Merging open knowledge extracted from text with MERGILO

Misaël Mongiovì a, Diego Reforgiato Recupero a, Aldo Gangemi a,b, Valentina Presutti a, Sergio Consoli a,d

a Semantic Technology Lab, ISTC-CNR, Rome and Catania, Italy
b Paris Nord University, Sorbonne Cité, CNRS UMR7030, France
c University of Cagliari, Via Ospedale 72, Cagliari 09124, Italy
d Philips Research, Data Science Group, High Tech Campus 36, 5656AE Eindhoven, The Netherlands

A R T I C L E   I N F O

Article history:
Received 15 November 2015
Revised 25 April 2016
Accepted 8 May 2016
Available online 10 May 2016

Keywords:
Knowledge reconciliation
Coreference resolution
Knowledge base integration
Graph alignment

A B S T R A C T

This paper presents MERGILO, a method for reconciling knowledge extracted from multiple natural language sources, and for delivering it as a knowledge graph. The underlying problem is relevant in many application scenarios requiring the creation and dynamic evolution of a knowledge base, e.g., automatic news summarization, human–robot dialogue, etc. After providing a formal definition of the problem, we propose our holistic approach to handle natural language input – typically independent texts as in news from different sources – and we output a knowledge graph representing their reconciliation. MERGILO is evaluated on its ability to identify corresponding entities and events across documents against a manually annotated corpus of news, showing promising results.

© 2016 Elsevier B.V. All rights reserved.

1. Introduction

This paper focuses on the problem of acquiring knowledge from multiple natural language (NL) sources and reconciling it in an integrated formal representation. This problem, referred to as knowledge reconciliation, is relevant in most application scenarios that require to create and evolve a knowledge base from multiple and dynamic NL sources, for example: (1) building an integrated knowledge view, e.g., a summary, about a specific event, e.g., the Opening of 2012 London Summer Olympics, by acquiring knowledge from different newspapers [1]; (2) supporting human–machine dialogue in the context of assistive robotics by collecting a patient’s personal memories, which are provided through NL inputs over time. Let us consider the following news from two different sources:

“Tony awards: ‘Fun Home’ and ‘Curious incident’ big winners.”

and

“On Broadway’s biggest night ‘Fun Home’ wins Tony award for Best Musical”

In an ideal scenario, the goal is to automatically produce an integrated knowledge graph 1 such as the one depicted in Fig. 1. 2

Solving this problem requires semantic parsing of multiple natural language texts, transforming them to a formal representation, and identifying common vs. different parts in order to reason over an integrated knowledge graph associated with its textual provenance. Regardless of the chosen order, all these tasks must be addressed. Transforming natural language to formal representations has been investigated in ontology learning [3] and machine reading [4,5]; recognizing common parts in multiple sources is differently addressed by means of text similarity [6], co-reference resolution [7–9], ontology matching [10], and knowledge base integration [11,12].

In this paper, we describe in detail and experiment MERGILO, an improved version of the method proposed in [13] to handle multiple NL inputs, typically short text inputs such as news, in order to output knowledge graphs representing the integrated knowledge that they express. Integrating knowledge from multiple NL sources is crucial in order to implement intelligent applications requiring the ability to evolve a multi-source and dynamic knowledge base. However, this problem is challenging, considering that natural language can use heterogeneous forms for expressing similar knowledge. To complicate the situation, evaluation is also hard since no gold standards are available and not even universal standards for knowledge representations exist (different representations would require different gold standards). In this paper, after formally introducing the problem and presenting MERGILO, we describe how we built a gold standard (we will also refer to it as the ground truth), through a semi-automatic process, and

1 Corresponding author.
E-mail address: misael.mongiovii@istc.cnr.it (M. Mongiovì).

2 This paper refers to RDF/OWL[2] as the primary knowledge representation language for knowledge graphs.

http://dx.doi.org/10.1016/j.knosys.2016.05.014
0950-7051/© 2016 Elsevier B.V. All rights reserved.
starting from an existing annotated corpus for Cross-Document Coreference Resolution. The methodology is generalizable to other formal representations, and consists in generating a set of yes/no questions that can be answered by non-skilled people, using a crowd sourcing platform (in our case CrowdFlower\(^3\)) to get the answers, and automatically generating the gold standard from the original corpus and the answers. In addition, we have tested our method against the generated gold standard, and compared the results to those produced by the existing baseline method.

**Fig. 2** shows the overall pipeline of MERGILIO: the input text sources are parsed and transformed into RDF knowledge graphs, then the knowledge graphs are reconciled by identifying their common parts. The first step is performed by reusing a state-of-the-art approach [5] (discussed in Section 3), while the second step is performed by means of a knowledge reconciliation method based on frame semantics and network alignment.

The rest of the paper is organized as follows: Section 2 discusses relevant related research. Section 3 presents our knowledge representation approach. Section 4 introduces our method to solve the knowledge reconciliation problem. Section 5 is dedicated to the evaluation of the proposed approach. Finally, Section 6 draws conclusions and shows directions where we are heading.

### 2. Related work

**Cross-document Coreference resolution.** The closest task to knowledge reconciliation, as defined in literature, is the NLP task known as Cross-document Coreference Resolution (CCR) [7]. CCR aims at associating mentions about a same entity (object, person, concept, etc.) across different texts. Relevant work addressing cross-document coreference resolution includes [14–16], [7] uses spectral clustering and graph partitioning, and [17] is based on bag of words, latent similarity and clustering techniques. This problem is defined and solved in terms of text fragments, rather than formal constructs such as those composing a knowledge graph. Therefore the results of CCR are “extractive”, and not applicable in “abstractive” tasks\(^4\) that require a machine-usable representation of knowledge. Trying to transfer the knowledge from a CCR output to an abstract representation is hard. The identification of text fragments for annotating mentions is not unambiguously defined. For example in the sentence “People said Reid’s representative Jack Ketsoyan confirmed...” of the EECB gold standard for CCR, the whole text fragment “Reid’s representative Jack Ketsoyan” is considered a mention (which clearly refers to “Jack Ketsoyan”). However, parts of this text – taken alone – refer either to the same entity (e.g., “Jack”, “representative”, “Ketsoyan”) or to other ones (e.g., “Reid”). Connecting that mention to the correct entity of an abstract representation is a non-trivial task that requires itself some degree of comprehension of the text. In contrast, solving the problem at an abstract level does not require handling text fragments, and has the further advantage of enabling the exploitation of additional information, including relations and semantic annotations, in order to improve the results.

