EnrichEuropeana+: MS8 - Final release of Enrichment Services

Version 1.0

Documentation Information

Action Number	2020-EU-IA-0075
Project Website	https://pro.europeana.eu/project/enricheuropeana
Contractual Deadline	30.11.2022
Nature	technical documentation
Author	Medina Andresel, Srdjan Stevanetic
Contributors	Sergiu Gordea, Philip Kahle
Reviewer	Mónica Marrero, Hugo Manguinhas, Frank Drauschke
Version	1.0
Date	09.12.2022

Co-financed by the Connecting Europe Facility of the European Union

Contents

Introduction

EnrichEuropeana + (fully titled 'Enriching Europeana through citizen science and artificial intelligence - unlocking the 19th century') aims to enhance Europeana Transcribe (www.transcribathon.eu) as a service for cultural heritage institutions. This document describes the new or updated functionality of the enrichment services introduced during the timeframe of the previous action[1] and further developed within the Activity 2: "Automated transcription and enrichment services" of the EnrichEuropeana+ action.

Scope

This milestone provides the technical documentation for services developed in the Task 2.2 "Transcription and metadata translation" and Task 2.3 "Topic detection and semantic enrichments". Important requirements concerning a better integration with the Europeana CSP, improved quality and enhanced scalability were taken into consideration when designing the new version of the machine translation and semantic enrichment services. The progress on the development of handwritten text recognition services (Task 2.1) was included in MS4: "First release of HTR Services".

Objectives

The main objectives of EnrichEuropeana+ action are:

- To engage public users and professionals in enhancing the semantic and multilingual description of Cultural Heritage objects by continuing the development of Europeana Transcribe.
- To increase accessibility of manuscripts related to historical events and societal transformations in Europe within the 19th Century through a new Citizen Science crowdsourcing campaign to stimulate user engagement for transcribing, translating, and adding semantic enrichments.
- To transform Europeana Transcribe into a service used by Cultural Heritage Institutions to crowdsource the enrichment of cultural object descriptions and improve the multilingualism of metadata.

The main objectives of the developments included in this milestone include the followings:

- Design a system architecture for scalable processing
- Implement authorization mechanism based on Europeana OAuth service
- Introduce support for content curation activities using Topic Detection technology
- Enhanced quality control for machine translations
- Use of semantic enrichment services for improving the quality of metadata for Europeana records

[1] https://github.com/EnrichEuropeana/Technical-Documentation/blob/master/MS7_final_version_of_semantic_enrichment.pdf

1. Scalable Infrastructure for Metadata Enrichment

As the amount of materials ingested into the Transcribathon tool is constantly increasing, the demand of processing capacity for the enrichment services is increasing as well. Additionally, the new developed services based on complex AI models impose more demanding requirements in terms of hardware configurations. By taking these aspects into consideration, the system architecture for enrichment services is following a modular approach and technology that has built-in scalability mechanisms.

Figure 1. presents an overview of the system architecture used by the semantic enrichment services. The runtime environment consists of the infrastructure required to operate the Enrichment Services and it's API, while the Asynchronous Processing module consists of the components used to train machine learning models. Several complementary technologies are employed in order to satisfy the functional and operational requirements. The goal is to provide an adaptive infrastructure, which can be easily scaled up with the hardware and software components required to process large amounts of data in an efficient way (e.g learning and executing complex AI models on large amounts of data). A more detailed description of the individual software components is provided in the following subsections.

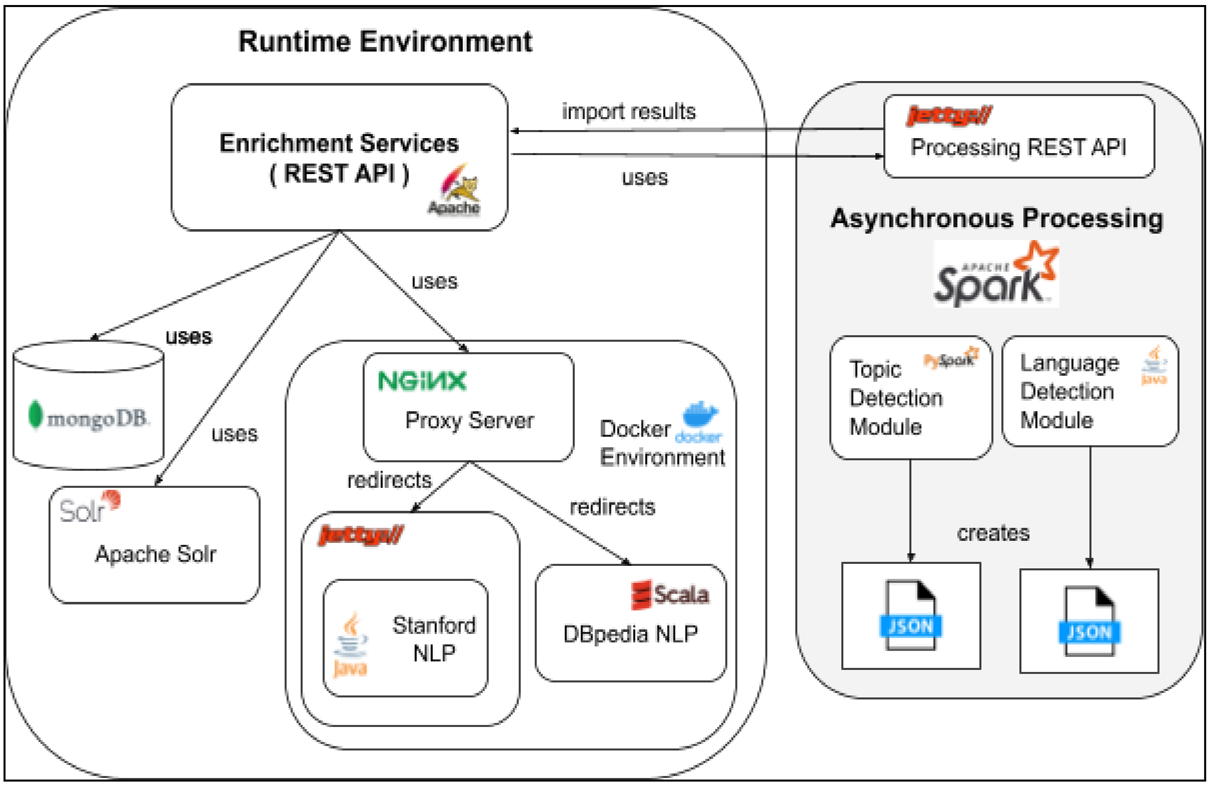


Figure 1. System architecture for semantic enrichment services

1.1 Runtime Environment

The runtime environment has the responsibility of providing efficient access to the Enrichments Services through their REST API, which is implemented using Spring technology and hosted on a Tomcat web server. Most of the enrichment services are optimized for read access, serving data stored in a mongoDB document database and search functionality powered by Solr based indices.

When applied to small sized texts, the machine translation and Natural Language Processing (NLP) services are able to provide real time performance. The translation services are accessed remotely, while the NLP services are packaged within docker containers being deployed on local server infrastructure. Load balancing and protected access to the NLP Services is ensured through a lightweight proxy server (i.e. using NGINX).

1.2 Asynchronous Processing

The Apache *Spark Server*[2] is used as a platform to run the scalable machine learning and data processing algorithms. The tasks executed on this infrastructure include computation of complex mathematical functions on large amounts of data. Consequently, this part of the infrastructure is used for asynchronous processing. The output of a machine learning algorithm is a computational model (i.e. AI Model), which is later used by the semantic enrichment services. Spark environment is offering the required functionality to run the AI Models and it also supports parallel task execution when applied to large data sets. The main programming languages used on the Spark engine are java and python, consequently, the results of the task execution use a programming language specific data representation. To ensure standardized data access and the interoperability of different system modules, the output of AI Models is converted to a JSON representation.

Similarly to the (externally hosted) machine translation services, the time required to execute the models is correlated to the size of the input. When applied to smaller amounts of data, like the description of Europeana records, (near) real time execution can be achieved. For such chases, the synchronous execution is emulated through the implementation of the Processing REST API. For larger processing tasks the results are computed offline and imported into the storage of the runtime environment, following the “write once read many” paradigm.

AI models trained by third parties can be also deployed on Spark infrastructure. This is also the case of the language detection model for which a wrapper was implemented

[2] https://spark.apache.org/

into the Processing REST API. More details will be provided in the following section presenting the enhancements of the previously developed semantic enrichment services.

Similarly, a REST API was implemented within the scope of the Action to provide remote access to the execution of HTR models. The HTR API was already described in the MS 4 Fist Release of HTR Services.

1.3 Authorization mechanism

The enrichment services are accessible through their REST API and make use of paid machine translation services. In order to prevent abusive use and to enable easy integration with the Transcribathon tool or Europeana CSP an authorization mechanism was implemented. The operations used to retrieve translations and enrichments from the database are open for anyone interested to use this data. However, the write operations like the computation of translations, named entity recognition, document topics, etc., are granted only to authorized users and applications. The authentication mechanism used in Europeana CSP APIs was integrated, access to operations that are updating the database are granted based on the JSON Web Tokens[3] issued by the Europeana Keycloak[4] server.

JWT is an open standard for securely transmitting the information between two parties using a JSON object. The tokens are digitally signed using public/private key pairs, and the signature guarantees that only a party who owns a private key signed it. Figure 2 illustrates the order of actions during a JWT authorization process.

[3] https://jwt.io/introduction
[4] https://github.com/europeana/keycloak-cf-docker

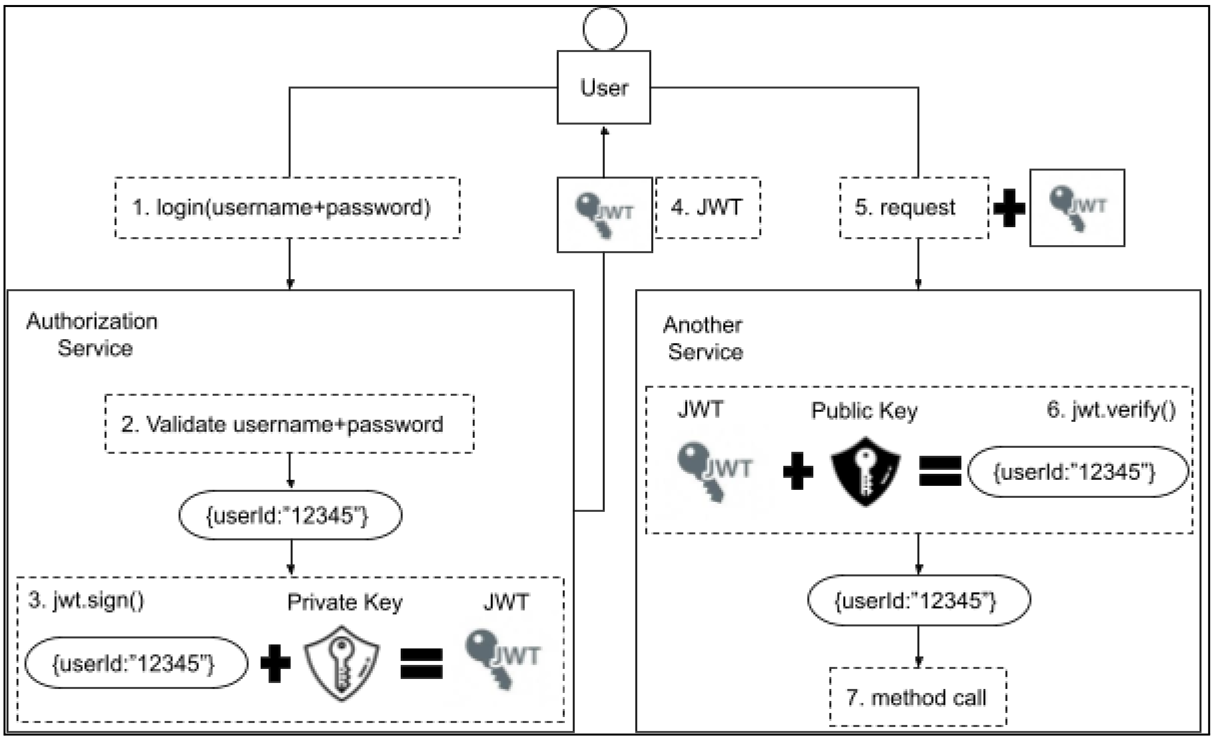


Figure 2. JWT authorization process

The API users or applications integrating the enrichment API are required to retrieve a valid JWT token in order to successfully access the enrichment API. User credentials (username and password) are used for requesting a JWT from the Keycloak Server, which signs the generated token with the Server's private key.

A generic authorization mechanism is provided within the Europeana api-commons[5] library. This solution can be reused and easily adapted for the needs of developed APIs, the basic functionality being activated through the configurations externalized in .properties files. In particular, the public key of the Keycloak server is used to prove the authenticity of the JWT token. An extension of the *AuthorizationService* interface is needed to verify if the authenticated users are allowed to access specific API methods (retrieve, create, update or delete) based on the roles assigned to them in the Keycloak server.

2 Topic Management Service

The extraction of textual information available from scanned documents into a plain text representation (i.e. which is easily understandable both by humans and machines) is a key activity for enabling its reuse by the communities of practice. Such activities are supported by online transcription tools like Transcribathon. Through the engagement of the user community in crowdsourcing campaigns, many historical documents were

[5] https://github.com/europeana/api-commons

transcribed and barrier free access to historical information was provided on the Europeana site.

However, stimulating user participation in crowdsourcing campaigns is still a challenging task. The impact assessment report[6] on Transcribathon campaigns indicate that the best engagement is achieved by organizing competitions on specific themes or *topics* which may raise the interest of individual users or local communities. The informational content from the selected documents plays an important role for the success of Transcribathon runs. However, the manual curation of very specific collections from Europeana remains a challenging and expensive task given the size of the repository, the multilingualism, and the heterogeneity of the data records.

