Narrative Annotation of Content for Cultural Legacy Preservation

Pierrick Bruneau(✉)

LIST, 5 Avenue des Hauts-Fourneaux, 4362 Esch-sur-Alzette, Luxembourg
pierrick.bruneau@list.lu

Abstract. An important, yet underestimated, aspect of cultural heritage preservation is the analysis of personal narratives told by citizens. In this paper, we present a server architecture that facilitates multimedia content storage and sharing, along with the management of associated narrative information. Via the exposition of a RESTful interface, the proposed solution enables the collection of textual narratives in raw form, as well as the extraction of related narrative knowledge. We apply it to a corpus related to the time of the European construction in Luxembourg. We disclose details about our conceptual model and implementation, as well as experiments supporting the interest of our approach.

1 Introduction

Collecting and collating personal views and stories of citizens is important in view of preserving the cultural heritage of a time and place. While the wide availability of social media will facilitate this work for future generations when they analyze our times, such means are not available for e.g. the European construction period (roughly 1945–1975). Adapted tools are needed to collect such testimonies by elderly people, as well as facilitate their collation and dissemination.

In this paper we cast our attention towards the time of the European construction in Luxembourg and the surrounding region. This work has been conducted in the context of a funded project in collaboration with elderly people organizations. In this context, witnesses of the time frame of interest (aged between 75 and 85 years old) have been interviewed in French, and their testimonies have been transcribed. An additional French corpus extracted and transformed using online platforms has also been created and studied - more details about corpora may be found in Sect. 4.

Eventually, the collected data is merely more than a set of short texts. To make this narrative data actionable, e.g. allow effective indexing, browsing, or exploitation by web applications, knowledge extraction techniques are needed in order to build relevant metadata from these stories, such as people, places and time frames involved. Hence in this paper we focus on the means to store, process and access such narrative information. More precisely, a dedicated data model and a back-end server are needed in order to model and store the collected stories.

J. Filipe et al. (Eds.): DATA 2017, CCIS 814, pp. 111–127, 2018.
https://doi.org/10.1007/978-3-319-94809-6_6

The rest of this article is organized as follows. Firstly, related work about narrative structures and knowledge is discussed in Sect. 2. Then, in Sect. 3, we define a data model appropriate for storing the collected content, jointly to its narrative metadata extracted by manual and automatic means. We disclose an API that facilitates the manipulation of this model. Next, Sect. 4 describes how knowledge extraction means can use the proposed API in order to annotate the collected content. The collected data is introduced, and the interest of our approach is supported by annotation examples.

2 Related Work

The study of the structure of narratives and stories has been applied to a variety of domains, e.g. emergency response [1], situational awareness [2], or collections of historical documents [3]. A major concern in this domain is to bridge the gap between raw text (i.e. the form under which testimonial stories are generally acquired) and structured information with semantic value, that would enable story linking and advanced queries.

Associating properties to entities and composing them is the core concern of ontology engineering. Well known ontologies include YAGO [4], the Google Knowledge Graph [5], and DBpedia [6]. These ontologies are often used in conjunction with controlled vocabularies such as Dublin Core [7] or FOAF [8], that facilitate the interoperability of data sources, most notably via the RDF representation language.

Rather than an ensemble of unrelated facts, narration implies relationships between atomic facts or events. [1] define a taxonomy of such links (compositionality, causality, correlation and documentation). These links are relevant to our context, but they consider events at a coarse level. Similarly to [3,9], they are mostly concerned by interoperability between different ontologies. Likewise, [2] emphasize link types, with little consideration of specific data structures for events. With close resemblance to the CIDOC CRM [3,10] define explicitly roles (also known as facets in [11]) applying to events (e.g. actors, dates and locations), which are appropriate for historical events in a broad sense (e.g. the French Revolution in [3]), but not for events as constituents of a subjective narrative. The objective is then to propose a standard metadata description space for historical artifacts, rather than exploring the structure of narration.

The contributions by [12] are the most closely related to our work. In their *Narrative Knowledge Representation Language* (NKRL), they define data structures and controlled vocabularies of predicates and links to support the analysis of non-fictional and factual narratives. They avoid the use of the term *story* as it has led to ambiguities in the literature. Rather, They define a set of events and facts as the *fabula*. The *plot* level adds chronological, logical and coherence links between events. The *presentation* level is about the form in which plots are shown. Some related work in narrative analysis and storytelling is concerned with mapping arbitrary stories to a classical narrative structure [13,14]. In our work, stories are potentially made of anecdotal testimonies, and as such cannot be expected to match these structures. More abstract properties, such as sentiment attached to stories, were also extracted in [15] in order to analyze the structure of books.

In [16], a simplified version of the model by [12], and premises of its implementation are presented. However, in the latter the narrative knowledge is laid out in the same schema as entities describing pieces of content. This complicates the extraction of narrative knowledge by multiple agents, e.g. algorithms or human annotators. In the multimedia processing literature, when establishing benchmarks for evaluation, it is common to decorrelate the managed content from its annotations [17]. In the context of the present paper, narrative knowledge could thus be considered as a special kind of annotations.

The way narrative entities, relationships and predicates are extracted is seldom considered in the literature reported above. Some authors explicitly assume that this mapping has to be performed manually [11], or via crowdsourcing [18]. Wikipedia page structure has also been exploited in [4]. Alternatively, a term-based heuristic is used in [19] to determine links between events, and the use of *Natural Language Processing* (NLP) techniques such as *Named Entity Recognition* (NER) to automatically extract facts and events has been evaluated in [3,20]. Entity types in event models such as SEM [2] are closely related to types extracted by standard NER methods such as [21] (e.g. people, locations, dates).

3 Narrative Annotation System Description

The model described in this section answers to complementary concerns: the formalization of narrative knowledge, and the management of multimedia content. The latter is a crucial aspect, as the content is the core set of legacy objects that are to be preserved (e.g. textual or audio testimonies, photographs).

