AN ENTITY REGISTRY: A MODEL FOR A REPOSITORY OF ENTITIES FOUND IN A DOCUMENT SET

Valerio Bellandi[1] and Stefano Siccardi[2]

[1]Department of Computer Science, Università degli Studi di Milano, Italy

[2]Consorzio Interuniversitario Nazionale per l'Informatica, Italy

ABSTRACT

This paper proposes a conceptual structure for a repository of entities that can be found by usual procedures of Natural Language Processing, that is the search for entities mentioned in text, their identification, possibly through the link to entries in Background Knowledge Basis (BKG) and theconstruction of a Knowledge Basis or Graph to host the information found in this process. We address applications where a BKG is of little help, because the involved entities are not so relevant to be included in any, being for instance ordinary people or small companies. Therefore, we rely on the entities' attributes and relationships for unique identification, disambiguation, knowledge checking and any other relevant operation. One of the final goals achieved by the proposed method is the ability to merge knowledge collected in separate bases, once they are referred to the same Entity Registry.

KEYWORDS

Named Entity Recognition, Named Entity Linking, Knowledge Basis, Knowledge Graphs

1. INTRODUCTION

The search for entities in documents is a frequently addressed task in Natural Language Processing (NLP). It may be described as a pipeline, composed by several steps, each focused fora specific task. First Named Entity Recognition (NER) is performed, to find the type of each entity that can be found in the document; Named Entity Linking (NEL) follows: it consists in searching each entity in a knowledge basis (KB) in order to recognize and identify the entity itself; finally, entities that are not present in the KB must be managed in the NIL mentions step. Often, however, some generic KBs (e.g. Wikipedia) are used to identify entities, and more context specific ones are used to keep track of entities found in the document sets at hand; in this perspective, we propose the introduction of a dedicated repository for the entities that have been identified in a context, the Entity Registry (ER), so that every fact in the context specific KB involves instances linked to entries in the entity registry. We will say that the KB is *referred to* the ER.

Even if the concept, in one form or another, is not new, it does not exist, at the best of our knowledge, a description of a suitable structure of the entity registry or a set of requirements of its functions. The present paper aims at filling this gap.

Our considerations are applicable to any context; however, we have in mind especially documents related to the legal domain, both lawsuits and criminal investigations.

David C. Wyld et al. (Eds): NIAI, MoWiN, AIAP, SIGML, CNSA, ICCIoT - 2023
pp. 01-12, 2023. CS & IT - CSCP 2023 DOI: 10.5121/csit.2023.130301

The structure of the paper is as follows: in section 2 we briefly summarize related work; in section3 we describe in details the structure of the proposed Entity Registry structure; in section 4 some techniques to deal with more than one ER are illustrated; section 5 contains an example of usage of an ER and section 6 contains our conclusions.

2. RELATED WORK

A typical pipeline to deal with entities extraction processes in documents, in order to extract semantic information is described in [1]

2.1. Named Entity Recognition

NER consists in finding mentions of entities belonging to a set of types, based on an ontology or just given as a list of labels. It can be considered a classical task in NLP, see for instance the survey [2]. It can be described as the task of recognizing information units like names of persons, organizations, locations etc., numerical expressions like money, dates, and some types of codes, identifying references to these entities in text. It is presently considered a standard function of many NLP tools, for instance Spacy ([3]).

2.2. Named Entity Linking

Basically, NEL consists in linking an entity, taken from a background KB to a mention in a document. Several strategies have been proposed, many relying on self-attention mechanisms ([4]). In many cases, mentions are encoded and compared to a list of candidate entities, that are assigned a rank. Sometimes graph based methods have been used ([5]) and it has been stressed the interplay between NER and NEL performances ([6]).

Normally, the NEL task supposes to deal with a well-structured KB, like for instance Wikidata, DBpedia, YAGO, KOG and so on. In the legal context we have in mind, however, one usually deals with ordinary people and their possessions (cars, phones, credit cards...), small companies and places with no special relevance. It is very unlikely, therefore that such entities can be found in one of the KBs normally used for NEL. On the contrary, one often has at disposal external entries for many entities, like social media profiles, company websites, citations in blogs or local news, but these are themselves document to be processed, instead of formal KBs.