\(^3\) http://www.crowdflower.com/

\(^4\) Abstractive means that the result of text analysis is not a (set of) text segment(s), but rather a representation of a text in a knowledge representation language, cf. [18] for a definition of abstractive techniques in NLP.
Event coreference. When extracted entities are events, the problem changes to resolution of event coreference across documents [19,20]. Authors in [20] jointly model named entities and events. Clusters of entity and event mentions are constructed and merged accordingly to a similarity threshold based on linear regression. Then, information flows between entity and event clusters through features that model semantic role dependencies. The system handles nominal and verbal events as well as entities, and the joint formulation allows information from event coreference to help entity coreference, and vice-versa. Joint entity and event cross-document co-reference is similar to our reconciliation problem. The main difference from our work is that we work at the knowledge graph level, while they work purely at the textual level. A recent work from Vossen et al. [21] leverages on RDF and language annotation frameworks for solving the event co-reference problem. Using a language-independent representation, they are able to find co-references not only across documents but also across different languages. Their system first processes the text (newspaper articles) and produces an interoperable representation of events in NAF format [22]. NAF represents several kinds of text annotations, including tokens, entities, semantic roles and time expressions. Elements are also disambiguated to DBpedia, FrameNet, WordNet and PropBank, and time expressions are converted to dates by leveraging on the date of publication of the article. Entities that are associated to the same DBpedia entity are considered co-referenced, while event co-references are detected by associating events whose predicates, places, participants and temporal references match. This work has several similarities with ours. We both leverage on RDF for representing knowledge extracted from text. We both disambiguate entities, word senses and semantic roles by linking to external sources. However, there are also many differences. Our approach does not leverage on time information (date of publication of the article) and is more flexible in associating entities and events. Indeed it enables entities that are not disambiguated to be co-referenced based on their association to other entities and events (see Section 5 for real examples). Furthermore, our approach enables events that are not temporally anchored (time information are not available) to be co-referenced, when sufficient information supports this hypothesis. This is obtained thanks to a global optimization on the whole knowledge graphs representing the text documents. In summary, the approach of Vossen et al. [21] and ours are complementary in that they focus on orthogonal aspects of graph-based event co-reference, i.e. compositionality of events and graph alignment, respectively.

From text to Linked data and RDF. Several approaches have been proposed to fill the gap between NLP and RDF by providing frameworks for linguistic annotation and abstract representation of text. Notable work for representing linguistic annotations includes NIF [23], OLiA [24] and DADA [25]. More specifically, GAF [22] is a recently proposed annotation framework for event representation. These frameworks are very helpful for integrating results from different NLP tools and have been leveraged by recent tools for event co-reference (e.g., [21]). Representing the meaning of text is more difficult and most of the work in this field has been focused on learning and populating specific (task oriented) ontologies. A nice survey, presented in [3], identifies seven systems representing the state of the art in the area, and describes the typical tasks addressed by ontology learning systems, as well as their functionalities and implemented techniques. Although it is a hard task, generating a high-level representation of the text has been proved to be effective in many fields including sentiment analysis [26–28], affective computing [29,30], common-sense reasoning [31,32]. Most ontology learning and population systems focus on deriving a schema-level formal representation of the knowledge expressed by a text source (e.g., concepts and taxonomical relations, axioms, etc.), while fact-level knowledge extraction is mainly addressed by ontology population tools, which require an existing target ontology and large-size text corpora. Many of them also need some manual intervention. Recently, more general purpose approaches have been proposed, including Abstract Meaning Representation (AMR) [33], which defines a semantic language to represent the meaning of thousands of English sentences (however, an implementation is not provided yet) and FRED [5,34] a method that transforms natural language text into RDF-OWL graphs by

---

5 [http://wit.istc.cnr.it/stlab-tools/fred](http://wit.istc.cnr.it/stlab-tools/fred)
leveraging the output of several NLP tools, and by using frame semantics [35] as a reference linguistic theory. Its main limit relates to schema-level axiomatization, i.e. it does not represent disjointness, and other OWL restrictions. All discussed approaches focus on knowledge representation, while we focus on knowledge integration [36]. However, they provide a level of organization of knowledge that makes abstract-level integration possible.

Knowledge base integration and ontology matching. A rich overview of ontology matching methods is provided by [10]. As for knowledge base integration, relevant work includes [12] that leverages the interplay between schema and instance matching. Similarly, [11] shows a simple greedy iterative algorithm for aligning knowledge bases with millions of entities and facts. These approaches are characterized by the preferred large size of the ontologies/datasets treated (for best performance), which rarely (probably never) derive from text sources. On the contrary, we aim at handling knowledge graphs derived from text sources, and modeled using a frame-semantics-based representation. They are aligned according to similarity measures that exploit frame semantics features, combined with an Integer Linear Programming (ILP) graph matcher. In general, ontology alignment and knowledge base integration methods have goals close to knowledge reconciliation. Besides the specificity of textual knowledge, the main difference is that they are designed for handling either schema-level entities (ontology concepts and relations), or large knowledge bases, respectively. In most cases, they require manual intervention for annotating seed examples, or huge corpora for training, in order to reach their best performance.

3. Knowledge extraction

A knowledge graph is a fully labeled multi-digraph, the RDF abstract data structure [2] being the primary example. It is characterized by multiple semantic layers, i.e. nodes and edges, which may represent schema entities, data entities, meta-data entities, linguistic entities, (named) sub-graphs, etc. NL constructions can be recognized from parsing text fragments, but their formal semantics needs to be represented as a knowledge graph in a formalization step. In our approach, we start by parsing and normalizing texts into RDF-OWL. We mainly expect to target relatively short texts (the size of documents in our benchmark varies from 457 to 1904 characters, with an average of 810 characters), and we need to represent concepts, relations, and factual knowledge, with less emphasis to schema-level axioms such as disjointness, cardinality restrictions, etc. Considering our requirements, FRED results to be the most appropriate as an open extractor (i.e. unsupervised and domain-independent) among the available tools. Indeed it handles corpora of short texts, producing libraries of related RDF named graphs including fact-level as well as basic schema-level triples, and is available as a REST service. It uses a simple frame semantics, which is well suited to our method. There are no limits in the size of documents that FRED can handle. The only thing to note may be represented by occasional errors of Boxer [37] (a tool in the FRED’s pipeline) when performing the deep parsing that might break the entire parsing of FRED.

We focus on four kinds of objects that are either denoted or expressed by linguistic constructions emerging from a semantic parsing of text in a frame semantics perspective:

1. Named and skolemized entities: entities with either a public name (named entities), or a machine-generated name (skolemized entities), which are assumed to be denoted/referenced by a

NL construction. They include individual persons, organizations, places, artifacts, theories, etc.

