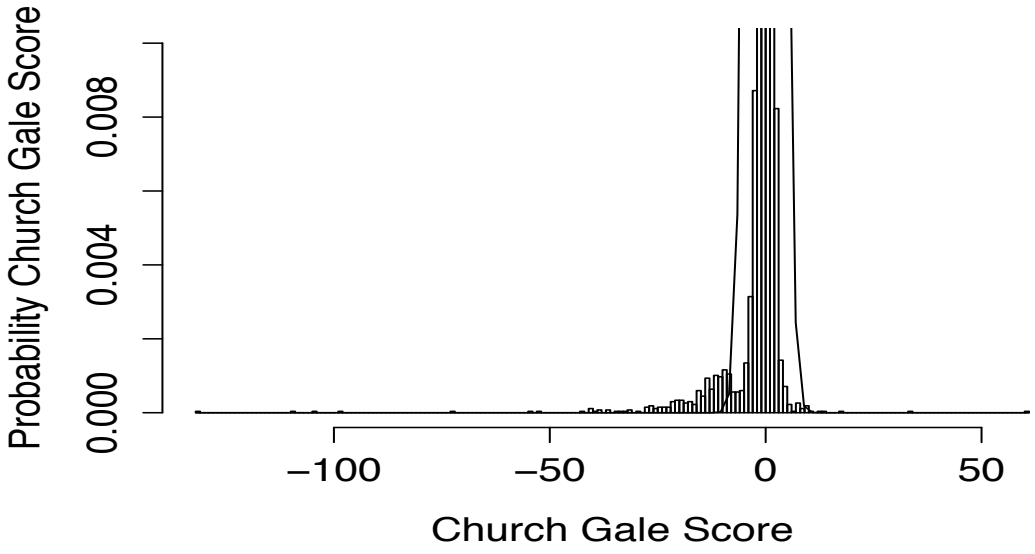

Figure 3.5 Distribution of Church Gale score in the Collaborative set relative to the normal distribution.

From the last four figures it should be already intuitively obvious that the distribution of Church Gale score in the MateCat set is closer to the normal distribution than the distribution of Church Gale score in the Collaborative Set. To justify this intuition we have plotted these distributions against the normal distribution using the Quantile to Quantile plot in figures 3.6 and 3.7. The Quantile to Quantile plot or Q-Q plot as is also known is used in statistics to compare two distributions by plotting their quantiles against each other.