On the other hand, we aim at facilitating high-level semantic tasks over the managed content. For example, users should be able to find content that relates to a given place or person, as well as find redundant or contradictory events. An adapted narrative knowledge model is thus necessary.

With regard to these context and constraints, we define a two-level model:

- The first level focuses on narrative modelling, with the definition of real-world entities such as places, people or times, as well as possible relationships among these structures. It elaborates on previous work on the topic [12,16]. We present it in Sect. 3.1.
- The second level is focused on content description. Aside wrappers to the managed pieces of content (e.g. stories, audio, images), it defines the *annotation* data structure, that is the core articulation between the abstract semantic model and the managed content. This facet of our system is disclosed in Sect. 3.2.

Section 3.3 focuses on the API that enables access to these models by tier applications.

3.1 Narrative Description Model

The role of the narrative data model exposed in this section is to describe entities such as places and people, and relationships that exist between these. Figure 1 shows the proposed model. We consider that instances from this model exist *per se*, i.e. independently of a specific piece of content.

The data model in Fig. 1 is heavily inspired by the model underlying NKRL [12], but exhibits decisive distinctions. The proposed structure was designed with flexibility in mind. For example, it enables partial specification - a typical narrative may occasionally omit spatial and/or temporal specifications. Notably, the proposed *TimeSpan* format allows loose specification, with all fields except year being optional. Both points and intervals in time can be described with the same format, simply by equaling *From* and *To* respective fields.

The layout of the schema in Fig. 1 emphasizes the complexity of narrative data: both simple (e.g. a person pictured in a multimedia document) and complex (e.g. plot structure in a story text) entities are supported.

In Fig. 1, we note that fields can directly hold references to other entities. This feature is considered in abstract terms for now. The means for its implementation are discussed at length in Sect. 4. When a field may feature several entity types, the generic *Entity* is indicated. It may be thought as the root *Object* type in some object-oriented programming languages, as all entity types in Fig. 1 derive directly from it.

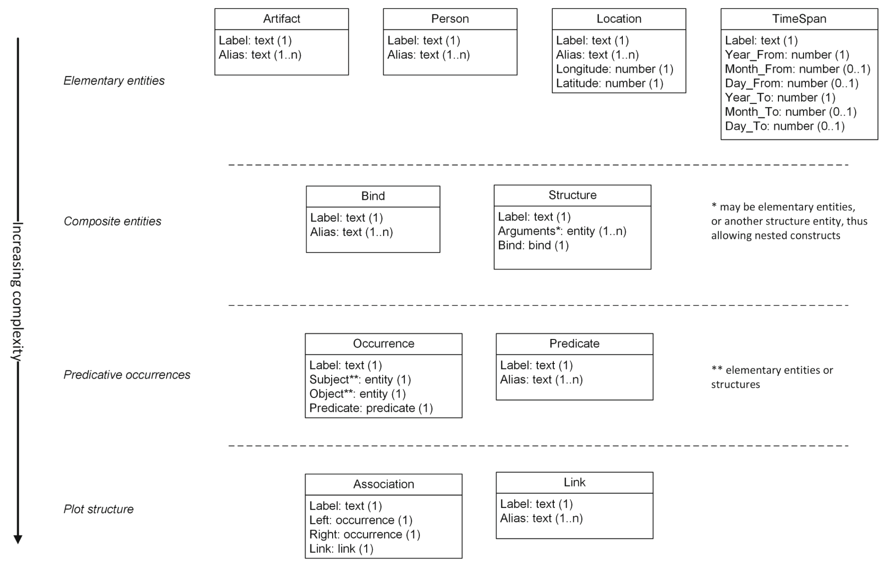


Fig. 1. Narrative data model. From top to bottom, entities are ordered by increasing complexity.

Entities from types depicted in Fig. 1 exist as single instances in the database. The *Alias* fields enable a single underlying instance to be designated by multiple writings. It is then up to the user managing the database (or knowledge extraction algorithms) to establish which alternate writings refer to which identical underlying instance.

This model was designed with expressivity and flexibility in mind, rather than mere consistency. This approach distinguishes from many works in ontology engineering which are mainly focused towards reasoning, i.e. inference of novel facts that can be deduced from the current fact base. The latter view can be valuable when actionable data is sought [1,2]. In narration, the focus is not so much on deduction than on facilitating the access and the presentation of the data. In our approach, ensuring the stored narrative knowledge is consistent accross the database is delegated to users and tier applications.

Following our simplified schema, complex narrative structures are stored using nested structures. Figure 2 shows an instanciation example. The complexity mentioned in Fig. 1 can be seen as the height that instances of a given entity generally takes.

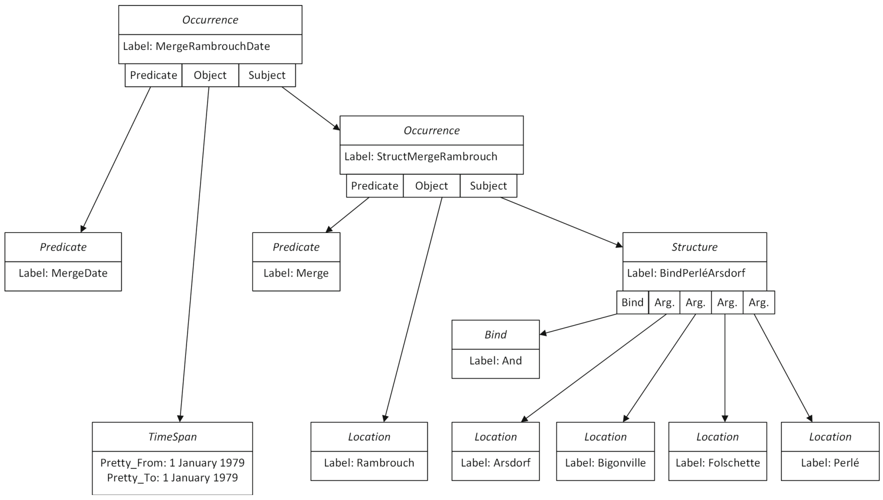


Fig. 2. Narrative instances extracted from the French text: *Perlé fut une commune jusqu'au 1er janvier 1979, date à laquelle elle a fusionné avec les communes d'Arsdorf, Bigonville et Folschette pour former la nouvelle commune de Rambrouch*. Entity reference fields are represented as sockets at the bottom of instances, with arrows pointing to the linked instance.