2. Event occurrences: entities that describe the occurrences of an event. In general an event is described by a set of triples \((E, P_i, O_i)\) with \(i \in \{1, n\}\), where \(E\) has a machine-generated name generated starting from the verb, and \(P_i\) is a semantic role classifying an argument \(O_i\) of \(E\). Event occurrences represent complex NL constructions (with dependency or categorical relations), and typically denote complex facts, e.g., \(\text{Anto gave a candy to Bilal; the officials report that Italy has an increase in corruption; Gibson is the author of Neuronomer; my breakfast with Goofy in the canteen, etc.}\)

3. Classes: entities with either a public name (words, terms), or a machine-generated name (set builders), which are assumed to represent a categorizing NL construction, and typically denote a set of individual entities or events, e.g., \(\text{dog, nation, city, breakfast, run, etc.}\)

4. Qualities: entities that are assumed to be inherent characteristics of an entity or event, e.g., \(\text{nice, strong, hardly, in a sweet way, etc.}\)

Since FRED performs word sense disambiguation and entity linking, some entities of the resulting graph are linked to external sources (DBpedia and VerbNet).

Properties of FRED’s graphs are divided in two macro-categories: roles and non-roles. Roles are outgoing edges from event nodes. Role edges are broadly classified into agentive, passive, and oblique roles. In [26] we have described in detail the three classes above, defining which role category falls in each of them. All other edges are non-role edges. Some of the non-role edges include \(\text{owl:sameAs, owl:equivalentClass, rdf:type and rdfs:subClassOf},\) with standard meaning from RDFS and OWL ontology specification languages.6

An example of FRED’s output is reported in Fig. 3 although the reader is invited to play with FRED online and check how it represents knowledge. The graph contains a number of individual entities (represented by a purple diamond) and classes (represented by a yellow circle), connected by relations. Individuals may represent concrete entities (persons, objects, places), or events or situations. For instance, the node \(\text{fred:begin}_1\) represents the event of starting of the pre-production process. It is connected to the entity \(\text{fred:Evil}\), which represents its agent, to the node \(\text{fred:process}_1\), which represents its theme (i.e. the pre-production process), and to the date of starting (green box). \(\text{fred:Evil}\) is associated to its corresponding DBpedia entity (node \(\text{dbpedia:Evil}\)), indicating that it is a publicly known concept. Each individual is also associated to its type, when known. For instance \(\text{fred:Evil}\) has type \(\text{schemao:Organization}\) and \(\text{schemao:MusicGroup}\).

4. Knowledge reconciliation

In the following we discuss in detail MERGIL0, our method for reconciling knowledge extracted from text using FRED. The main issue in reconciling two FRED graphs consists in detecting nodes of the two graphs that correspond to the same entity. Specifically our problem can be stated as follows:

Definition 1. Given two FRED graphs \(G_1 = (V_1, E_1, P_1)\) and \(G_2 = (V_2, E_2, P_2)\), where \(V_1\) and \(V_2\) represent nodes (entities), \(E_1\) and \(E_2\) represent edges (relations), and \(P_1\) and \(P_2\) represent edge labels (properties), find a complete list of node pairs \((v_1, v_2)\) (cross-graph co-references), with \(v_1 \in V_1\) and \(v_2 \in V_2\), such that each pair of the list refers to the same entity.

6 http://wit.istc.cnr.it/stlab-tools/fred/api
7 http://www.w3.org/TR/rdf-schema/, http://www.w3.org/TR/owl2-overview/
8 Prefix fred: stands for http://www.ontologydesignpatterns.org/ont/fred/
Complete means that if two entities $v_1 \in V_1$ and $v_2 \in V_2$ represent the same entity, the list of cross-graph co-references must include $(v_1, v_2)$.

In the above definition we assume that two mentions of the same entity in the same sentence are identified by a single node in the corresponding FRED graph. In real cases there might be different RDF entities that are recognized as equivalent (e.g., by a sameAs relation). We assume that such entities have been collapsed into one single entity. The problem of finding co-referenced entities within a text document is solved by FRED by means of Boxer [37] and CoreNLP. [9]

Figs. 4–6 depict the pipeline of our graph-alignment-based method for solving the problem defined above for two given sentences supplied as an example. After the sentences are parsed by machine reading (Fig. 4), the resulting graphs are first compressed by merging nodes and removing unnecessary URIs. The two compressed graphs are aligned by establishing a 1–1 correspondence between nodes of the first graph and nodes of the second graph that maximizes a score function, which combines the similarity between aligned nodes and the similarity between aligned edges (Fig. 5). Maximizing the score function has the effect of aligning nodes that have high similarity and that are in turn connected to edges with high similarity. Therefore both element similarities and structural information are considered. At the end, the aligned nodes are mapped to individuals in the original graphs and sameAs relations are added between aligned nodes. The final output of our knowledge reconciliation method, MERCULO, for the two input sentences is reported in Fig. 6.

In the following subsections we give details on (i) graph compression, (ii) node and edge similarity and (iii) the graph alignment algorithm.

4.1. Graph compression

Graph compression aggregates clusters of nodes in order to obtain abstracted graphs with less, more informative, nodes. This step is necessary for two reasons. First, the same entity may be represented by different equivalent nodes. Collapsing all equivalent nodes reduces the number of cross-graph associations to be found and increases their quality. Second, it enables aggregating type information to nodes, therefore increasing the amount of information that helps associating nodes across graphs. The abstraction process has two steps: aggregating equivalent nodes and aggregating types.

Aggregating equivalent nodes. We aggregate all equivalent nodes (e.g., nodes connected by sameAs relations). More in detail, we consider the set of connected components of the input graph restricted to only edges of the type owl:sameAs,
owl:equivalentTo, coref or other_coref and collapse each connected component into one single node. The aggregated node will contain all URIs of the original nodes. The aggregated nodes will also inherit all connections (edges) of the original nodes except rdf:type, rdfs:subClassOf, owl:sameAs, owl:equivalentTo, coref and other_coref.

Aggregating types. For every individual node \( v \) we collect all nodes that are rdf:type of \( v \) (i.e. all related class nodes). They include all nodes that are rdfs:subClassOf any node that is rdf:type of \( v \) (recursively). We then augment the URIs of the corresponding aggregated node with the labels of collected nodes, except very general URIs (such as dul:Event). Then we remove the original class nodes. Note that the same URI of a type node may be duplicated in more than one aggregated node. This is possible since two distinct individuals may refer to the same type node.

For instance, from the graph in Fig. 3, the compressed graph generated would have nine nodes with the following URIs:

1. fred:Begin_1; fred:Begin; vn.data:Begin_55010100;
2. 2009-02-01;
3. fred:Evile; dbpedia:Evile; schemaorg:Organization;
   schemaorg:MusicGroup;
4. fred:process_1; fred:Pre-productionProcess;
   fred:Process;
5. fred:Pre-production;
6. fred:album_1; fred:SecondAlbum; fred:Album;
7. fred:Second;
8. fred:thing_1; boxer:Thing;
9. fred:Russ_russel; dbpedia:Russ_Russel.

