

FAIR Knowledge Graph construction from text, an approach applied to fictional novels

Diego Rincon-Yanez¹, Sabrina Senatore¹

¹University of Salerno Via Giovanni Paolo II, 132 - 84084 Fisciano SA, Italy

Abstract

A Knowledge Graph (KG) is a form of structured human knowledge depicting relations between entities, destined to reflect cognition and human-level intelligence. Large and openly available knowledge graphs (KGs) like DBpedia, YAGO, WikiData are universal cross-domain knowledge bases and are also accessible within the Linked Open Data (LOD) cloud, according to the FAIR principles that make data findable, accessible, interoperable and reusable. This work aims at proposing a methodological approach to construct domain-oriented knowledge graphs by parsing natural language content to extract simple triple-based sentences that summarize the analyzed text. The triples coded in RDF are in the form of subject, predicate, and object. The goal is to generate a KG that, through the main identified concepts, can be navigable and linked to the existing KGs to be automatically found and usable on the Web LOD cloud.

Keywords

Knowledge Graph, Natural Language Processing, Semantic Web Technologies, Knowledge Graph Embeddings, FAIR

1. Introduction

Due to the complex nature of real-world data, defining a data knowledge model within the specification of a domain is not a trivial task. Semantic Web Technologies (SWT) promise formal paradigms to structure data in a machine-processable way by connecting resources defining semantic relations among them. The result is a rich information representation model whose information is well defined, uniquely identifiable, and accessible as a knowledge base. However, the knowledge representation (KR) task is becoming more challenging every day due to the increasing need to improve, extend, and re-adapt that knowledge to the real-world evolution.

Knowledge Graphs (KGs) provide a way of represent knowledge at different levels of abstraction and granularity; generally it is described [1] as composed of a knowledge base (KB) and a reasoning system with multiple data sources. The KB is a directed network model whose edge represent semantic properties; traditionally those network models can be classified depending on the relations nature into a (1) homogeneous graph [2] $G = (V, E)$, whose edges (E) represents the same type of relationship between the vertices (V), and (2) heterogeneous graph $G = (V, E, R)$ [3, 2], whose edges (E) describe a set of relations (R), i.e., natural connections


Text2KG 2022: International Workshop on Knowledge Graph Generation from Text, Co-located with the ESWC 2022, May 05-30-2022, Crete, Hersonissos, Greece

✉ drinconyanez@unisa.it (D. Rincon-Yanez); ssenatore@unisa.it (S. Senatore)

🆔 0000-0002-8982-1678 (D. Rincon-Yanez); 0000-0002-7127-4290 (S. Senatore)



© 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

between the nodes.

Thanks to the W3C Resource Description Framework (RDF) standard¹, knowledge can be expressed as a network of atomic assertions, also called facts, described as triples composed of *Subject (S)*, *Predicate (P)* and *Object (O)*, $T = (S, P, O)$. In the light of the traditional definition of heterogeneous graph $G = (V, E, R)$, a KB can be described by mapping $V \equiv (S \cup O)$ and $R \equiv P$ as the semantic relation between nodes, leaving the description of vertices as $\forall e \in E : e = (V_x, R_z, V_y)$. On the other hand, the KG Reasoning System (KRS) can be interpreted as the combination of one or more computing or reasoning (r) techniques to read or write the statements in the KB, and it can be expressed as $KG = (KRS, KB)$ or $KG = (KRS, (V, E, R))$ where $KRS = \{r_1, r_2 \dots r_n\}$ and $n \geq 1$ e.g.: $R_i : KGE(KB)e_{R_j} = (e_s, e_p, e_o)$. In a nutshell, a Knowledge Graph (KG) is composed of a Knowledge Base (KB) and a Knowledge Reasoning System (KRS) $KG = (\{r_1, r_2 \dots r_n\}, (V, E, R))$.

The FAIR [4]; Findability, Accessibility, Interoperability and Reuse principles refer to the guidelines for good data management practice: resources must be more accessible, understood, exchanged and reused by machines; these principles in the context of SWT and particularly in the field of Knowledge Graph, encourage communities such as Open Linked Data Community (OpenLOD) and Semantic Web to join efforts into the Fair Knowledge Graph topology [4] building. On the other hand, the Open Knowledge Graphs (OpenKGs) are semantically available on the Web [2, 5], they can be focused on universal domains, e.g., DBpedia [6], WikiData [7], or domain-specific knowledge such as YAGO [8] or WordNet²; most of them are integrated and can be found through the OpenLOD Cloud³, and typically, they can be queried by SPARQL interfaces. Nowadays, Linked Open Data and FAIR principles have played an essential role in the KG spreading. These proposals catapulted the Knowledge Base Construction (KBC) process by providing enormous leverage helping to cross-reference and improve the new generated KBs accuracy, increasing the number of available queryable sources on the Web.

This work proposes a methodological approach implemented as a library to build domain-specific knowledge graphs compliant with FAIR principles from an automated perspective without using knowledge-based semantic models. The method leverages traditional Natural Language Processing (NLP) techniques to generate simple triples as a starting point and then enhance them using external OpenKG, e.g., DBpedia, to annotate, normalize, and consolidate an RDF-based KG. As a case study, novels in literature were explored to validate the proposed approach by exploiting the features from the narrative construction and relationships between characters, places and events in the story.

The remainder of this work is structured as follows: Section 2 provides an overview of relevant approaches, knowledge base construction foundations, and KG construction approaches. Section 3 presents the proposed methodological approach at a high abstraction level. Then, Section 4 describes details of the developed tool and the case study results. Experimental evidence and evaluation is reported in Section 5. Finally, the conclusions and the future work are highlighted in Section 6.

