NEC TM Technical Description

Version 1.00

0. Revision Control

Author	Description
Alex Helle	Initial Version of Specification (Sept 25th 2018)
Amando Estela	First Revision (Sept 26th 2018)
Carmen Herranz-Carr and Carolina Herranz-Carr	Second Revision (Sept 28th 2018)

1. INTRODUCTION

1.1 Purpose

This software design document describes the architecture and system design of NEC TM based on ActivaTM, cloud-based Translation Memory tool. The target audience are developers, managers and advanced users of the tool.

1.2 Scope

ActivaTM is a fast, highly-scalable cloud-based Translation Memory tool developed by Pangeanic as a part of the EXPERT EU project. The goal of ActivaTM is to seamlessly integrate all available TMs to serve vendor-independent tools in the standard translation flow. ActivaTM integrates (via plugins) with multiple manual and machine translation software (SDL Studio, PangeaMT, NeuralMT, Bing Translate).

NEC TM Data project will provide the centralised infrastructure for efficient data sharing, TM matching, TM retrieval, and domain categorisation of resources generated by Member States/ EEA. This will enable the development of NEC TM, an open source software developed from Pangeanic's translation memory database ActivaTM.

1.3 Overview

The document describes design and architecture of ActivaTM and it subcomponents

1.4 Definitions and Acronyms

TM - Translation Memory

ES - ElasticSearch

CAT - Computer-Assisted Translation

Translation unit - data structure, containing two (or more) text segments of different language along with additional properties. Sometimes used interchangeably with segment

Segment - single-language text string, part of translation unit

1.5 Dependencies

ElasticSearch - 2.4.0

Apache Spark - 1.6.3

PostgreSQL - 9.4

Celery (Redis) - 2.*

During the scope of the project some of these dependencies will be upgraded.