Within the scope of this activity, we provided automated support for content curation activities in the Transcribathon platform by employing Machine Translation (MT) and Machine Learning (ML) technologies. The MT technology is employed within the pre-processing pipeline to aggregate appropriate input for learning a topic detection model. Given the lack of annotated materials, we performed experiments based on non-supervised solutions. The first version of the topic detection functionality is based on *Latent Dirichlet Allocation* (LDA) method. Within this process, LDA selects the most relevant terms for automatically detected topics, which can be further used for searching new materials related to individual topics.

An overview of the process used for learning topic models and making them available through the Topic Management API is presented in Figure 3. The details on the model training process are described in the following Model Training and Evaluation section, while the individual API requests for using the automatically identified topics are presented in the Topic Management API section.

[6] Impact Assessment Report: Enrich Europeana / Transcribathon | Europeana Pro

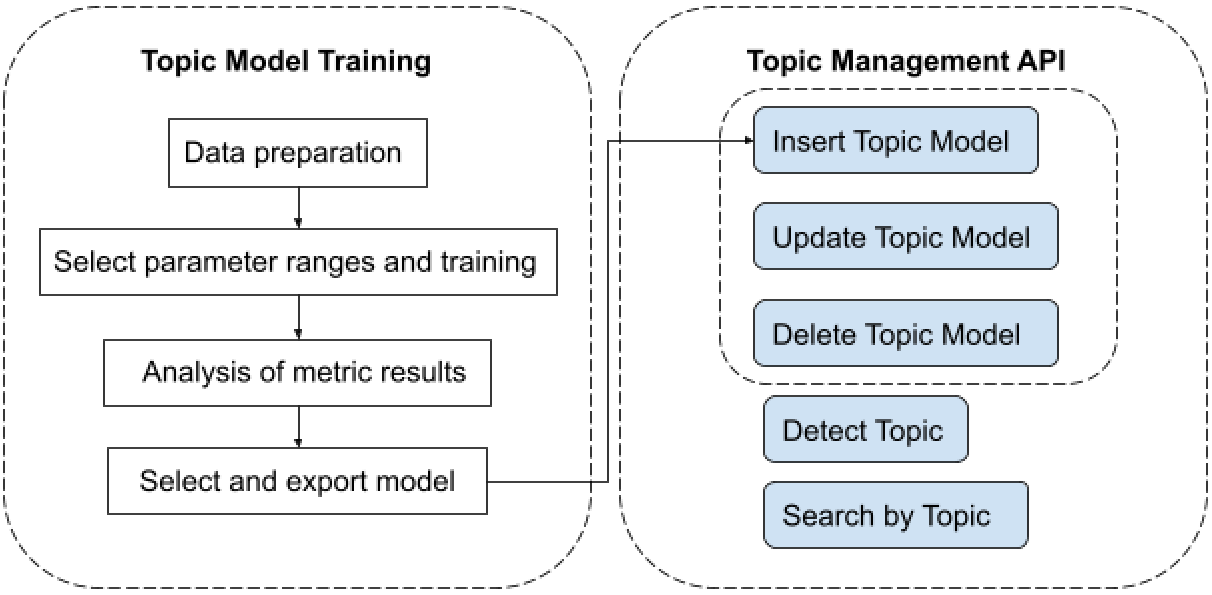


Figure 3. Learning and enactment of topic detection models

2.1 Model Training and Evaluation

Even if the topic detection technology is language agnostic, the learned models depend on the input data, which in the case of the Europeana repository includes descriptions available in one of the official European languages, but also Hebrew, Russian and Norwegian. By employing automatic language detection and machine translation, the proposed approach is able to automatically group Europeana records in clusters of similar content, independent from the language of the original object descriptions. In order to achieve this goal the preprocessing pipeline illustrated in Figure 4 is applied.

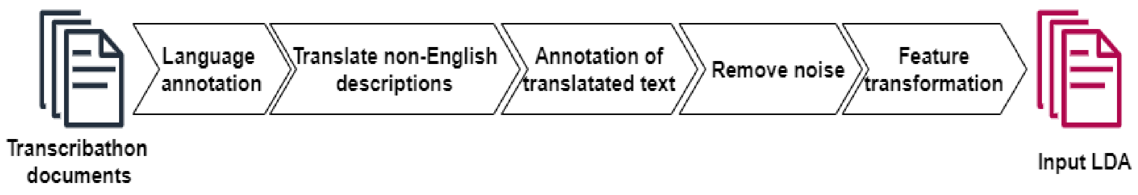


Figure 4. Text preprocessing pipeline

The first step in the pipeline is used to convert the original document descriptions into plain text by removing the available HTML markups, followed by the automatic detection of the language used within the original text. All non-English descriptions are translated to English by using the machine translation services. While the translation service still has some difficulties in translating mixed languages (i.e. which are sometimes still present in the document description), the language detection is once again applied on the translated text, and the failed translations are removed from the dataset. In the final preprocessing step, the textual descriptions are transformed into a *document term count vector*, in which a predefined list of stop words are excluded (i.e. a list of very common words with very low semantic information). The *document term count vector* represents the input for LDA model learning.

LDA uses two configuration parameters: **K –** the number of topics to be learned and **I –** the number of iterations run for building the topic model. In each iteration, the probability distribution of terms to topics and of documents to topics are adjusted to improve the document to topic assignments. To find the most appropriate values for model learning, we run several iterative experiments with different combinations for **K** and **I**.
The entire process used for identifying the best topic model and storing the description of the best model into the database is illustrated in Figure 5. This process generates different LDA models (i.e. named **LDA_K_I**) for each combination of the **K** and **I** values, from which we aim to determine the most appropriate one for the given dataset. This assessment is made by using the two complementary metrics, the *coherence* - **coh(LDA_K_I)-** which indicates how well the documents assigned to the same topic relate to each other and the *inter-topic distance* – **itdist(LDA_K_I)** - which is an indicator on how well the document clusters are separated from each other. The better models are maximizing the values for both these metrics, however, the maximal values for these complementary metrics are not achieved with the same **K** and **I** configuration. As the inter-topic distance is a surjective function with respect to **K**, we use the derivative of the function with respect to **K** to assess the quality of the models.

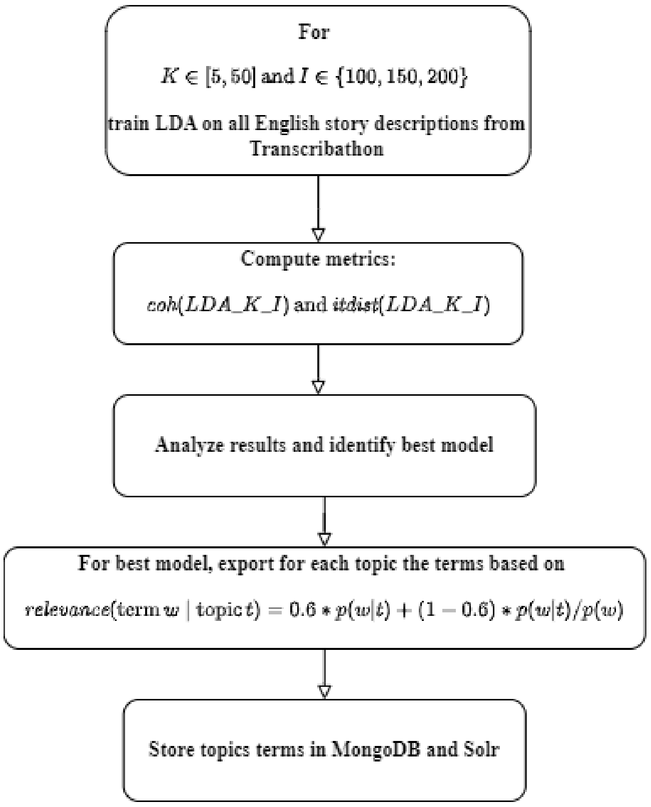

For
$K \in [5,50]$ and $I \in \{100, 150, 200\}$
train LDA on all English story descriptions from Transcribathon
Compute metrics:
$coh(LDA\_K\_I)$ and $itdist(LDA\_K\_I)$
Analyze results and identify best model
For best model, export for each topic the terms based on
$relevance(\text{term } w \mid \text{topic } t) = 0.6 * p(w|t) + (1-0.6) * p(w|t)/p(w)$
Store topics terms in MongoDB and Solr

Figure 5. Procedure for finding the best topic models for the Transcribathon dataset.

The representation of the topics learned within the selected model are meant to be used for assigning new incoming documents to the existing topics and for searching new candidate documents in external repositories. For this reason, we are storing the description of the K identified Topics into the database of the Enrichment API and index them in a Solr collection for facilitating retrieval based on free text search.

The effectiveness of using the topic descriptions for searching candidate documents for a given topic was instrumenting by using the Precision@10 metric (i.e. fraction of relevant documents in the top 10). When searching the candidate documents for a given topic, we take in consideration a number **N** of search results and reorder them by running the topic model to compute the probabilities of the document to belong to the previously identified models. By variating **N** in the range of [300,600] with a step of 50, we aim at identifying how many search results need to be analyzed in order to achieve a good precision in top 10 results. In order to decide if a document is relevant to a given topic, we apply 2 different thresholds. When using a lower threshold of 0.3, we consider a document to be relevant for the current topic, even if this is not the main topic of the document. When using a threshold of 0.5, the retrieved documents are considered to be relevant for at most one topic. For each topic in the model, **Precision@10** is computed by running a Solr searching query that includes the most relevant terms of the topic.

The decisions on selecting the most appropriate topic model and topic representation are based on the assessment and the observations of the experimental results. The first observation is that for each **I** in **[100, 300]**, the coherence and inter-topic distance metrics vary the least when the number of topics is 15. This means that according to both metrics, the overall optimum is achieved at this point as shown in Figures 6 and 7. For all other **K** values, the variation is much larger, meaning that the model performance is not stable with respect to **I**. This motivates our selection of **LDA_15_150** as the best model for the Transcribathon dataset. In Figure 8, a visual representation of the selected model is presented.

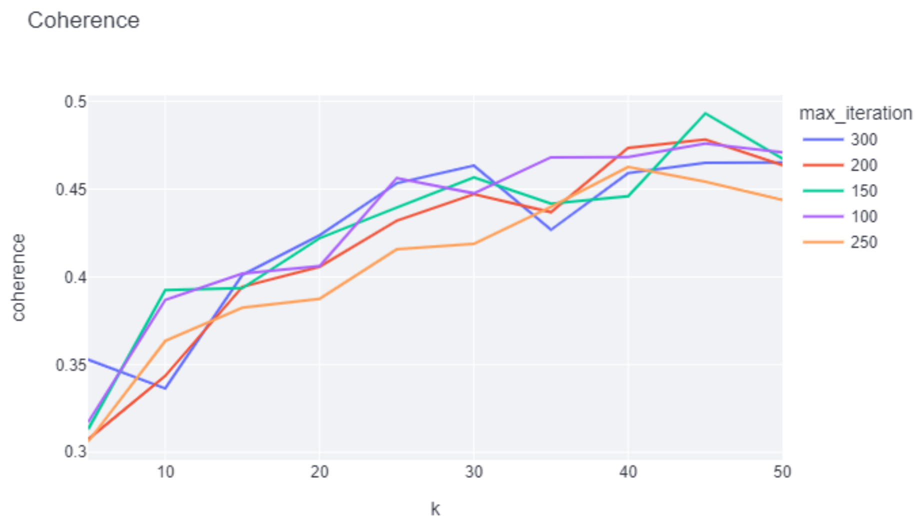


Figure 6. Coherence based on ***K*** (number of topics) and ***I***(number of iterations)

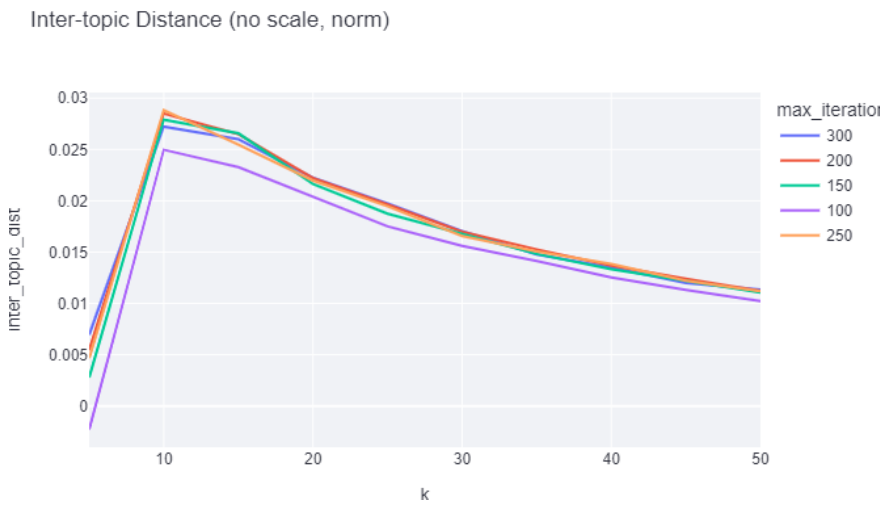


Figure 7. Normalized inter-topic Distance based on ***K*** (number of topics) and ***I*** (number of iterations)

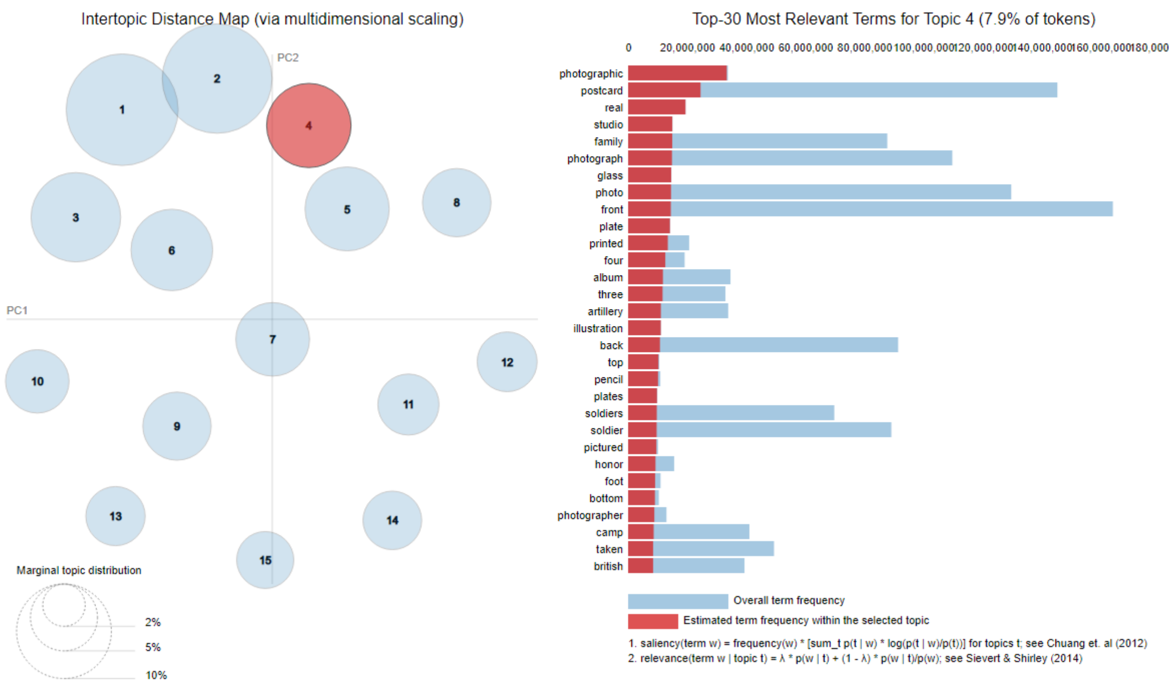


Figure 8. Visualization of **LDA_15_150** (15 topics and 150 iterations).

