DELIVERABLE

Project Acronym: FLAVIUS

Grant Agreement number: ICT-PSP-250528

Project Title: Foreign LAnguage Versions of Internet and User generated Sites

---

D4.4 Translation Memory Module able to provide translation

Revision: 1.0

Authors:

Antoine Sauzay (Softissimo)

Christophe Brun-Franc (Softissimo)

Project co-funded by the European Commission within the ICT Policy Support Programme		
Dissemination Level		
C	Confidential, only for members of the consortium and the Commission Services	

Revision History

Revision	Date	Author	Organization	Description
0.1	July, 4th	Christophe Brun-Franc	Softissimo	Draft version
0.2	July, 20th	Antoine Sauzay	Softissimo	Updated Draft version
0.3	August, 17th	Théo Hoffenberg	Softissimo	Revised version
0.4	August, 23th	Antoine Sauzay	Softissimo	Updated version
0.5	Sept, 8th	Joël Benchitrit	Softissimo	Revised version
0.6	Sept, 14th	Constantin Walter	Across	Revised version
1.0	Sept, 14th	Antoine Sauzay	Softissimo	Official version

Statement of originality:

This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

Contents

1. Introduction 5
   1.1 Related Documents 5
   1.2 Glossary 5
2. Translation Memory description 7
   2.1 Translation memory main concepts 7
   2.2 Parameters needed to call Across from Flavius 8
   2.3 Relation attribute management 8
   2.4 Creation of relation attribute 9
   2.5 User-attribute 9
   2.6 Handling numbers in the ACROSS TM 9
   2.7 Handling HTML tags in the ACROSS TM 11
3. Translation memory in the Flavius workflow 12
   3.1 Working scope 12
   3.2 Preparation of the data 12
   3.3 Searching for matching translation in the ACROSS TM 13
   3.4 Receiving the result from the TM 15
   3.5 Saving the translation 16
4. Implementation in the current version 17
   4.1 FLAVIUS user interface to import a TMX file 17
   4.2 Import of TMX and ACROSS TM configuration 19
   4.3 Managing translation Memories through FLAVIUS 21
5. Remaining questions for the future version 25
   5.1 Import partial TMX file 25
   5.2 General memory for all users 25

5.3 Sentences in place of Segments 26
5.4 Adding ACROSS TM entries through post-editing 27
6. Appendix: Technical aspects 28
6.1 Connecting the ACROSS TM 28
6.2 Implementation through an interface 28
6.3 The ITranslationMemory Interface 30
6.4 Performance Tests 31
6.4.1 Context and methodology 31
6.4.2 Results 31
6.4.3 Conclusion 33

1. Introduction

This document describes the implementation of the Translation Memory in the FLAVIUS platform.

The aim of the translation memory feature in Flavius is to enhance the quality of the automatic translation through the identification of text segments that have been previously translated by the user.

1.1 Related Documents

This document uses the following documents:

- The Flavius deliverable D4.3 released by Across.
- The API documentation released by Across.

1.2 Glossary

This section describes the different specific terms that will be used in the following document.

Translation unit: A translation unit is an Across TM object containing the source text and its translation. A translation unit is characterized by its source language, its target language, and a set of relation attributes.

Relation attribute: In the ACROSS system, a relation is an attribute attached to a translation unit, binding additional information to this translation unit, like the creator name of this unit, or a description tag for a domain like "bank". Each unit can have more than one relation. The addition of relations for each translation unit allows the system to propose a filtering system in addition to the text search engine.

Search example

We suppose that we have a translation memory and a user named Translator1. If this user searches in the translation memory for the source text: "Enter your PIN code". The system will only return:

- an answer if this user has created this entry in the memory
- nothing if the entry does not exist
- nothing if the entry exists without the relation " Translator1" (saying Translator1 is not the owner of this entry)

2. Translation Memory description

2.1 Translation memory main concepts

A translation memory like ACROSS TM is a data set containing translations units. This data set helps a user or an automatic system (we will call them "user") to remember what was the translation specifically validated by a user for a specific source text.

Each time a user creates a translation that could be reused, he inserts the new translation unit in the Translation memory. The system adds relation attributes to this entry to segregate information in the TM between users.

We must take care about the relation creation process as it impacts different points:

- the retrieved result: two users can have two different translations for the same source
- the privacy security (some information could be confidential in the memory)
- the search performance

A translation memory can be used and accessed in various ways:

- as a basis for automatic pre-translation of sentences that are 100% matches
- as a basis for the translator in the case of a partial match with the sentence
- as a basis for a concordance search (all the sentences containing this expression)

A translation memory can be filled with two manners:

- The integration of source text and their translation post-edited after translation with SMT.
- The import of texts with their translation performed manually.

As the FLAVIUS process must be as automatic as possible, the system will use the first functionality of the translation memory (100% matches). The 100% matches will include variants as seen in the following paragraphs (2.6 and 2.7)

2.2 Parameters needed to call Across from Flavius

The Across TM component is based on a client / server communication. The client part needs the following parameters to call the server:

- The unique identifier of the ACROSS TM (for instance the ACROSS Server unique identifier - GUID -)
- The credential information (login and password of the account used to open a connection to the Translation Memory server).
- The confidence threshold used to ensure that a translation provided by the Translation Memory server is "good enough".