¹W3C RDF Official Website - <https://www.w3.org/RDF/>

²WordNet Official site - <https://wordnet.princeton.edu>

³Open LOD Cloud - <https://lod-cloud.net/>

Table 1

KBC process classification; Knowledge Source defines the characteristics of where the knowledge is located, e.g., Human, source design and Schema Fixed lexicon of entities and relations where the data will be stored, adapted from [13]

Knowledge Source	Method	Schema
Curated	Manual	Yes
Collaborative	Manual	Yes
Structured	Automated	Yes
Semistructured	Automated	Yes
Unstructured	Automated	No

2. Related Work

The Knowledge Base (KB) lifecycle [9] can be described by three main phases: the Knowledge Creation, in charge of discovering initial knowledge; in this phase, initial statements are generated systematically to feed the KB. Then, the Knowledge Curation phase annotates and validates the generated triples. Finally, the Knowledge Deployment phase is accomplished once the KB is mature and can be used along with a reasoning system (KRS) to perform inferring operations.

2.1. Knowledge Base Construction

A knowledge base construction (KBC) can be achieved by performing the first two stages of the KB life cycle, populating a KB [10] and then annotate the knowledge with semantic information [11]. The annotation process can be approached from an automated or a human-supervised perspective. More specifically, the human-centered methods are categorized as (1) curated, meaning a closed group of experts or (2) collaborative, by an open group of volunteers; these two methods must be sustained by one or more knowledge schemes (e.g., ontology-based population). Meanwhile, automatized approaches can uphold the previous schema-based[12], but also a schema-free approach [13], this categorization is detailed in Table 1.

The KB is often a Commonsense Knowledge (CSK) since it refers to generic concepts from daily life; classifying and getting more specific information can enhance the CSK by adding new statements annotated semantically. In [14] a three-step architecture is proposed: (1) Retrieval, (2) Extraction, (3) Consolidation are the steps aimed at fact construction to extract multiple key-value statements pairs from the consolidated assertions. Also, Open KGs are used to match and connect terms from existing ontologies. Since Open KGs, like DBpedia, are composed of large amounts of triples, retrieving data by SPARQL queries or other retrieval systems is time and resource-consuming. Working with DBpedia On-Demand [15] allows accessing a lightweight version of the KB, by using the semi-structured fields of DBpedia articles called info-boxes (structured knowledge from the ontology).

Ontological reasoning, on the other hand, needs synergistic integration: in [16], for example, lexical-syntactic patterns are extracted from sentences in the text, exploiting an SVM (Support Vector Machines) algorithm and then finding matches with DBpedia; the resulting matches are then integrated by a Statistical Type Inference (STI) algorithm to create a homogeneous

RDF-based KB. Furthermore, Open Information Extraction (Open IE) provides atomic units of information, with individual references (URI) to simplify conceivable queries on external databases, such as DBpedia [17].

The KB construction has evolved along with the knowledge representation requirement for data ingestion as well as human consumption of data [18]. This evolution has increased the complexity in terms of data dimensionality, number of sources, and data type diversity [19]. For these additional challenges [20], semantic tools [21] can enhance the collected data by providing cross-context, allowing inference from native data relationships or external data sources, such as status reports or supply chain data.

Once an initial KB has been deployed, the Knowledge Completion can be accomplished, leveraging on the open-world assumption (OWA) also by using masking functions [22] to connect unseen entities to the KG. Specifically, a masking model uses text features with the structure $T = (S, P, ?)$ to learn embeddings of entities and relations via high-dimensional semantic representations based on the known Knowledge Graph Embedding (GKE) models such as TransE neural network [23]. This approach can also be used for link prediction by exploiting existing GKE models such as DistMult, ComplEX, to perform inference using the masked form $T = (S, ?, O)$.

2.2. Knowledge Graph Embeddings (KGE)

Knowledge Graph Embeddings (KGE), also called Knowledge Graph Representation Learning, has become a powerful tool for link prediction, triple classification, over an existing KG. These approaches encode entities and relations of a knowledge graph into a low-dimensional embedding vector space; these networks are based on Deep Learning techniques arriving at a score by minimizing a loss function \mathcal{L} .

KGE provides a comprehensive framework to perform operations [24] over a KB, usually provided in the KB deployment phase. TransE [23], being one of the most studied KGE models, $fTransE = -\|e_s + r_p - e_o\|_n$, computes the similarity between the $e_{subject}$ translated by the embedding of the predicate e_p and the object e_o . TransE model is often exploited for applications such as a knowledge validation model and achieves state-of-the-art prediction performance.

TorusE is a TransE-based embedding method that, combined with the use of Lie Groups knowledge base (KGLG) [25], is used to validate an inferred KB; it has been tested using standard de-facto datasets to evaluate the multidimensional distance between spaces of computed embeddings. Other KGE methods such as DistMult [26] $fDistMult = \{r_p, e_s, e_o\}$ use a trilinear dot product to capture and extract semantic relations, while the method ComplEX [27] $fCompEX = Re(\{r_p, e_s, e_o\})$ uses a Hermitian dot product.