In Figure 9 the precision in top 10 is illustrated for each topic of the selected model. This shows that the proposed approach can still identify relevant documents, without the need to compute the LDA document-topic assignments for the documents available in external repositories. We can achieve a good recommendation precision in the top 10 by retrieving between 500 to 600 documents which contain topic terms.

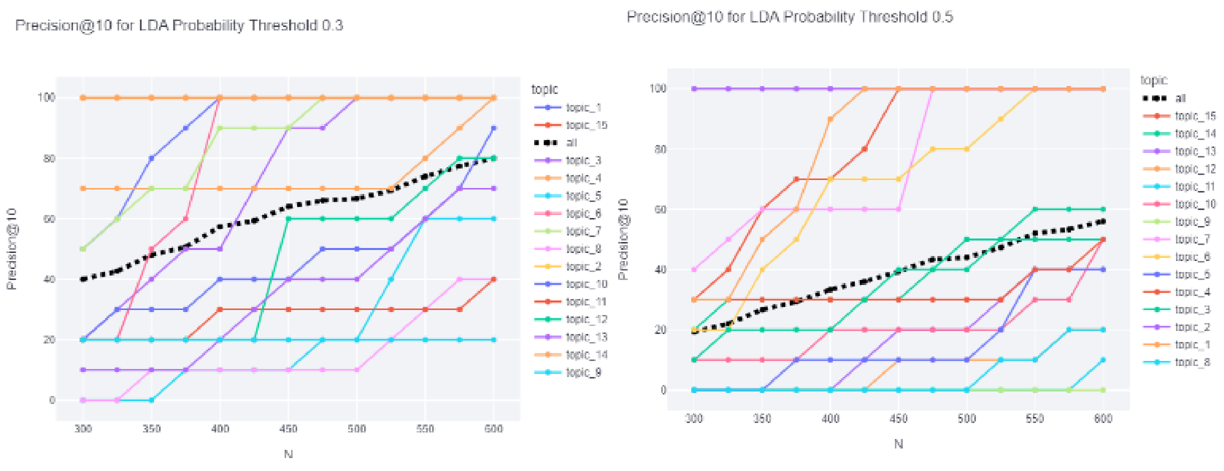


Figure 9. Precision@10 for each topic based on probability threshold of 0.3, respectively 0.5 and **N** in [300, 600]

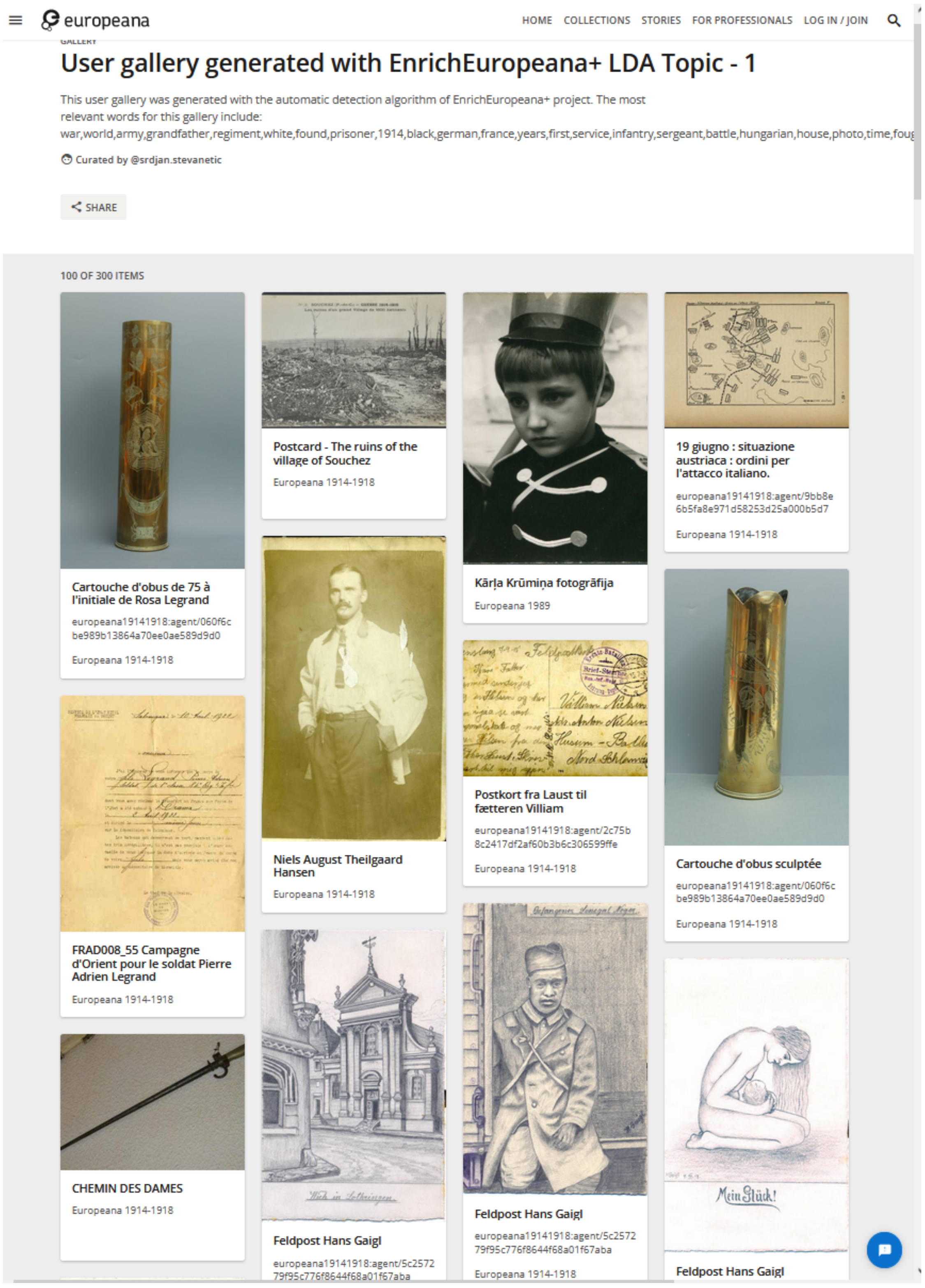
2.2 Visualization as User Galleries

The visualizations of topics presented in the previous sections are using the tools developed by researchers as means for assessing the experimental results from a technical perspective. They are using statistical information made available through the model learning process. Consequently, only information used as input for the machine learning algorithms is taken into account. Such visualizations are useful for data scientists to identify the most promising topic models, however, they are not able to estimate the effectiveness of the topics from the end user perspective.

Users are empowered to curate their own galleries in Europeana Portal[7]. This functionality is powered by the Europeana User Sets API[8] and it is available to be used upon request by the partners from Generic Services projects. By using the User Set API, the visualization of the learned topics was made available for interval verification by project partners in the Europeana portal. For each of the 15 topics generated with the selected algorithm, a user gallery was created including the top 300 Europeana records and the most relevant words associated with the topic (see Figure 10).

As a result of employing automatic translations, the topic detection algorithm is able to identify related europeana content independent from the language of the original metadata. This visualization will be used by the project partners to assess the opportunity of curating new galleries on more specific themes (e.g. postcards from the First World War, maps depicting war situations, etc.) in the Europeana portal.

[7] https://pro.europeana.eu/page/user-galleries
[8] https://docs.google.com/document/d/1bYjEb15RuxBeZRm2jKTBzQli_DVnRJk4_Jp9NYYtvK0/edit



GALLERY

User gallery generated with EnrichEuropeana+ LDA Topic - 1

This user gallery was generated with the automatic detection algorithm of EnrichEuropeana+ project. The most relevant words for this gallery include: war,world,army,grandfather,regiment,white,found,prisoner,1914,black,german,france,years,first,service,infantry,sergeant,battle,hungarian,house,photo,time,foug

Curated by @srdjan.stevanetic

100 OF 300 ITEMS

Cartouche d'obus de 75 à l'initiale de Rosa Legrand
europeana19141918:agent/060f6cbe989b13864a70ee0ae589d9d0
Europeana 1914-1918

FRAD008_55 Campagne d'Orient pour le soldat Pierre Adrien Legrand
Europeana 1914-1918

CHEMIN DES DAMES
Europeana 1914-1918

Postcard - The ruins of the village of Souchez
Europeana 1914-1918

Niels August Theilgaard Hansen
Europeana 1914-1918

Feldpost Hans Gaigl
europeana19141918:agent/5c257279f95c776f8644f68a01f67aba

Kārļa Krūmiņa fotogrāfija
Europeana 1989

Postkort fra Laust til fætteren Villiam
europeana19141918:agent/2c75b8c2417df2af60b3b6c306599ffe
Europeana 1914-1918

Feldpost Hans Gaigl
europeana19141918:agent/5c257279f95c776f8644f68a01f67aba
Europeana 1914-1918

19 giugno : situazione austriaca : ordini per l'attacco italiano.
europeana19141918:agent/9bb8e6b5fa8e971d58253d25a000b5d7
Europeana 1914-1918

Cartouche d'obus sculptée
europeana19141918:agent/060f6cbe989b13864a70ee0ae589d9d0
Europeana 1914-1918

Feldpost Hans Gaigl

Figure 10. Example of User Gallery generated using LDA Topic

2.3 Topic Management API

The Topic Management API, has the goal to offer support for curation and visualization of data records that describe similar content. Topic detection technology provides the means for building AI models and running them to evaluate which topics fit better to a given text (e.g. descriptions of europeana records). Still, the curators of Europeana user galleries or Transcribathon datasets are working on a higher level, working with the whole representation of cultural heritage materials (e.g. Europeana records or Trasncribathon stories). Consequently, the topic management API is implementing an endpoint which allows the integration of topic models into the curation processes available in Europeana or Transcribathon portal.

The API requests implemented within the topic management API are described in the following subsections, providing access to functionality that includes:

- Storing, updating and deletion of topic descriptions
- Search topics
- Detect topics for an object description

In order to support efficient use of topics learned through the experiments presented in the previous sections, the topic descriptions are imported into the runtime environment of enrichment services (see Section Create Topic Request). While the LDA method uses all terms found in the corpus (i.e. currently 10K+ and increasing when new documents are added to the corpus), we integrate an additional processing step to select only the most relevant terms for each imported topic.