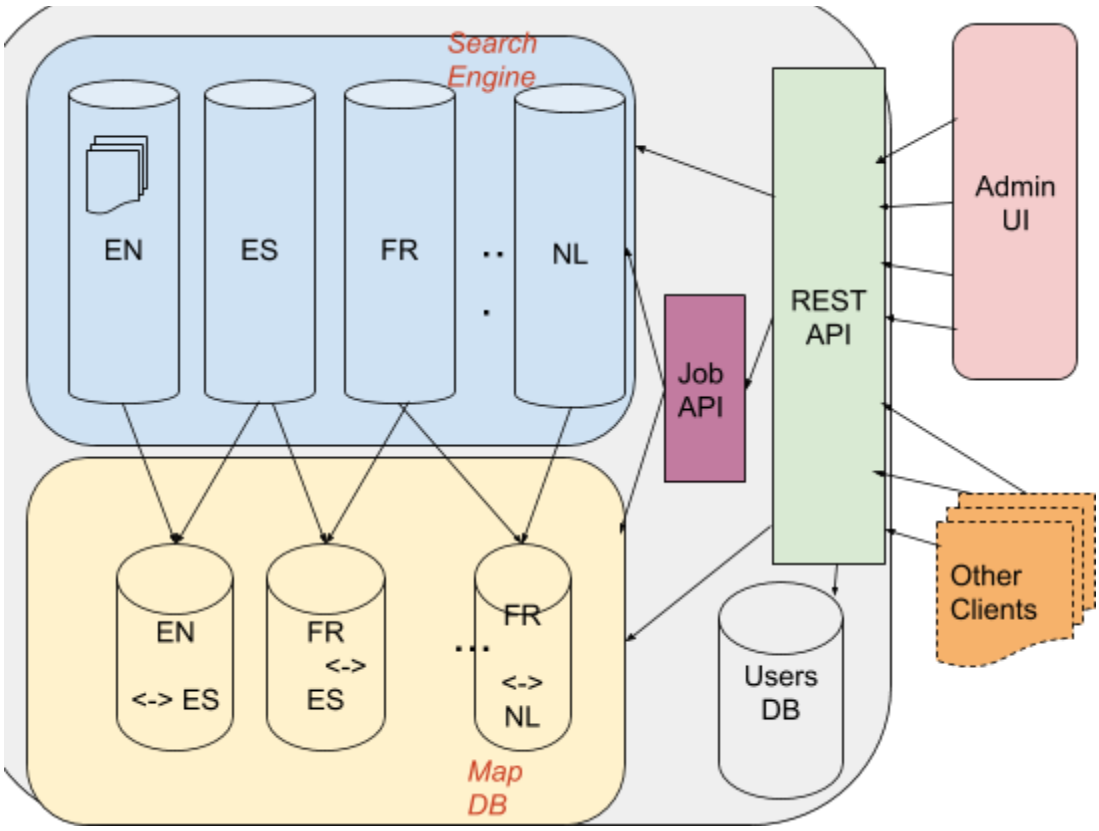
2. SYSTEM OVERVIEW

Translation Memory (TM) is a collection of segments helping to improve quality of translation process in CAT tools (such as Trados Studio) by reusing translation of phrases, sentences or even paragraphs (commonly known as segments) from previously known translations. Usually, automatic machine translation tools use translation memory in-memory (loaded from standard TMX format) or by using web service. The number of segments can be huge if all possible European language combination pairs are taken into account. Current non-commercial solutions appear to be either non-scalable (such as dealing with TMX files) or in early stages of development, such as TinyTM database. This document describes the design of a fast and scalable cloud-based TM database with a goal to overcome the limitations of current solutions for TM storage & query.

3. SYSTEM ARCHITECTURE

3.1 Architectural Design

This diagram depicts the main components of ActivaTM:



Search Engine (*based on ElasticSearch*[1]) holds monolingual segment text in a separate index (aka monolingual index), e.g. for example, all English segments are stored separately no matter what language they have been mapped with in the source TM data. This enables a significant conservation of memory, as each and every unique segment is stored only once. The segments are stored along with their corresponding properties, such as industry, type, organization etc. Furthermore each segment is allocated a unique id (UUID, calculated by hashing a segment string) once added to the system. Sample EN segment looks similar to the following snippet (JSON representation):

```
{
        "text" : "Connect the pipe to the female end of the T.",
        "id" : "1b45648d-5b94-525a-ba75-0b16d3ce3d7e",
```

[1] "Elastic." https://www.elastic.co/. Accessed 11 Sep. 2018.

```
        "industry" : ["Automotive Manufacturing"],
        "type" : ["Instructions for Use"],
        "organization" : ["pangeanic"],
        "language" : ["en-GB"]

}
```


Sample corresponding ES segment is stored in a corresponding index (JSON representation):


```
{
        "text" : "Conecte la tubería al extremo hembra de la T.",
        "id" : "b9049718-157b-5a7b-9c6a-f2a5ac8265cb",
        "industry" : ["Automotive Manufacturing"],
        "type" : ["Instructions for Use"],
        "organization" : ["pangeanic"],
        "language" : ["es-ES"]

}
```


Once the search engine is queried for the source language segment, it will look first for an exact match, if an exact match is not found, the system will widen the search conditions to support fuzzy matching and regular expressions. The selected search engine should support all necessary query combinations: exact match, fuzzy match and regular expressions.

Map DB (*based on ElasticSearch*) holds collections of segment mappings for each pair of languages. Each language pair is stored in a separate ES index (aka bilingual index). This way when we find a match in a source language segment DB (indexed by search engine), we will quickly retrieve corresponding the UUID of the target language segment. The text of a target segment will be retrieved by querying a corresponding search engine index. **Map DB** supports a quick bulk update operation enabling future updates of a segment, for example, modifications such as data changes. Sample mapping of the previously mentioned segments are stored in Map DB in a following way as a part of “EN-ES” database:


```
{
        "source_id" : "1b45648d-5b94-525a-ba75-0b16d3ce3d7e",
        "target_id" : "b9049718-157b-5a7b-9c6a-f2a5ac8265cb",
        "creation_date" : “20090914T114346Z”,
        "change_date" : “20090914T114346Z”,
}
```


Each language pair is held in a separate ES index named after a language pair, e.g. “EN <-> ES” language mappings are stored in the“EN-ES” collection, thus identifying the source and target language of stored segments.

Users DB (*based on PostgreSQL*[2]) - contains users and their permissions (scope). For each user defined in the system, the UsersDB stores a username, encrypted password, list of scopes (permissions) and user settings. The scopes are associated with a user and stored in a separate table containing a list of permitted language pairs, domains, usage count (number of times the user has utilised a scope), limit (maximum number of times the scope can be used) along with start and end date of the scope.

Job API (*based on Celery*[3] *and Apache Spark*[4]) is a mechanism for parallel background processing of lengthy client requests (such as adding, deleting and cleaning segments) preventing the blocking of the REST API. The mechanism is triggered by REST API requests, queuing job to Celery and executed by Apache Spark which processes the data in the background

REST API (*based on Flask*[5]) is a programming interface for interacting with ActivaTM content(adding, deleting, updating, cleaning translation units etc.), managing users and managing background jobs. The user is authorized by accessing authorization endpoint (with username and password). Access to other endpoints is granted based on JWT token return by authorization endpoint and depending on user role (admin or user).