4.2. Node and edge similarity

Similarity measures for nodes and edges are used by the optimizer to define the alignment score function. The similarity can be positive or negative. Elements that have negative similarity tend not to be associated, while elements with positive similarity tend to be associated. Note that the alignment algorithm performs a global optimization, and hence local parts of the alignment may be penalized in favor of a global reward. For instance, two edges with positive similarity may not be aligned because this would imply aligning their endpoint nodes with negative similarity. Similarly, two nodes with negative similarity may be aligned to enable aligning incident edges with positive similarity.

We consider a subset of inter-graph node pairs (the same for edge pairs) that are associable and define a similarity measure between such nodes (or edges). If two nodes (or edges) are not associative we say that their similarity is \(-\infty\). As an example of associative nodes, a named entity fred:Evile (a musical band) in a graph (e.g., Fig. 3) may be associated with a skolemized entity fred:band_1 in another graph. Instances of associative edges might be vnrole:Agent and vnrole:Actor (cf. both agentive roles). They would be actually associated if both pairs of their source nodes and destination nodes are associated as well.

Node similarity. We distinguish among three kinds of node pairs: relevant, compatible and incompatible. We first check if both nodes refer to named entities. If so, we check whether they refer to the same named entity or to different ones. Labels of named entities are compared both by string matching and by their alignment to public resources (DBpedia). If the labels are equal or are associated with the same DBpedia entity, the pair of nodes is considered relevant. Otherwise the nodes are considered incompatible. If one of the two nodes does not refer to a named entity, we check the similarity of all cross-node pairs of labels except those of skolemized entities (i.e. individuals that are not named entities, e.g., fred:process_1 in Fig. 3; we remind the reader that a node may have more than one URI) to see if the nodes share equivalent or similar concepts. We discuss below how similarity between URIs is computed. If the two nodes share the same URI or refer to words with similarity higher than a predefined threshold that we call similarity threshold, they are considered compatible. In all other cases, the nodes are considered incompatible.
To compute URI similarity, we first check whether the two URIs correspond to nodes of the same type (both events or both skolemized entities or both qualities or both classes etc.). If they do not have the same type, they are considered not similar. Otherwise, corresponding labels are extracted and label similarity is computed by semantic word-to-word similarity. We remind that URIs are collected (during the compression) from equivalent nodes, type nodes and subclasses. A URI is generated by FRED as a string with a common part and a variable part. The variable part is typically a word or a short text extracted from the input document or elaborated by reasoning (e.g., a node of the compressed graph that represents a reference “her” to “Tara Reid” will have as a label the variable part “person”, which is the same as the entity of a generic word “person” in the text). We strip off the common parts and apply the WordNet Lesk-Tanim similarity provided by SEMILAR [38] – a popular tool for text similarity – on the variable parts. We experimented with similarity thresholds in the range between 0.0 and 0.9 and got the best performance with 0.7.

Based on the node pairs classification above, the similarity between two nodes $v_1 \in G_1$ and $v_2 \in G_2$ is assigned as follows:

$$sim(v_1, v_2) = \begin{cases} 
1 & \text{if } v_1 \text{ and } v_2 \text{ are relevant} \\
-1 & \text{if } v_1 \text{ and } v_2 \text{ are compatible} \\
-\infty & \text{if } v_1 \text{ and } v_2 \text{ are incompatible}
\end{cases}$$

Although simple, we found experimentally that this scoring schema works better than other more complex ones. More complex score functions might work better in some cases but worse in other cases, compromising the overall performances. The rationale behind our scoring schema can be expressed intuitively as: $1 = \text{associate unless there are better choices}, -1 = \text{associate only if you have valid reasons to do it}, -\infty = \text{do not associate}.$

**Edge similarity.** The similarity between two edges is defined in terms of their type. Specifically, we distinguish between compatible and incompatible edges based on their property type and possibly their thematic role (cf. Section 3). If both edges are non-role edges, they are considered compatible. If both edges are role edges, they are considered compatible only if their roles are both agentive (AGNT) or passive (PTNT). In all other cases the edges are considered incompatible.

The similarity between two edges is defined as:

$$sim(e_1, e_2) = \begin{cases} 
1 + \omega + \epsilon & \text{if } e_1 \text{ and } e_2 \text{ are compatible} \\
-\infty & \text{if } e_1 \text{ and } e_2 \text{ are incompatible}
\end{cases}$$

where $\epsilon$ is a very small number (0.001) introduced to break ties (when different alignments produce the same score, we prefer the one with the highest number of aligned edges) and $\omega$ is a parameter that enables associating sets of compatible nodes if they are connected by sufficiently high numbers of edges. Just to give an intuitive meaning of $\omega$, to have a positive reward in associating two stars\footnote{A star is a subgraph with a central node connected to a set of peripheral (degree 1) nodes} of compatible (not relevant) nodes, the degree of the two centers must be at least $1/\omega$. In our experiments we set exactly $\omega = 1/3$.

**4.3. Alignment**

Once the similarity among nodes and edges has been defined, our problem can be described in terms of a graph alignment prob-
Graph alignment is a widely studied problem that has many applications in several fields [11,39–41]. It can be formulated as a quadratic assignment problem [39] and reduced to Integer Linear Programming [40]. The problem formulation we adopt in MERGILIO is designed specifically for directed multi-graphs (a pair of nodes can be connected by more than one edge) and is similar to other previously proposed formulations [42].

We denote with \( V(G) \) the set of nodes in \( G \) and with \( E(G) \) the set of triples \( e = (v_1, v_2, p) \) in \( G \) (edges of \( G \)) where \( v_1, v_2 \in V(G) \) and \( p \) is a property, drawn from the vocabulary \( P \) of properties.

An alignment between two graphs is defined as \( A = (AV, AE) \) where:

- \( AV \) is a set of pairs \((v_1, v_2)\), with \( v_1 \in V(G_1) \) and \( v_2 \in V(G_2) \), that defines a 1–1 correspondence between nodes in \( G_1 \) and nodes in \( G_2 \) (i.e., such that there are no couples of pairs \((v_1, v_2)\) and \((u_1, u_2)\) such that \( v_1 = u_1 \) or \( v_2 = u_2 \));
- \( AE \) is a set of pairs \((e_1, e_2)\), with \( e_1 \in E(G_1) \) and \( e_2 \in E(G_2) \), that defines a 1–1 correspondence between edges in \( G_1 \) and edges in \( G_2 \), and such that if \( e_1, e_2 \in AE \) then the endpoints of \( e_1 \) and \( e_2 \) form two pairs in \( AV \).