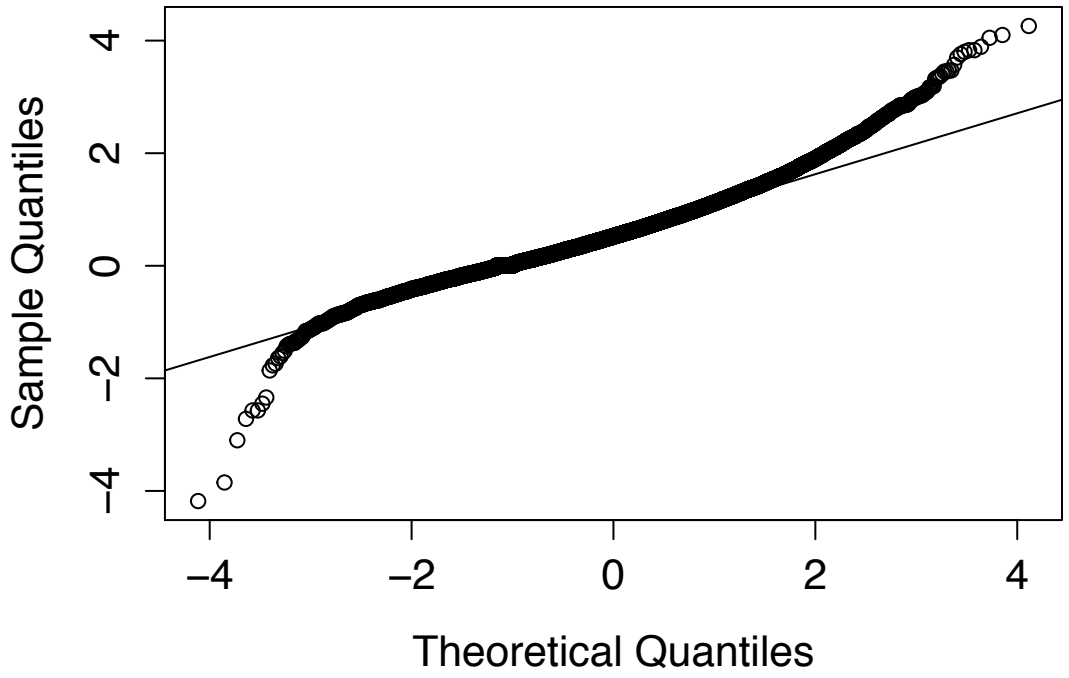

Figure 3.6. The Q-Q plot for the Matecat set

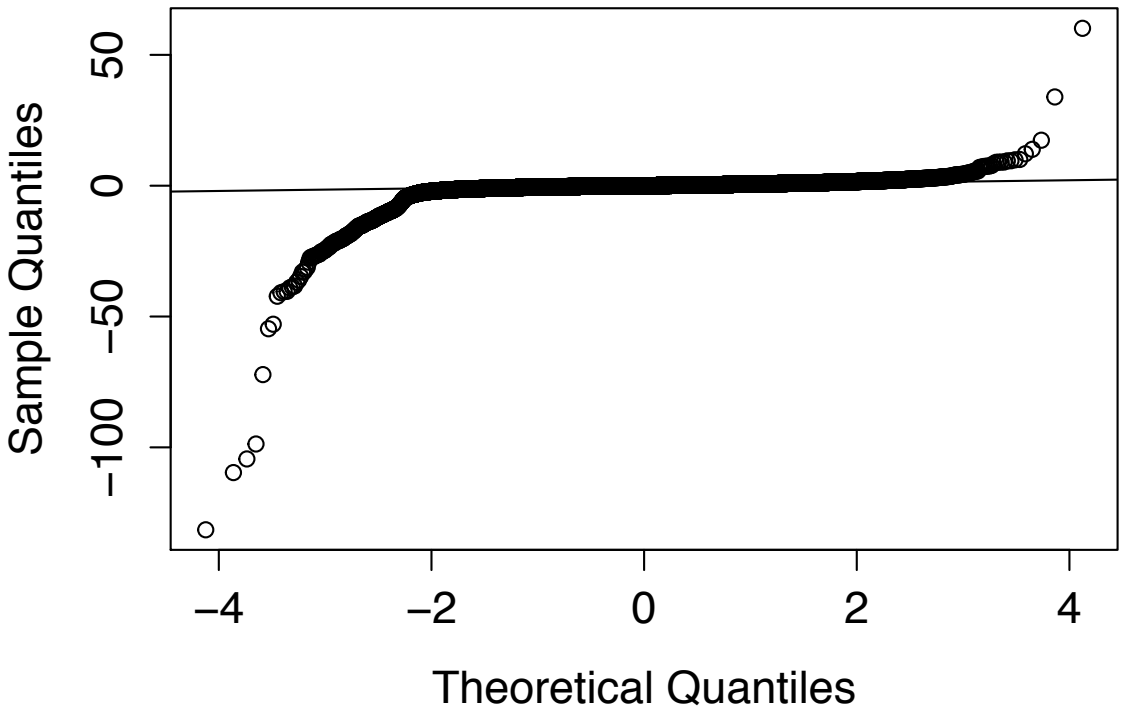

Figure 3.7. Q-Q plot for the Matecat set.

In the Collaborative Set the scores that have a low probability could be a source of errors. Of course it is not always true that low probability score signals an error, otherwise finding a class of faulty segments errors would be trivial. A case when a low probability score does not show and error is when the acronyms present in the source language are expanded in the target language.

Given the above considerations we first draw random bisegments from the Matecat Set. As said before the bisegments in the Matecat Set contain mainly positive examples. Second, we draw random bisegments from the Collaborative Set biasing the sampling to the bi-segments that have scores away from the centre of the distribution. In this way we hope that we draw enough negative segments. After manually validating the examples we created training set and a test set where the proportion of the negative examples is approximately 30%:

- **Training Set**. It contains 1243 bi-segments and has 373 negative examples.
- **Test Set**. It contains 309 bi-segments and has 87 negatives examples.