2.3. NIL Mentions Management

Unlike NER and NEL, that are very commonly considered in systems dealing with entities in documents, NIL mentions are only occasionally managed. We distinguish two main points: NIL prediction and NIL clustering.

NIL prediction can be considered a classification task, with a reject option that is NIL or unlikable (to any known entities). Several options have been proposed for this task, and its quality is evaluated in terms of the usual indicators for classification tasks (see [4], [7]).

NIL clustering aims at grouping together mentions referring to the same entity. Several algorithms have been proposed, based on GNN ([8]) or hierarchical clustering ([9]). Incremental management to add NIL mentions to a KB has been considered in [10]

3. THE ENTITY REGISTRY STRUCTURE

We describe the ER structure in terms of a graph in order to be able to manage the concepts, but they can be implemented with any suitable data structures, like triples or even tabular data. The ER consists of a metamodel, describing the entity types, as can be found in the NER step, and of the instances, considered in the NEL step.

3.1. The Metamodel

Entity types are described in the ER metamodel. Each entity has a unique name in the metamodel and a set of attributes. The main goal of the attribute specification, for the aims of the ER, is to define how an entity can be identified with certainty. Therefore, the metamodel must describe, for each entity type, the sets of attributes that uniquely identify an instance, that is the unique keys. We will sometimes call "partial keys" attributes that make part of a key. On the other hand, the metamodel does not contain entity attributes that are not involved with the task of identifying an entity.

The two observations above also describe the main difference between the ER metamodel and an ontology describing the same entity types, as the ontology should describe all the possible attributes without specifying which ones are keys for the entity.

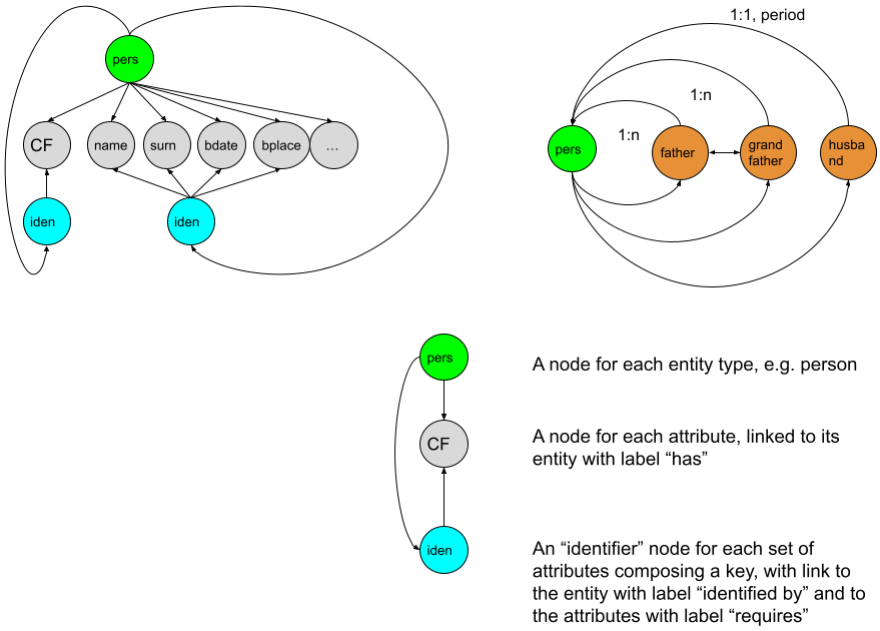

Figure 1. Graph representation of the metamodel

Figure 1 shows a graph for a section pf the metamodel. In the example, the entity type "person" can be uniquely identified by the code "CF" alone or by the combination of "name", "surname", "birth date" and "birth place". Relationships are included as in a separate subgraph for the sake of clarity; see below for details.

We note that, when a background KB is used, an identifying attribute may be defined as a link to some entry in the KB. Moreover, in some context, an entity could be identified by its relationships too. For instance, persons could be identified by their names, surnames, relationships with fathers and mothers. We however do not consider identifying relationships in the metamodel, substituting

them with attributes. In the example above, we would consider "father name and surname" and "mother name and surname" as identifying attributes. The reason is not only the sake of simplicity: considering identifying relationships would imply to add instances that could be useless or difficult to uniquely identify in turn, adding uncertainty to the KB. On the contrary, attributes can be usually found in the documents themselves.