For information, these data are stored in the FLAVIUS platform as General parameters and are identical for all the Flavius users. They are accessible through the Administration interface of the platform.

2.3 Relation attribute management

The entries stored in the ACROSS TM can be confidential or specific to a user. The entries cannot always be shared with everyone.

In the ACROSS TM, this issue is solved like this:

- A relation attribute is added to each translation unit inserted from the FLAVIUS platform. This relation attribute is set to a unique identifier link to the owner of the entry (Flavius user account unique GUID).
- Each translation unit retrieves from the Translation Memory through the FLAVIUS platform will be filtered using this relation attribute (The search function will only return the entries corresponding to your ID)

2.4 Creation of relation attribute

A relation attribute (a GUID) has to be created in the TM server before being used in a search request or in an insertion (segment by segment or through a TMX file import). The Flavius platform will call the ACROSS method GetorCreateRelationAttribute each time it is needed to ensure the relation attribute exists in the server list before using it.

This method creates a GUID if nothing exists corresponding to the given name and only return the existing one if the name already exists.

2.5 User-attribute

Each entry of the ACROSS TM can also be tagged with a user-attribute. User-attributes are custom attributes that must be first created in ACROSS TM (with their possible values) and allow to add descriptive data to an entry.

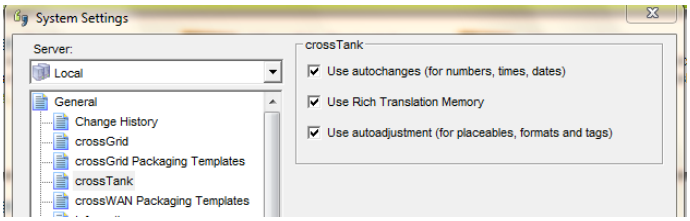
In Flavius, a user-attribute "origin" will be created to store the origin of the entry:

- the name of the TMX file used to import the entry
- "post-editing" when the entry is added from the post-editing interface.

This user-attribute will be used to manage ACROSS TM from the FLAVIUS platform.

2.6 Handling numbers in the ACROSS TM

The ACROSS TM can be configured to automatically replace numbers in the matching translation (see picture below).



Autochange configuration

In this case, if a matching segment is found without the correct numbers, the matching translation will be retrieved and the numbers will be replace with the source text values.

En 2009, il a littéralement tiré vers le haut la croissance et la rentabilité du Groupe, représentant près de 20 % du chiffre d'affaires et 26 % du résultat brut d'exploitation.	In 2009, our growth and profitability were both pulled sharply higher by operations outside France, which accounted for nearly 20% of premium income and 26% of EBIT.
En 2009, il a littéralement tiré vers le haut la croissance et la rentabilité du Groupe, représentant près de 18 % du chiffre d'affaires et 26 % du résultat brut d'exploitation.	In 2009, our growth and profitability were both pulled sharply higher by operations outside France, which accounted for nearly 18% of premium income and 26% of EBIT.
En 2009, il a littéralement tiré vers le haut la croissance et la rentabilité du Groupe, représentant près de 18 % du chiffre d'affaires et 30 % du résultat brut d'exploitation.	In 2009, our growth and profitability were both pulled sharply higher by operations outside France, which accounted for nearly 18% of premium income and 30% of EBIT.
En 2010, il a littéralement tiré vers le haut la croissance et la rentabilité du Groupe, représentant près de 18 % du chiffre d'affaires et 30 % du résultat brut d'exploitation.	In 2010, our growth and profitability were both pulled sharply higher by operations outside France, which accounted for nearly 18% of premium income and 30% of EBIT.

Number sample

Project co-funded by the European Commission within the ICT Policy Support Programme		
Dissemination Level		
C	Confidential, only for members of the consortium and the Commission Services	

2.7 Handling HTML tags in the ACROSS TM

The entries added to the ACROSS TM can contain HTML tag. This must be taken into account when searching matching translation in the ACROSS TM.

The FLAVIUS platform is able to process segment texts that contain HTML layout (such as BOLD, LINK…) but the formatting can have an impact on the search as the ACROSS TM can apply a penalty to the confidence level when a different HTML tag is detected in the text segment.

- If you apply the penalty, the TM will only return the entries that have exactly the same tags. In this case you have a perfect match but you will only match perfect formatting.
- If you do not apply the penalty, the TM will return additional matches, but with a different formatting.

We decided at this stage not to apply the penalty for formatting in order to provide additional matches. Indeed the objective is to provide a better translation through the use of the TM.

3. Translation memory in the Flavius workflow

This chapter describes the implementation of the translation memory module in the FLAVIUS platform. At this stage the translation memory allows to translate a source text.