Our way of encoding predicative occurrences is reminiscent of RDF-based ontologies, that rely on binary relations, and reification to represent more complex semantics. In the context of general ontologies, [22] indeed show that using reification, complex facts can be unfolded as several atomic facts. In brief, with reification, an instance of a binary relation $a\mathcal{R}_1b$ can act as the argument of another relation, effectively allowing e.g. $(a\mathcal{R}_1 b)\mathcal{R}_2 c$. This design has been subject to debate in the literature. For example, [12] advocates the usage of pure n-ary relations. Our eventual choice has been motivated by its greater flexibility, and better compliance with the technologies chosen for its implementation (see Sect. 4).

An important terminological nuance lies between our schema proposition and those also using reified facts: in the latter, properties are linked to entities by a static set of *predicates*, when in our proposition the *predicate* is a full-fledged entity type, with no constraints on the values its instances take. This choice is guided by the need to enable greater expressivity, as required in the context of narratives. Dealing with such an open vocabulary yields additional difficulties, addressed using NLP means as discussed to a greater extent in Sect. 4.

3.2 Content Description Model

In [16], the narrative entities introduced in Sect. 3.1 are laid out in the same schema as entities describing pieces of content. Here we choose to separate entities that support the narrative knowledge formalization, from those that describe the actual managed content. The content schema is given in Fig. 3.

We distinguish two main types of documents: stories, that are essentially made of a string of text, and media, that wrap actual media files along with a

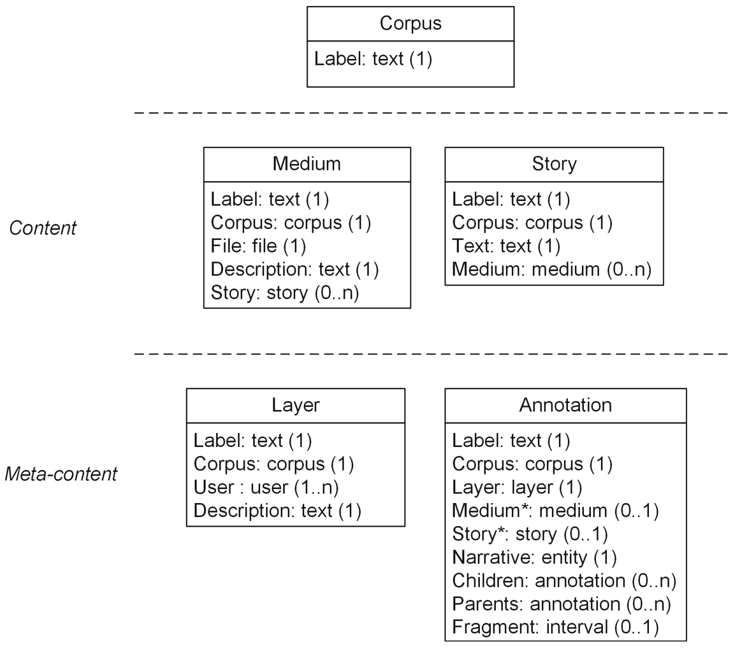


Fig. 3. Content description model. It distinguishes the actual managed content, from meta-content, essentially made from annotations.

formal description. Optional direct links may exist between these entities: one or more stories may act as the caption for a medium, as well as one or more media may decorate a given story.

Following principles developed in the Camomile project [17], the articulation between narrative entities discussed in Sect. 3.1 and content description is made by *annotations*. Roughly, via their *narrative* field, they associate narrative entities from the schema in Fig. 1 to whole or fragments of managed multimedia content. Specifically, the *fragment* field, basically an array of integers, optionally allows the annotation to be specific of a fragment of a document. Its meaning depends on the context - e.g. starting and ending character positions for a story, beginning and ending timestamp in seconds for audio and video, or a pixel bounding box for pictures. A simple example illustrating the distinction between content, annotations and narrative entities in given in Fig. 4.

The root concept in Fig. 3, the *corpus*, abstracts a group of homogeneous documents. This will enable the separate treatment of the two distinct corpora presented in Sect. 4.1. *Layer*s are meant to gather annotations of identical nature, w.r.t. content of a given corpus. For example annotations by a given user or group of users during a specific manual annotation session, or by a given automatic annotation algorithm, may fall in distinct layers. This addresses a very pragmatic concern: the fact that the same story or medium might be annotated several times, possibly differently, by multiple agents. A layer may assemble annotations of several narrative types, covering any part of the corpus (i.e. its media and stories).

In addition to the guidelines introduced in [17], the annotation entity in Fig. 3 offers the possibility to form annotation hierarchies, via the *children* and *parents* fields. The intent is to mirror the potential complexity of constructions following the narrative data model (see e.g. Fig. 2), and facilitate retrieval operations in the context of annotation tasks.