On the other hand, we require that the ER contains descriptions of any relationships that must be checked in the KBs referred to the ER (relationship types). We distinguish several cases:

- Relationships with cardinality 1:1 or 1:n, for instance "being husband of" or "being father of", that must be checked for the uniqueness constraint on one or both involved nodes

- Relationships with cardinality 1:1 or 1:n whose uniqueness must be checked together with additional attributes or attributes' functions. This happens usually when time is involved in the relationship. For instance, "being husband of" may link an individual to two or more others when a "time period" is taken into consideration

- Incompatible relationships, including relationships that cannot be bidirectional, "being father of" cannot be bidirectional and is incompatible with "being grandfather of"

Relationships that do not require checking need not to be included in the ER.

In some cases, attributes may be related or dependent on each other. For instance, the tax code ("codice fiscale"), that is attributed in Italy to every individual at birth time, is built via standard rules using name, surname, date and place of birth, so that this tuple of attributes and the code contain basically the same information. We do not consider this type of information in the metamodel, as we prefer to assign the task of checking and managing information to context specific systems.

Finally, one could consider relationships between entity types, for instance hierarchical type – subtype relationships, like "Person" with subtypes "Lawyer", "Clerk", "Teacher" etc. We require, for such relationships, a link between the type and each subtype, and also between their homologous attributes, this latter may be an implicit one, like the convention of using the same names. However, we encourage the definition of simple, general types, independent of context and time, keeping at a minimum their relationship. In fact, the spirit of the ER is to manage the entity identity, not their roles. For instance, an individual could be "lawyer" in a subset of documents, "witness" in another and "victim" in a third one and so on. Instead of defining multiple types, we suggest that labels are attached to the entity instances; this process lays at the border between the ER itself and the KB or graphs using the ER to manage information extracted from documents.

3.2. The Entities in the ER

Each entity has an entry in the ER. The entries hold information about their entity type as an attribute or, if recorded in the same physical graph, as a link. Each entity is assigned a unique id and attributes from the set defined for its type.

Optionally, some labels may be attached to the entities; they are not checked by the ER and may be intended as shortcuts for information found in KBs referred to the ER. This may be an alternative strategy to sub types discussed above. For instance, an entity of type person could have labels lawyer, witness and so on. These the usage of these labels, however, should be taken to a minimum, as they can be in all cases deduced from the individual KBs.

We do not consider relationships between entities, as we do not replicate in the ER information contained in the KBs, but a few special relationships related to the entities identification:

- The “merged” relationship, that means that two entities have been recognized to represent the same object in the real world
- The “split” relationship, that means that two entities have been created from a single one, because it has been recognized that two objects in the real world are involved. This operation corresponds to the so-called disambiguation of entities
- The “similar”, with a similarity degree and possibly type, meaning that two entities share either some characteristics

We consider the “similar” relationship as an optional and context dependent one; for instance one could define similarity based on the number of partial key attributes shared (e.g. persons sharing their name and surname with unknown date and place of birth), possibly with attribute weights. This similarity can be checked using the ER only. One could consider KB based similarity measures instead, for instance based on the number of facts shared.

3.3. The ER Functions

The ER must be able to provide a minimal set of functionalities or operations. Metamodel related functions are:

- Creation of a metamodel, whose input is the description of entity types with attributes and keys and of relationships with cardinality, attributes and incompatibility constrains. Input could be derived from an ontology
- Update of a metamodel, to add a new entity type or a new attribute for an existing type
- Query to obtain the list and details of entity types and relationships

We do not require delete operations for the metamodel, if they will be provided by a specific implementation, however, they must ensure the integrity of the ER, that is only attributes, entity types and relationship types having no instances may be deleted.

Entity related functions are:

- Create a new entity, given its type and some attributes. The ER will assign a unique ID to the new entity
- Query entities, given either their specific IDs or logical expression based on their attributes.
- Update an entity, that is change the value of an attribute or add a new attribute from the list of attributes for its type
- Delete an entity; this operation should either require that no instance can be found in the KBs referred to the ER or that they must be invalidated and processed again

- Merge two entities, that is create a relationship of type “merge” in the ER. This implies that no reprocessing of the KB is needed after a merge.