3. The proposed approach

The proposed approach provides an incremental unsupervised methodology, compliant with the FAIR principles, to build a knowledge graph on a specific domain, without exploiting any ontology schema, but achieving the discovery, construction, and integration of triples $T = (S, P, O)$, extracted from an unstructured text, with zero initial knowledge. The approach

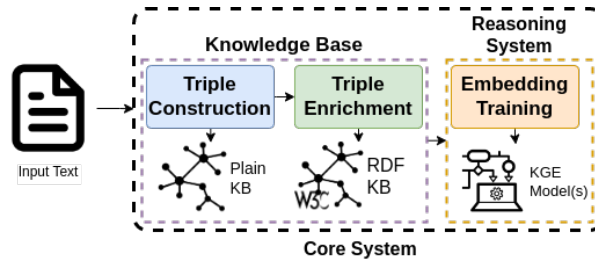


Figure 1: Proposed Methodology from a high level view. Pipeline steps, positioned related with the Knowledge Graph definition.

is described by the pipeline of Figure 1 shows the interaction of the main components of the proposed framework.

From a high-level perspective, the input is represented by the natural language text sources and feeds the *Triple Construction* component, in charge of extracting triples from the analysis of sentences in the text. The text parsing and the triples extraction are tailored to exploit the specific prosaic text features, starting from the narrative properties and the implicit relations created by the recurring entities described in the text. The facts extracted from textual data sources are plain triples $\langle subject, predicate, object \rangle$ that, are incorporated as formal statements in the KB. The triple elements are annotated semantically, by querying external semantic knowledge bases such as DBpedia or Wikidata. This task is accomplished by the *Triple Enrichment* component, generating a RDF-compliant KB. Afterward, this triple set is input to *Embedding Training* component responsible for the KB completion. It accomplishes the reasoning on the KB, exploiting a KGE model, to infer and validate the collected KB.

Next subsections provides additional details about the introduced components of the framework pipeline of Figure 1.

3.1. Triple Construction

NLP-based parsing tasks are applied to extract plain triple-based sentences. The *Triple Construction* component initially carries out the *Basic POS annotation* on the input raw text, consisting of traditional preliminary text analysis, namely, the tokenization, stop-word removal and -tagging routines to identify and annotate the basic part of speech entities.

Once the *Basic POS annotation* is accomplished, parallel activities, namely *Information Extraction* and *Chunking Rules* and *Tagging Pattern Execution* are carried out, as shown in Figure 2.

3.1.1. Information Extraction:

This block is composed of three basic tasks necessary to get an further processing of the tagged text yield by the *Basic POS Annotation*. The text is indeed given to the Named Entity Recognizer (*NER*) which identifies named entities according to some specific entity types: person, organization, location, event. Then a *Dependency Parsing* task allow locating named entities into *subject* or *object* and finally, the *Relation Extraction* task is in charge of the triple composition,

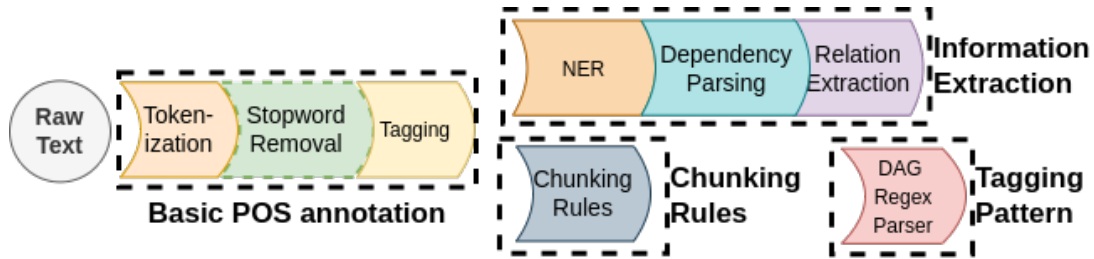


Figure 2: Basic plain Triple Construction: from raw text to the POS annotation, chunking rules to grammar dependency rules. The dashed lines represent a process that is not executed, applies for the Tagging Pattern Execution step.

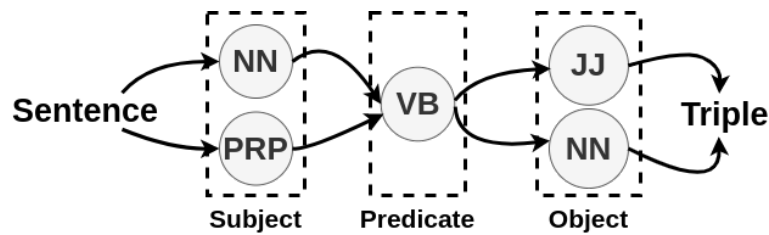


Figure 3: Directed Acyclic Graph (DAG) describing the Tagging Pattern task

starting by the detected entities. Let us remark that the named entities has a crucial role in the triple identification and generation.

3.1.2. Tagging Pattern:

This task accomplish ad-hoc text processing, considering the nature of the text from a linguistic perspective. Linguistic patterns were defined specifically for processing the prosaic narrative and for this reason, for example, some stop words are not discarded. Those patterns can be implemented to discriminate additional triple-based statement to feed the KB; in particular, a finite-state machine is proposed, implementing three evaluation steps, one for each triple component $T = (S, P, O)$. Figure 3 sketches a few of the implemented state transitions used in the development of this methodology.