3.2 API Description

This section describes the main REST API functions for the services described in the Functional Description:

3.2.1 Fuzzy Match Services

3.2.1.1 Search translation memory segments

```
GET -> https://localhost/api/v1/tm
```

Permission: user

Field	Type	Description

[2] "PostgreSQL." https://www.postgresql.org/. Accessed 11 Sep. 2018.
[3] "Homepage | Celery: Distributed Task Queue." http://www.celeryproject.org/. Accessed 11 Sep. 2018.
[4] "Apache Spark™ - Unified Analytics Engine for Big Data." https://spark.apache.org/. Accessed 11 Sep. 2018.
[5] "Flask." http://flask.pocoo.org/. Accessed 11 Sep. 2018.

token	String	Token returned by auth endpoint.

Parameter

Field	Type	Description
q	String	String to query
slang	String	Source language
tlang	String	Target language
limit optional	Number	Limit output to this number of segments
min_match optional	Number	Return only match above or equal to given threshold (0-100) Default value: 75
domain optional	String	Filter segments by domain(s)

Success 200

Field	Type	Description
Translation	String/ Json	units matching the query

3.2.1.2 Batch search translation memory segments

GET -> https://localhost/api/v1/tm/query_batch

Permission: user

Field	Type	Description
token	String	Token returned by auth endpoint.

Parameter

Field	Type	Description
q	String	String to query (multiple values allowed)
slang	String	Source language.
tlang	String	Target language.
limit optional	Number	Limit output to this number of segments Default value: `10`
min_match optional	Number	Return only match above or equal to given threshold (0-100) Default value: `75`

domain optional	String	Filter segments by domain(s).

Success 200

Field	Type	Description
Translation	String/Json	units matching the query

3.2.2 User Management Services

3.2.2.1 Add/update user details

POST -> https://localhost/api/v1/users/<username>

Permission: admin

Field	Type	Description
token	String	Token returned by auth endpoint.

Parameter

Field	Type	Description
username	String	Name of User
password	String	User password

role	String	User role: admin or user

Error 4xx

Name	Type	Description
403	String	Insufficient permissions

3.2.2.2 Delete user

DELETE -> https://localhost/api/v1/users/<username>

Permission: admin

Field	Type	Description
token	String	Token returned by auth endpoint.

Parameter

Field	Type	Description
username	String	

Error 4xx

Name	Type	Description
403	String	Insufficient permissions

404	String	User doesn't exist

3.2.2.3 Get user details

GET -> https://localhost/api/v1/users/<username>

Permission: admin

Field	Type	Description
token	String	Token returned by auth endpoint.

Parameter

Field	Type	Description
username optional	String	Name of User, if not user specified, all users' details are returned

Error 4xx

Name	Type	Description
404	String	User doesn't exist

Success 200

Field	Type	Description

User details	String/ Json	users matching the query

3.2.2.4 Add/update user scope

POST -> https://localhost/api/v1/users/<username>/scopes

Permission: admin

Field	Type	Description
token	String	Token returned by auth endpoint.

Parameter

Field	Type	Description
username	String	
lang_pairs optional	String	List (separated with comma) of allowed language pairs to query, ex.: en_es,es_en,es_fr. By default, allows all pairs
domains optional	String	List (separated with comma) of allowed domains to query, ex.: Health,Insurance. By default, only public domains
can_update optional	Boolean	Allow/forbid update TM (add a translation unit) in the given scope. Default is false.

can_import optional	Boolean	Allow/forbid import TM in the given scope. Default is false.
can_export optional	Boolean	Allow/forbid import TM in the given scope. Default is false.
usage_limit optional	Integer	Limit allowed usage (queries number). Default: no limit
start_date optional	Date	Scope starts at that date. Default - not limited
end_date optional	Date	Scope ends at that date. Default - not limited

Error 4xx

Name	Type	Description
403	String	Insufficient permissions
404	String	User doesn't exist

3.2.2.5 Delete user scope

DELETE -> https://localhost/api/v1/users/<username>/scope

Permission: admin

Field	Type	Description

token	String	Token returned by auth endpoint.