III. Feature Identification and Classification algorithms

In this section we discuss the features computed for the training and the test sets. There are two ways these features can be used: either a manual classification algorithm is devised based on the data examination or a machine learning approach is taken. Because the data in MyMemory is diverse and comes from many sources we favour the machine learning approach. Therefore this sections ends with a brief presentation of the algorithms used for classification and the rationale for using them. But first let's enumerate the features:

- *same*. This feature takes two values: 0 and 1. It has value 1 if the source and target segments are equal. There are cases specifically in the collaborative part of MyMemory when the source segment is copied in the target segment. Of course there are perfectly legitimate cases when the source and target segments are the same (e.g. when the source segment is a name entity that has the same form in the target language), but many times the value 1 indicates a spam attempt.

- *cg score*. This feature is the Church-Gale score described in the equation 1. This score reflects the idea that the length of the source and destination segments that are true translations is correlated. We expect that the classifiers learn the threshold that separates the positive and negative examples. However, relying exclusively on the Church-Gale score is tricky because there are cases when a high Church Gale score is perfectly legitimate. As said in the previous section this happens when the acronyms in the source language are expanded in the target language.

- *has url*. The value of the feature is 1 if the source or target segments contain an URL address, otherwise is 0.

- *has tag*. The value of the feature is 1 if the source or target segments contain a tag, otherwise is 0.

- *has email*. The value of the feature is 1 if the source or target segments contain an email address, otherwise is 0.

- *has number*. The value of the feature is 1 if the source or target segments contain a number, otherwise is 0.

- *has capital letters*. The value of the feature is 1 if the source or target segments contain words that have at least a capital letter, other- wise is 0.

- *has words capital letters*. The value of the feature is 1 if the source or target segments contain words that consist completely of capital letters, otherwise is 0. Unlike the previous feature, this one activates only when there exists whole words in capital letters.

- *punctuation similarity*. The value of this feature is the cosine similarity between the source and destination segments punctuation vectors. The intuition behind this feature is that source and target segments should have similar punctuation vectors if the source segment and the target segment are true translations.

- *tag similarity*. The value of this feature is the cosine similarity between the source segment and destination segment tag vectors. The reason for introducing this feature is that the source and target segments should contain very similar tag vectors if they are true translations. This feature combines with has tag to exhaust all possibilities (e.g., the tag exists/ does not exist and if it exists is present/is not present in the source and the target segments)

- *email similarity*. The value of the feature is the cosine similarity between the source segment and destination segment email vectors. The reasoning for introducing this feature is the same as for the feature tag similarity. This feature combines with the feature

has email to exhaust all possibilities.

- *url similarity*. The value of the feature is the cosine similarity between the source segment and destination segment url addresses vectors. The reasoning for introducing this feature is the same as for the feature tag similarity.

- *number similarity*. The value of the feature is the cosine similarity between the source segment and destination segment number vectors. The reasoning for introducing this feature is the same as for the feature tag similarity.

- *bisegment similarity*. The value of the feature is the cosine similarity between the destination segment and the source segment translation in the destination language. It formalizes the idea that if the target segment is a true translation of the source segment then a machine translation of the source segment should be similar to the target segment.

- *capital letters word difference*. The value of the feature is the ratio between the difference of the number of words containing at least a capital letter in the source segment and the target segment and the sum of the capital let- ter words in the bi-segment. It is complementary to the feature has capital letters.

- o*nly capletters dif*. The value of the feature is the ratio between the difference of the number of words containing only capital letters in the source segment and the target segments and the sum of the only capital letter words in the bi-segment. It is complementary to the feature has words capital letters.

- *lang dif*. The value of the feature is calculated from the language codes declared in the segment and the language codes detected by a language detector. For example, if we expect the source segment language code to be ”en” and the target segment language code to be ”it” and the language detector detects ”en” and ”it”, then the value of the feature is 0 (“en”- “en”, ”it”-“it”). If instead the language detector detects ”en” and ”fr”

then the value of the feature is 1 (“en”-“en”, ”it”-“fr”) and if it detects ”de” and ”fr” (“en”-“de”,”it”-“fr”) then the value is 2.