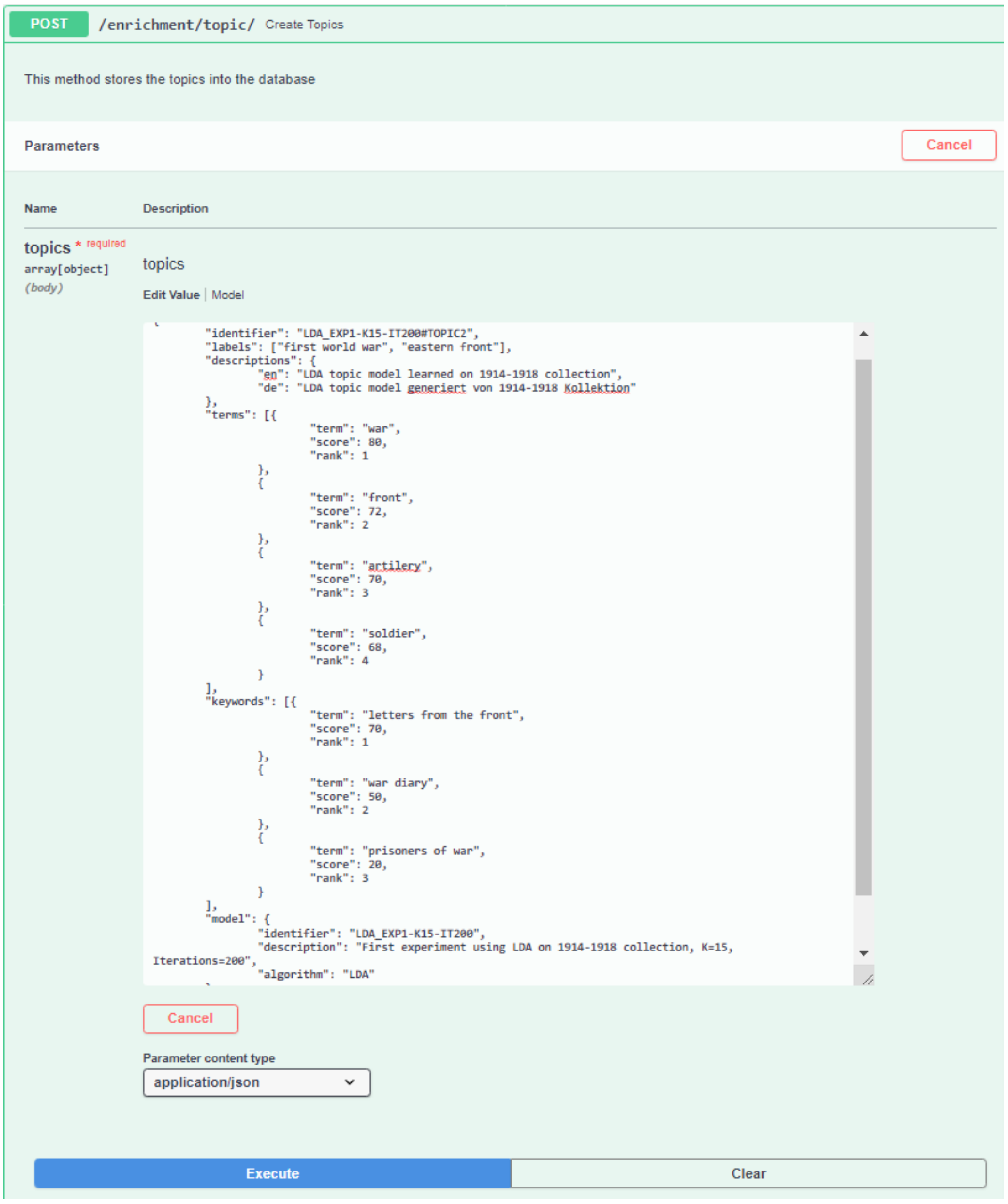
The relevance function indicated in equation (2) is used to identify the most relevant terms to be stored within the topic descriptions. The long tail of terms with low contribution to the topic is removed by retaining only the top 500 most relevant terms.

$$relevance(term\, w \mid topic\, t) = \lambda * p(w|t) + (1-\lambda) * p(w|t)/p(w) \qquad \textbf{(2)}$$

The relevance function adjusts the probability of a term in a topic to take into consideration also the frequency of the term across the corpora. Intuitively, for each term w in topic t, if w is very frequent in the corpus, then the relevancy of w to t is downgraded by a factor of λ.The optimal results were obtained for λ=0.6 which is used when exporting the topic terms. LDA models compute the probability distribution for all words used available in the dataset – **p(w)**, and for the words belonging to a topic – **p(w|t)**.

Create Topic Request

For storing a topic description into the Topic Management API, a specific endpoint is provided. By using a Http Post Request, a topic description can be submitted to be stored into MongoDB and Solr collections. The topic description is submitted in json format containing an externally generated identifier, a description, a list of labels, the list of topic terms together with their score and relevance rank, most frequent keywords associated with the Transcribathon items grouped in the topic, the topic model that created this topic (see also the topic learning session) and it's creation date. An example of the create topic request and the corresponding api response are presented in Figures 11 and 12, respectively.



/enrichment/topic/ Create Topics

This method stores the topics into the database

Parameters

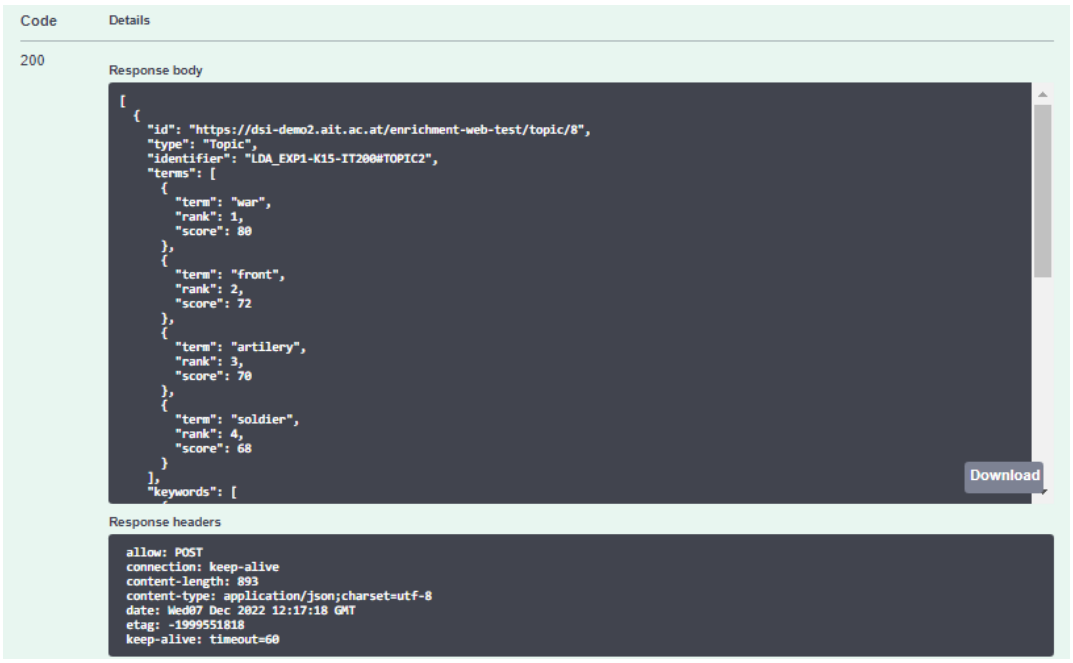
Name	Description
topics * required array[object] *(body)*	topics

Edit Value | Model

```
        "identifier": "LDA_EXP1-K15-IT200#TOPIC2",
        "labels": ["first world war", "eastern front"],
        "descriptions": {
                "en": "LDA topic model learned on 1914-1918 collection",
                "de": "LDA topic model generiert von 1914-1918 Kollektion"
        },
        "terms": [{
                        "term": "war",
                        "score": 80,
                        "rank": 1
                },
                {
                        "term": "front",
                        "score": 72,
                        "rank": 2
                },
                {
                        "term": "artilery",
                        "score": 70,
                        "rank": 3
                },
                {
                        "term": "soldier",
                        "score": 68,
                        "rank": 4
                }
        ],
        "keywords": [{
                        "term": "letters from the front",
                        "score": 70,
                        "rank": 1
                },
                {
                        "term": "war diary",
                        "score": 50,
                        "rank": 2
                },
                {
                        "term": "prisoners of war",
                        "score": 20,
                        "rank": 3
                }
        ],
        "model": {
                "identifier": "LDA_EXP1-K15-IT200",
                "description": "First experiment using LDA on 1914-1918 collection, K=15,
Iterations=200",
                "algorithm": "LDA"
```

Parameter content type: application/json

Figure 11. Insert topic request



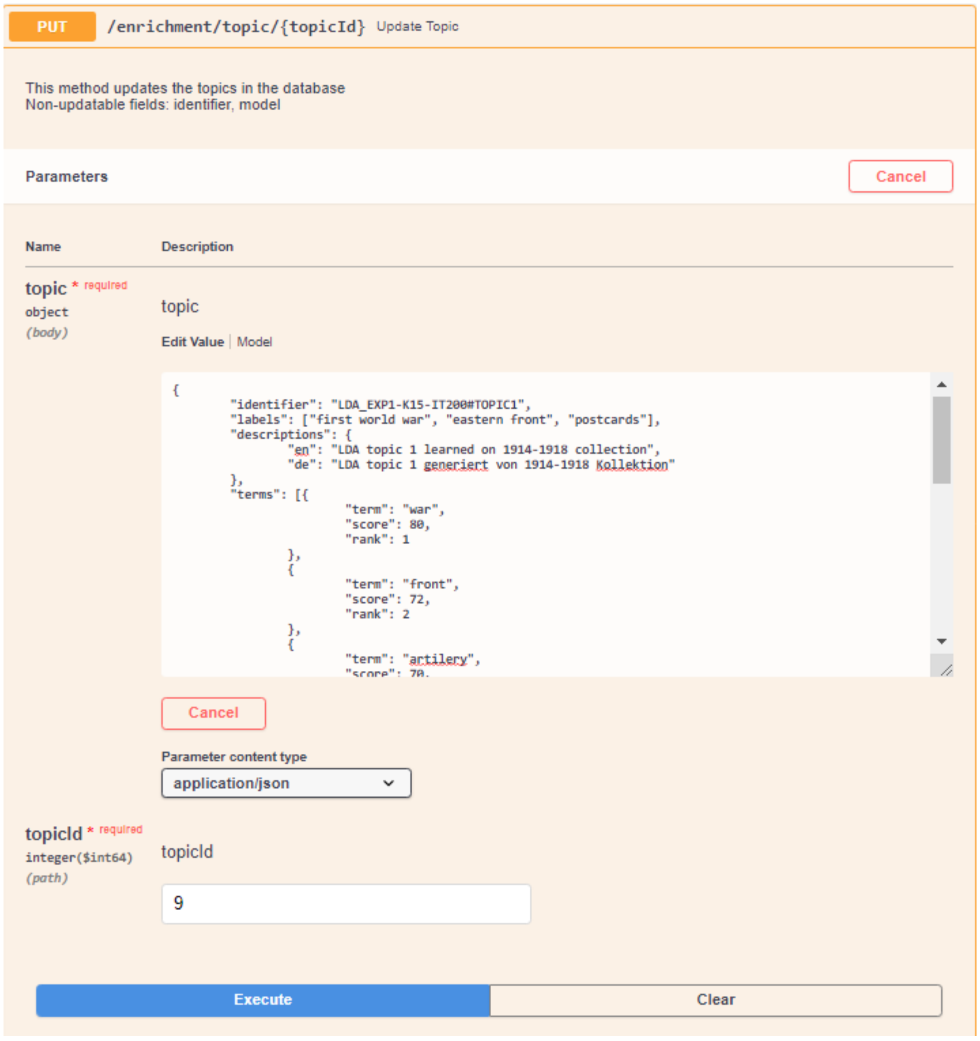
```
[
  {
    "id": "https://dsi-demo2.ait.ac.at/enrichment-web-test/topic/8",
    "type": "Topic",
    "identifier": "LDA_EXP1-K15-IT200#TOPIC2",
    "terms": [
      {
        "term": "war",
        "rank": 1,
        "score": 80
      },
      {
        "term": "front",
        "rank": 2,
        "score": 72
      },
      {
        "term": "artilery",
        "rank": 3,
        "score": 70
      },
      {
        "term": "soldier",
        "rank": 4,
        "score": 68
      }
    ],
    "keywords": [
```

```
allow: POST
connection: keep-alive
content-length: 893
content-type: application/json;charset=utf-8
date: Wed07 Dec 2022 12:17:18 GMT
etag: -1999551818
keep-alive: timeout=60
```

Figure 12. Insert topic response

Update Topic Request

Whenever an update of an existing topic description is required, this can be performed through the update topic request. This method has the same input as the create method, with the difference that the id of the topic to be updated is submitted through the url. An example of the request for updating the topic created in the previous section is presented in Figure 13. The result of the request is the updating of the description and the labels for the topic as shown in Figure 14.



PUT /enrichment/topic/{topicId} Update Topic

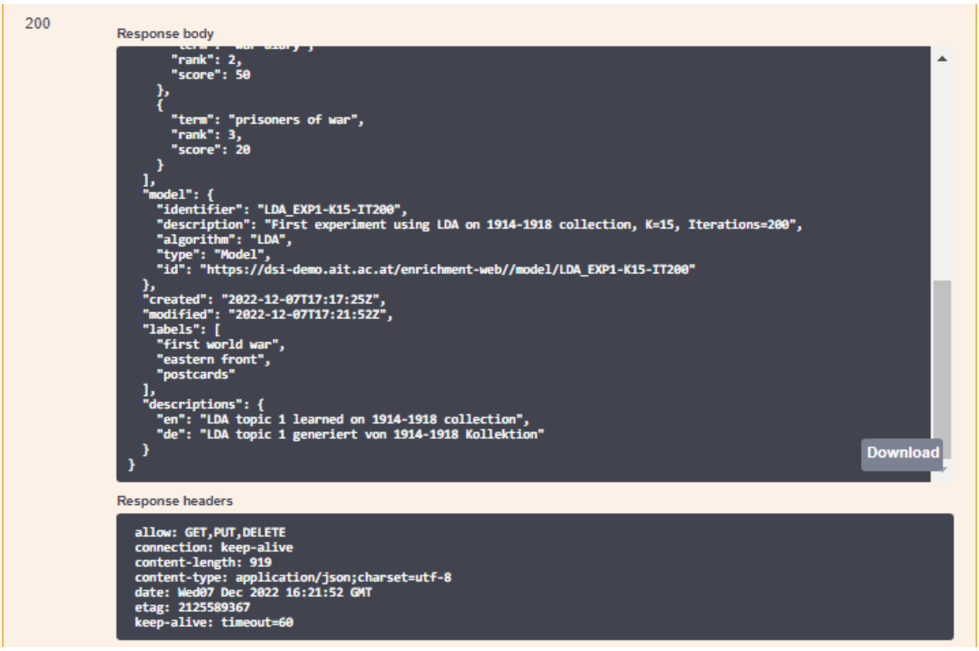
This method updates the topics in the database
Non-updatable fields: identifier, model

Parameters

Name	Description
topic * required object *(body)*	topic Edit Value | Model Parameter content type: application/json
topicId * required integer($int64) *(path)*	topicId 9

```
{
        "identifier": "LDA_EXP1-K15-IT200#TOPIC1",
        "labels": ["first world war", "eastern front", "postcards"],
        "descriptions": {
                "en": "LDA topic 1 learned on 1914-1918 collection",
                "de": "LDA topic 1 generiert von 1914-1918 Kollektion"
        },
        "terms": [{
                        "term": "war",
                        "score": 80,
                        "rank": 1
                },
                {
                        "term": "front",
                        "score": 72,
                        "rank": 2
                },
                {
                        "term": "artilery",
                        "score": 70,
```