3.1.3. Chunking Rules:

This task achieve a parallel shallow syntactic analysis where certain sub-sequences of words are identified as forming special phrases. In particular, a regular-expression has been defined over the sequence of word tags to select and extract basic triple for enriching the KB, such as the following tag string $TP : \{ < DT >? < JJ* > < NN >? < VB* > < DT >? < JJ* > < NN > \}$. This regular expression pattern says that a triple pattern TP chunk should be formed by the subject chunker which finds an optional determiner (DT) followed by any number of adjectives (JJ) and then a noun (NN); then the predicate-chunker composed of the verb (in its tenses) an finally the object-chunker that is similar to the subject chunker, except that it could

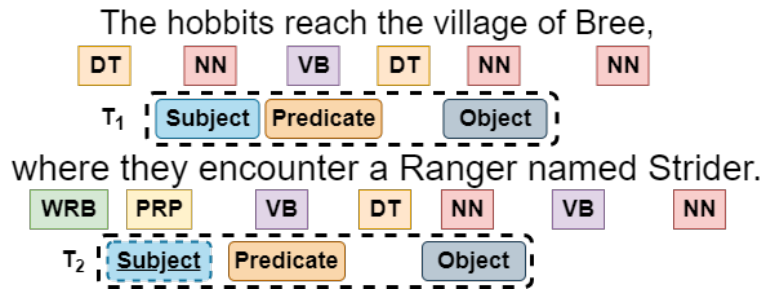


Figure 4: Chunking Rule Evaluation Example.

be a pronoun besides the noun. Subject and object chunkers are typical noun phrases. Some rule example application is shown in Figure 4.

As final result of the *Triple Construction* component, all the generated data are integrated into a set of plain triples, $T_{Plain} \equiv T_{IE} \cup T_{Pattern} \cup T_{Chunking}$, where the T_{IE} represents resulting triples of Information Extraction task, the $T_{Pattern}$, the result of Tagging Pattern and $T_{Chunking}$ is the Chunking Rules result, once redundant and duplicated are discarded. Finally, the resulting triple set T_{Plain} is stored into a plain format file.

3.2. Triple Enrichment

Once the construction of the triple has been completed and stored, the *Triple Enrichment* component (Figure 1) is in charge of annotating semantically the triple elements. First, each element of triples is a candidate to be enriched semantically, by assigning a URI to it; for this purpose, SPARQL queries whose arguments are the individual subject and object of the triples are submitted to OpenKGs such as DBpedia or Wikidata, to retrieve the relative URIs, when they are available. Similarly, the predicate is also looked up in the semantically enhanced lexical database WordNet, once the verb has been transformed into its infinitive form.

Algorithm 1 Triple enrichment Algorithm

Input: Plain Triple Data set

Output: RDF Triple Data set

- 1: **for all** $[S, P, O] : AllTriples$ **do**
 - 2: **for** (S and O) as Entity **do**
 - 3: $[URI, Label] = QueryOpenKG(Entity)$
 - 4: $triples.replace(Entity, URI)$
 - 5: $triples.append((URI, rdfs:label, label))$
 - 6: **end for**
 - 7: $O = QueryWordNet(tokenization(O))$
 - 8: **end for**
 - 9: **return** RDF Triples
-

The general process of the *Triple Enrichment* component is described in Algorithm 1. It

takes the triple collection as input and for each triple *subject* and *object*, search for the corresponding URI in a OpenKG; and then, once retrieved, replaced the URI in the triple, along with the associated *rdfs:label*. Let us notice that not only the external OpenKG URI is extracted from the query, but also the *rdfs:label* for each entity added to the KB. A similar query is carried out on WordNet, for the *predicate*: a simple disambiguation task is accomplished considering the other triple elements, by similarity between terms in the WordNet definition and triple words. Otherwise, the first sense is retrieved for default. Future development are taken into account to select the right sense after a more accurate disambiguation task. The output of the Triple Enrichment component is a semantic version of the collected triples: $T_{RDF-KB} \equiv T_{RDF-IE} \cup T_{RDF-Pattern} \cup T_{RDF-Chunking}$, where T_{RDF-IE} , $T_{RDF-Pattern}$, $T_{RDF-Chunking}$ are the RDF-annotated triples coming from the corresponding tasks of the *Triple Construction* component.

3.3. KG Embeddings Training

Last component of the described pipeline is the *Embedding Training* that is responsible to train the KGE model on the processed triples. The KGE model provides a mechanism to perform operations with the discovered KB, such as fact checking, question answering, link prediction and entity linking.

According to the close-world knowledge assumption (CWA) [22] of the knowledge graph, in general no new statements (triples) can be added to an existing KG, except that predicting relationships between existing, well-connected entities. In the light of this consideration, it is essential to integrate some reasoning mechanism (KRS) that allows the KG to perform operation within the statements inside the KB, to reach the so called closed-world Knowledge Graph Completion. At the same time, once all the possible statements are saturated, additional new statement could be added by considering external information sources, in order to enrich the KB.

4. Implementation detail: A case study

The approach has been implemented and validated, considering a case study relative to the narrative domain, specifically books and article synopses describing short stories, fairy tales, adventure stories. This section describes the methodology at a technical level to identify the main characters, capture the relationships among them, places and events where they are involved by exploiting linguistic features and qualities of the narrative prose.

An enlarged view of the proposed pipeline of Figure 1 is shown in Figure 5 where the whole high-level framework is described. A synergy between different methodologies and technologies has been achieved: NLP activities are integrated with the Machine Learning models, designed according to the principles of Information Retrieval and FAIR, through the Semantic Web technologies (RDF, Turtle, OWL and SPARQL).