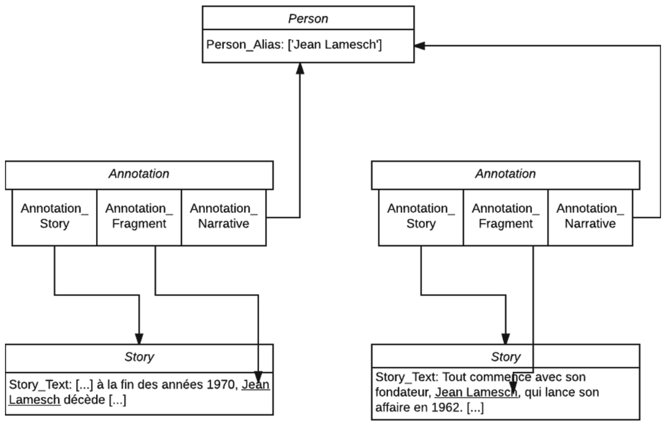

Person
Person_Alias: ['Jean Lamesch']
Annotation
Annotation_ Story
Annotation_ Fragment
Annotation_ Narrative
Annotation
Annotation_ Story
Annotation_ Fragment
Annotation_ Narrative
Story
Story_Text: [...] à la fin des années 1970, Jean Lamesch décède [...]
Story
Story_Text: Tout commence avec son fondateur, Jean Lamesch, qui lance son affaire en 1962. [...]

Fig. 4. Example showing the link between two stories (on bottom, extracted from the *web* corpus described in Sect. 4.1) and the same underlying entity (person, on top) ensured by annotations.

3.3 API Description

The models exposed in Figs. 1 and 3 are implemented under the Drupal Content Management System [16]. Specifically, entity instances are implemented as *nodes* in Drupal. Our narrative and content entities are implemented as specializations of this baseline, notably supporting optional and variable-sized fields. Beyond primitive types (e.g. text strings, numbers), Drupal offers the possibility to define fields holding *entity references*: this feature enables links between entities as displayed in Fig. 2 for example.

Drupal also features a RESTful module [23], that allows to bind entity types as defined in a Drupal backend to REST API routes. In brief, REST is a protocol based on the HTTP protocol and the JSON format, to allow consumption, creation and update of the data held by a Drupal backend. The RESTful module manages the conversion to and from entity references and node IDs. The general architecture in which the REST API and the underlying model situate is described by Fig. 5. In brief, the REST API is the interface to the backend data stored according to the schema described in Sects. 3.1 and 3.2. It enables the consumption and production of the data by both automatic (e.g. NER techniques) and interactive programs (e.g. web-based annotation tools).

Figures 6 and 7 show the REST API routes mirroring the entities shown in Figs. 1 and 3, respectively. Without any filter, a route returns all entities of the associated type. Query filters can be applied following the syntax presented in [24], using node IDs or primitive values. The *annotation_narrative_type* filter in the *annotation* route allows filtering according to entity types shown in Fig. 1. *annotation_narrative_type* and *annotation_narrative* filters are matched by inspecting the entity referenced by the respective *narrative* field.

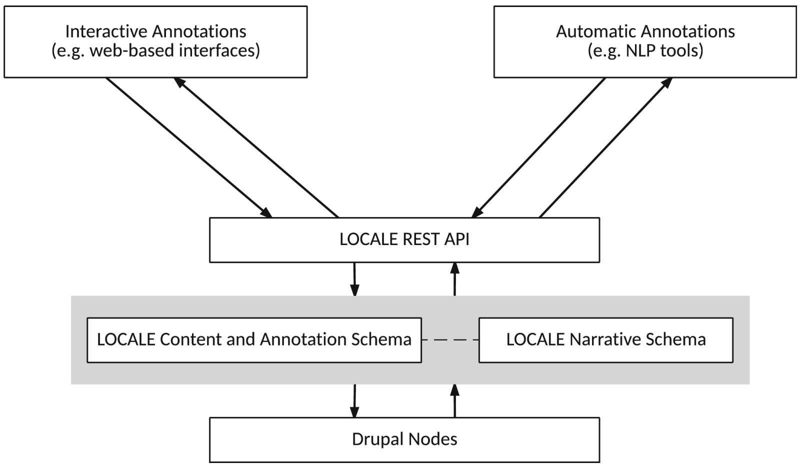


Fig. 5. General architecture of the LOCALE platform, from the REST API perspective.

ROOT_URL/artifact?filter[id, artifact_alias^{1}]
ROOT_URL/person?filter[id, person_alias]
ROOT_URL/location?filter[id, location_alias, location_longitude, location_latitude^{2}]
ROOT_URL/timespan?filter[id, timespan_year_from, timespan_month_from, timespan_day_from, timespan_year_to, timespan_month_to, timespan_day_to^{2}]
ROOT_URL/bind?filter[id, bind_alias]
ROOT_URL/structure?filter[id, structure_bind, structure_arguments^{3}]
ROOT_URL/occurrence?filter[id, occurrence_subject, occurrence_object, occurrence_predicate]
ROOT_URL/predicate?filter[id, predicate_alias]
ROOT_URL/association?filter[id, association_left, association_right]
ROOT_URL/link?filter[id, artifact_alias]

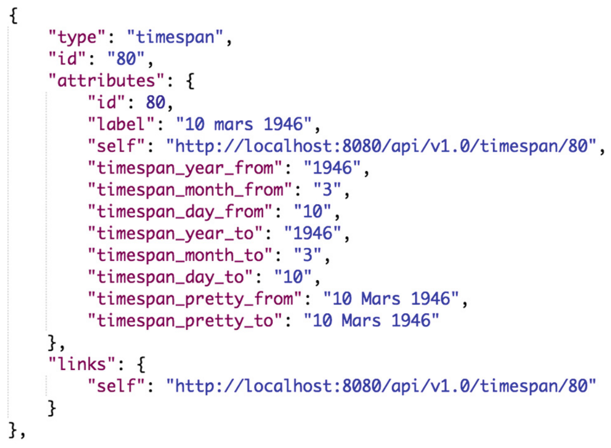
Fig. 6. Routes related to narrative information exposed by the REST API. Superscripted numbers are explained in the text of Sect. 3.3.