Figure 13. Update topic request



200

Response body

```
      "rank": 2,
      "score": 50
    },
    {
      "term": "prisoners of war",
      "rank": 3,
      "score": 20
    }
  ],
  "model": {
    "identifier": "LDA_EXP1-K15-IT200",
    "description": "First experiment using LDA on 1914-1918 collection, K=15, Iterations=200",
    "algorithm": "LDA",
    "type": "Model",
    "id": "https://dsi-demo.ait.ac.at/enrichment-web//model/LDA_EXP1-K15-IT200"
  },
  "created": "2022-12-07T17:17:25Z",
  "modified": "2022-12-07T17:21:52Z",
  "labels": [
    "first world war",
    "eastern front",
    "postcards"
  ],
  "descriptions": {
    "en": "LDA topic 1 learned on 1914-1918 collection",
    "de": "LDA topic 1 generiert von 1914-1918 Kollektion"
  }
}
```

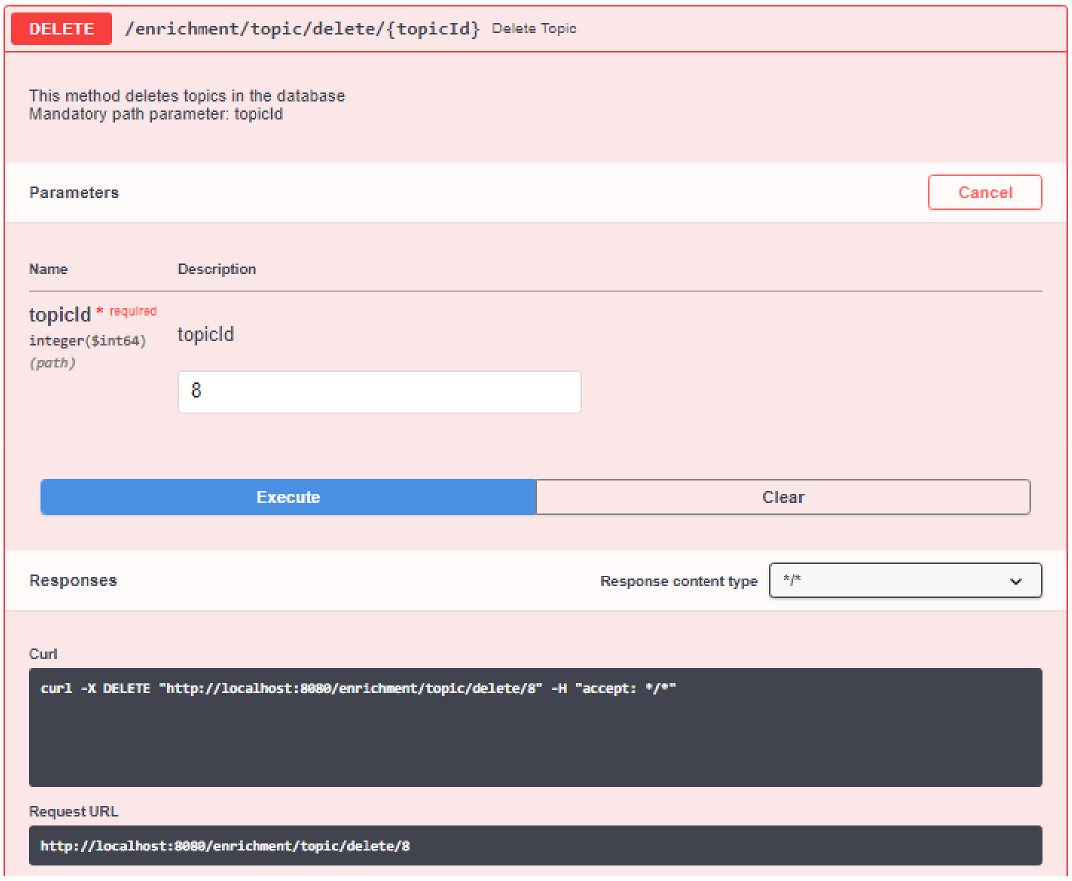
Response headers

```
allow: GET,PUT,DELETE
connection: keep-alive
content-length: 919
content-type: application/json;charset=utf-8
date: Wed07 Dec 2022 16:21:52 GMT
etag: 2125589367
keep-alive: timeout=60
```

Figure 14. Update topic response

Delete Topic POST Request

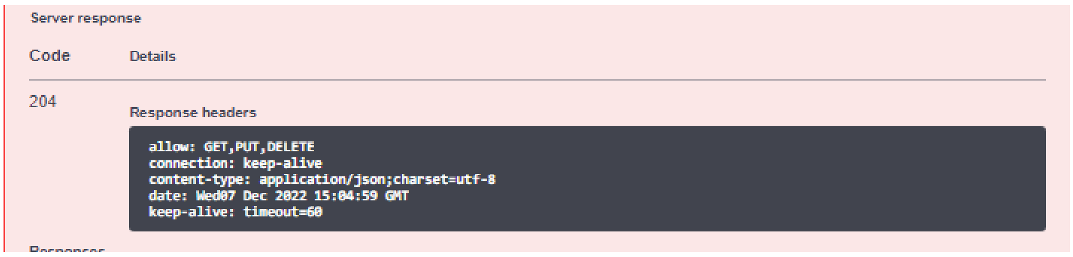
An existing topic stored in the database can be deleted by using its id. An example of the request for deleting the previously created topic is presented in Figure 15, while the response indicating the successful deletion is presented in Figure16, respectively.



```
curl -X DELETE "http://localhost:8080/enrichment/topic/delete/8" -H "accept: */*"
```

```
http://localhost:8080/enrichment/topic/delete/8
```

Figure 15. Delete topic request



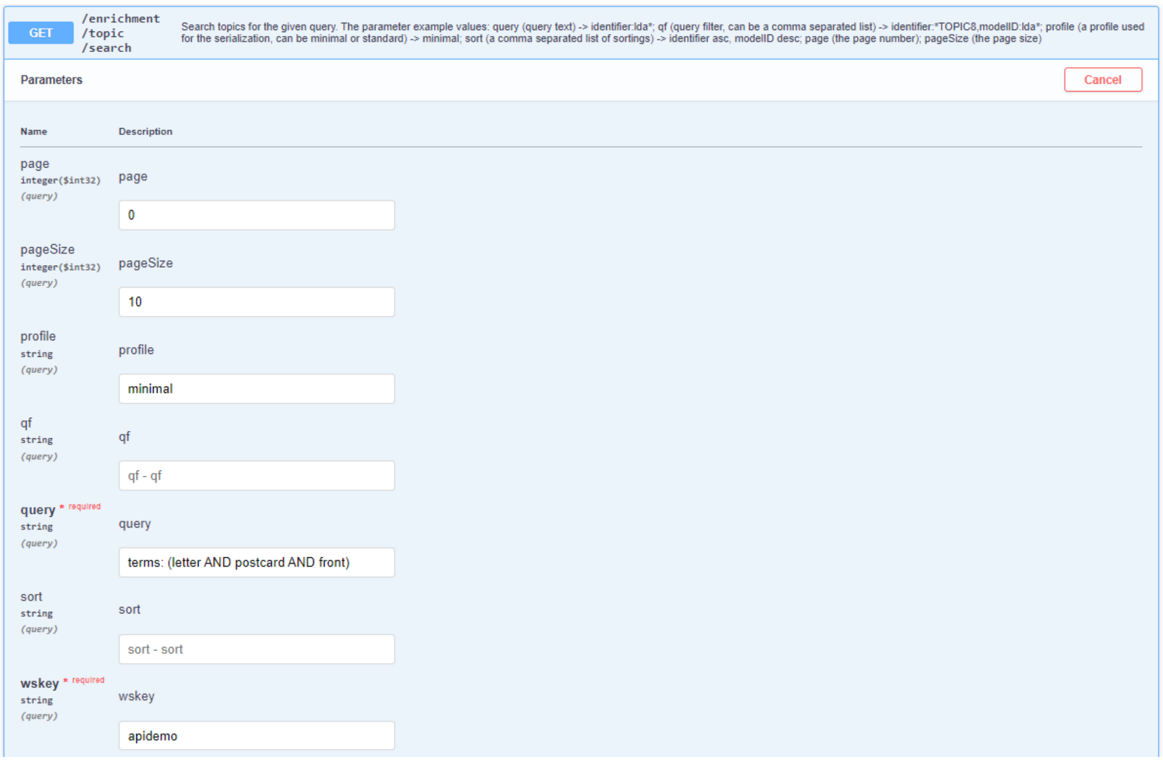
```
allow: GET,PUT,DELETE
connection: keep-alive
content-type: application/json;charset=utf-8
date: Wed07 Dec 2022 15:04:59 GMT
keep-alive: timeout=60
```

Figure 16. Delete topic response

Search Topics Request

This API offers the functionality to search topics by using Solr queries (either free text or using the Lucene search syntax). As the retrieval of the topic is based on their ids, which are typically not known to the users of the api, the search method is offering the main entry point to discovering the topics available in the database. The parameters of the search request are aligned with the common specifications for search methods in Europeana APIs[9]. Figure 17 shows the input parameters available for the search requests. The mandatory query parameter is used to provide solr search queries. In the provided example, the user is searching the ids of the topics (i.e. minimal profile) containing the terms: letter, postcard and front.

Figure 18 presents the corresponding response with the topic ids shown in the *items* property and the pagination information generated according to the *page* and *pageSize* parameters.


GET /enrichment /topic /search

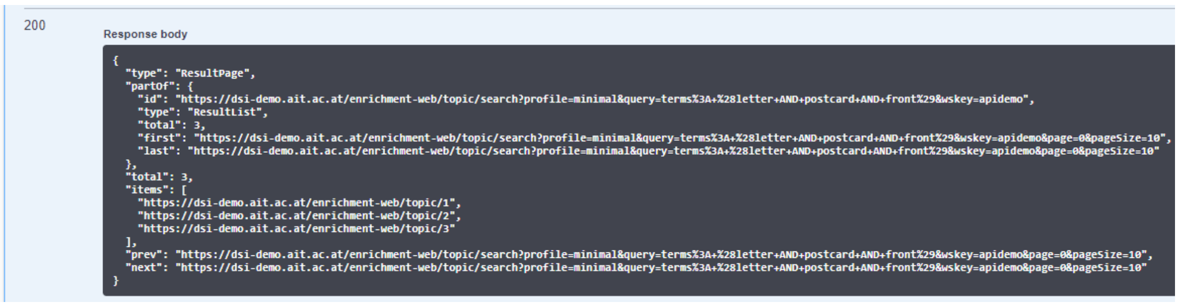
Search topics for the given query. The parameter example values: query (query text) -> identifier:lda*; qf (query filter, can be a comma separated list) -> identifier:*TOPIC8,modelID:lda*; profile (a profile used for the serialization, can be minimal or standard) -> minimal; sort (a comma separated list of sortings) -> identifier asc, modelID desc; page (the page number); pageSize (the page size)

Parameters

Name	Description
page integer($int32) (query)	page: 0
pageSize integer($int32) (query)	pageSize: 10
profile string (query)	profile: minimal
qf string (query)	qf: qf - qf
query * required string (query)	query: terms: (letter AND postcard AND front)
sort string (query)	sort: sort - sort
wskey * required string (query)	wskey: apidemo

Figure 17. Search topic request

[9]see https://docs.google.com/document/d/1UuzMvWE8AnjW1TOnv7uwiwxfZ8ADfmCCUEVjNu9JlkI/edit



200

Response body

```
{
  "type": "ResultPage",
  "partOf": {
    "id": "https://dsi-demo.ait.ac.at/enrichment-web/topic/search?profile=minimal&query=terms%3A+%28letter+AND+postcard+AND+front%29&wskey=apidemo",
    "type": "ResultList",
    "total": 3,
    "first": "https://dsi-demo.ait.ac.at/enrichment-web/topic/search?profile=minimal&query=terms%3A+%28letter+AND+postcard+AND+front%29&wskey=apidemo&page=0&pageSize=10",
    "last": "https://dsi-demo.ait.ac.at/enrichment-web/topic/search?profile=minimal&query=terms%3A+%28letter+AND+postcard+AND+front%29&wskey=apidemo&page=0&pageSize=10"
  },
  "total": 3,
  "items": [
    "https://dsi-demo.ait.ac.at/enrichment-web/topic/1",
    "https://dsi-demo.ait.ac.at/enrichment-web/topic/2",
    "https://dsi-demo.ait.ac.at/enrichment-web/topic/3"
  ],
  "prev": "https://dsi-demo.ait.ac.at/enrichment-web/topic/search?profile=minimal&query=terms%3A+%28letter+AND+postcard+AND+front%29&wskey=apidemo&page=0&pageSize=10",
  "next": "https://dsi-demo.ait.ac.at/enrichment-web/topic/search?profile=minimal&query=terms%3A+%28letter+AND+postcard+AND+front%29&wskey=apidemo&page=0&pageSize=10"
}
```