This translation process is performed in 4 steps:

- Preparation of the data
- Searching for matching translation in the ACROSS TM
- Receiving the result from the TM
- Saving the translation

3.1 Working scope

In this stage of the Flavius platform progress, the memory will be only used in the XML file scenario. For the moment, the HTML pages are processed in one call and are not segmented. This will be automatically fixed when the parsing module will be enriched with a HTML parser.

3.2 Preparation of the data

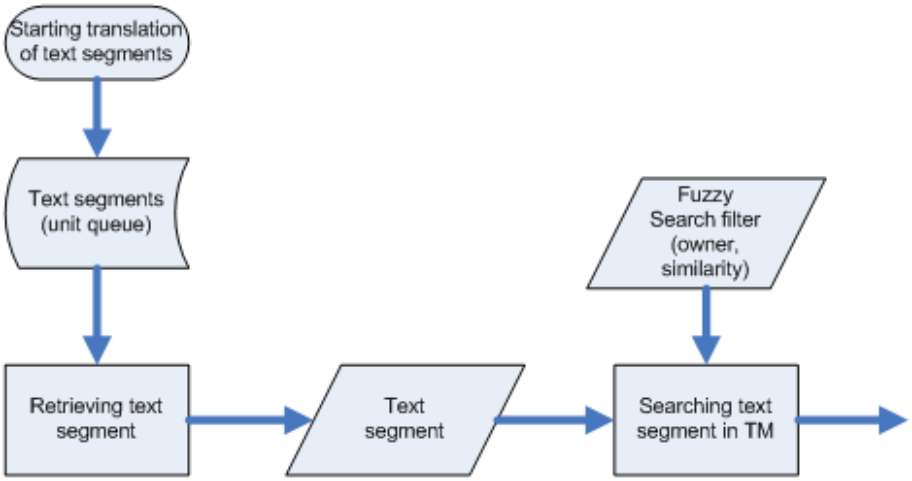
Before the translation phase (translation memory and statistical machine translation), the source text (for now, the XML) that is provided to the FLAVIUS platform has to be processed.

The aim of this processing is to extract from the source text (that can be composed of several segments) the text segments that must be translated. This process is described in parsing phase in the architecture document.

3.3 Searching for matching translation in the ACROSS TM

The FLAVIUS platform performs a search on each text segment (using the FuzzySearch method of the FLAVIUS Interface with a confidence level equal to 100% and possibly variants on numbers) to retrieve the matching translations.

This process is describes in the following diagram:



Searching for matching translation

The search call is configured with a confidence threshold and a filter. This filter is composed of one or more attribute.

The ACROSS TM responds to the search with one or more segment unit that matches the following criteria:

- The level of confidence of the translation is higher than the confidence threshold (100% in our case)
- The relation attributes of the segment unit must match the relation attributes of the filter.

Project co-funded by the European Commission within the ICT Policy Support Programme		
Dissemination Level		
C	Confidential, only for members of the consortium and the Commission Services	

If there is more than one relation attribute, the operator OR is applied. It means that, for instance, if the relation attributes of the filter are set to user1 and user2, both segment units with the relation attribute set to user1 or/and user2 are retrieved from the ACROSS TM.

As the translation memory is used in an automatic mode, the threshold is set to the maximum value. We will only have 100% matches (with potential differences on numbers, see chapter 2.6)

Error handling

There are three types of error that can occur when searching text segment in the ACROSS TM:

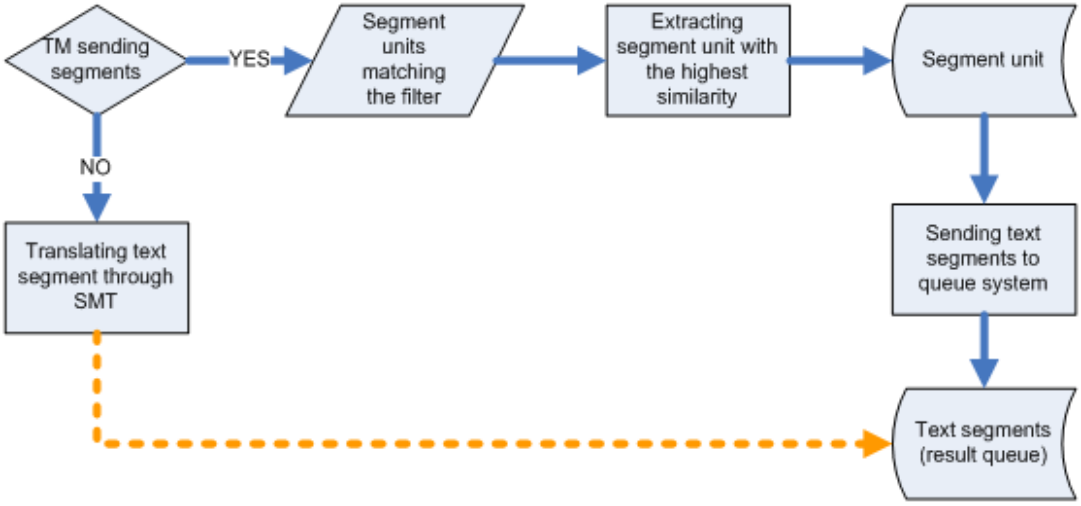
- The ACROSS TM is not responding
- The ACROSS TM is responding but the FuzzySearch method returns an error code
- The ACROSS TM is responding but the FuzzySearch method throws an exception

These cases are handled by the FLAVIUS platform. In one of these errors happens, the process is not aborted. The FLAVIUS platform sends this text segment directly to the SMT.