ROOT_URL/corpus?filter[id]
ROOT_URL/layer?filter[id, layer_corpus]
ROOT_URL/story?filter[id, story_corpus, story_medium]
ROOT_URL/medium?filter[id, medium_corpus, medium_story]
ROOT_URL/annotation?filter[id, annotation_corpus, annotation_layer, annotation_story, annotation_medium, annotation_narrative, annotation_narrative_type]

Fig. 7. Routes related to content exposed by the REST API.

The routes described in Fig. 7 return information describing managed media and their annotations. Actual narrative entities behind annotations, i.e. semantic values that annotate documents, can be accessed via the routes shown in Fig. 6 using the fetched IDs. The *Location* entities can be queried by filling the *Longitude* and *Latitude* filters. *Longitude* and *Latitude* can be specified as relative floats. By convention, positive latitudes (resp. longitudes) denote the northern part of the Earth (resp. eastern part w.r.t. Greenwich meridian). Filters mirroring the *TimeSpan* entity fields can also be specified.

All routes in Figs. 6 and 7 support the GET (i.e. retrieving entities) verb, as well as PATCH (i.e. updating an entity) or POST (i.e. creating a new entity) whenever appropriate. The Drupal RESTful modules allows the fairly straightforward customization of the input and outputs of the routes defined. We used this facility to implement a *pretty* date format, that allows the conversion of textual dates to and from the schema defined in Fig. 1 (see Fig. 8 for an example).



```
{
    "type": "timespan",
    "id": "80",
    "attributes": {
        "id": 80,
        "label": "10 mars 1946",
        "self": "http://localhost:8080/api/v1.0/timespan/80",
        "timespan_year_from": "1946",
        "timespan_month_from": "3",
        "timespan_day_from": "10",
        "timespan_year_to": "1946",
        "timespan_month_to": "3",
        "timespan_day_to": "10",
        "timespan_pretty_from": "10 Mars 1946",
        "timespan_pretty_to": "10 Mars 1946"
    },
    "links": {
        "self": "http://localhost:8080/api/v1.0/timespan/80"
    }
},
```

Fig. 8. Date record example, as retrieved from the *ROOT_URL/timespan* route. The POST and PATCH verbs also parse dates provided under the *pretty* format, if provided.

In the following list, we elaborate about specific points highlighted in Fig. 6:

1. Alias fields have variable size: as a result, e.g. *filter[artifact_alias]* = *STRING_VALUE* matches against any value of the respective array.
2. Filtering other than equality can be specified e.g. with *filter[location_{longitude|latitude}][operator]=FLOAT_RADIUS_VALUE*. This also applies to point 1, where an integer from 0 to n could indicate approximate matching using edit distance.
3. Arguments may be queried directly via their node ID.

4 Experiments

4.1 Corpora Description

First we describe two corpora that have been loaded in our backend:

– 266 short stories (avg. 140 characters) related to the period of 1945–1975 in Luxembourg and the surrounding region. The stories were selected and extracted automatically using the the Google Custom Search Engine API[1]. More precisely, the Google API was invoked for a list of heuristical queries about well known Luxemburgish locations or companies (e.g. *Kirchberg, Luxair*) for the targeted time frame. Results originate from several web portals (i.e. Wikipedia, http://www.industrie.lu, http://www.vdl.lu). Each story is associated to a date and a location name. This kind of indexing is easily handled by the model described in Fig. 1. Each location name has been mapped with latitude and longitude using the Google Maps Geocoding API (See footnote 1). We refer to this corpus as *web* later on. This set of stories is described to further extent in [25].

[1] https://developers.google.com.

– Interviews were conducted with elderly people that have lived in Luxembourg in the time frame of interest (1945–1975). We used photograph collections from the spatio-temporal time frame of interest in order to trigger memories, as well as the *web* subset. We obtained approximately 5 h of audio recordings, by 5 participants, among which 2 h have been manually transcribed and segmented into 9 stories (avg. 3964 characters). We refer to this subset as *interviews* later on.

Unique labels are simple to initialize for *Location* and *TimeSpan* narrative entities (the name itself and the date format illustrated in Fig. 8, respectively). As distinct stories might have the same n initial words, in our implementation, unique story labels are generated as MD5 hashes [26] from the respective text.

4.2 Knowledge Extraction

As exposed in Sect. 2, the most frequent setting in the literature is to consider that the mapping of a presentation to a plot structure is performed manually. In this section, we describe means to extract, at least approximately, the narrative information out of this initial representation. Performing this automatic mapping operation can have various utilities in the context of the project described in the introduction. First, as mentioned in Sect. 4.1, testimonies in our data corpus are recorded and transcribed manually, but exhibit no structure that facilitate their presentation in context, and exploration. Offline extraction of the narrative structure would avoid tedious manual efforts. As mentioned in the introduction, our research context copes with testimonies collected in French. This constrained the technologies discussed later on in this section.

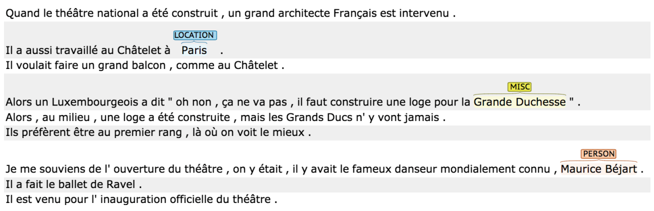
Named Entity Recognition consists in detecting entities such as people and places automatically in text. For example, the LIA tools [21] recognize people, locations, organizations, socio-political groups, quantities, dates and products in French text. Such facilities can then be a crucial initial step towards feeding the model described in Fig. 1, of which people, places and time specifications are the core. The most renowned general purpose natural language processing system, the Stanford CoreNLP suite [27], also provides such NER functionalities for several languages including French.