The implementation is based on Python language and cover the core system shown in Figure 5. Input textual resources are downloaded from Kaggle narrative books, available in plain text format; at the same time, the synoptic version of the book was retrieved from Wikipedia. Our case study is applied to the book *The Lord of the Rings*. The text was extracted using a simple

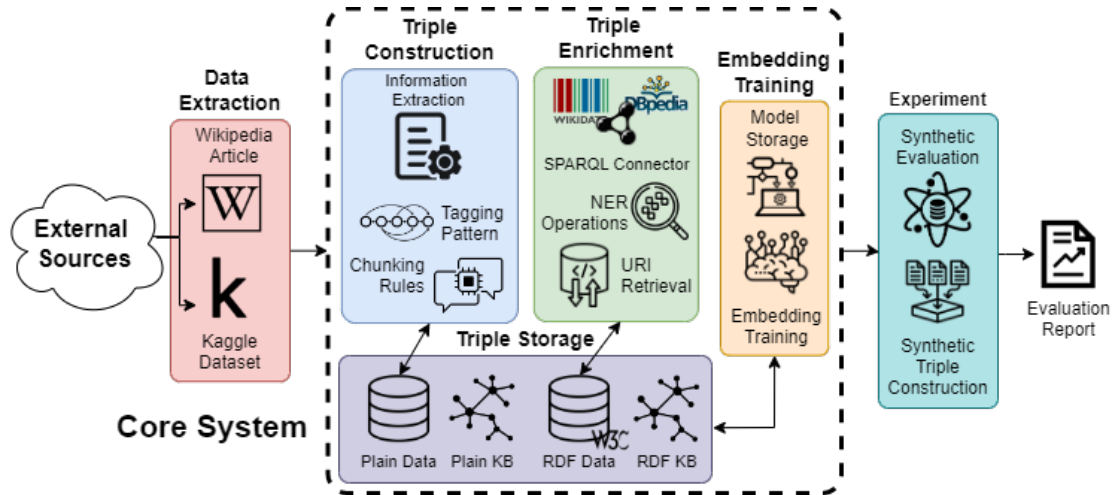


Figure 5: Methodology implementation detailed overview.

web crawler that retrieve the content of the URL page; the data is cleaned using a standard HTML/XML parser (*Data Extraction*).

As shown in Figure 5, the *Triple Construction* component achieves the traditional POS-tagging activities (Figure 2). The information extraction is performed by the Stanford Open-IE server tool combined with Spacy library (*en_core_web* pre-trained models in English) for the extraction of triples.

The *Tagging Pattern* and *Chunking Rules* tasks from *Triple Construction* are ad-hoc designed; these tasks also were implemented by exploiting WordNet. To give an example of pattern extraction described in Figure 3 the text "But Frodo is still wounded in body and spirit" is analyzed according to the regular expression $T = (NN*, VB*, NN*)$, where the wildcard character is a placeholder for possible variations of that POS-tags. The generated triple is $T = (Frodo, wounded, spirit)$.

The triple extraction is also aimed at generating the widest set of triples from the sentence analysis. Along with the definition of regular expression patterns for discovering simple phrase chunks, the co-reference is also achieved: primary sentence with the same subject in secondary sentences or simply sentences with pronoun are re-arranged to get triples with noun instead of pronoun. The rule (PRP, VB, NN) in Figure 4 allows us to obtain the pronoun *PRP* that will be replaced by the right a noun (in the previous sentence $NN S = hobbits$), to get the triple $T_2 = (hobbits, encounter, Ranger)$.

Once the *Triple Construction* process is concluded, the generated triples are rearranged, discarding duplicates (coming from the parallel tasks). The remaining triples are ordered according to the same subject and then predicate and then stored in a plain CSV file for further use.

The *Triple Enrichment* component uses the CSV-generated file to carry out Algorithm 1. An initial Named Entity Recognition task allows discriminating important entities among subject and object, in the triples in order to discard irrelevant triples (i.e., triples with no named entity)

Algorithm 2 SPARQL Query example to URI and Label extraction

```
SELECT *
WHERE {{
  {{ dbr:{word} rdf:type dbo:FictionalCharacter .
     dbr:{word} dbp:name ?label .
  }} UNION {{
    {{ dbr:{word} dbo:wikiPageRedirects ?URI }}
    UNION
    {{ dbr:{word} dbo:wikiPageDisambiguates ?URI }} .
    ?URI rdf:type dbo:FictionalCharacter .
    ?URI rdfs:label ?label
  }}
  FILTER (lang(?label) = 'en')
}}
```

and more important, to use that entities as parameter for the query, according to Algorithm 1. The individual entities are looked at DBpedia by submitting SPARQL queries, to retrieve the semantic annotation (URI and label) corresponding to that entity. The goal is to make the subject or object on each triple accessible resources, according to the Linked Data principles.

The query shown in Algorithm 2 was submitted to the public SPARQL end-point⁴ by implementing the SPARQLWrapper to retrieve the DBpedia URI of the selected entities and the *rdfs:label*. Let us notice that when a URI of an entity is not available, i.e., no result is returned by the query, a customized URI is generated. In a similar way, the *predicate* is processed using WordNet after reducing it to the infinitive form, e.g., the several tenses *kills*, *killed*, *killing* become the infinite form *kill*. After these activities, a semantic knowledge base (called RDF KB) should be available.

The resulting RDF KB is passed to the KGE models (**Embeddings Training** in Figure 5 to get the embeddings on the RDF triple-based KB. Embedding models such as ComplEX, TransE, and DistMult were trained to compare and evaluate the results across this implementation. The training was achieved and evaluated by using the evaluation proposed by the KG Embeddings of the framework Ampligraph [28].

4.0.1. Triple Storage:

An important role was assumed by the storage component, that allows to store all the intermediate version of the triples generated. In a nutshell, triples generated in the *Triple Construction* component (Section 3.1) are stored in CSV format (to reuse them later); then, after the the *Triple Enrichment* (Section 3.2), the resulting triples are stored semantically, namely in RDF/Turtle languages. RDF-based triples can be imported into the NEO4J⁵ by the neosemantics plugin and then the resulting knowledge graph can be graphically displayed into the GUI This last

⁴DBpedia SPARQL end-point - <https://dbpedia.org/sparql>

⁵<https://neo4j.com/v2/>

storage task is crucial for reusing the KB for later validation, charts generation, query execution, and further operations.