3.4 Receiving the result from the TM

Then, the FLAVIUS platform analyses the result to get the TM translation of text segment.

This process is describes in the following schema:



Retrieving the best translation

Three cases can occur:

- There is only one proposal:
  →This translation is kept and the translation of the text segment through the SMT engine is not performed.

- There is more than one proposal:
  →The one with the highest level of confidence is kept and the translation of the text segment through the SMT engine is not performed.

- There is no proposal: →The text segment is translated through the SMT engine.

3.5 Saving the translation

The translation is performed in the FLAVIUS platform either by the ACROSS TM or the SMT engine.

As it is relevant to identify the origin of the translation (SMT engine or ACROSS TM) both for statistical reasons and for post-editing, this information will be kept.

This information will be saved as an attribute in the XML pivot file.

- SMT: the translation was performed by the SMT engine
- TM: the translation was performed by the ACROSS TM.

4. Implementation in the current version

4.1 FLAVIUS user interface to import a TMX file

TMX file is a standard XML file used to import and export translation memories. The Flavius platform will accept the revision 1.3 and 1.4 of the TMX standard.

This import process is performed in 3 steps:

- Providing the TMX file and parameters through the FLAVIUS platform
- Checking the validity of the TMX
- Importing the TMX file using ACROSS TM API

The import of TMX files, only available for authenticated user, is reachable through the button "Import TMX file" available in the FLAVIUS Translation memory page.

Providing data for importing a TMX file

The import of a TMX file requires providing:

- The TMX file to import
- The source and target languages

The import of the TMX file is started when the "Import" button is clicked.

Checking the validity of the TMX

The FLAVIUS platform first checks if the TMX file provided:

- has the extension .TMX
- is a valid TMX file (v1.3 or v1.4)
- does not exceed the maximum size.

If the TMX file does not match one criterion, the FLAVIUS platform refuses the import and requests the user to select another TMX file.

The maximum size of XML file accepted will be configured in the FLAVIUS platform for each role. The user will inherit the maximum size of its role.

Moreover, the FLAVIUS platform compares the length of the text segments included in the TMX file with the minimum and maximum size accepted for text segment. If at least one segment does not match the size limit, the FLAVIUS platform informs the user that the import is rejected.

The maximum and minimum length of text segments will be configured in the FLAVIUS platform for each role. The user will inherit the maximum size of its role.

Importing the TMX

Then the TMX import is started:

- the XML configuration file required by the TMX import method of ACROSS TM is created (see the dedicated chapter Import of TMX)
- the import of TMX is performed (see the dedicated chapter Import of TMX).

Reporting of the TMX import

Once the import of the TMX file is finished, a report is displayed to inform the user about the TMX import.

This report contains:

- the total number of entries
- the number entries added to the TM
- the number entries not added to the TM (target segment empty, length of text segments)

Error handling

As for the translation, there are three types of error that can occur when importing a TMX file in the ACROSS TM:

- The ACROSS TM is not responding
- The ACROSS TM is responding but the import TMX module returns an error code
- The ACROSS TM is responding but the import TMX module throws an exception

These cases are handled by the FLAVIUS platform. In one of these errors happens, a message will be returned to inform the user about the problem.

4.2 Import of TMX and ACROSS TM configuration

Import of TMX through API

The TMX import will be performed using the asynchronous method CrossTankManager.ImportTMX which is provided as part of the CROSS API SI.

This method accepts as parameter:

- The TMX file
- The source and target language
- a XML structure used to configure the behaviour of the TMX import

This XML structure - generated by the FLAVIUS platform for each call to ImportTMX - will be used to set the value of the relation attribute corresponding to the FLAVIUS user who imports

the TMX file into the ACROSS TM (and if the user wants to share it with the community) Thus, this value will be added for each entry added to the ACROSS TM.

Also, the value of the user-attribute "origin" is set to the name of TMX file for each entry added to the ACROSS TM.

Then, each entry added to the ACROSS TM is categorized by the following data:

- The user-attribute "origin"
- The relation attribute

These two data represent the unique identifier of an entry in the ACROSS TM (in the scope of the FLAVIUS platform). If there is more than one entry with the same information, the oldest one will be automatically removed from the ACROSS TM.

Configuration of ACROSS TM

The ACROSS TM will be configured to allow duplicate entries. This configuration will be shared by all the FLAVIUS users.

This will allow having for a same source text segment, different translations provided by different users (each of the translation linked to a specific relation attribute).

If there is more than one translation with the same relation attribute and "origin" user-attribute linked to a same source text, the more recent translation will be retrieved.