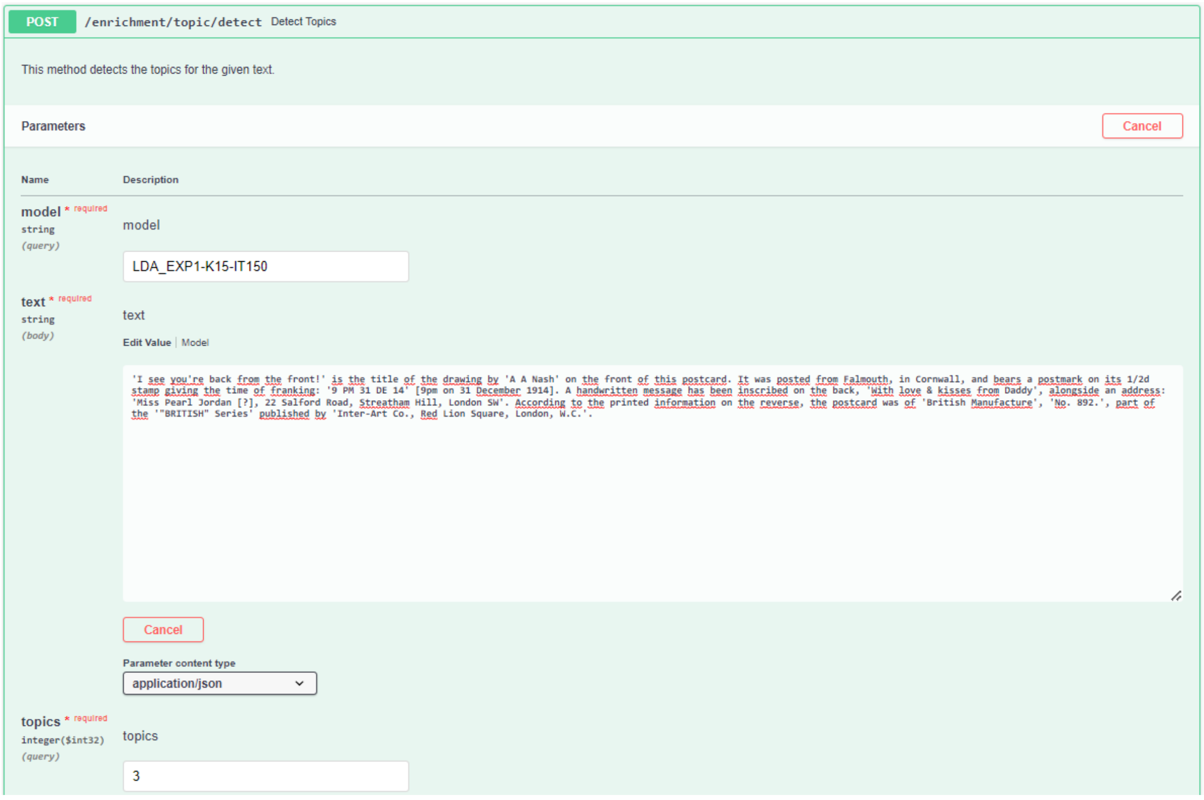
Figure 18. Search topic response

3.4 Using topics for content curation purposes

The topics identified by machine learning algorithms can be used for supporting content curation activities, like identifying new incoming documents if they are closely related to the existing topics, or actively searching for additional documents to include into a collection created through the use of automatically detected topics. Such use cases are described in the following subsections.

Topic Detection Request

The Topic Management API offers the functionality of verifying if a document description, which was not used in model learning, is related to the existing Topics. This service relies on an existing, previously learned LDA model. The required parameters are: *model* being the identifier of the previously learned model, *topics* representing the number of top topics to be included in the response, and a *text* for which the topic probabilities are computed using the given learned model. The response contains pairs consisting of the topic identifier with the associated probability. Figures 19 and 20 illustrate an example of the input parameters and the response respectively.



POST `/enrichment/topic/detect` Detect Topics

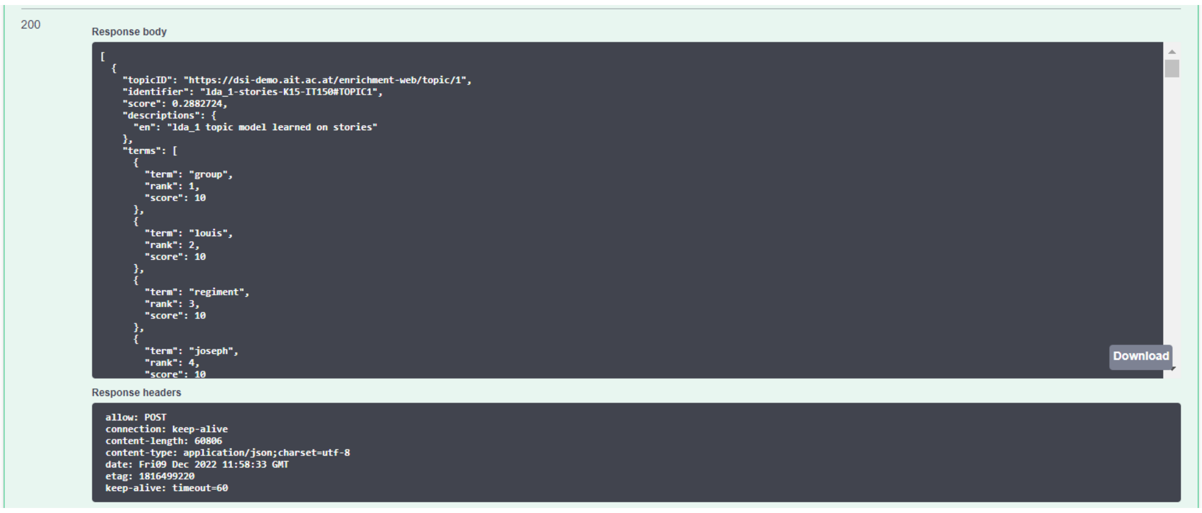
This method detects the topics for the given text.

Parameters

Cancel

Name	Description
model * required string *(query)*	model LDA_EXP1-K15-IT150
text * required string *(body)*	text **Edit Value** | Model 'I see you're back from the front!' is the title of the drawing by 'A A Nash' on the front of this postcard. It was posted from Falmouth, in Cornwall, and bears a postmark on its 1/2d stamp giving the time of franking: '9 PM 31 DE 14' [9pm on 31 December 1914]. A handwritten message has been inscribed on the back, 'With love & kisses from Daddy', alongside an address: 'Miss Pearl Jordan [?], 22 Salford Road, Streatham Hill, London SW'. According to the printed information on the reverse, the postcard was of 'British Manufacture', 'No. 892.', part of the '"BRITISH" Series' published by 'Inter-Art Co., Red Lion Square, London, W.C.'. Cancel Parameter content type: application/json
topics * required integer($int32) *(query)*	topics 3

Figure 19. Detect topic request



200

Response body

```
[
  {
    "topicID": "https://dsi-demo.ait.ac.at/enrichment-web/topic/1",
    "identifier": "lda_1-stories-K15-IT150#TOPIC1",
    "score": 0.2882724,
    "descriptions": {
      "en": "lda_1 topic model learned on stories"
    },
    "terms": [
      {
        "term": "group",
        "rank": 1,
        "score": 10
      },
      {
        "term": "louis",
        "rank": 2,
        "score": 10
      },
      {
        "term": "regiment",
        "rank": 3,
        "score": 10
      },
      {
        "term": "joseph",
        "rank": 4,
        "score": 10
```

Response headers

```
allow: POST
connection: keep-alive
content-length: 60806
content-type: application/json;charset=utf-8
date: Fri09 Dec 2022 11:58:33 GMT
etag: 1816499220
keep-alive: timeout=60
```

Figure 20. Detect topic response

Search by Topic in Europeana Collections

The goal of indexing the terms for each topic is to enable fast search for related CH objects in Europeana collections. A Solr query is generated using the topic terms, where the relevance score can be used for boosting individual search terms. Note that Solr computes the similarity between query terms and documents based on cosine similarity, which is not well correlated with the LDA model which uses conditional probabilities (which is currently not realizable using the standard Solr search). Consequently, recommending new documents from Europeana for a given topic follows a two-step approach. In the first instance, relevant documents are pre-selected by searching the repository and the LDA probabilities are computed for re-ranking the results. The documents with low LDA probability are removed from the recommendation list. This functionality was used to evaluate the effectiveness of LDA models to curate collections of objects from Europeana.

A serious limitation of this approach is related to the fact that the descriptions of Europeana records are not yet translated to English. For this reason, the functionality is only at experimental level, and it is not included in the topic management API. However, it is expected that the English description will be made available in Europeana in the coming years. Current efforts are invested to assess effectiveness of using the topic detection based on a more advanced BerTopic[10] approach, which offers also a better correlation with the free text search available in Europeana API.

3 Enhancements for existing enrichment services

A first version of the enrichment services was developed in the previous Action and the main functionality was documented in the specific project milestone[11]. In the following subsections we are presenting the enhancements developed within the scope of the EnrichEuropeana+ Action.

3.1 Machine Translation API

The challenges we face in the machine translations are manyfold, starting with outdated writing and constructs from historical texts, and continuing with mixed languages in the same document, spelling errors (Sajudis instead of Sąjūdis), very short texts and annotations, etc. To evaluate the quality of text translations on the documents used in the transcribathon Tool, we implemented support for an additional engine provided by DeepL translator[12] (support for Google translation and eTranslation[13] was implemented

[10] https://maartengr.github.io/BERTopic/index.html

[11] https://github.com/EnrichEuropeana/Technical-Documentation/blob/master/MS7_final_version_of_semantic_enrichment.pdf

[12] https://www.deepl.com/translator

[13] https://webgate.ec.europa.eu/etranslation/public/welcome.html

in the previous action). DeepL uses convolutional neural networks, and it is claimed to be more accurate than google translate on particular european languages (and particular datasets)[14]. The machine translation API acts as a proxy for the free version of the DeepL API with integrated authorization and caching mechanism which facilitates its use for the enrichment activities within the scope of the action.

3.2 Named Entity Recognition and Linking

The Named Entity Recognition and Linking (NERL) aims at identifying the named entities (e.g. places, persons) mentioned in natural language texts such as descriptions and transcriptions of the documents. Wikidata is the largest openly accessible repository of named entities, therefore, linking with Wikidata entities is the main task of the NERL service.

In the previous deliverable we provided a detailed description of the APIs, i.e. their parameters and graphical user interfaces, here we provide the details of the updated implementation of the NERL algorithm.

[14] https://techcrunch.com/2017/08/29/deepl-schools-other-online-translators-with-clever-machine-learning/

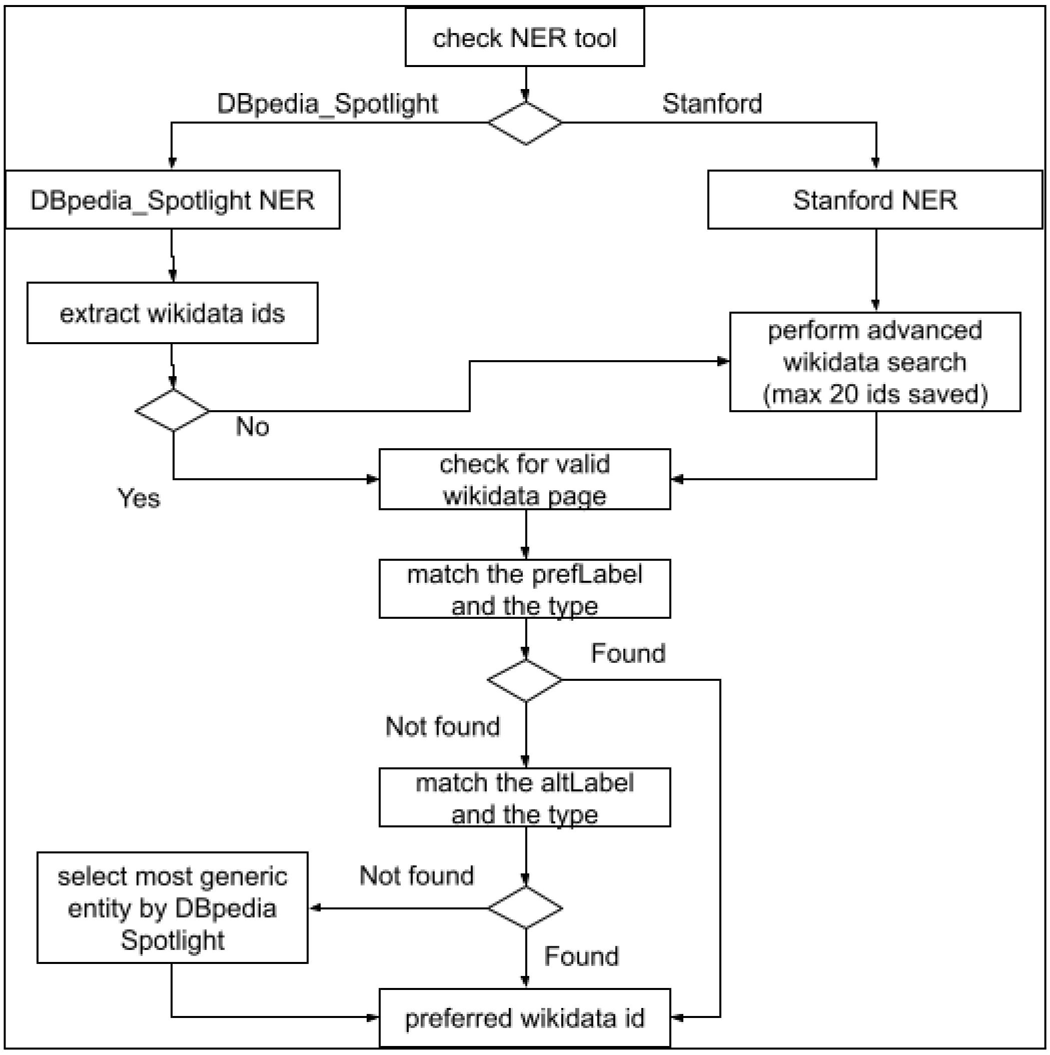
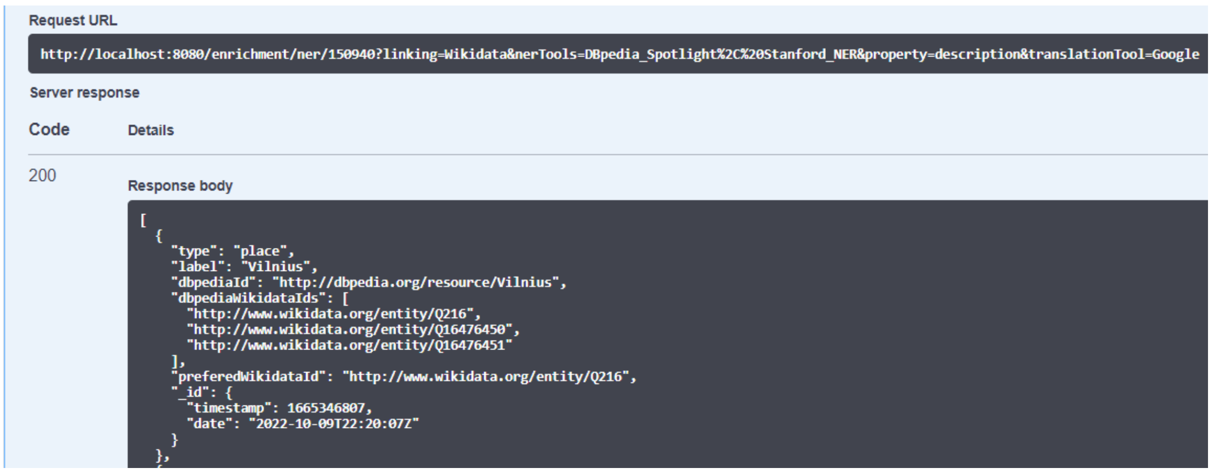

check NER tool
DBpedia_Spotlight
Stanford
DBpedia_Spotlight NER
Stanford NER
extract wikidata ids
perform advanced wikidata search (max 20 ids saved)
No
Yes
check for valid wikidata page
match the prefLabel and the type
Found
Not found
match the altLabel and the type
select most generic entity by DBpedia Spotlight
Not found
Found
preferred wikidata id