Parameter

Field	Type	Description
username	String	
id	Integer	Scope id

Error 4xx

Name	Type	Description
403	String	Insufficient permissions
404	String	User doesn't exist

3.2.2.6 Authorization

POST -> https://localhost/api/v1/auth

Parameter

Field	Type	Description
username	String	Name of User
password	String	Password for account

Error 4xx

Name	Type	Description
401	String	Invalid credentials

Success 200

Field	Type	Description
access_token	String	Authorization token for use it other endpoints

3.2.3 Domain Management Services

3.2.3.1 Add/update domain

POST -> https://localhost/api/v1/domains/<domain>

Permission: admin

Field	Type	Description
token	String	Token returned by auth endpoint.

Parameter

Field	Type	Description

domain	String	Name of Domain
type	String	Domain type: private, public or unspecified

Error 4xx

Name	Type	Description
403	String	Insufficient permissions

3.2.3.2 Delete domain

DELETE -> https://localhost/api/v1/users/<username>

Permission: admin

Field	Type	Description
token	String	Token returned by auth endpoint.

Parameter

Field	Type	Description
domain	String	Name of Domain

Error 4xx

Name	Type	Description
403	String	Insufficient permissions
404	String	Domain doesn't exist

3.2.3.3 Get domain details

`GET -> https://localhost/api/v1/domains/<domain>`

Permission: user

Field	Type	Description
token	String	Token returned by auth endpoint.

Parameter

Field	Type	Description
domain optional	String	Name of Domain, if not domain specified, all domain' details are returned. Admin will see all domains, and Users will only see allowed domains in the scopes.

Error 4xx

Name	Type	Description
404	String	Domain doesn't exist

Success 200

Field	Type	Description
Domain details	String/ Json	domains matching the query

3.2.4 Import Services

3.2.4.1 Import translation memory segments from TMX file

`PUT -> https://localhost/api/v1/tm/import`

Permission: admin

Field	Type	Description
token	String	Token returned by auth endpoint.

Parameter

Field	Type	Description
file	File	Zipped TMX file to import.
domain	String	Domain name of the imported file.
lang_pair optional	String	Language pair to import (for multilingual TMX files). 2-letter language codes join by underscore. By default, import first pair in each segment

3.2.4.2 Add new translation memory unit

POST -> https://localhost/api/v1/tm

Permission: user

Field	Type	Description
token	String	Token returned by auth endpoint.

Parameter

Field	Type	Description
stext	String	Source segment text.
ttext	String	Target segment text..
slang	String	Source language.
tlang	String	Target language.
domain	String	Translation unit domain.
file_name optional	String	File name (or source name)

3.2.5 Export Services

3.2.5.1 Export translation memory segments to TMX file

https://localhost/api/v1/tm/export

Permission: admin

Field	Type	Description
token	String	Token returned by auth endpoint.

Parameter

Field	Type	Description
slang	String	Source language.
tlang	String	Target language.
file_name optional	String	Filter segments by filename(s).
domain optional	String	Filter segments by domain(s).

Success 200

Field	Type	Description
binary	File	Content of zipped TMX file(s)

3.3 Design Rationale

For Search Engine, we looked mainly on Lucene-based search engines which dominate the open-source search engine market : Solr and ElasticSearch both satisfy our basic requirements, supporting out-of-the box exact matching, fuzzy matching and regular expressions. Moreover, both engines are very popular in industry and in the academy. ElasticSearch has a couple of major advantages which makes it our top pick:

ElasticSearch is scalable almost transparently when Solr requires more effort and additional software (ZooKeeper) to set up clusters and shards

ElasticSearch has a powerful Query DSL (structured JSON language to build complex queries easily)

Several alternatives were considered to serve as a Map DB. We looked primarily at NoSQL solutions such as:

- ElasticSearch - full-text search engine but can be used as key-value storage
- MongoDB - a document-oriented database
- CouchDB - another document-oriented database
- Redis - fast in-memory key-value database, used mainly as a cache

All of the databases provide key-value mapping interface via HTTP and suitable for purposes of ActivaTM and scale horizontally rather easily due to their nature of being NoSQL solutions. Consequently, the major differentiation should be their performance of bulk adding and one-by-one key querying as well as memory consumption. The benchmark experiments were conducted on English-Spanish TMX files having overall 440K translation memories. The following table demonstrates the recorded execution times and memory consumption:

Metric/Engine	**ElasticSearch**	**MongoDB**	**CouchDB**	**Redis**
Bulk adding (47K segments)	83s	432s	***67s***	458s
Bulk adding (440K segments)	858s	6112s	644s	***621s***
Querying (1K segments)	***11s***	23s	52s	51s
Querying (10K segments)	***51s***	187s	458s	72s