4.3 Managing translation Memories through FLAVIUS

The user will be able to manage the entries added to the ACROSS TM:

- get the number of TM entries according their origins (post-editing, specific TMX file)
- delete entries according their origins (post-editing, specific TMX file import)

List of TM entries

This FLAVIUS interface will display to the user the following information for each languages pair:

- The name of the TMX file (or “Post-Editing” when the corresponding module will be in place)
- The date of the TMX file import (or the date of the last entry added through post-editing)
- The total number of entries added
- The number of entries with the tag "Shareable" (future version, see dedicated chapter)
- The number of entries with the tag "General" (future version)

Deleting the entries from the ACROSS TM

The user will be able to delete for a specific language pair all the entries linked to a TMX file import (or all the post-edited entries).

Assuming that a user wants to delete the entries added by the import of TMX file called "myTMX.tmx" for the language pair French / English, he will click on the delete button in the corresponding line.

The entries that will be really deleted in the Across TM are:

- The French/English segment units that have the relation attribute only set to the user tag and the user-attribute "origin" set to "myTMX".

- The French/English segment units that have the relation attribute only set to the user tag, the "shareable" tag and the user-attribute "origin" set to "myTMX" (future version)

- The French/English segment units with the relation attribute set to the user tag and the "general" tag are not deleted, but the user tag is removed. These units will only be part of the General group. (future version)

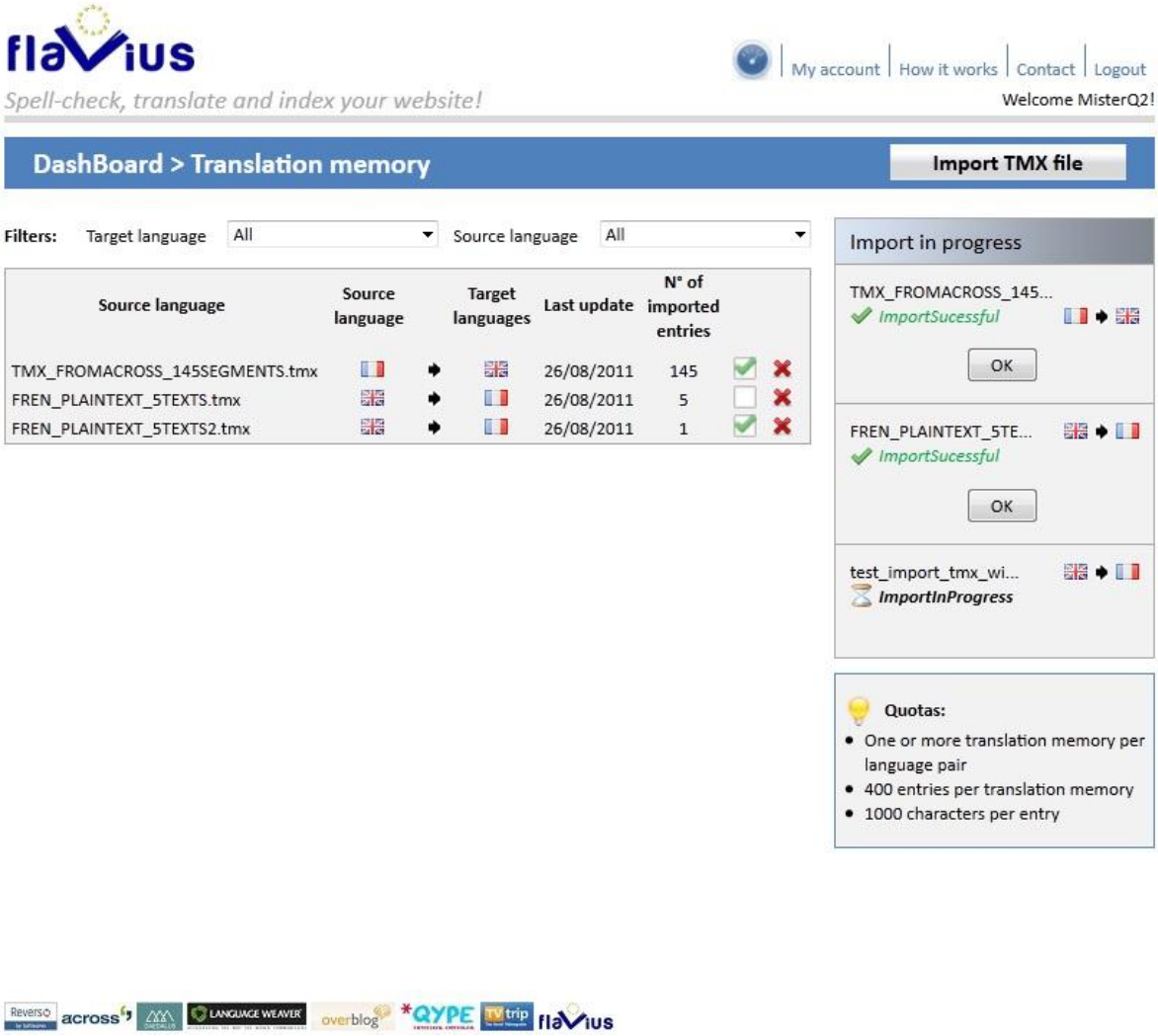
These entries of the ACROSS TM are deleted when the "Delete my entries" button, linked to the TMX file import and the specific languages pair, is clicked.