- Split an entity in one keeping the original ID and a new one

The query and split functions deserve a deeper discussion. Queries must output all or a subset of attributes as required by the caller, as would happen in any standard database; if requested by the caller, they must also output all the possibly merged or similar entities. In the latter case, type and degree of similarity should be added to the answer set. Another point is that sometimes NER tools can recognize entities in documents, but are not able to identify values of specific attributes. Thisis especially likely for deep-learning based programs. A typical case is when pair of words is labelled as a person, but it is impossible for the NER program to distinguish the person’s name and surname. It is requested therefore that the query functions can deal both with the usual situation when each attribute has its related requested value and the more complex one when it is given a list of attributes (like [name, surname]) and a list of values (like [Mary, Ann, Scott, Fitzgerald]). In the latter case the query function should try all the attributes – values combinations. We note in passing that, at entity creation time, the caller must choose a value for each attribute, even if in a casual way.

The split function must have in input the attributes that must be kept in the original node, those that must be copied to the new one and if they must be voided in the original. It must be checked, moreover, that the split nodes can be distinguished. This implies that, during the split operation, at least a partial key must be assigned different values in the entities. If the split is required because of an attribute that is not currently considered in the metamodel, it must be added to it before the split may happen. For instance, suppose that in the metamodel the entity type person has a key consisting of name, surname, birth date and birth place. In a first document we find a person with name=X, surname=Y, birth date= DD, create an entity with id=ID1 and relate it to the mention we found. In a second document, we find a person with same name and surname and relate the mention to the same entity. However, later on we find that the birthdate in the second document is different; as it is a partial key, we can split entity ID1 without further processing. On the other hand, suppose that neither of the documents contains birth dates information, but in the first another person ID2 is reported as father, while in the second document the father is ID3. We do not have any key related to fathers in the metamodel, but a later analysis, based on the relationship types definition, may spot the contradiction and request a split. At this point, a partial key attribute, like father name and surname, must be added to the metamodel before the split can be performed.

3.4. The Three Layers Structure

The ER, together with the KBs referred to it, creates a three-layers structure, where each individual piece of information can be tracked back to the pertaining entity, that is to something existing in the real world.

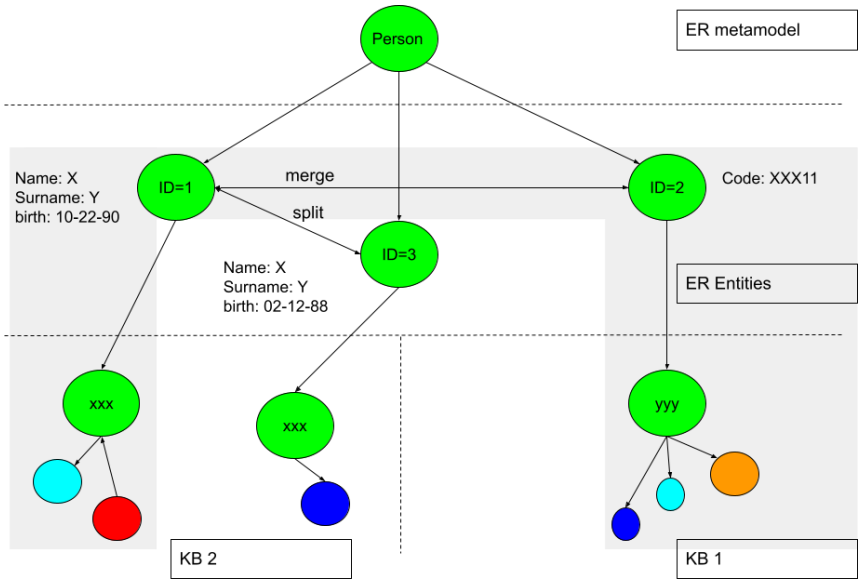

Figure 2. The three-layers structure

Figure 2 shows an illustrative example. The metamodel of the ER describes which entity types are considered; the second layer contains the entities that were found in the documents, specifying that entity with ID=1 was merged with entity with ID=2, and then split originating the new entity with ID=3. The third layer contains Knowledge Basis 1 in the left part and Knowledge Basis 2 in the right. Each node in this layer is related to an entity in the ER (specific attributes of nodes in the KBs are not shown for the sake of clarity). The shadowed area highlights all the information that the structure can relate to the same real world entity represented by the merged nodes with ID=1 and ID=2.