Querying (440K segments)	1400s	6451	19647	***1210s***
Process size (440K segments)	4.9G [1]	549M	771M	***148M***

[1] *ElasticSearch process maintains indexes of monolingual texts too and thus can't be directly compared in terms of memory consumption of MapDB only*

4. DATA DESIGN

4.1 Data Description

Mainly, ActivaTM deals with **translation units, a** data structure containing source segment (text in source language), target segment (text in target language), multiple timestamps (creation date, modification date) and various properties (industry, type, organization). Translation units are currently parsed from TMX file and translated into internal ActivaTM data. This section describes the internal data structures of ActivaTM.

ActivaTM has the following databases:

- Monolingual indexes (Search Engine, stored in ElasticSearch)
- Bilingual indexes (Map DB, stored in ElasticSearch)
- Users & scopes (stored in PostgreSQL)

4.1.1 Monolingual index

Monolingual index stores text segments of a single language along with few additional auxiliary properties (list of target languages, token count). ElasticSearch index is named as "tm_<language code>", for example **tm_en** and has the following schema:

- ***_id*** - UUID-based unique id, allocated by hashing the text[6]. Same-text segments will have the same id (though, there is a minor chance of collision, currently ignored)
- ***text*** - contains raw text, the field is analyzed (e.g. tokenized and searchable)
- ***target_language*** - contains codes of target languages for which this segment has corresponding translation unit. For example, if translation unit is EN-ES ("hello" ← → "hola"), then English segments will have 'es' as a target language. Similarly, Spanish segments such as "hola" will have 'en' as a target language. If there is an additional EN-FR translation unit ("hello" ← → "bonjour"), then "hello" segment's

[6] "22.20. uuid — UUID objects according to RFC 4122 — Python 3.7.0 ...." https://docs.python.org/3/library/uuid.html. Accessed 12 Sep. 2018.

target_language field will contain the list: *["en", "fr"]*. This field is mainly used to Generate algorithms

- ***token_cnt*** **-** number of tokens in text. Used for internal matching algorithm

4.1.2 Bilingual index (Map DB)

Bilingual index stores all fields of translation units needed to recreate TMX file properly. Additionally, it contains IDs of its source and target texts so that search algorithms can map the found source segment text to actual translation units stored in bilingual index. The ElasticSearch index is named as "map_<lang1>_<lang2>", for example "map_en_es". The same index serves for EN->ES and ES->EN queries, imports, exports etc. The index has the following schema:

- ***_id*** - UUID-based unique id, allocated by hashing the concatenation of source and target text segments[7]. Same-text translation units will have the same id (though, there is a minor chance of collision, currently ignored)
- ***source_id, target_id*** **-** IDs of correspondent text segments in monolingual indexes
- ***source_text, target_text*** **-** source and target segment texts, stored here as not-analyzed, to speed up TMX generation and to avoid accessing monolingual indexes
- ***source_language, target_language*** **-** source and target languages, including locale (en-GB, es-AR etc)
- ***domain*** **-** list of domains assigned to the translation unit
- ***filename*** **-** list of filename from where the translation unit comes
- ***Other*** **-** other optional fields (industry, type and organization) can be added during import if they are specified in the TMX file.

To emphasize the difference, bilingual index data is usually accessed by ID and not by free-text search, for that monolingual index is used.

4.1.3 Users & Scopes

Users and their permission scopes (in short, scopes) are stored in PostgreSQL in the following tables:

- Table: **users**
  - *username*
  - *password*
  - *role* - either "user" or "admin". Admin has access to all ActivaTM interfaces, User is limited to querying and limited statistics
  - *token_expires* - whether JWT expires within 24 hours or stays valid indefinitely

[7] "22.20. uuid — UUID objects according to RFC 4122 — Python 3.7.0 ...." https://docs.python.org/3/library/uuid.html. Accessed 12 Sep. 2018.

(useful for automatic API clients such as crawler)

- *is_active* - whether user is active. Inactive user's access will be denied

- Table: **user_scopes**
  - *id*
  - *username* - user to whom this scope belongs
  - *lang_pairs* - list of language pairs, for them the scope is applicable
  - *domains* - list of domains, for them the scope is applicable
  - *usage_limit* - maximal number of user queries applicable to this scope. If the user exceeds this number, access will be denied
  - *usage_count* - current number of user queries matching this scope
  - *start_date* - beginning of this scope validity
  - *end_date* - end of this scope validity