We aim at finding the alignment \( A = (AV, AE) \) that maximizes a suitable score function that considers both nodes and edges similarity. We define the score function as:

\[
\text{score}(G_1, G_2, AV, AE) = \sum_{(v_1, v_2) \in AV} \text{sim}(v_1, v_2) + \sum_{(e_1, e_2) \in AE} \text{sim}(e_1, e_2)
\]

### 4.3.1. Solving the alignment

Computing the optimal alignment is a NP-hard problem [42] and hence no polynomial-time algorithm for it is known. However, since the size of knowledge graphs generated from text is not very high, and this kind of graphs is usually sparse, standard optimization techniques are affordable. We reduce our problem into ILP (Integer Linear Programming) and use a standard solver for the optimization. ILP optimizers often converge to optimal solutions on small or medium size problem instances and provide good approximations with proved error bounds on larger instances.

We consider \( RV, RE \) as the sets of all possible associations of nodes and edges, respectively. We denote with \( x_{v_1,v_2} \), a variable that has value 1 if \((v_1, v_2) \in AV\), 0 otherwise, and with \( y_{e_1,e_2} \) a variable that has value 1 if \((e_1, e_2) \in AE\), 0 otherwise. The ILP formulation of our alignment problem is:

\[
\begin{align*}
\max & \quad \sum_{(v_1, v_2) \in AV} \text{sim}(v_1, v_2) \cdot x_{v_1,v_2} + \sum_{(e_1, e_2) \in AE} \text{sim}(e_1, e_2) \cdot y_{e_1,e_2} \\
\text{s.t.} & \quad \sum_{v_2 \in v_1} x_{v_1,v_2} \leq 1 \quad \forall v_2 \in V(G_2) \\
& \quad \sum_{v_1 \in v_2} x_{v_1,v_2} \leq 1 \quad \forall v_1 \in V(G_1) \\
& \quad \sum_{e_2 \in e_1} y_{e_1,e_2} \leq 1 \quad \forall e_2 \in E(G_2) \\
& \quad \sum_{e_1 \in e_2} y_{e_1,e_2} \leq 1 \quad \forall e_1 \in E(G_1) \\
& \quad x_{u_1,u_2} \geq \text{sim}(u_1, u_2) \cdot \forall (u_1, u_2) \in RE : (u_1, u_2) \in RV \\
& \quad y_{v_1,v_2} \geq \text{sim}(v_1, v_2) \cdot \forall (v_1, v_2) \in RE : (v_1, v_2) \in RV \\
\end{align*}
\]

Note that, although we do not explicitly aim at aligning edges, edge alignment is necessary for solving the graph alignment problem. Indeed a good graph alignment must conserve the structure as better as possible, i.e., edges in the first graph must have (as much as possible) a correspondence in the second graph and vice-versa.

We use a standard solver (IBM ILOG CPLEX 12.6.1) to find the optimal alignment between the input graphs. In most cases we find the optimal solution in fractions of a second. For larger problem instances it is possible to apply known efficient heuristics in change of a slight loss in quality [39].

Although we described the reconciliation of pairs of documents, we can handle multiple documents as well by multiple applications of pairwise document knowledge reconciliation. We first reconcile the first two documents, producing an initial knowledge base. Then, we iteratively integrate the abstract representation of the other documents to the existing knowledge base one document by one.

### 5. Experimental analysis

We implemented MERGILIO as a Python tool[12] on top of FredLib. [13] We used IBM ILOG CPLEX 12.6.1 for solving the Integer Linear Program.

A simple baseline method for our problem is to apply existing knowledge extraction tools on a single document built by appending all input documents one after another. However, this approach would produce a sparse representation that misses associations among concepts, entities and events across documents. An alternative approach consists in building multiple knowledge graphs, one per input document, and integrates them by existing knowledge base integration tools. Recent knowledge base integration tools [11] are based on graph alignment and hence they are somehow similar to our approach. However, a crucial part of these methods is the definition of rewards and penalties of aligned nodes and edges (node and edge similarity), which relies on similarity between label texts, and often requires manual intervention. Similarity functions defined for knowledge bases such as Yago, IMDB and Freebase are not adequate to compare the rich repertoire of entities and relations produced by knowledge extraction tools such as FRED. Our tool is specifically designed for this kind of graphs. Another possibility is to employ the tool by Vossen et al. [21] to find corresponding entities and events across documents. Although we expect high precision results when time information is available (e.g., we have the date of publication of articles), in our scenario, where time information is not available, this method is not able to match events and hence it would not do any better than simply associating entities disambiguated to the same DBpedia entity.

Besides the fact that no competing approaches are available, we do not have a benchmark for evaluation either. Instead of building a benchmark with ground truth annotations from scratch, we adapted an existing corpus for CCR. Among the corpora available (EECB [20], ECB++ [43], MEANTIME[14]), we chose a cluster of the EECB 1.0 [20] corpus (cluster 1). The EECB gold standard is a well-established extension of ECB [19], a corpus annotated with event co-references, that also contains entity co-reference annotations. We are currently considering evaluating our tool against the other corpora as well.

We employed a semi-automatic process to transfer the co-references from mentions into co-references between entities. The manual intervention is necessary since there is no well-assessed method to map a mention (text span) to an entity (that is usually associated to a single word in the text) with 100% reliability. For example in the sentence “Reid’s representative Jack Ketsoyan” the whole text is a mention that refers to “Jack Ketsoyan”. However, our abstract representation (RDF graph) does not contain any entity associated to the whole text span. In contrast it contains three different entities associated with “Reid”, “representative” and “Jack Ketsoyan”, respectively. It is not easy to distinguish which of

---

[11] http://bit.uni-stuttgart.de/stlab-tools/mergilio/  
[12] http://bit.uni-stuttgart.de/stlab-tools/fredlib  
[13] http://www.newsreader-project.eu/results/data/wikinews/
Fig. 7. An example of question for CrowdFlower workers. The correct answer to this question is “Yes” since both the highlighted text and the red underlined text refer to Tara Reid. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article).

![Graph showing precision and recall](image)

**Fig. 8.** Precision and recall obtained by varying the word similarity threshold.

these entities correspond to the mention above; it would require some degree of comprehension of the text.

We constructed the corpus by performing the following steps:

- building the RDF graph of each document;
- mapping RDF entities with mentions in the EECB gold standard;
- building clusters of entities from clusters of mentions.