In the next sessions, we will illustrate some methods to manage ERs in order to fully exploit the potentialities of the concept.

4. DEALING WITH SEVERAL ENTITY REGISTRIES

It may happen that different documents have been processed using different ERs either because the processing contexts were not the same, or for any other reason. We consider in detail two possible scenarios for dealing with more than a single ER.

4.1. The Hierarchy Concept

In this scenario, we have two or more ERs and we want an upper level one, that is their union. For instance, two KBs may be derived from the documents of two criminal investigations and be referred to different ERs; each one will contain suspected people, victims, places, etc. of a specific investigation. We want to create a upper-level ER, containing entities of both the cases. The main use case of this operation is to gather form a number of KBs all the available information about entities.

In order to create this upper level ER, we suppose that the ERs share the same metamodel and perform a union operation. For each entity in each ER, it will check if in the other there is an ER

with the same (complete) key. If not (even if they share just a partial key), the entity will be just added to the upper-level ER, otherwise it will be added with a "merge" relationship to the other entity sharing its key. At this point it will also be checked if some contradiction arises. This will happen if the entities share a key, but have different values for some other partial keys, for instance two persons having the same unique "tax code" in both ERs, but different birth dates. The union operation will output a warning message urging the user to check the correctness of information.

After the upper-level ER has been created, downstream checks of the KBs referred to the original ERs will be possible (see section 5.)

4.2. The Mappings between an ER and Another

In this case, we consider two ERs having different metamodels and want to relate their common instances. This step is needed when we want to merge information of several KBs, referred to ERs that differ not only for the contained entities, but for the metamodels too.

The mapping must relate to each other entity types that are considered in both ERs. We considera number of cases of increasing complexity:

- Homologous types with different names, having the same set of attributes with different names: a 1-1 map between types and attributes may be built (e.g., Persons – Individuals; name, surname – first name, last name, etc.)

- Homologous types with different names, having complementary sets of attributes: a 1-1 map between entity types and a union of attributes are considered (e.g., Persons / Individuals with name / first name, surname / last name and birth date, birth place in ER1, father name, mother name in ER2)

- Homologous types with different names and with some equivalent attributes expressed ina different way in the ERs, (e.g. attributes name, surname in the first, attribute surname-name in the other): suitable functions may be considered for the attribute translation

- Non homologous types with non-empty intersection, for instance: ER1 has type Person and ER2 has types Actor, Politician, Singer etc. Or ER1 has types related to amateur sport practitioners (tennis, football, ski…) and ER2 has types related to professional roles (lawyer, physician, teacher...). These cases should be treated individually, the general idea being that of creating a super-type with links to types in the ER; subtypes or inferred labels might be considered, as discussed in section 3.2, or information about the specific roles could be confined to the KBs referred to the ER with supertype