All feature values are normalized between 0 and 1. The most important features are bisegment similarity and language difference. The other features are either sparse (e.g. relatively few bi-segments contain URLs, emails or tags) or they do not describe the translation process very accurately. For example, we assumed that the punctuation in the source and target segments should be similar, which is true for many bi-segments. However, there are also many bisegments where the translation of the source segment in the target language lacks punctuation.

The translation of the source English segment to Italian is performed with the Bing API. The computation of the language codes for the bisegments is done with the highly accurate language detector Cybozu[11] .

We have tried the most important classification algorithms. The first one is one of the oldest classification algorithms: **Decision Tree**. Even if they are known to overfit the training data they have the advantage that the rules inferred are readable by humans. This means that we can tamper with the automatically inferred rules and at least theoretically create a better classifier. We have also tried a more robust version of Decision Tree. The **Random Forests** are ensemble classifiers that consist of multiple decision trees and have a lower probability to overfit the data than the classical decision tree. The final prediction is the mode of individual tree predictions.

The **Logistic Regression** works particularly well when the features are linearly separable. In addition, the classifier is robust to noise, avoids overfitting and its output can be interpreted as probability scores. Of course we do not know a priori if the features are linearly separable and this is the reason the algorithms like logistic regression should be tested. Another classification algorithm we have tried from the same class with Logistic Regression is **Support Vector Machines**. If the conditional independence that the naive Bayes class of algorithm postulates holds, the training converges faster than logistic regression and therefore an algorithm like **Gaussian Naive Bayes** needs less training instances. This size of the training set is particularly important when the annotation process is expensive. Finally we have tested a non-parametric algorithm: the **K- Nearst Neighbors**. This algorithm classifies a new instance based on the

11 https://github.com/shuyo/language- detection/blob/wiki/ProjectHome.md

distance it has to k training instances and its prediction output is the label that classifies the majority of training instances. The algorithm gives good results in classification problems where the decision boundary is irregular.

IV. Results

The machine learning algorithms have been evaluated in two ways. The first evaluation is a three-fold stratified classification on the training set. All classification algorithms are evaluated against two baselines. The first baseline, called Baseline Uniform, generates predictions randomly. The second baseline, called Baseline Stratified, generates predictions respecting the training set class distribution. The results of the three-fold stratified evaluation are given in the table 3.1:

Algorithm	**Precision**	**Recall**	**F1**
Random Forest	0.95	0.97	0.96
Decision Tree	0.98	0.97	0.97
SVM	0.94	0.98	0.96
K-Nearst Neighbors	0.94	0.98	0.96
Logistic Regression	0.92	0.98	0.95
Gaussian Naïve Bayes	0.86	0.96	0.91
Baseline Uniform	0.69	0.53	0.60
Baseline Stratified	0.70	0.73	0.71

Table 3.1: The results of the three-fold stratified classification

With the exception of the Gaussian Naive Bayes classifier all other algorithms have excellent results. All other algorithms beat the baselines by a significant margin: at least 20 points.

The second evaluation is performed against the test set. The baselines are the same as in three-fold evaluation above and the results are in table 2.

Algorithm	Precision	Recall	F1
Random Forest	0.85	0.63	0.72
Decision Tree	0.82	0.69	0.75
SVM	0.82	0.81	0.81
K-Nearst Neighbors	0.83	0.66	0.74
Logistic Regression	0.80	0.80	0.80
Gaussian Naïve Bayes	0.76	0.61	0.68
Baseline Uniform	0.71	0.72	0.71
Baseline Stratified	0.70	0.51	0.59

Table 3.2: The results of the classification performed on the test set

The results for the second evaluation are worse than the results for the first evaluation. For example, the difference between the F1-scores of the best performing algorithm: SVM and the stratified baseline is of 10%: twice lower than the difference between the best performing classification algorithm and the same baseline for the first evaluation. This fact might be explained partially by the great variety of the bi-segments in the Matecat and Web Sets. Obviously this variety is not fully captured by the training set.