The RDF graphs are built by running FRED on the documents of the EECB corpus. The hardest task is to establish the correspondence between RDF entities and text mentions. Once this correspondence is available, clusters of entities can be easily built by taking all entities that correspond to mentions in the same cluster of mentions (from the EECB annotations) and grouping them together.

To map RDF entities to EECB mentions we take advantage of entity-associated text spans generated by FRED during the construction of the RDF graph. Note that text-spans is the only information we get from FRED’s graphs, in order to avoid bias in the resulting gold standard. Each text span maintains the character offset of the part of original text associated to an entity. Often this text span differs from the corresponding mention in the gold standard. For example in the following sentence:

*Tara Reid, 33, who starred in ‘American Pie’ and appeared on U.S. TV show ‘Scrubs’, has entered the Promises Treatment Center*
FRED creates an entity fred:Tara_reid and connects it to the text span corresponding to “Tara Reid”. In contrast, in the EECB gold standard the whole text “Tara Reid, 33, who starred in ‘American Pie’ and appeared on U.S. TV show ‘Scrubs’” is associated to a mention that refers to Tara Reid. In this example the FRED’s text span is wholly contained in the EECB mention, but this is not always true in general. Indeed containment is neither a necessary nor sufficient condition for a FRED’s text span and an EECB mention to correspond.

To solve the mapping, we resort to a partially manual approach based on CrowdFlower. We recruited a number of non-skilled people (workers) to establish the correspondence between mentions. Specifically, each worker had to answer a collection of questions asking whether two text spans in a certain sentence correspond to the same entity (person, thing, event or concept) or not. An example of question is shown in Fig. 7. Questions are generated by considering all pairs (FRED’s text span / EECB mention) that partially overlap. Fully overlapping pairs are automatically assigned as corresponding, while non-overlapping pairs are assumed to be discordant.

The CrowdFlower job is composed by 280 questions that require a yes/no answer. Each question was solved by at least 3 people (when full agreement was reached) and up to 5 people (when there was low agreement). The agreement varied from 0.52 to 1.0 depending on the question. Agreement is computed as the sum of the trust score of workers that gave the aggregated answer divided by the total trust score of workers that answered that question. The trust score is a value between 0 and 1 assigned to each worker to measure its ability to solve the job. Completing the job costed 24 US dollars.

The obtained corpus is publicly available. It contains 19 RDF graphs with the abstract representation of the 19 input documents, 19 extended RDF graphs that also contain annotations of RDF entities with text spans in the original documents, and a file with the cross-graph correspondences among RDF entities. Corresponding entities are grouped into clusters. A cluster corresponds to a real-word entity or event. The corpus contains 43 clusters covering 145 RDF entities and 558 pairwise cross-graph correspondences.

We aligned pairs of documents from the corpus in all possible ways, and evaluated the results for each pair (171 pairs in total). We computed precision, recall and F-measure of the aligned

**Fig. 9.** Precision and recall obtained when varying the word similarity threshold when aligning entities only.

---

15 http://wit.istc.cnr.it/stlab-tools/mergilo
16 Note that the all-to-all alignment is just for evaluation purpose. Aligning multiple documents can be done by a linear number of pairwise alignments, as described in Section 4.
pairs of nodes across graphs. Since there might be several nodes in an RDF graph that correspond to the same entity (this is due to the way FRED builds RDF representations of input sentences), and we are interested in measuring the quality of the alignment across graphs (not within graphs), we collapsed all equivalent nodes within a graph into single nodes. We consider two nodes in a graph as equivalent if they are identified as equivalent by either the gold standard or FRED (e.g., connected by a sameAs relation, which by definition is created when the detected entity is found on DBpedia). Precision was computed as the percentage of aligned pairs identified by the tool that are also gold pairs. Recall was computed as the percentage of gold pairs that are also identified by the tool. The F-measure was computed as twice the product of precision and recall divided by their sum.

We run our tool with several values of the node similarity threshold (parameter introduced in Section 4.2) in the range between 0.1 and 1.0. The results are compared with a baseline method that only aligns named entities with the same name and entities that are linked to the same DBpedia entity by FRED (FRED uses TAGME [44] for linking to DBpedia entities). Precision and recall of MERGILO are shown in Fig. 8 together with the result of baseline (the average values among all tests is considered).

Our tool has a recall constantly higher than baseline, with a precision slightly lower than baseline. The best performances are obtained with a similarity threshold of 0.7 (0.85 precision, 0.51 recall and 0.61 F-measure). Low values of the similarity threshold penalize the precision while maintaining the recall almost constant. This is expected since words that have different meanings tend to be associated, thus increasing the number of false positives. With a similarity threshold of 1.0, MERGILO is equivalent to the baseline method, since only named entities with the same name and entities that are linked to the same DBpedia entity can be associated. The baseline method has a slightly higher precision than MERGILO (0.90) but a lower recall (0.41) and a significantly lower F-measure (0.53). The higher precision is expected since the baseline method only aligns entities with high confidence (linked to the same DBpedia entity or with the same name). Our method is able to reach a significantly higher recall with a slight loss in precision. Note that, in contrast with the baseline, our method is able to align events. Figs. 9 and 10 show the performances in aligning only entities and only events, respectively.

In aligning entities our method reaches the maximum precision (0.88) and recall (0.57) with a similarity threshold of 0.7. Again the method is stable when varying the parameter, with only small variations in the range 0.2–0.7. Precision is close to baseline (0.88 vs. 0.90) while recall is significantly higher (0.57 vs. 0.54), indicating that our method is able to align correctly entities that are not disambiguated to the same DBpedia entity. In aligning events,
both precision and recall increase slowly at increasing the similarity threshold, reaching the best performances at 0.8. With this similarity threshold precision is 0.33 and recall is 0.28, with F-measure equal to 0.29. Despite far from optimal, these performances are much better than baseline, whose precision and recall is zero, as expected. Note that these results are obtained without considering neither temporal information nor domain knowledge. Mapping corresponding events without further knowledge on the topic of discussion is a very hard task even for humans.

Our tool took one hour and forty-five minutes to process all 171 document pairs with a similarity threshold of 0.7. We did not perform extensive running time tests on large datasets since our first goal is accuracy. Our implementation is not optimized for efficiency, and contains several bottlenecks (word-similarity computation, rest interfaces among components, etc.). However, they can be handled by optimizing the implementation for efficiency (pre-computing word-similarity, making all components in-process, etc.). The main scalability concern of our method involves the numerical optimization based on ILP, since its resolution is NP-complete, and therefore the time complexity might be exponential in the worst case. Although the algorithm is exponential in the general case, real scenarios are relatively easy to solve thanks to the sparsity of the graphs and the small set of potential associations across graphs. Just to give an idea, the ILP optimization took fractions of seconds for each document pair. For large documents the running time might increase significantly. In this case it is possible to apply known scalable heuristics for graph alignment [39].