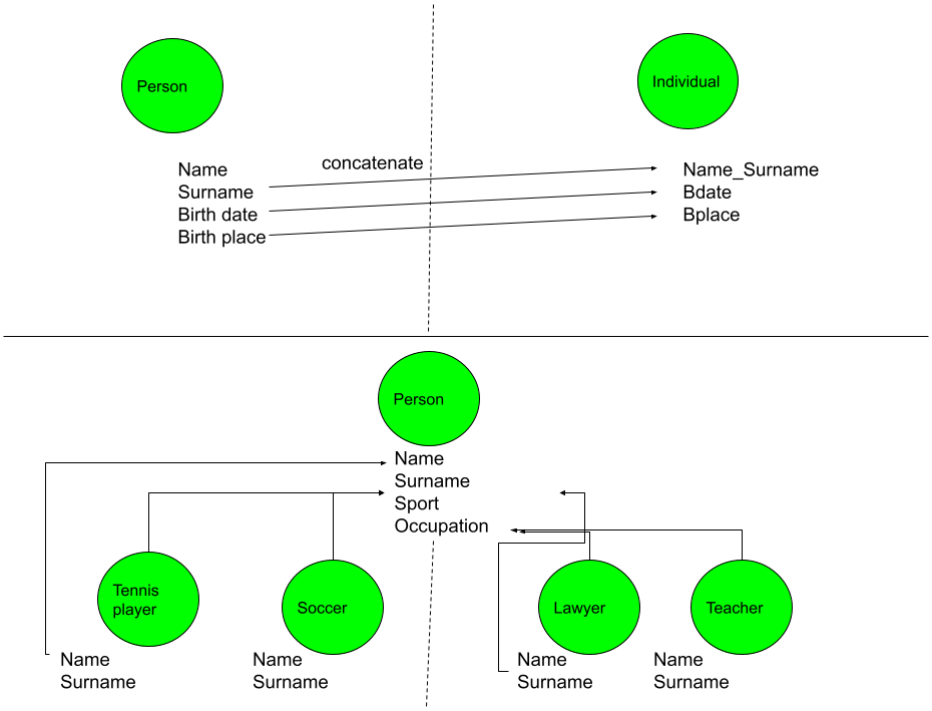

Figure 3. Examples of ER mapping

Figure 3 illustrates some cases of mapping between ERs: the upper part represents a case where simple entity types and attribute mappings can be found, as there is a one to one correspondence, but for the name and surname attributes, that can be managed by a concatenation function.

In the lower part a more complex situation is illustrated, where the mapping is achieved using a super type (Person), with attributes name and surname that are found in all the original entity types and new attributes sport (taken from the entity type label in the ER on the left) and occupation (taken from the entity type label in the ER on the right).

5. BASIC USAGE OF AN ENTITY REGISTRY

In this section we give some examples of an entity registry at work, when it is used to check KBs referred to it. For the sake of concreteness, we will represent the KBs as graphs, that is we will consider more specifically Knowledge Graphs (KG), our consideration will have however a general character and the translation in terms of KB in other forms should be straightforward.

We suppose that the KBs we consider are referred to the same ER, as we can always suppose that necessary operations described in the previous section have been carried on, in case they were originally referred to different ERs. This means that each node has attributes representing its entity type and entity ID. We make the additional hypothesis that all the attributes of nodes and relationships in both the KGs have the same names, or that we have at disposal a mapping between them.

We consider the task of comparing information about nodes, in two KGs, that are related to the same entities, based on attributes and relationships in each KG. We consider the following cases:

- The node related to entity E1 in KG1 has the same attributes as the node related to E1 in KG2, in this case nothing has to be done
- The node related to entity E1 in KG1 has some more attributes than the node related to E1 in KG2, and the attributes are compatible; in this case we can merge the knowledge
- The node related to entity E1 in KG1 has some attributes not compatible with those of the node related to E1 in KG2, and in at least one of the KGs the attributes are not a full key; in this case the checking procedure suggests an entity split based on the conflicting attributes
- The node related to entity E1 in KG1 has some attributes not compatible with those of the node related to E1 in KG2, and in both the KGs the attributes are a full key; in this case the checking procedure suggests an information revision, to find the correct attributevalues
- The node related to entity E1 in KG1 has the same relationships as the node related to E1 in KG2, in this case nothing has to be done
- The node related to entity E1 in KG1 has some more relationships than the node related to E1 in KG2, and the relationships are compatible; in this case we can merge the knowledge
- The node related to entity E1 in KG1 has some relationships not compatible with those of the node related to E1 in KG2, and in at least one of the KGs the attributes are not a full key; in this case the checking procedure suggests an entity split based on attributes that can be deduced by the conflicting relationships
- The node related to entity E1 in KG1 has some relationships not compatible with those of the node related to E1 in KG2, and in both the KGs the attributes are a full key; in thiscase the checking procedure suggests an information revision, to find the correct relationships

We note that relationship compatibility must be checked against both cardinality and incompatible relationship types, taking into account, if necessary, the relationship attributes that may constrain the conflicts. All the relevant information should be available in the metamodel.

Once all checking has been performed and possible conflicts have been resolved either with node splits or with information revision, it will be possible to query all the KG / KB using the entity identifiers. This actually results in a merge of all the available knowledge.

Finally, we mention two optional operations, to check similarity of two entities: the first consists in evaluating the percentage of common attributes of a node related to entity E1 and one related to E2, in order to compute a similarity degree of attributes. The second consists in evaluating the percentage of common links of the pair of nodes, in order to compute a similarity degree of relationships. In both cases, weight for the importance of attributes or relationships may be used.