Once the entries are deleted, a report is displayed to inform the user. This report contains:

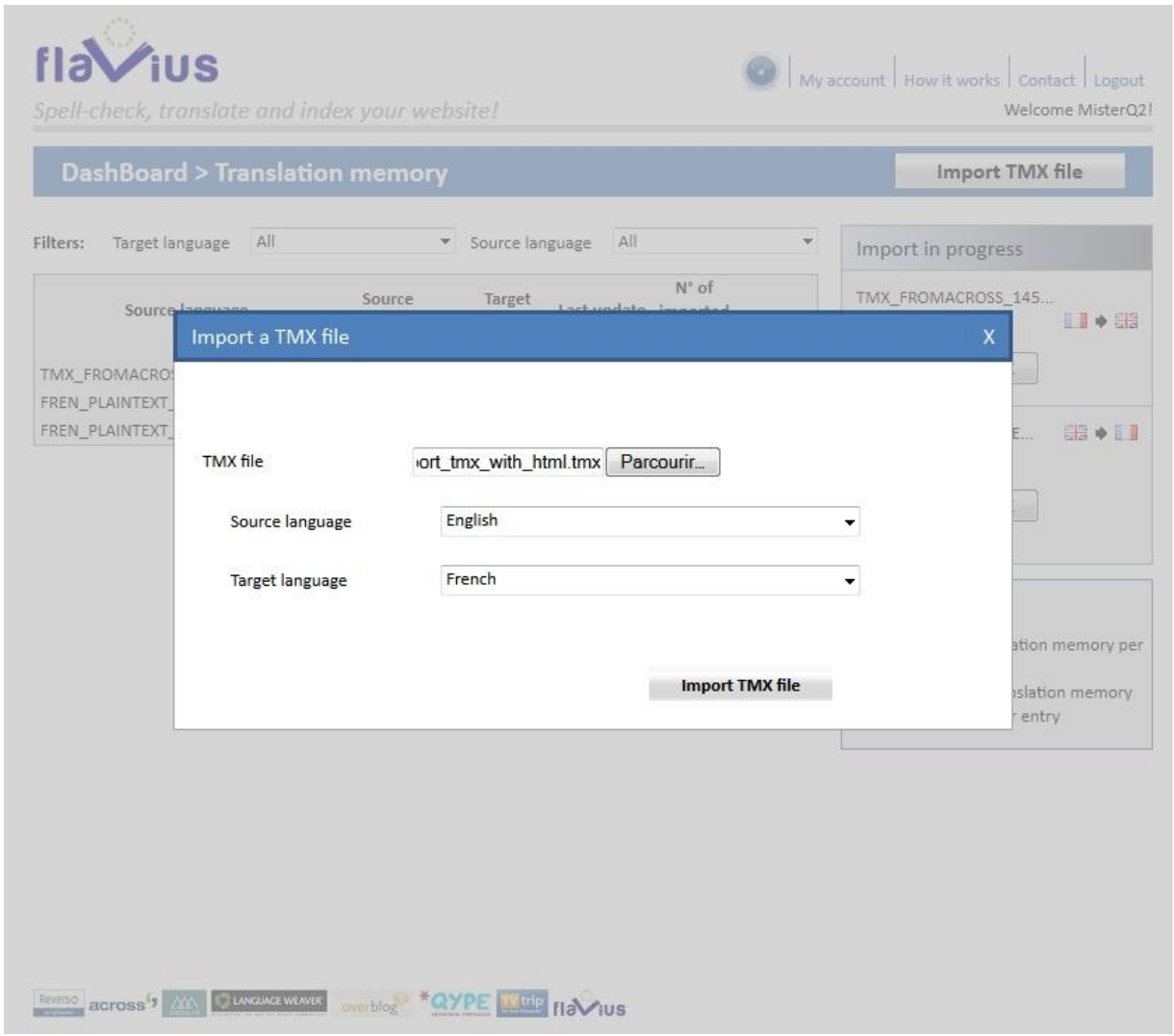
- the number of entries fully deleted
- the number of entries with only its own tag removed
- the number of entries with its own tag and the "shareable" tag removed

The two followings images show a screen shot of the translation memories management interface.

- The first one, when the last import is finished.
- The second one shows when an import is asked



Project co-funded by the European Commission within the ICT Policy Support Programme		
Dissemination Level		
C	Confidential, only for members of the consortium and the Commission Services	



Project co-funded by the European Commission within the ICT Policy Support Programme		
Dissemination Level		
C	Confidential, only for members of the consortium and the Commission Services	

5. Remaining questions for the future version

5.1 Import partial TMX file

At this stage of the development, if one translation unit is not valid, the TMX is considered as rejected.

To enhance the usability in a future version, the system will propose to the user to decide:

- either to abort the TMX import and to provide a new TMX file
- or to continue the TMX import.