4.0.2. Source Code:

A demo for the implementation of this project is available on GitHub. The usage instructions are consigned on the repository⁶ and the produced results (KBs) should be published under the CC-BY-SA license.

4.1. Knowledge Graph Completion Evaluation

The KB validation is performed by implementing a cross-graph representation learning approach [29] by embedding triples based on the semantic meaning of the KB; the representation is done by estimating a confident score in each statement based on a degree of correctness. This estimation is calculated by exploiting the KGE models and the corresponding evaluation metrics over the discovered KB.

First, the validation is accomplished by considering the output of each tasks of the *Triple Construction* component and setting a pair-wise comparison among T_{RES-IE} , $T_{RES-Pattern}$, $T_{RES-Chunking}$ through the trained KGE models, with the purpose of measure the correspondence between the discovered knowledge. Then, to assess the generated knowledge base, new triples are added to the KGE models once trained the trustworthiness (i.e., how they are true) of these triples. These new triples are called "synthetic", because they are manually generated, are used to perform link predictions queries of the form $T = (?, P, ?)$. The synthetic triples are divided into two groups, positive (ST_{Pos} , existing) and negatives (ST_{Neg} , not existing, or unlikely to be true). Known learning-to-rank metrics to assess the performance of KGE models are as follows.

$$\mathbf{MRR} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{q} \quad (1) \quad \mathbf{HITS@N} = \frac{|q \in Q : q < n|}{|Q|} \quad (2)$$

The *Mean Reciprocal Rank* (MRR) is a statistical measure for assessing the quality of a list of possible answers to a sample of queries, sorted by probability of correctness. MRR assume value equal to 1 if a relevant answer was retrieved at rank 1; the intention is to measure the correctness of each of the proposed scenarios using *MRR* to validate source knowledge against the validation dataset. The *HITS@N* measures the percent of triples that were correctly ranked in the top N . Besides, the combination with *HITS*, with two different coefficients $N = 5$ and $N = 3$ to measure the ranking of the entities part of the statement (S, O) in the validation dataset at the time of link prediction.

5. Experiment Design and Results

5.1. Triple Discovery

For the validation of the KBC process, let us focus on the KB generated by the *Triple Construction* component described in Section 3.1. After the basic POS tagging tasks, the three sub-tasks,

⁶Github Repository - <https://github.com/d1egopro/Text2KG>

Information Extraction, *Chunking Rules* and *Tagging Pattern Execution* generates statements for the graph construction, the statements are grouped in T_{IE} , $T_{Pattern}$, $T_{Chunking}$, respectively.

The triples generated by using the Information Extraction (T_{IE}) are approximately 75% of the discovered triples. The remaining 25% can be split between the triples (about 10% of the total KB generated) coming from the Chunking Rules ($T_{Chunking}$) and those (around 15%) from Tagging Pattern components ($T_{Pattern}$). Overlapping statements represent 1% of the triples discovered.

5.2. KG Completion evaluation approach

Table 2 shows the results of the evaluation process, by pair-wise comparison of the statements generated by the components and tasks, as described in Section 4.1. The first two columns of the table show, in fact, the set of triples (statements) generated by two data sources, compared with each other. In other words, the first column represent the reference triple set that is used to validate the triple set from the second column. Then, there are the presented metrics and the relative values obtained for each KGE model used, for the three KGE models analyzed.

Let us notice that in general there is an important dependence between the ranked triple list compared with the expected ordering of the reference triple set for most of pairs reported in the table. The worst MRR values are returned for the ranked triples $T_{RES-Pattern}$ returned by all the three models, with respect to $T_{RES-Chunk}$ and vice-versa. This is due to the small amount of triples generated by the corresponding tasks, compared to the equally small reference dataset. The highest scores of the MRR are instead returned for the number of triples generated by using the Information Extraction (T_{IE}) and even for the *Chunking Rules* task, namely $T_{RES-Chunk}$. The percentage of the top three and five ranked triples are also shown in the table, confirming that using triples from the Information Extraction task (T_{IE}) for both reference and validation often produces good triple rankings. To consider the validation of the external knowledge, the last rows of the table show the result of the comparison with the synthetic triples ST_{Pos} and ST_{Neg} .

The results suggest that the validation of the synthetic triples produces high values (about 0.95) of MRR for the positive ST_{Pos} triples and 0 for the negative ST_{Pos} triples. The synthetic statements appear to accurately reflect the knowledge discovered by the methodology and the correspondence implementation, and the positive statements are potential candidates to feed into the existing knowledge base.

6. Conclusions and Future Work

This paper proposes unsupervised knowledge construction from domain-specific knowledge given a natural language text. In particular, a case study is designed on a narrative text, exploiting features unique to prosaic text. The text was parsed and basic triples, representing the core part (subject, predicate and object) of the main sentences are extracted. By Semantic Web technologies, the triple are semantically annotated to be linked and navigable into the Linked Data cloud. At that point, Knowledge graph embedding models have been exploited to extend the potential of the collected triples by verifying and validating the so generated knowledge. Performance is assessed by cross-validation, scoring functions and accuracy on unseen triples.