Unlike in the first evaluation, in the second one we have two clear winners: Support Vector Machines (with the linear kernel) and Logistic Regression. They produce F1-scores around 0.8. The results might seem impressive, but they are insufficient for automatically cleaning MyMemory. To understand why this is the case we inspect the results of the confusion table for the SVM algorithm. From the 309 examples in the test set 175 are true positives, 42 false positives, 32 false negatives and 60 true negatives. This means that around 10% of all examples corresponding to the false negatives will be thrown away. Applying this method to the MyMemory database would result in the elimination of many good bi-segments. We should therefore search for better methods of cleaning where the precision is increased even if the recall drops.

There are two potential solutions to this problem. The first solution is to improve the performance of the classifiers. In the future we will study ensemble classifiers that can boost the performance of the classification task. The idea behind the ensemble classifiers is that with differently behaving classifiers one classifier can compensate for the errors of other classifiers. If this solution does not give the expected results we will focus on a subset of bi-segments for

which the classification precision is more than 90%. For example, the Logistic Regression classification output can be interpreted as probability. Our hope is that the probabilities scores can be ranked and that higher scores correlate with the confidence that a bi-segment is positive or negative.

Another improvement will be the substitution of the machine translation module with a simpler translation system based on bilingual dictionaries. The machine translation module works well with an average numbers of bi-segments. For example, the machine translation system we employ can handle 40000 bi-segments per day. However, this system is not scalable, it costs too much and it cannot handle the entire MyMemory database. Unlike a machine translation system, a dictionary is relatively easy to build using an aligner. Moreover, a system based on an indexed bilingual dictionary should be much faster than a machine translation system. We have already compiled an English-Italian dictionary that has around 1 million entries. We have implemented an algorithm based on the well-known Aho-Corasick (Aho and Corasick, 1975) algorithm that translates from English to Italian using the 1 million entries dictionary. However for now the results are not encouraging because the dictionary does not have a way to model the word ambiguity.

V. Cleaning Europarl

Europarl (Koehn, 2005) is a multilingual corpus (in 21 languages) extracted from the proceedings of the European Parliament. The corpus is offered as a set of documents containing the source language lines and the target language lines such as the line "*n*" in document x corresponds to the line "*n*" in document y (where x and y are parallel documents).

We have inspected a sample of bi-segments extracted from the parallel corpus and we have noticed that the most frequent error verifies when the source language segments (sometimes) or destination language segments (more often than not) do not match the language of the document. For example in the Romanian side of the English-Romanian parallel corpus we can find segments in Spanish, Italian etc.

The algorithm that cleans Europarl identifies such segments and eliminates them from the corpus. The natural way to proceed is to identify the actual language of the segments and

compare it with the expected language code. In fact our classifier incorporate this decision and a meta-rule:

- First the language detector identifies the language of the source and target segments. For this task we use Cybozu. Cybozu outputs a distribution of probabilities over the language codes. To restrict the set of possible language identified we initialize the language detector with the language profiles corresponding to the 21 European languages part of Europarl. In this way we avoid spurious confusions between closely related languages that are not among languages in Europarl (for example Bulgarian which is in Europarl and Russian which is not). Moreover we use the language profiles trained on short fragments because many segments in Europarl have few words.
- If one of the language codes returned by Cybozu is identical with the expected language code then we assign this language code to the segment. Otherwise the language code assigned is the most probable language code in the probability distribution returned by Cybozu.

The results of cleaning are given in the following table where NSC stays for Number of Segments after the cleaning and NS stays for the initial number of segments.

Language Pair	**NSC/NS**
English-Estonian	643248/651746
English-German	1903768/1920209
English-Czech	641298/646605
English-Latvian	633931/637599
English-Lithuanian	628461/635146
English-Romanian	395685/399375
English-Slovene	616117/623490

English-Polish	625298/632565
English-Slovak	634935/640715
English-Hungarian	619163/624934
English-Bulgarian	403431/406934

Table 3.3 Language Pairs in Europarl that need cleaning

We notice that 11 language pairs from the 21 language pairs in Europarl have been identify to contain segments in other languages. The cleaned corpus has been offered to the Europarl team and has been uploaded to the MyMemory database.