These operations reconnect the ER to the more usual procedures of evaluating NEL candidates, that can be used for this task. On the other hand, to compare relationships a large number of methods can be found, related to exact and inexact graph matching, pattern recognition and so on (see e.g. [11]). Both techniques pave the way to the introduction of uncertainty in the evaluation of mentions in association with referred entities, and specific methods for uncertainty management (as outlined for instance in [12]) might be used to deal with the resulting scenarios.

6. CONCLUSIONS

The present paper describes the conceptual architecture, functions and application of a repository aiming at uniquely identifying entities mentioned in documents, extracted and become part of Knowledge Bases or Knowledge Graphs. This repository, that we name Entity Registry, is specifically intended for local contexts, involving entities that normally cannot be found in the big Knowledge Bases used in the usual Entity Linking process. On the contrary, an ER is thought as being gradually populated with entity found in specific document sets, like those pertaining a single criminal investigation or lawsuit, as each new document is considered.

We showed how different ER can be related to each other and merged, with the goal of putting together increasing knowledge corpora, relevant for a given application.

In a future work, we will strengthen the links between Entity Registries and more usual procedures, like NER and NEL, in order to provide, where possible, automatic analytic tools.

REFERENCES

[1] Batini, Carlo & Bellandi, Valerio & Ceravolo, Paolo & Moiraghi, Federico & Palmonari, Matteo & Siccardi, Stefano. (2021). "Semantic Data Integration for Investigations: Lessons Learned and Open Challenges." In *2021 IEEE International Conference on Smart Data Services(SMDS)*, pp.173-183.

[2] Nadeau, David & Satoshi Sekine. (2007) "A survey of named entity recognition and classification." *Lingvisticae Investigationes* Vol. 30, pp. 3-26.

[3] Honnibal, M., & Montani, I. (2017). "spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing." *To appear*.

[4] Sevgili, Özge & Shelmanov, Artem & Arkhipov, Mikhail & Panchenko, Alexander & Biemann,Chris (2022). "Neural entity linking: A survey of models based on deep learning." *Semantic Web* Vol. 13, No. 3, pp. 527–570.

[5] Hachey, B. & Radford, W. & Curran, J.R. (2011). "Graph-Based Named Entity Linking with Wikipedia." In: *Bouguettaya, A., Hauswirth, M., Liu, L. (eds) Web Information System Engineering – WISE 2011. WISE 2011*. Lecture Notes in Computer Science, vol 6997.

[6] Tedeschi, Simone & Conia, Simone & Cecconi, Francesco & Navigli, Roberto. (2021). "Named Entity Recognition for Entity Linking: What Works and What's Next." In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pp. 2584–2596

[7] Hachey, Ben & Radford, Will & Nothman, Joel & Honnibal, Matthew & Curran, James R.. (2013). "Evaluating Entity Linking with Wikipedia." *Artificial Intelligence* Vol. 194, pp. 130–150

[8] Logan, Robert L & McCallum, Andrew & Singh, Sameer & Bikel, Dan. (2021). "Benchmarking Scalable Methods for Streaming Cross Document Entity Coreference". In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing* Vol.1, pp. 4717-4731

[9] Monahan, Sean & Lehmann, John & Nyberg, Timothy & Plymale, Jesse & Jung, Arnold. (2011) "Cross-Lingual Cross-Document Coreference with Entity Linking." *Theory and Applications of Categories*

[10] Pozzi, Riccardo & Moiraghi, Federico & Lodi, Fausto & Palmonari, Matteo (2022). "Evaluation of Incremental Entity Extraction with Background Knowledge and Entity Linking". In *Proceedings of the 11th International Joint Conference on Knowledge Graphs(IJCKG 2022)*

[11] Carletti, Vincenzo. (2016) “Exact and Inexact Methods for Graph Similarity in Structural Pattern Recognition” *PhD thesis of Vincenzo Carletti.. Computer Vision and Pattern Recognition [cs.CV]. Université de Caen; Universita degli studi di Salerno*

[12] Bellandi, Valerio & Frati, Fulvio & Siccardi, Stefano & Zuccotti, Filippo (2022) “Management of Uncertain Data in Event Graphs”. In *Information Processing and Management of Uncertainty in Knowledge-Based Systems. IPMU 2022. Communications in Computer and Information Science* Vol. 1601