Table 2

KG Completion approach results, exploiting the three KGE models TransE, DistMult, ComplEx. Src is the reference data; Val. are the data to validate with Src. All the KB are the final T_{RES} for each process

		TransE			DistMult			ComplEx		
Src.	Val.	MRR	H@3	H@5	MRR	H@3	H@5	MRR	H@3	H@5
T_{IE}	T_{CH}	0.69	0.53	0.85	0.63	0.58	0.71	0.65	0.59	0.71
T_{IE}	T_{PT}	0.55	0.40	0.74	0.48	0.41	0.56	0.55	0.47	0.65
T_{CH}	T_{IE}	0.53	0.42	0.67	0.52	0.48	0.57	0.57	0.54	0.59
T_{CH}	T_{PT}	0.33	0.28	0.42	0.30	0.28	0.31	0.33	0.31	0.36
T_{PT}	T_{IE}	0.54	0.40	0.70	0.53	0.47	0.58	0.55	0.49	0.61
T_{PT}	T_{CH}	0.34	0.28	0.42	0.33	0.31	0.33	0.29	0.28	0.31

		TransE			DistMult			ComplEx		
Src.	Val.	MRR	H@1	H@3	MRR	H@1	H@3	MRR	H@1	H@3
T_{KB}	ST_{Pos}	0.95	0.92	0.99	0.92	0.85	0.99	0.99	0.98	0.99
T_{KB}	ST_{Neg}	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01

The work provides a general methodological approach to automatically build a knowledge base, on a preferred domain, given in textual resources. It allows the creation of different types of KBs. It can be leveraged to build ad-hoc knowledge bases, tailored to a specific domain. Question-answering systems can act as validation systems when verification of the veracity and reliability (degree of truth) of a given statement is required, for example, for the discovery of fake news or just for fact-checking.

As future work, the generation and enrichment of triples by querying existing Open Knowledge bases will be improved, along with the more challenging goal of enhancing the degree of confidence in the KB by implementing an automatic validation method.

References

- [1] L. Ehrlinger, W. Wöß, Towards a definition of knowledge graphs, in: Proceedings of 12th International Conference on Semantic Systems (SEMANTiCS2016), volume 1695, CEUR-WS, 2016, pp. 1–4.
- [2] A. Hogan, E. Blomqvist, M. Cochez, C. D’Amato, G. de Melo, C. Gutierrez, S. Kirrane, J. E. L. Gayo, R. Navigli, S. Neumaier, A.-C. N. Ngomo, A. Polleres, S. M. Rashid, A. Rula, L. Schmelzeisen, J. Sequeda, S. Staab, A. Zimmermann, Knowledge Graphs, number 2 in Synthesis Lectures on Data, Semantics, and Knowledge, Morgan & Claypool, 2021. doi:10.2200/S01125ED1V01Y202109DSK022.
- [3] B. Lee, S. Zhang, A. Poleksic, L. Xie, Heterogeneous Multi-Layered Network Model for Omics Data Integration and Analysis, *Frontiers in Genetics* 0 (2020) 1381. doi:10.3389/FGENE.2019.01381.
- [4] M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J. W. Boiten, L. B. da Silva Santos, P. E. Bourne, J. Bouwman, A. J. Brookes, T. Clark, M. Crosas, I. Dillo, O. Dumon, S. Edmunds, C. T. Evelo, R. Finkers, A. Gonzalez-

- Beltran, A. J. Gray, P. Groth, C. Goble, J. S. Grethe, J. Heringa, P. A. t Hoen, R. Hooft, T. Kuhn, R. Kok, J. Kok, S. J. Lusher, M. E. Martone, A. Mons, A. L. Packer, B. Persson, P. Rocca-Serra, M. Roos, R. van Schaik, S. A. Sansone, E. Schultes, T. Sengstag, T. Slater, G. Strawn, M. A. Swertz, M. Thompson, J. Van Der Lei, E. Van Mulligen, J. Velterop, A. Waagmeester, P. Wittenburg, K. Wolstencroft, J. Zhao, B. Mons, Comment: The FAIR Guiding Principles for scientific data management and stewardship, *Scientific Data* 3 (2016) 1–9. doi:10.1038/sdata.2016.18.
- [5] P. N. Mendes, M. Jakob, C. Bizer, DBpedia: A multilingual cross-domain knowledge base, in: *Proceedings of the 8th International Conference on Language Resources and Evaluation, LREC 2012*, European Language Resources Association (ELRA), Istanbul, Turkey, 2012, pp. 1813–1817.
- [6] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, Z. Ives, DBpedia: A Nucleus for a Web of Open Data, in: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 4825 LNCS, Springer, Berlin, Heidelberg, 2007, pp. 722–735. doi:10.1007/978-3-540-76298-0_52.
- [7] H. Turki, T. Shafee, M. A. Hadj Taieb, M. Ben Aouicha, D. Vrandečić, D. Das, H. Hamdi, Wikidata: A large-scale collaborative ontological medical database, *Journal of Biomedical Informatics* 99 (2019) 103292. doi:10.1016/j.jbi.2019.103292.
- [8] F. M. Suchanek, G. Kasneci, G. Weikum, YAGO: A Core of Semantic Knowledge, in: *Proceedings of the 16th international conference on World Wide Web - WWW '07*, ACM Press, New York, New York, USA, 2007, pp. 697–706. URL: <https://dl.acm.org/doi/10.1145/1242572.1242667>. doi:10.1145/1242572.
- [9] U. S. Simsek, K. Angele, E. Kärle, J. Opdenplatz, D. Sommer, J. Umbrich, D. Fensel, Knowledge Graph Lifecycle: Building and Maintaining Knowledge Graphs, in: *Proceedings of the 2nd International Workshop on Knowledge Graph Construction, CEUR-WS, 2021*, p. 16.
- [10] C. Ré, A. A. Sadeghian, Z. Shan, J. Shin, F. Wang, S. Wu, C. Zhang, Feature Engineering for Knowledge Base Construction, . (2014). arXiv:1407.6439.
- [11] V. Leone, G. Siragusa, L. Di Caro, R. Navigli, Building semantic grams of human knowledge, in: *LREC 2020 - 12th International Conference on Language Resources and Evaluation, Conference Proceedings, 2020*, pp. 2991–3000.
- [12] D. R. Yanez, F. Crispoldi, D. Onorati, P. Ulpiani, G. Fenza, S. Senatore, Enabling a Semantic Sensor Knowledge Approach for Quality Control Support in Cleanrooms, in: *CEUR Workshop Proceedings*, volume 2980, CEUR-WS, 2021, pp. 1–3. URL: <http://ceur-ws.org/Vol-2980/paper409.pdf>.
- [13] M. Nickel, K. Murphy, V. Tresp, E. Gabrilovich, A Review of Relational Machine Learning for Knowledge Graphs, *Proceedings of the IEEE* 104 (2016) 11–33. doi:10.1109/JPROC.2015.2483592. arXiv:1503.00759.
- [14] T.-P. Nguyen, S. Razniewski, G. Weikum, Advanced Semantics for Commonsense Knowledge Extraction, in: *Proceedings of the Web Conference 2021, ACM, New York, NY, USA, 2021*, pp. 2636–2647. doi:10.1145/3442381.3449827.
- [15] M. Brockmeier, Y. Liu, S. Pateer, S. Hertling, H. Paulheim, On-Demand and Lightweight Knowledge Graph Generation – a Demonstration with DBpedia, in: *Proceedings of the Semantics 2021*, volume 2941, CEUR-WS, 2021, p. 5. arXiv:2107.00873.