In order to structure recognized entities and annotations according to the schema described in Fig. 3, and possibly extract non-named entities (i.e. *Artifacts* in Fig. 1), syntactic cues are needed. *Part-Of-Speech* (POS) tagging is about estimating the function of words in text (e.g. adjective, verb, determinant). *Semantic Role Labeling* (SRL) builds upon POS-tagging in order to extract higher-order structures (e.g. subject, object, verbal forms), that are very close to the syntactic cues expected in our model. Actually this is not surprising insofar as the same seminal references in language analysis are foundational both for narratology [12] and SRL [28]. POS-tagging facilities are available in French both in the LIA tools [21] and the CoreNLP suite [27]. The latter also offers facilities in SRL, which are used by examples shown in Sect. 4.

NLP tools presented above allow extracting predicates and structural information. However, as we consider open vocabularies for *Predicate*, *Bind* and *Link* instances, semantics still have to be attached to them, e.g. allowing queries for instances similar to a template. Prepositions and coordination markers take their values in sufficiently small sets, so term-based queries are acceptable then. But as verbs are very numerous, a proxy is needed to allow queries for similar *Predicate* instances. Word embedding techniques associates a value in a high-dimensional numeric space to words, in a way that reflects word semantics [29]. We propose to use the implicit structure reflected by the word embedding space. Such mapping functions can be implemented locally using models from libraries such as TensorFlow [30] trained with a corpus in French. We used the complete French Wikipedia archive[2]. It contains approximately 2.3M articles in XML format. We converted them to a plain text format as expected by the training algorithm using the tool proposed by [31]. Punctuation and other non-textual characters were then removed using regular expressions. No stemming is required as all forms of a given word are embedded separately, and generally end up in close vicinity to each other.

4.3 Automatic Annotation Scenario

In this section, we illustrate how our REST API can be used in conjunction with NLP means in order to extract and attach narrative knowledge to the managed data. The following excerpt from the *interviews* corpus is used to support the illustration: *Quand le théâtre national a été construit, un grand architecte Français est intervenu. Il a aussi travaillé au Châtelet à Paris. Il voulait faire un grand balcon, comme au Châtelet. Alors un Luxembourgeois a dit "oh non, ça ne va pas, il faut construire une loge pour la Grande Duchesse". Alors, au milieu, une loge a été construite, mais les Grands Ducs n'y vont jamais. Ils préfèrent être au premier rang, là où on voit le mieux. Je me souviens de l'ouverture du théâtre, on y était, il y avait le fameux danseur mondialement connu, Maurice Béjart. Il a fait le ballet de Ravel. Il est venu pour l'inauguration officielle du théâtre.* In brief, the story is about the new theatre that opened in Luxembourg in 1964.



Quand le théâtre national a été construit , un grand architecte Français est intervenu .
Il a aussi travaillé au Châtelet à Paris .
Il voulait faire un grand balcon , comme au Châtelet .
Alors un Luxembourgeois a dit " oh non , ça ne va pas , il faut construire une loge pour la Grande Duchesse " .
Alors , au milieu , une loge a été construite , mais les Grands Ducs n' y vont jamais .
Ils préfèrent être au premier rang , là où on voit le mieux .
Je me souviens de l' ouverture du théâtre , on y était , il y avait le fameux danseur mondialement connu , Maurice Béjart .
Il a fait le ballet de Ravel .
Il est venu pour l' inauguration officielle du théâtre .

Fig. 9. Named entities extracted by the CoreNLP server, visualized using *brat* [32].

[2] https://dumps.wikimedia.org/frwiki/latest/.

Using a script, this piece of content was retrieved using the *story* REST route (see Fig. 7), and submitted to a local CoreNLP server [27] in order to tentatively identify named entities. Extracted entities are shown in Fig. 9. The extracted set is neither complete (e.g. *Grands Ducs* and *théâtre national* are not detected as *Person* and *Location* entities, respectively), nor fully accurate (e.g. *Grande Duchesse* is set in the *miscellaneous* category, when it is truly a *Person* entity). Nevertheless, the extracted entities can be used to create respective *Person* and *Location* entities. The resulting narrative and content entity structure, mirroring the example in Fig. 4, is shown in Fig. 10, and stored using our REST API.

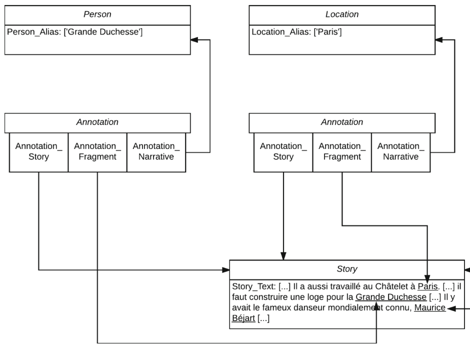

Person
Person_Alias: ['Grande Duchesse']
Location
Location_Alias: ['Paris']
Person
Person_Alias: ['Maurice Béjart']
Annotation
Annotation_ Story
Annotation_ Fragment
Annotation_ Narrative
Story
Story_Text: [...] Il a aussi travaillé au Châtelet à Paris. [...] il faut construire une loge pour la Grande Duchesse [...] Il y avait le fameux danseur mondialement connu, Maurice Béjart [...]

Fig. 10. Annotations and narrative entities resulting from the NER.