Finally, we report some interesting results found by our method. The following two sentences are fragments extracted from two documents of our benchmark that refer to the same event of reporting information operated by the representative of Tara Reid, Jack Ketsoyan.

“Perennial party girl Tara Reid checked herself into Promises Treatment Center, her rep told People.”

“A publicist says Tara Reid has checked herself into rehab.”

The abstract representation of these two events, built up from the words “told” in the first sentence and “says” in the second one, are correctly associated by our method. Such an association is possible since the agent is the same and the topic corresponds to an event that is in turn associated to co-referenced entities (the event of checking into Promises Treatment Center operated by Tara Reid). Our method is able to find the most suitable assignment through a global optimization of the match between the two abstract representations. Later on, in the same text documents, we find the expression “her family’s privacy”, where “her” refers to Tara Reid. Our method correctly associates the abstract representation of “family” and “privacy”. Again, only by exploring the relations of “family” and “privacy” with other entities, it is possible to perform the correct match.

6. Conclusions

The major challenge in automatic knowledge reconciliation is to make sense of similarity of multiple graphs, at the same time representing knowledge across schema-level, instance-level, temporal, spatial, and context-bound entities and relations. It is a long-term programme that proves sometimes difficult even for humans when processing e.g., a few news items, but more and more critical with the overwhelming amount of information delivered daily to our brains.

In this paper we presented MERGILIO, a method for generating and integrating knowledge graphs extracted from multiple text documents. Our tool relies on FRED, a machine reading tool for generating abstract representations of text documents, and integrates the generated knowledge by means of an optimization technique for graph alignment. We assessed the performance of our tool in identifying entities that correspond across documents. We showed a methodology to annotate a corpus and thus creating a gold standard by using the CrowdFlower platform. The results, obtained by comparing MERGILIO with a baseline on the generated corpus, show that our method is effective in integrating knowledge from multiple sources by correctly identifying co-referent entities and events. Ongoing work concentrates upon: (1) increasing the recall of reconciliation across multiple heterogeneous documents; (2) managing time-indexable relatedness, especially with reference to events; (3) combining ideas from the method by Vossen et al. [21] with our method in order to increase precision; (4) integrating MERGILIO into vertical applications, for example within the understanding component of a companion robot under development in an European project supporting active ageing of people with dementia. In the latter, the goal is to recognize and associate entities and relations referenced by a patient during robot–human dialogue.

Acknowledgments

The research leading to these results has received funding from the European Union Horizons 2020 the Framework Programme for Research and Innovation (2014–2020) under Grant agreement 643808 Project MARIO Managing active and healthy aging with use of caring service robots.

References

[1] R. Cattoni, F. Corcoglinetti, C. Girardi, B. Magnini, L. Serafini, R. Zanoli, The knowledgestore: an entity-based storage system, in: Proceedings of Language Resources and Evaluation Conference, LREC, 2012, pp. 3639–3646.
[2] R. Cyganiak, D. Wood, M. Lanhamer, RDF 1.1 Concepts and Abstract Syntax, W3C Recommendation 25 February 2014, online, 2014, http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/.
[3] W. Wong, W. Liu, M. Bennamoun, Ontology learning from text: a look back and into the future, ACM Comput. Surv. 44 (4) (2012) 1–36.
[4] A. Etzioni, M. Banko, M.J. Cafarella, in: Machine Reading, AAAI Press, 2006, pp. 1517–1519.
[5] V. Presutti, F. Draiaichio, A. Gangemi, Knowledge extraction based on discourse representation theory and linguistic frames, in: EKAW: Knowledge Engineering and Knowledge Management that Matters, in: Lecture Notes in Computer Science, vol. 7603, Springer, Berlin, Heidelberg, 2012, pp. 114–129.
[6] W.H. Gomaa, A.A. Fahmy, A survey of text similarity approaches, Int. J. Comput. Appl. 68 (13) (2013) 13–18.
[7] S. Singh, G. Wiekum, Cross-document co-reference resolution using sample-based clustering with knowledge enrichment, Trans. Assoc. Comput. Linguist. 3 (1) (2015) 15–28.
[8] M. Garcia, P. Camallo, An entity-centric coreference resolution system for per-entities with rich linguistic information, in: Proceedings of the 25th International Conference on Computational Linguistics, COLING 2014, ACL, 2014, pp. 741–752.
[9] V. Stoyanov, J. Eisner, Easy-first coreference resolution, in: Proceedings of the 24th International Conference on Computational Linguistics, COLING 2012, ACL, 2012, pp. 2519–2534.
[10] J. Euzenat, P. Shvaiko, Ontology Matching, second ed., Springer-Verlag, Heidelberg, 2013.
[11] S. Lacroix-Julien, K. Palla, A. Davies, G. Kasneci, T. Graepel, Z. Ghahramani, Sigma: simple greedy matching for aligning large knowledge bases, in: Proceedings of International Conference on Knowledge Discovery and Data Mining, KDDI2013, ACM, New York, USA, 2013, pp. 572–580.
[12] F.M. de la Manque, A. Bibeau, P. Senellart, PADIS: probabilistic alignment of relations, instances, and schema, in: Proceedings of the VLDB Endowment, vol. 5, VLDB Endowment, 2011, pp. 157–168.
[13] M. Mongiovi, D.R. Becupero, A. Gangemi, V. Presutti, A.G. Nuzzolese, S. Consoni, Semantic reconciliation of knowledge extracted from text through a novel machine reader, in: Proceedings of the 8th International Conference on Knowledge Capture, K-CAP 2015, ACM, New York, USA, 2015, pp. 25:1–25:4.
[14] D. Rao, P. McNamara, M. Dredze, Streaming cross document entity coreference resolution, in: Proceedings of the 23rd International Conference on Computational Linguistics, COLING 2010, ACL, Stroudsburg, PA, USA, 2010, pp. 1050–1058.
[15] M. Shishkov, A. Subramanyan, F. Pereira, A. McCallum, Large-scale cross-document coreference using distributed inference and hierarchical models, in: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, HLT ’11, vol. 1, ACL, Stroudsburg, PA, USA, 2011, pp. 793–803.
[16] N. Andrews, J. Eisner, M. Dredze, Robust entity clustering via phylogenetic inference, in: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, vol. 1, ACL, Stroudsburg, PA, USA, 2014.
[17] A.-C. Ngonza Ngomo, M. Röder, R. Usbeck, Cross-document coreference resolution using latent features, in: Proceedings of the Second International Conference on Linked Data for Information Extraction (LD4IE), in: CEUR Workshop Proc., vol. 1267, CEUR-WS.org, 2014.