- [16] T. Kliegr, O. Zamazal, LHD 2.0: A text mining approach to typing entities in knowledge graphs, *Journal of Web Semantics* 39 (2016) 47–61. doi:10.1016/j.websem.2016.05.001.
- [17] J. L. Martinez-Rodriguez, I. Lopez-Arevalo, A. B. Rios-Alvarado, OpenIE-based approach for Knowledge Graph construction from text, *Expert Systems with Applications* 113 (2018) 339–355. doi:10.1016/j.eswa.2018.07.017.
- [18] Y. Liu, T. Zhang, Z. Liang, H. Ji, D. L. McGuinness, Seq2RDF: An end-to-end application for deriving Triples from Natural Language Text, in: *Proceedings of the 17th International Semantic Web Conference, CEUR-WS, 2018*, p. 4. URL: <http://ceur-ws.org/Vol-2180/paper-37.pdf>.
- [19] A. Ratner, C. Ré, P. Bailis, Research for practice, *Communications of the ACM* 61 (2018) 95–97. URL: <https://dl.acm.org/doi/10.1145/3233243>. doi:10.1145/3233243.
- [20] S. Tiwari, F. N. Al-Aswadi, D. Gaurav, Recent trends in knowledge graphs: theory and practice, *Soft Computing* 25 (2021) 8337–8355. URL: <https://link.springer.com/10.1007/s00500-021-05756-8>. doi:10.1007/s00500-021-05756-8.
- [21] G. Fenza, M. Gallo, V. Loia, D. Marino, F. Orciuoli, A cognitive approach based on the actionable knowledge graph for supporting maintenance operations, in: *2020 IEEE Conference on Evolving and Adaptive Intelligent Systems, EAIS 2020, Bari, Italy, May 27-29, 2020*, IEEE, 2020, pp. 1–7. doi:{10.1109/EAIS48028.2020.9122759}.
- [22] B. Shi, T. Weninger, Open-World Knowledge Graph Completion, *Proceedings of the AAAI Conference on Artificial Intelligence* 32 (2018).
- [23] A. Bordes, N. Usunier, A. Garcia-Durán, J. Weston, O. Yakhnenko, Translating embeddings for modeling multi-relational data, in: *Advances in Neural Information Processing Systems, NIPS'13, Curran Associates Inc., Red Hook, NY, USA, 2013*, pp. 2787–2795.
- [24] K. Scharei, F. Heidecker, M. Bieshaar, Knowledge Representations in Technical Systems – A Taxonomy, . (2020). arXiv:2001.04835.
- [25] T. Ebisu, R. Ichise, Generalized translation-based embedding of knowledge graph, *IEEE Transactions on Knowledge and Data Engineering* 32 (2020) 941–951. doi:10.1109/TKDE.2019.2893920.
- [26] B. Yang, W.-t. Yih, X. He, J. Gao, L. Deng, Embedding Entities and Relations for Learning and Inference in Knowledge Bases, in: Y. Bengio, Y. LeCun (Eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015*, p. 15.
- [27] T. Trouillon, J. Welbl, S. Riedel, E. Gaussier, G. Bouchard, Complex Embeddings for Simple Link Prediction, in: M. F. Balcan, K. Q. Weinberger (Eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, PMLR, New York, New York, USA, 2016, pp. 2071–2080.
- [28] L. Costabello, S. Pai, C. L. Van, R. McGrath, N. McCarthy, AmpliGraph: a Library for Representation Learning on Knowledge Graphs, GitHub (2019). URL: <https://zenodo.org/record/2595049>. doi:10.5281/ZENODO.2595049.
- [29] Y. Wang, F. Ma, J. Gao, Efficient Knowledge Graph Validation via Cross-Graph Representation Learning, in: *Proceedings of the 29th ACM International Conference on Information & Knowledge Management, CIKM '20, ACM, New York, NY, USA, 2020*, pp. 1595–1604. doi:10.1145/3340531.3411902. arXiv:2008.06995.