In view of extracting predicative occurences that involve the named entities identified above, we apply the SRL facilities available in the CoreNLP server to the respective sentences. The result from this analysis is depicted in Fig. 11. Then we apply the following procedure:

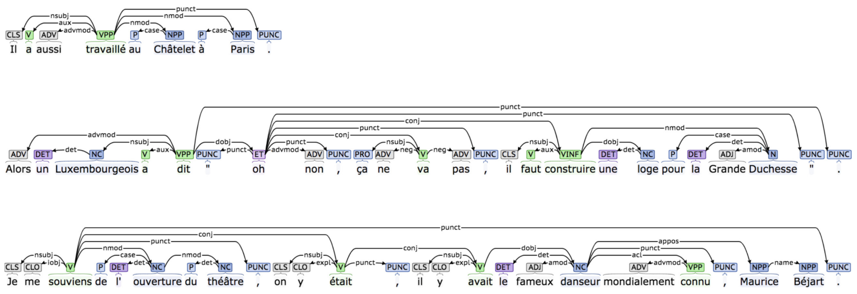

Il a aussi travaillé au Châtelet à Paris .
Alors un Luxembourgeois a dit " oh non , ça ne va pas , il faut construire une loge pour la Grande Duchesse " .
Je me souviens de l' ouverture du théâtre , on y était , il y avait le fameux danseur mondialement connu , Maurice Béjart .
nsubj
aux
advmod
punct
nmod
case
dobj
conj
neg
det
amod
expl
iobj
acl
appos
name

Fig. 11. SRL results, visualized using *brat* [32].

- Identify the verbal tokens that are parent of at least a group where a noun is involved (e.g. *travaillé*, *dit*, *construire*, *souviens* and *avait* in Fig. 11).
- Form predicative occurences with predicates that have subject (*nsubj* in Fig. 11) and/or object (*dobj* in Fig. 11) links in the SRL output.

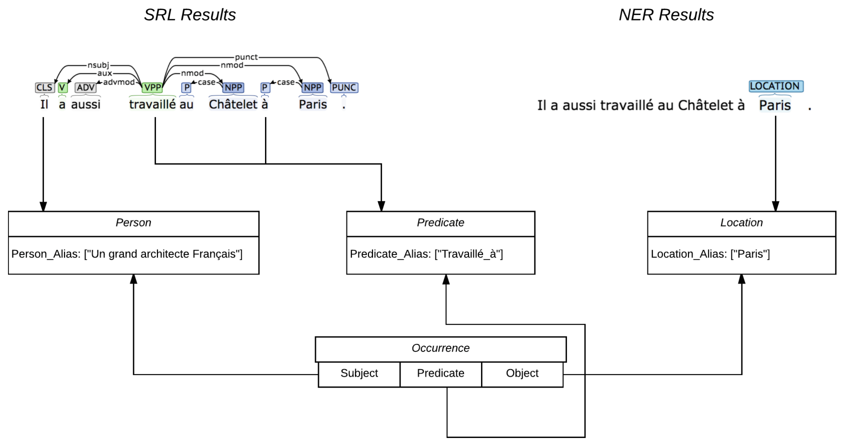
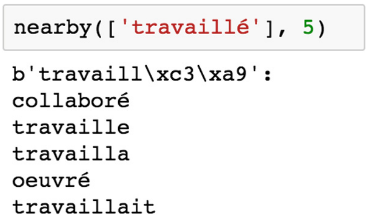

SRL Results
NER Results
nsubj
aux
advmod
punct
nmod
nmod
case
case
CLS V ADV VPP P NPP P NPP PUNC
Il a aussi travaillé au Châtelet à Paris .
LOCATION
Il a aussi travaillé au Châtelet à Paris .
Person
Person_Alias: ["Un grand architecte Français"]
Predicate
Predicate_Alias: ["Travaillé_à"]
Location
Location_Alias: ["Paris"]
Occurrence
Subject
Predicate
Object

Fig. 12. Construction of a predicative occurrence using SRL and NER results.

– If nominal modifiers are found (*nmod* in Fig. 11), create predicates that concatenate the verbal token and the respective preposition - if they do not already exist - and create predicate occurences accordingly.
– If coordination links are found between predicates (i.e. *conj* in Fig. 11), create associations between occurences accordingly.

Figure 12 illustrates the construction of predicative occurences using this procedure. Annotations linking the occurrences to the respective content are then created accordingly. As discussed in Sect. 4.2, prepositions (e.g. *à*, *pour*, *du*) and coordination markers (e.g. comma, *mais*) take their values in small sets. When appropriate, we concatenate the latter with the respective predicates. Later on, when querying the backend for similar predicative occurences, we may deparse the verb from the constructed predicate, and perform a query using nearest neighbors in a word embedding space as index. Figure 13 gives an example of such a query.



```
nearby(['travaillé'], 5)
```

```
b'travaill\xc3\xa9':
collaboré
travaille
travailla
oeuvré
travaillait
```

Fig. 13. The 5 nearest neighbors of the word *travaillé* (passive form of the verb *work*) in the word embedding space.

5 Conclusion

We described a schema and an API that facilitate the storage, processing and sharing of personal narratives related to the period of 1945–1975 in Luxembourg. We tested it with real data collected from the web and interviews conducted with witnesses of the spatio-temporal time frame of interest, and showed how our API can be used in conjunction to NLP means in order to extract and manage narrative knowledge. As NLP tools are prone to errors and omissions, semi-automatic means to annotate and curate the corpora presented in Sect. 4.1 would be the most natural extension to this work. A variant of the latter application would be the semi-automatic input of a story. When a user types a story in an interface, text would be sent on the fly to processing services. Based upon extracted entities, related stories can then be displayed live to the client as a contextual help.

A key extension would be also to enable the resolution of conflicts between stories and plots, as well as the resolution of entities having alternative writings. Classical use of reasoning is to enable deduction of novel facts if adequate rules are defined [4], but this range of techniques has also been used to detect contradictions [33].

Finally, we will study the issue of subjectivity: personal narratives often contain implicit references to the narrator (e.g. me, my brother). Experiments in Sect. 4.3 show these references can only be indirectly detected. Simple heuristics could be implemented by using a closed list of keywords along with the extracted cues.

Acknowledgements. This work was achieved in the context of the LOCALE project, funded by the Luxembourgish *Fonds National de la Recherche*. It used data from the *Centre National de l'Audiovisuel*. We would also like to thank elderly volunteers from *AMIPERAS* and *Konviksgaart* organizations, who contributed to the creation of the *interviews* corpus.