If the user finally decides to import the TMX, only the text segments that match the size limit will be imported.

5.2 General memory for all users

In a second step of development we propose to implement the concept of a general memory usable by all the users:

- All the information is not confidential and some translation unit can be useful for all the community. The web page will propose to the user to share these segments with the community (a check box in the TMX import and in the post edition page)
- If the check box is ticked, the segments will be also tagged with a user-attribute Privacy equal to "**Shareable**"
- An administrator of the Flavius TM will review periodically the "**Shareable**" translation units to transform the user-attribute Privacy to "**General**" if it will be useful for the community.

- In the user profile page, a check box will propose to filter the TM entries with his own ID only or with his own name and the user-attribute Privacy equal to "**General**". In the second case, he will be able to retrieve entries imported by other users but shared for all the community.

5.3 Sentences in place of Segments

For the moment, the queries sent to the TM are the segments extracted from the XML documents by the parser. These segments can contain more than one sentence. The size of the segment will reduce the number of matches in the TM.

The Across API proposes a dedicated method to extract sentences from a given text. This method references the module used by Across in its commercial product to detect sentences in the documents.

The future improvement of the Flavius parsing module will include a call to this additional tokenization step to ensure that the segments are only sentences.

The sentence rules used by Flavius will be maintained directly in the Across database system.

The translation memory integration in the translation process will not be modified by this enhancement. Indeed, the sentence analysis will be part of the parsing module and the translation module will only receive sentence instead of paragraph, increasing the the number of matching segments.

5.4 Adding ACROSS TM entries through post-editing

The ACROSS TM will be also enriched through the post-editing interface.

Each text segment translated through the ACROSS TM and updated by the user during post-editing will be added to the ACROSS TM.

Each entry added through post-editing will be tagged with:

- the relation attribute set to the owner's tag (its unique identifier)
- the user-attribute "origin" set to "post-editing"

A functionality will be added to give the possibility to export the post-edited entries in a TMX file.

6. Appendix: Technical aspects

6.1 Connecting the ACROSS TM

To call Across TM through the API, the Flavius system needs to sign up with an authorization method.

The connection with the Translation Memory server and the different applicative sessions (such as CrossTank session,…) are opened at the beginning of the processing of a source text, and remains open until the whole text segments that composed the source text are not translated.

For performance reason, the translation of the different segments is parallelized through a multi-threading mechanism. The implementation of the communication with the Translation Memory will take into account this particularity. This process will enhance the translation speed.

Technically, the main process will queue all the segments, and the threads of a thread pool will process the requests to the Across TM. (About the number of threads, see the performance test (paragraph 6.4)).

6.2 Implementation through an interface

The implementation of the communication with the Translation Memory server is standardized through a C# Interface (ITranslationMemory).

This interface defines the main functionalities needed for the Flavius implementation, independently of the Across API. In this architecture, the Flavius Translation memory management is not dependent on the Across API implementation. If the Across API is modified, only the interface implementation needs to be modified.

The TM interface of the FLAVIUS platform is currently defining the following methods:

- **Connect**: Open a connection with the Translation Memory Server using the credential information (see configuration)
- **IsConnected**: Return if the connection is open
- **Close**: Close the connection and free the allocated resources.
- **FuzzySearch**: Perform a fuzzy search. This method accepts the following parameters: a list of sentences, a confidence threshold (100% in our case), a relation attribute filter. This filter can contain more than one relation attribute. The FuzzySearch method returns a list that contains for each provided text segment the following information :
    - the source text,
    - the translation (or **nothing** if no translation were found in the ACROSS TM),
    - the level of confidence
- **CheckTMX**: Return if the TMX is validated by the Flavius checker.
- **ImportTMX**: Insert entries in the Translation Memory Server from a TMX file. This method accepts the following parameters: the name of a TMX file, a relation attribute list that will tag all the translation units.
- **IsImportTMXFinished**: Give a status about the current import. The memory import is asynchronous.
- **CloseImport:** Finalize the import procedure.
- **GetTMEntriesCount:** Return the entry count for a particular origin (a specific TMX file for example).
- **DeleteTM**: Delete the list of entries in the Translation Memory Server based on a filter (For example the name of a previously imported TMX).
- **UpdateTM:** Update an attribute for a list of entries in the Translation Memory Server based on a filter, like the name of a previously imported TMX.