4. Conclusions

In this deliverable we have presented the procedure for acquiring a large multilingual corpus from the web. Up-to-date we have collected approximately 400 millions parallel words. We have also presented a machine learning procedure to clean translation memories. In the future we will continue crawling the web and we will integrate the results of mapping the parallel web documents found using the heuristics presented in section 2.V.

In the end, we intend to deliver a database containing parallel documents crawled and pending legal issues the translation memories (TMX files) obtained from the parallel documents.

References

Aho, Alfred V., and Corasick, Margaret J. (June 1975). "Efficient string matching: An aid to bibliographic search". *Communications of the ACM*

B. L. Humphreys and D. A. Lindberg. 1993. The UMLS project: making the conceptual connection between users and the information they need. Bull Med Libr Assoc, 81(2): 170–177, April.

D. Varga, L. Németh, P. Halácsy, A. Kornai, V. Trón, V. Nagy (2005). Parallel corpora for medium density languages. *In Proceedings of the RANLP 2005*, pages 590-596

Eduard Barbu. (2015) "Spotting false translation segments in translation memories". In the *Proceedings of the 1st International Workshop on Natural Language Processing for Translation Memories (NLPTM)*, pages 9-16. September 2015, Hissar, Bulgaria.

Esplà-Gomis, M. (2009), Bitextor: a Free/Open-source Software to Harvest Translation Memories from Multilingual Websites, *in* 'Proceedings of MT Summit XII'.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Pretten- Hofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12: 2825–2830.

Fabienne Braune, Alexander Fraser, Improved unsupervised sentence alignment for symmetrical and asymmetrical parallel corpora, Proceedings of the 23rd International Conference on Computational Linguistics: Posters, p.81-89, August 23-27, 2010, Beijing, China

Jassem, K. and Lipski, J. (2008). A new tool for the bilingual text aligning at the sentence level. *Intelligent Information Systems*, pages 279–286.

Jorg Tiedemann. 2011. Bitext Alignment. Number 14 in Synthesis Lectures on Human Language Technologies. Morgan & Claypool, San Rafael, CA, USA.

Marco Trombetti. 2009. Creating the world's largest translation memory.

Papavassiliou, Vassilis, Prokopidis, Prokopis, and Thurmair, Gregor. (2013). A modular open-source focused crawler for mining monolingual and bilingual corpora from the web. In *Proceedings of the Sixth Workshop on Building and Using Comparable Corpora*, pages 43–51, Sofia, Bulgaria. Association for Computational Linguistics.

Philipp Koehn. 2005. A parallel corpus for statistical machine translation. In Proceedings of MT-Summit.

Philipp Koehn and Jean Senellart. 2010. Convergence of translation memory and statistical machine translation. In Proceedings of AMTA Workshop on MT Research and the Translation Industry, pages 21–31.

Raivis Skadins, Jorg Tiedemann, Roberts Rozis, and Daiga Deksne. 2014. Billions of parallel words for free: Building and using the EU bookshop corpus. In *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC-2014)*, Reykjavik, Iceland. European Language Resources Association (ELRA).

Ralf Steinberger, Andreas Eisele, Szymon Klocek, Spyridon Pilos, and Patrick Schluter. 2013. Dgt-tm: A freely available translation memory in 22 languages. CoRR, abs/1309.5226.
Resnik, Philip and Smith, Noah A. (2003). The Web as a parallel corpus. *Computational Linguistics*, 29(3): 349–380.

Ruopp, A., Xia, F.: Finding parallel texts on the web using cross-language information retrieval. In: Proceedings of the Second International Workshop On Cross Lingual Information Access Addressing the Information Need of Multilingual Societies, pp. 18–25 (2008)

Uwe Reinke. 2013. State of the art in translation memory technology. Translation: Computation, Corpora, Cognition, 3(1).

Ventsislav Zhechev and Josef van Genabith. 2010. Maximizing tm performance through sub-tree alignment and smt. In Proceedings of the Ninth Conference of the Association for Machine Translation in the Americas.

William A. Gale and Kenneth W. Church. 1993. A program for aligning sentences in bilingual corpora. COMPUTATIONAL LINGUISTICS.
Zhifei Li and Sanjeev Khudanpur. 2009. Large-scale discriminative n-gram language models for statistical machine translation. In Proceedings of AMTA.