Figure 21. Named entity recognition and linking workflow

Figure 21 illustrates the steps of the NER algorithm. The effectiveness of the linking algorithm is improved with each of the individual steps used for discovering links between the named entities identified within the original text and the wikidata entities. First, based on the type of the NER tool (i.e. Stanford and/or DBpedia_Spotlight) the corresponding NER analysis is performed. As an output, a list of named entities comprising the entity label (e.g. Paris), its type (e.g. place) and the position in the analyzed text is returned. When using the DBpedia_Spotlight tool, the linking to Wikidata (i.e. discovery of Wikidata IDs) is computed in one step. In cases, when no Wikidata IDs are found, the linking using the Wikidata advanced search is performed. The same search is also performed when using Stanford NER, which does not provide any linking by itself. After retrieving the candidate entities for the linking process, the automatic validation process is started. First, it is verified that the wikidata URIs are referencing valid named entities (i.e. links to disambiguation pages or entities without descriptions are dropped). The valid wikidata ids are iterated to find the preferred entity to be used for the linking decision. If any of the wikidata entities has a preferred label (*prefLabel* wikidata field) and the type indicated by the NER tools (see details on the type matching follow below) it is selected for the preferred linking. Otherwise the matching with the alternative labels (*altLabel* wikidata field) is performed. If no entity matches the criteria, the first candidate by DBPedia Spotlight is selected as the preferred entity for linking.

In the following we present additional implementation details for the above mentioned steps. The advanced wikidata search[15] is used for discovering entities based on the labels discovered within document descriptions and transcriptions by Stanford NER. The advanced search incorporates multiple techniques for searching entities, including the matching on alternative labels, writing variants (e.g. Sąjūdis found in case of Sajudis), reverse order of person's names (e.g. Moore Lane instead of Lane Moore) and returns a ranked list of results. Since DBpedia is already providing the linking, we do not additionally perform the advanced search linking afterwards, and the priority in choosing the preferred wikidata id is given to the DBpedia wikidata ids, in which case if they are found, one of them is definitely selected as a preferred wikidata id. DBpedia Spotlight performs the context analysis in contrast to the entity search based on entity linking, additionally, the linking to wikidata entities is automatically performed.

By using the new implementation of the NERL workflow, we analyzed the english translations for the descriptions of all stories available in Transcribathon (i.e. 32K+ stories). 51,143 references to named entities were found. An example of the NERL output is presented in Figure 22. For all detected named entities, the position in the original text is stored in a separate database collection. The algorithm verifies first if the detected entities are already known to the system, otherwise the new entries are saved in the database. In case of DBpedia, named entities are considered to be the same based on the dbpediaId, while in case of Stanford NER, the label and the detected type is used.


Request URL

```
http://localhost:8080/enrichment/ner/150940?linking=Wikidata&nerTools=DBpedia_Spotlight%2C%20Stanford_NER&property=description&translationTool=Google
```

Server response

Code Details

200 Response body

```
[
  {
    "type": "place",
    "label": "Vilnius",
    "dbpediaId": "http://dbpedia.org/resource/Vilnius",
    "dbpediaWikidataIds": [
      "http://www.wikidata.org/entity/Q216",
      "http://www.wikidata.org/entity/Q16476450",
      "http://www.wikidata.org/entity/Q16476451"
    ],
    "preferedWikidataId": "http://www.wikidata.org/entity/Q216",
    "_id": {
      "timestamp": 1665346807,
      "date": "2022-10-09T22:20:07Z"
    }
  },
```

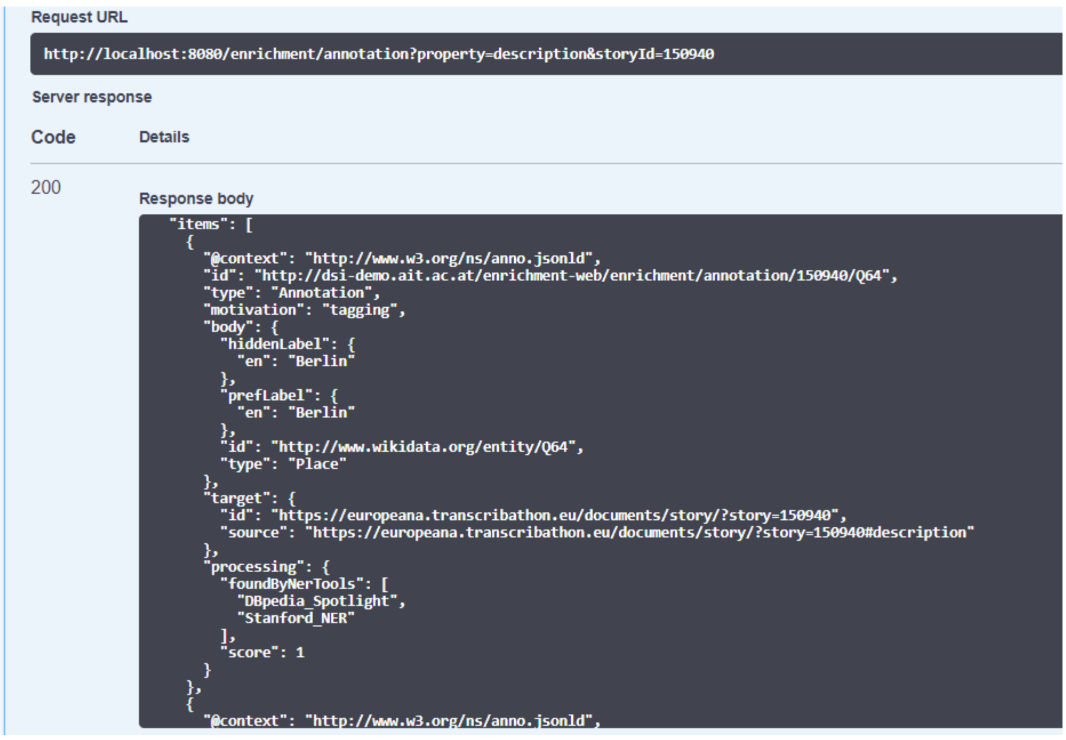
Figure 22. An example of a named entity

[15] https://www.wikidata.org/w/index.php?search=&title=Special:Search&profile=advanced&fulltext=1&ns0=1

3.3 Annotation Data Model

The NERL services exposed the complete information computed by the internal process, however, in terms of semantic enrichment of data records only the linking of the documents with named entities are relevant for the client applications. In order to support easy integration in transcribathon tool, the enrichments are made available using the W3C Web Annotations Data Model[16].

In this section we provide the details on the improved representation of enrichments collected as result of the NERL analysis.



Request URL

```
http://localhost:8080/enrichment/annotation?property=description&storyId=150940
```

Server response

Code Details

200 Response body

```
  "items": [
    {
      "@context": "http://www.w3.org/ns/anno.jsonld",
      "id": "http://dsi-demo.ait.ac.at/enrichment-web/enrichment/annotation/150940/Q64",
      "type": "Annotation",
      "motivation": "tagging",
      "body": {
        "hiddenLabel": {
          "en": "Berlin"
        },
        "prefLabel": {
          "en": "Berlin"
        },
        "id": "http://www.wikidata.org/entity/Q64",
        "type": "Place"
      },
      "target": {
        "id": "https://europeana.transcribathon.eu/documents/story/?story=150940",
        "source": "https://europeana.transcribathon.eu/documents/story/?story=150940#description"
      },
      "processing": {
        "foundByNerTools": [
          "DBpedia_Spotlight",
          "Stanford_NER"
        ],
        "score": 1
      }
    },
    {
      "@context": "http://www.w3.org/ns/anno.jsonld",
```

Figure 23. Annotation Data Model

Figure 23 shows a serialization of the *NamedEntityAnnotation* class object. The *id* field captures the story id (e.g. 150940) together with the wikidata identifier of the preferred wikidata id (e.g. Q216), assigned to the named entity during the NER linking analysis. The field *target* is of type *SpecificResource* and it captures the story for which the annotation is generated and the document fragment from which the enrichment is derived (i.e. *source* property). The field *body* captures the wikidata identifier, the type of the entity found in the NER analysis, the *body.hiddenLabel* field which is the label identified by the NER tool, and the *body.prefLabel* field which is the preferred label from the wikidata id page. The *processing* field is represented by the *Processing* class and it includes the NER tools used for computing the enrichment (i.e. the *foundByNerTools* field) and a *score* indicating the level of trust for the automatically computed enrichments. This level of trust is computed using the following scoring function:

[16] https://www.w3.org/TR/annotation-model/

$$score = 0.3 * stanfordNer() + 0.4 * linkedByDBpedia() + 0.3 * linkedByWikidataSearch()$$

The function is summing the contribution of individual NERL components to the final score. Currently we employ the boolean decisions for the $stanfordNer()$, $linkedByDBpedia()$, and $linkedByWikidataSearch()$ functions. Consequently the return value of these functions is 1 if the corresponding component is able to recognize or link entities, and 0 otherwise. Note that currently the *stanforNer()* return always value 1 for the detected named entities. As future work, we plan to investigate the reuse of probabilities delivered by AI models within the scoring function.

We computed the annotations for all story descriptions and discovered a total of 90132 references to named entities. Please note that different annotations may link the same entity to different records, but only one link to an entity is generated for a data record, even if the entity is mentioned multiple times..

4 Support Services

Additional services were developed to enhance individual steps in the metadata enrichment and enrichment validation processes. These services are not intended to be exposed for public use, the access being granted only for the developed tools and for the project partners. The language detection service and the keywords linking validation user interface fall into this category.

4.1 Language Detection Spark Service

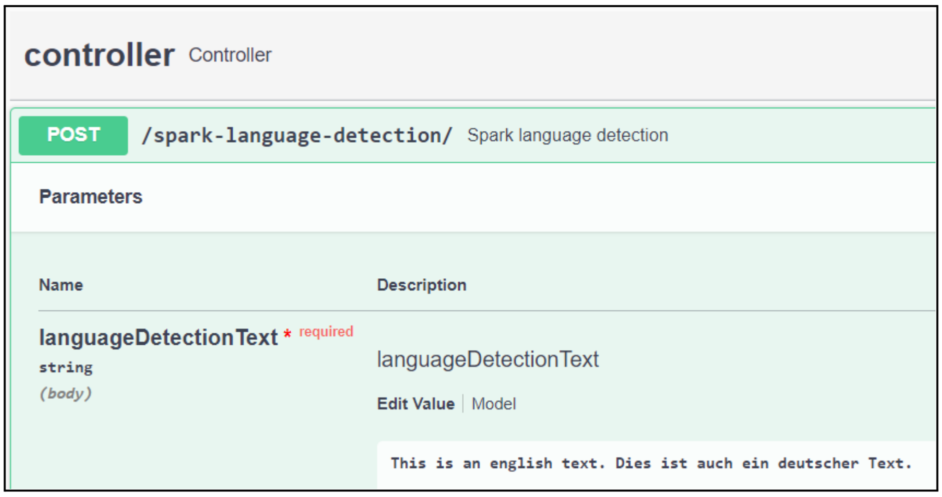
Some of the machine translation services require the submission of the language in which the text is written to compute the translation to English. Additionally, the machine translations are paid services, therefore the texts which are already written in the English language shouldn't be submitted for translation. However, in many cases, the text from story descriptions don't have a reliable language attribution, they have multiple language attribution or they even don't have it at all. As the translations to English are a prerequisite for the NERL workflow, we implemented an automatic language detection service (LDS) to address the language attribution issues.

The LDS service is designed with a REST API endpoint, which is a wrapper performing synchronous invocation of the language detection model executed within the spark environment. The core of the service is the *spark-nlp* library which implements the Deep Learning models trained on the large datasets from Wikipedia and Tatoeba and evaluated on the Europarl dataset[17].

Figure 24 shows the Swagger UI[18] of the LDS endpoint with an example for which the provided text is written in English and German languages. Figure 25 depicts the service

[17] https://nlp.johnsnowlabs.com/2020/12/05/detect_language_21_xx.html
[18]

response in which both the English and the German languages are detected with the same probability.