6.3 The ITranslationMemory Interface

```
public interface ITranslationMemory : IDisposable
{
        bool Connect(string server, string username, string password);

        bool IsConnected();

        void Close();

        ITMImportResult ImportTMX(string tmxPath, int source_lcid, int target_lcid,
 Dictionary<TMParameter.Parameter, string> tmxParams);

        bool CloseImport(Dictionary<TMParameter.Parameter, string> tmxParams);

        ITMImportResult IsImportTMXFinished(string jobGuid);

        int GetTMEntriesCount(int source_lcid, int target_lcid, Dictionary<TMParame
ter.Parameter, string> tmxParams);

        List<ITranslationSegment> FuzzySearch(List<ITranslationSegment> translation
segments, int similarity, int source_lcid, int target_lcid, Dictionary<TMParameter.
Parameter, string> tmxParams);

        ITMUpdateResult UpdateTM(string[] tmServerParams, Dictionary<TMParameter.Pa
rameter, string> tmxParams);

        ITMDeleteResult DeleteTM(string[] tmServerParams, Dictionary<TMParameter.Pa
rameter, string> tmxParams);

        ITMCheckResult CheckTMX(string tmxFilePath, string[] dtdFilePath, int MaxSi
zeEntries, int source_lcid, int target_lcid);
}
```


Project co-funded by the European Commission within the ICT Policy Support Programme		
Dissemination Level		
C	Confidential, only for members of the consortium and the Commission Services	

6.4 Performance Tests

6.4.1 Context and methodology

This chapter describes the performance evaluation done on the ACROSS Translation Memory Server FuzzySearch method to verify the performance of our client / server architecture in term of multithreading and to define a first best range for the number of thread in the Flavius Translation memory module.

The evaluation was performed on the ACROSS TM Server installed on a dedicated server located in the Reverso premises.

The ACROSS TM contained 40 000 entries (20 000 French to English entries, 20 000 entries linked to others directions). 970 entries have been extracted from this memory to be used during the requests.

The corpus was composed of 50% of long sentences (more than 25 words) and 50% of short sentences.

The evaluation was done through the ACROSS API and a testing application implemented in C#. This application was designed to measure the time spend for each TM request and for the global request.

We tested the French to English language direction.

6.4.2 Results

970 text segments / 2 threads in parallel

- The total time needed to receive a response for all the requests is 4.416 seconds. If we divide by the number of requests, we need 0.0045 seconds per search in this configuration.
- The maximum time is 0.149 seconds. This is for the first request.
- The minimum time is 0.004 seconds.
- 80% of translations were found in less than 0.01 seconds
- 19% of translations were found in between 0.01 seconds and 0.02 seconds
- 7 translations were found in between 0.02 seconds and 0.03 seconds
- 2 translations were found in more than 0.03 seconds (the two first requests).

970 text segments / 5 threads in parallel

- The total time needed to receive a response for all the requests is 2.304 seconds. If we divide by the number of requests, we need 0.0024 seconds per search in this configuration.
- The maximum is 0.185 seconds. This is for the first request.
- The minimum time is 0.005 seconds.
- 60% of translations were found in less than 0.01 seconds
- 39% of translations were found in between 0.01 seconds and 0.02 seconds
- 10 translations were found in between 0.02 seconds and 0.03 seconds
- 5 translations were found in more than 0.03 seconds (the first requests).

970 text segments / 10 threads in parallel

- The total time needed to receive a response for all the requests is 1.803 seconds. If we divide by the number of requests, we need 0.0018 seconds per search in this configuration.
- The maximum is 0.170 seconds. This is for the first translation requested.
- The minimum time is 0.006 seconds.
- 12% of translations were found in less than 0.01 seconds
- 80% of translations were found in between 0.01 seconds and 0.02 seconds
- 7% of translations were found in between 0.02 seconds and 0.03 seconds
- 14 translations were found in more than 0.03 seconds (the two first translations).

970 text segments / 20 threads in parallel

- The total time needed to receive a response for all the requests is 2.254 seconds. If we divide by the number of requests, we need 0.0023 seconds per search in this configuration.
- The maximum is 0, 223 seconds.
- The minimum time is 0,006 seconds.
- 10% of translations were found in less than 0.01 seconds
- 65% of translations were found in between 0.01 seconds and 0.02 seconds
- 15% of translations were found in between 0.02 seconds and 0.03 seconds
- 10% of translations were found in more than 0.03 seconds.

1000 text segments / 10 threads in parallel / No matching translation

- These segment texts were selected to have no matching translation in the ACROSS TM.
- The total time needed to receive a response for all the requests is 0.407 seconds. If we divide by the number of requests, we need 0.0004 seconds per search in this configuration.
- The maximum is 0.006 seconds.
- The minimum time is 0.001 seconds.
- 25% was performed in less than 0.002 seconds
- 65% was performed in 0.003 seconds
- 9% was performed in 0.004 seconds
- 11 were performed in 0.005 seconds and 0.006 seconds

6.4.3 Conclusion

This first evaluation of the FuzzySearch performance through the ACROSS API has shown that:

- Once the connection is established, the FuzzySearch call is fast. Most of the translations (from 92% to 99%) are found in less than 0.02 seconds.
- When there is no matching translation, the search is very fast: 99 % of the queries performed in less than 0,004 seconds.
- The queries to the FuzzySearch API can be parallelized. The total time to translate the same 970 text segments depends directly on the number of parallelized queries.
  - 2 queries: 4.416 seconds
  - 5 queries: 2.304 seconds
  - 10 queries: 1.803 seconds
  - 20 queries: 2.254 seconds

  We see than the minimum time is reached for 10 queries. This is probably due to bottleneck generated by the number of processors on the server, and on the client machine. The value will be refined once the full architecture will be in place.