References

1. Scherp, A., Franz, T., Saathoff, C., Staab, S.: F-A model of events based on the foundational ontology DOLCE+DnSUltralite. In: K-CAP 2009, pp. 137–144 (2009)
2. Van Hage, W., et al.: Abstracting and reasoning over ship trajectories and web data with the Simple Event Model (SEM). Multimedia Tools Appl. **57**(1), 175–197 (2012)
3. Segers, R., et al.: Hacking history: automatic historical event extraction for enriching cultural heritage multimedia collections. In: K-CAP 2011 (2011)
4. Suchanek, F.M., et al.: Yago: a large ontology from Wikipedia and WordNet. Web Semant. Sci. Serv. Agents WWW **6**(3), 203–217 (2008)
5. Google: Introducing the knowledge graph (2012). http://tinyurl.com/zofw8fb
6. Mendes, P.N., Jakob, M., García-Silva, A., Bizer, C.: DBpedia spotlight: shedding light on the web of documents. In: ACM Semantics, pp. 1–8 (2011)
7. Weibel, S., Kunze, J., Lagoze, C., Wolf, M.: Dublin core metadata for resource discovery (no. rfc 2413) (1998)

8. Brickley, D., Miller, L.: Foaf vocabulary specification 0.91. Technical report, ILRT Bristol (2007)
9. van der Meij, L., Isaac, A., Zinn, C.: A web-based repository service for vocabularies and alignments in the cultural heritage domain. In: Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) ESWC 2010. LNCS, vol. 6088, pp. 394–409. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13486-9_27
10. Doerr, M.: The CIDOC conceptual reference module: an ontological approach to semantic interoperability of metadata. AI Mag. **24**(3), 75–92 (2003)
11. Mulholland, P., Wolff, A., Collins, T.: Curate and storyspace: an ontology and web-based environment for describing curatorial narratives. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) ESWC 2012. LNCS, vol. 7295, pp. 748–762. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-30284-8_57
12. Zarri, G.: Representation and Management of Narrative Information. Theoretical Principles and Implementation. Springer, London (2009). https://doi.org/10.1007/978-1-84800-078-0
13. Tilley, A.: Plot Snakes and the Dynamics of Narrative Experience. University Press of Florida, Gainesville (1992)
14. Yeung, C., et al.: A knowledge extraction and representation system for narrative analysis in the construction industry. Expert Syst. Appl. **41**(13), 5710–5722 (2014)
15. Min, S., Park, J.: Mapping out narrative structures and dynamics using networks and textual information. arXiv preprint arXiv:1604.03029 (2016)
16. Bruneau, P., Parisot, O., Tamisier, T.: Storing and processing personal narratives in the context of cultural legacy preservation. In: DATA (2017)
17. Poignant, J., Budnik, M., Bredin, H., Barras, C., Stefas, M., Bruneau, P., Adda, G., Besacier, L., Ekenel, H., Francopoulo, G., Hernando, J., Mariani, J., Morros, R., Quénot, G., Rosset, S., Tamisier, T.: The CAMOMILE collaborative annotation platform for multi-modal, multi-lingual and multi-media documents. In: LREC Conference (2016)
18. Bollacker, K., et al.: Freebase: a collaboratively created graph database for structuring human knowledge. In: SIGMOD, pp. 1247–1250 (2008)
19. Gaeta, A., Gaeta, M., Guarino, G.: RST-based methodology to enrich the design of digital storytelling. In: IEEE INCOS 2015, pp. 720–725 (2014)
20. Van Hooland, S., et al.: Exploring entity recognition and disambiguation for cultural heritage collections. Digit. Scholarsh. Humanit. **30**(2), 262–279 (2015)
21. Favre, B., Béchet, F., Nocéra, P.: Robust named entity extraction from large spoken archives. In: HLT/EMNLP 2005, pp. 491–498 (2005)
22. Hoffart, J., Suchanek, F.M., Berberich, K., Weikum, G.: YAGO2: a spatially and temporally enhanced knowledge base from Wikipedia. Artif. Intell. **194**, 28–61 (2013)
23. Burstein, A., Bosch, M.: RESTful best practices for Drupal (2014). https://github.com/RESTful-Drupal/restful
24. Burstein, A., Bosch, M.: Applying a query filter (2017). https://github.com/RESTful-Drupal/restful/blob/7.x-2.x/docs/api_url.md#applying-a-query-filter
25. Parisot, O., Tamisier, T.: A corpus of narratives related to Luxembourg for the period 1945–1975. In: 28th International Workshop on Database and Expert Systems Applications, pp .113–117 (2017)
26. Rivest, R.: The MD5 Message-Digest Algorithm. RFC 1321, MIT and RSA Data Security (1992)

27. Manning, C.D., Surdeanu, M., Bauer, J., Finkel, J.R., Bethard, S., McClosky, D.: The stanford CoreNLP natural language processing toolkit. In: ACM Semantics, pp. 55–60 (2014)
28. Jurafsky, D., Martin, J.H.: Semantic role labeling. In: Speech and Language Processing. Pearson (2014)
29. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: NIPS, pp. 3111–3119 (2013)
30. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al.: TensorFlow: a system for large-scale machine learning. In: 12th USENIX conference on Operating Systems Design and Implementation, pp. 265–283 (2016)
31. Attardi, G.: A tool for extracting plain text from Wikipedia dumps (2016). https://github.com/attardi/wikiextractor
32. Stenetorp, P., Pyysalo, S., Topić, G., Ohta, T., Ananiadou, S., Tsujii, J.: BRAT: a web-based tool for NLP-assisted text annotation. In: Proceedings of the Annual Meeting of the Association for Computational Linguistics, pp. 102–107 (2012)
33. Paulheim, H.: Knowledge graph refinement: a survey of approaches and evaluation methods. In: Semantic Web, pp. 1–20 (2016)