[18] D.R. Adevé, E. Hovy, K. McKown, Introduction to the special issue on summarization, Comput. Linguist. 28 (4) (2002) 399–408.

[19] C. Bejan, S. Harabagiu, Unsupervised event coreference resolution with rich linguistic features, in: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, ACL, Uppsala, Sweden, 2010, pp. 1412–1422.

[20] H. Lee, M. Recasens, A. Chang, M. Surdeanu, D. Jurafsky, Joint entity and event coreference resolution across documents, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL, ACL, Stroudsburg, PA, USA, 2012, pp. 489–500.

[21] L. Bos, E. Laparra, L. Aldabe, G. Rigau, Interoperability of cross-lingual and cross-document event detection, in: Proceedings of the 3rd Workshop on Event's at the NAACL-HLT, 2015, pp. 108–116.

[22] A. Fokkens, M. Van Erp, P. Vossen, S. Tonelli, W.R. Van Hage, B. SynerScope, L. Serafini, R. Sprinzl, J. Hoeksema, Gaf: a grounded annotation framework for events, in: Proceedings of North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL HLT, vol. 2013, 2013, pp. 11–20.

[23] S. Hellmann, J. Lehmann, S. Auer, M. Brümer, Integrating NLP using linked data, in: Proceedings of the 12th International Conference on The Semantic Web, ISWC '13, in: Lecture Notes in Computer Science, vol. 8219, Springer Berlin Heidelberg, 2013, pp. 98–113.

[24] C. Chiarcos, An ontology of linguistic annotations, in: Proceedings of the LDV Forum, vol. 23, 2008, pp. 1–16.

[25] S. Cassidy, An RDF realisation of LAF in the DADA annotation server, in: Proceedings of the Fifth Joint ISO-ACL/SIGSEM Workshop on Interoperable Semantic Annotation, ISA-5, 2010, pp. 1–4.

[26] D. Reforgiato Recupero, V. Presutti, S. Consoli, A. Gangemi, A.G. Nuzzolese, SenticSpace: frame-based sentiment analysis, Cogn. Comput. 7 (2) (2014) 211–225.

[27] E. Cambria, A. Hussain, Sentic Computing: A Common-sense-based Framework for Concept-level Sentiment Analysis, vol. 1, Springer, Cham, Switzerland, 2015.

[28] A.G. Nuzzolese, M. Mongiovì, Detecting sentiment polarities with sentilo, in: Semantic Web Evaluation Challenges, Springer, 2015, pp. 244–250.

[29] E. Cambria, Affective computing and sentiment analysis, IEEE Intell. Syst. 31 (2016) 102–107.

[30] E. Cambria, A. Hussain, C. Havasi, C. Eckl, SenticSpace: visualizing opinions and sentiments in a multi-dimensional vector space, in: Proceedings of the 14th International Conference on Knowledge-Based and Intelligent Information and Engineering Systems, Part IV, in: Lecture Notes in Computer Science, vol. 6279, Springer, Berlin, Heidelberg, 2010, pp. 385–393.

[31] E. Cambria, T. Mazzocco, A. Hussain, C. Eckl, Sentic Medoids: Organizing Affective Common Sense Knowledge in a Multi-dimensional Vector Space, in: Lecture Notes in Computer Science, vol.6577, Springer, Berlin, Heidelberg, 2011, pp. 601–610.

[32] S. Poria, A. Gelbukh, E. Cambria, A. Hussain, G.-B. Huang, EmsenticSpace: a novel framework for affective common-sense reasoning, Knowl.-Based Syst. 69 (2014) 108–123.

[33] L. Banarescu, C. Bonial, S. Cai, M. Georgescu, K. Griffith, U. Hermjakob, K. Knight, P. Koehn, M. Palmer, N. Schneider, Abstract meaning representation for Sembanking, in: Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse, 2013, pp. 178–186.

[34] D. Reforgiato Recupero, A.G. Nuzzolese, S. Consoli, V. Presutti, M. Mongiovì, S. Peroni, Extracting knowledge from text using SHELDON, a semantic holistic framework for linked ontology data, in: Proceedings of the 24th International Conference on World Wide Web, WWW '15 Companion, 2015, pp. 235–238.

[35] C.J. Fillmore, Frame semantics, in: Linguistics in the Morning Calm, Hanshin Publishing Co., Seoul, South Korea, 1982, pp. 111–137.

[36] S. Poria, A. Gelbukh, E. Cambria, P. Yang, A. Hussain, T. Durran, Merging SenticNet and WordNet-Affect emotion lists for sentiment analysis, in: Proceedings of the 11th IEEE International Conference on Signal Processing, ICSP, vol. 2, IEEE, 2012, pp. 1251–1255.

[37] J. Bos, Wide-coverage semantic analysis with boxer, in: Proceedings of the 2008 Conference on Semantics in Text Processing, STEP '08, ACL, Stroudsburg, PA, USA, 2008, pp. 277–286.

[38] V. Rus, M.C. Lintean, R. Banjade, N.B. Niraula, D. Stefanescu, SEMILAR: the semantic similarity toolkit, in: Annual Meeting of the Association for Computational Linguistics, ACL (Conference System Demonstrations), 2013, pp. 163–168.

[39] J.T. Vogelstein, J.M. Controy, V. Lyzinski, L.J. Podrazik, S.C. Katzner, E.T. Harley, D.E. Fishkind, E.J. Vogelstein, C.E. Priebe, Fast approximate quadratic programming for graph matching, PloS One 10 (4) (2015) e0121002. PMID: 25886624.

[40] G.W. Klau, A new graph-based method for pairwise global network alignment, BMC Bioinform. 10 (1) (2009) 1–9.

[41] M. Mongiovì, R. Di Natale, R. Giugno, A. Pulvirenti, A. Ferro, R. Sharan, Sigma: a set-cover-based inexact graph matching algorithm, J. Bioinform. Comput. Biol. 8 (02) (2010) 199–218.

[42] M. Mongiovì, R. Sharan, Global alignment of protein–protein interaction networks, Data Min. Syst. Biol.: Methods Protoc. 939 (2013) 21–34.

[43] A. Cybulski, P. Vossen, Using a sleighhammer to crack a nut? Lexical diversity and event coreference resolution, in: Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14), European Language Resources Association (ELRA), Reykjavik, Iceland, 2014, pp. 4545–4552.

[44] F. Ferragina, U. Scalisi, Tagme: on-the-fly annotation of short text fragments (by wikipedia entities), in: Proceedings of the 19th ACM International Conference on Information and Knowledge Management, CIKM ’10, ACM, New York, USA, 2010, pp. 1625–1628.