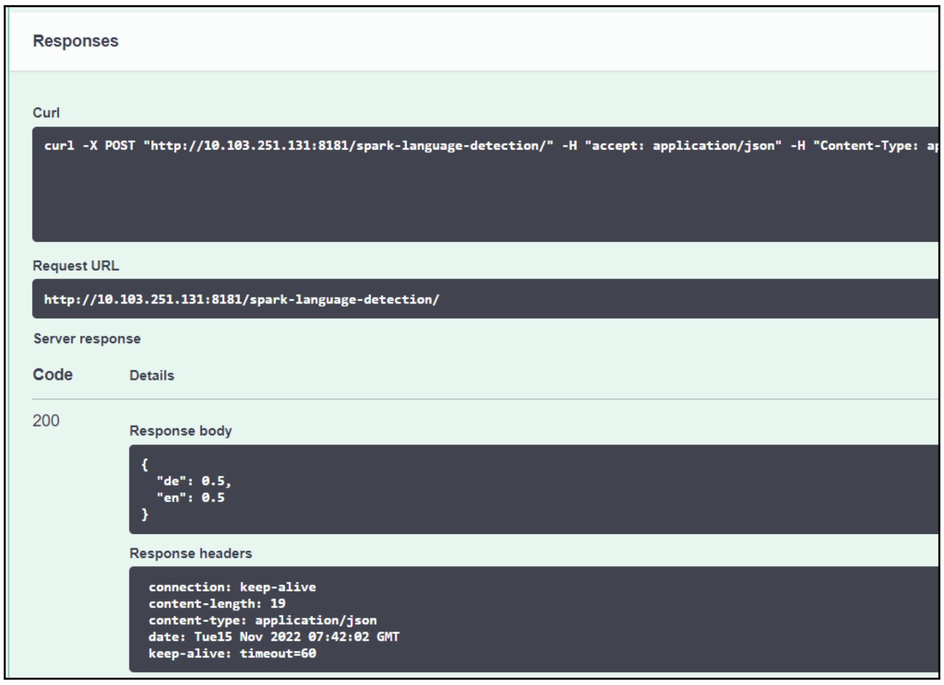
controller Controller

POST `/spark-language-detection/` Spark language detection

Parameters

Name	Description
languageDetectionText * required string (body)	languageDetectionText Edit Value | Model `This is an english text. Dies ist auch ein deutscher Text.`

Figure 24. Swagger UI for LDS with an example of a request



Responses

Curl

```
curl -X POST "http://10.103.251.131:8181/spark-language-detection/" -H "accept: application/json" -H "Content-Type: ap
```

Request URL

```
http://10.103.251.131:8181/spark-language-detection/
```

Server response

Code Details

200

Response body

```
{
  "de": 0.5,
  "en": 0.5
}
```

Response headers

```
 connection: keep-alive
 content-length: 19
 content-type: application/json
 date: Tue15 Nov 2022 07:42:02 GMT
 keep-alive: timeout=60
```

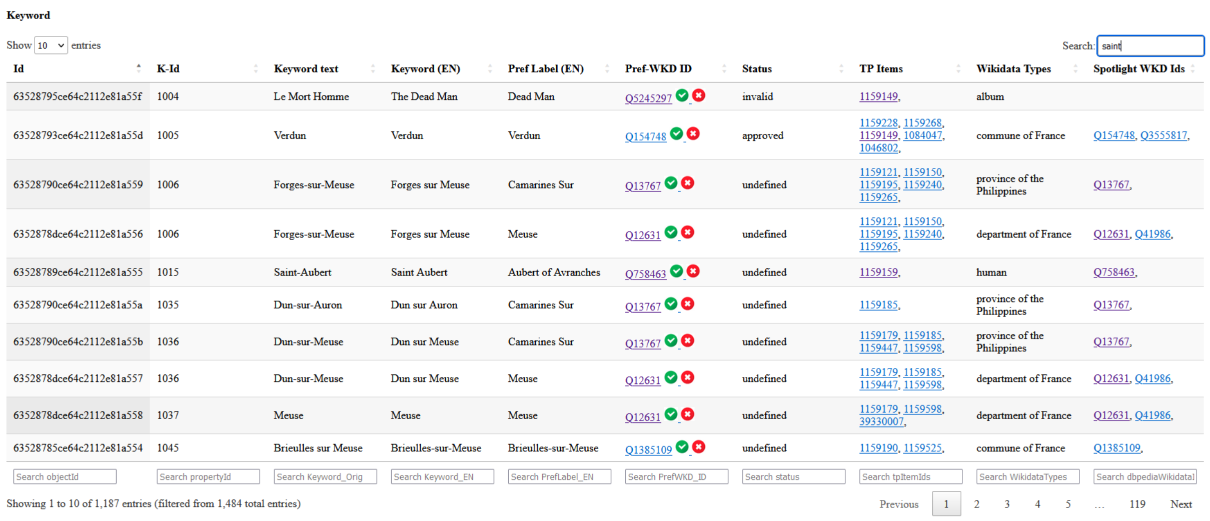
Figure 25. Swagger UI for LDS with a response from the request in Figure 24.

4.2 Linking Transcribathon keywords with Named Entities

The keywords assigned by end users to documents in Transcribathon platform carry important information related to the main messages and the subject of the documents. However, these keywords are represented by simple words in different European languages. By applying the NERL workflow we aim at transforming the existing keywords into named entities which have a more powerful meaning in terms of semantic enrichments. The machine translation service is employed to convert all keywords into their corresponding terms in English language, which is a prerequisite for successful execution of NERL workflow.

Named entity recognition approach is relying on the context in which the named entities are used, which in the case of keywords is often not available (i.e. the keywords are often not present in the document description, or they might be associated to very short textual annotations written on photos or postcards). Consequently, the automated linking approach is not able to provide a high precision, still it is able to provide important information which will allow end users to validate or reject the output of the algorithm.

By developing a graphical user interface we offer the required means to verify and validate the linking of keywords with wikidata entities (see Figure 26). The user interface is listing the automatically computed information and displays it to the end user for making a decision on their approval or rejection. The information computed in the intermediary steps such as the english translation (i.e. Keyword (EN) column) and the links discovered by the algorithm (i.e. Pref WKD ID and Spotlight WKD Ids) are displayed together with a brief information for the entity selected by the algorithm and the references from the Transcribathon platform (i.e. column TP Items links to the Trasncribathon documents which are tagged with the keyword). For the automatically detected wikidata entity (i.e. Pref WKD ID) we also display the english preferred language (i.e. PrefLabel (EN)) and the types associated with the Entity in wikidata. The buttons displayed near the automatically selected wikidata Id allows the end user to approve or reject the linking, which is reflected in the update of the status. Approved wikidata links can be used to enrich the records with semantic entities.


Keyword

Show 10 entries

Search: saint

Id	K-Id	Keyword text	Keyword (EN)	Pref Label (EN)	Pref-WKD ID	Status	TP Items	Wikidata Types	Spotlight WKD Ids
63528795ce64c2112e81a55f	1004	Le Mort Homme	The Dead Man	Dead Man	Q5245297 ✓ ✗	invalid	1159149,	album	
63528793ce64c2112e81a55d	1005	Verdun	Verdun	Verdun	Q154748 ✓ ✗	approved	1159228, 1159268, 1159149, 1084047, 1046802,	commune of France	Q154748, Q3555817,
63528790ce64c2112e81a559	1006	Forges-sur-Meuse	Forges sur Meuse	Camarines Sur	Q13767 ✓ ✗	undefined	1159121, 1159150, 1159195, 1159240, 1159265,	province of the Philippines	Q13767,
6352878dce64c2112e81a556	1006	Forges-sur-Meuse	Forges sur Meuse	Meuse	Q12631 ✓ ✗	undefined	1159121, 1159150, 1159195, 1159240, 1159265,	department of France	Q12631, Q41986,
63528789ce64c2112e81a555	1015	Saint-Aubert	Saint Aubert	Aubert of Avranches	Q758463 ✓ ✗	undefined	1159159,	human	Q758463,
63528790ce64c2112e81a55a	1035	Dun-sur-Auron	Dun sur Auron	Camarines Sur	Q13767 ✓ ✗	undefined	1159185,	province of the Philippines	Q13767,
63528790ce64c2112e81a55b	1036	Dun-sur-Meuse	Dun sur Meuse	Camarines Sur	Q13767 ✓ ✗	undefined	1159179, 1159185, 1159447, 1159598,	province of the Philippines	Q13767,
6352878dce64c2112e81a557	1036	Dun-sur-Meuse	Dun sur Meuse	Meuse	Q12631 ✓ ✗	undefined	1159179, 1159185, 1159447, 1159598,	department of France	Q12631, Q41986,
6352878dce64c2112e81a558	1037	Meuse	Meuse	Meuse	Q12631 ✓ ✗	undefined	1159179, 1159598, 39330007,	department of France	Q12631, Q41986,
63528785ce64c2112e81a554	1045	Brieulles sur Meuse	Brieulles-sur-Meuse	Brieulles-sur-Meuse	Q1385109 ✓ ✗	undefined	1159190, 1159525,	commune of France	Q1385109,
Search objectId	Search propertyId	Search Keyword_Orig	Search Keyword_EN	Search PrefLabel_EN	Search PrefWKD_ID	Search status	Search tpItemIds	Search WikidataTypes	Search dbpediaWikidata]

Showing 1 to 10 of 1,187 entries (filtered from 1,484 total entries)

Figure 26. Keyword data model for linking it with wikidata

5 Improving the quality of Europeana records

Different enrichment services developed within the course of the actions were employed to compute automatic enrichments for the Europeana records. In the following we describe how the machine translation, NERL and HTR technologies were used to compute automatic enrichments in form of metadata translation, linking with named entities, generation of document transcriptions.

Enrichment of existing UGC materials from Europeana 1914-1918 Collection. The enrichment services are used to enrich Europeana records through different channels. The main use is through the Transcribathon tool, where the automatically computed enrichments are displayed for manual validation by end users. However, additional requirements were identified during the implementation of the Action such as the need to reingest the 14-18 collection to replace the image files with their IIIF representation and the improvement of the metadata with language tags and English translation of the object description. The language tags and the English translation for the story descriptions were computed with the use of the machine translation service, and the UGC materials from the Europeana 1914-1918 collection are now prepared for reingestion into Europeana. Additionally the English translation of the description for all stories available in Transcribathon tool were computed and they are prepared for being displayed to the end users.

Automatic semantic enrichment for all Transcribathon stories. In the previous action the computation of the semantic enrichments was focusing on the transcription text and the descriptions contributed by end users on the individual items. Within the scope of Activity 2, the implementation was extended to compute enrichments based on the text available in the description of the Transcribathon stories (i.e. which corresponds to the description of Europeana records).

The crowdsourcing campaign organized by the Action aims at collecting transcriptions and manual metadata enrichments for the content contributed by the project partners. The keywords attached by the users to the transcribed documents contain important information for classification and understandability of these materials, however they are not yet considered semantic enrichments as they are not explicitly linked to semantic entities. Therefore, the named entity recognition service was applied in order to link previously contributed keywords with Wikidata named entities.

For the total number of 31,957 story descriptions, 51,143 named entities were discovered by the 2 NLP tools: Stanford and DBpedia_Spotlight. Out of those 22,945 entities are successfully linked with wikidata, 7,702 linked entities are of type person, and 15,243 of type place. The keywords used to enrich documents in the Transcribathon tool were analyzed as well. From a total of 6,236 keywords, references to 1,484 named entities were detected, 1,241 of them being linked with wikidata entities, 276 of type agent, and 965 of type place.

Automatic transcription using HTR technology. Manual transcriptions collected within the transcription campaign are also used for enhancing the automated transcription technology. Concretely, the transcriptions for the materials contributed by Dublin City Council and State Archives in Zagreb were used to generate new HTR models, which are better suited for generating automatic transcriptions for the large part of these collections which were not manually transcribed.

For the training of the handwriting model for the Dublin City Council minutes, 324 images of manuscript pages, already transcribed in Transcribathon, were imported into Transkribus. The available plain text transcriptions were matched to the layout detected by Transkribus and yielded a corpus with 10.873 transcribed lines and 55.902 words, where about 95% were used as training data and the rest as validation data.
Two trainings were carried out on the basis of this data using the PyLaia HTR engine[19]:

- The first training was started from scratch and stopped at 2,7% character error rate (CER) on the training set and 4,7% CER on the validation set.
- For the second training, an available model ("Transkribus English Handwriting M3", trained on 2.125.253 of miscellaneous English writing from the 18-19th century) was used as a starting point. By tuning it to the material at hand, the new model produced stopped at 1% CER on the training set and 3,61% CER on the validation set. The lower error rates suggest this model as a candidate for generating the transcriptions for an upcoming Transcribathon run.

The State Archives in Zagreb provided a set of 2.448 ration cards to be transcribed in Transkribus and used for training a handwriting model. This material includes tabular forms and a third-party company, Bonsai Tech[20], identified the coordinates of the table cells including handwriting beforehand. Both the scanned cards and the region information were imported into Transkribus. A webinar was carried out to teach employees of the State Archives Zagreb how to produce training data for this material using Transkribus Lite[21].
To the date of this writing all cards have been transcribed (42.349 table cells including 47.604 words) and the model training for this corpus is in preparation.

[19] https://github.com/jpuigcerver/PyLaia
[20] https://bonsai.tech/
[21] https://lite.transkribus.eu/

6 Conclusions

This document presents the work carried out for achieving the milestone 8 of EnrichEuropeana+ Action. It presents the new developments in the enrichment APIs, including the newly developed topic management service, the enhancements for the NERL services developed in the previous action, and the internal services employed in the semantic enrichment workflows. The system architecture was designed to be scalable, able to process large amounts of data.

These services, together with the HTR technology presented in Milestone 4 of the Action, are used to compute automatic semantic enrichments for Europeana records. The enrichment use cases and the results obtained so far for the selected Europeana collections are presented in a dedicated section of the document. In the next months of the project we will focus on the integration of these services in the workflow of the Trascribathon tool and for transferring validated enrichment into the Europeana platform.