2021-12-01

KnowText: Auto-generated Knowledge Graphs for custom domain applications

Bojan Bozic
Technological University Dublin, bojan.bozic@tudublin.ie

Jayadeep Kumar Sasikumar
Technological University Dublin

Tamara Matthews
Technological University Dublin

KnowText: Auto-generated Knowledge Graphs for custom domain applications

BOJAN BOŽIĆ, JAYADEEP KUMAR SASIKUMAR, and TAMARA MATTHEWS, Technological University Dublin, Ireland

While industrial Knowledge Graphs enable information extraction from massive data volumes creating the backbone of the Semantic Web, the specialised, custom designed knowledge graphs focused on enterprise specific information are an emerging trend. We present "KnowText", an application that performs automatic generation of custom Knowledge Graphs from unstructured text and enables fast information extraction based on graph visualisation and free text query methods designed for non-specialist users. An OWL ontology automatically extracted from text is linked to the knowledge graph and used as a knowledge base. A basic ontological schema is provided including 16 Classes and Data type Properties. The extracted facts and the OWL ontology can be downloaded and further refined. KnowText is designed for applications in business (CRM, HR, banking). Custom KG can serve for locally managing existing data, often stored as "sensitive" information or proprietary accounts, which are not on open web access. KnowText deploys a custom KG from a collection of text documents and enable fast information extraction based on its graph based visualisation and text based query methods.

CCS Concepts: • **Information systems → Web Ontology Language (OWL)**; Query languages for non-relational engines; • **Human-centered computing → Graph drawings**.

ACM Reference Format:
Bojan Božić, Jayadeep Kumar Sasikumar, and Tamara Matthews. 2021. KnowText: Auto-generated Knowledge Graphs for custom domain applications. In *The 23rd International Conference on Information Integration and Web Intelligence (iiWAS2021), November 29-December 1, 2021, Linz, Austria.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3487664.3487803

1 INTRODUCTION

Knowledge Graphs (KG) enable fast information extraction from unstructured text using the graph "metaphor" as knowledge representation [19] – matching that of the human brain at both visual and logic level. Automatic generation of Knowledge Graphs from unstructured text is a long standing goal in artificial intelligence research as KG can gather facts from text, organize these according to a background ontology and then enable queries, much like the human brain organizes facts into a logical structure. Knowledge Graphs [4], intended to leverage computer understanding of semantics for the Semantic Web [1] can also enhance human perception of information from unsupervised text by rendering the network of facts. By using the "graph" visualisation metaphor, KG present facts as syntactic "triples" of ⟨Subject, Predicate, Object⟩ extracted from text and visualised as an acyclic graph, where the nodes are Subjects or Objects and the edges are Predicates. An increase in "readability" of such information is achieved by linking the extracted triples to Named Entities (NE) which are already classified in an automatically extracted ontology.

Knowledge Graphs [25] assign a graph representation to information organized into ontological structures. Enabling easy access to a visual representation of the KG and to query its inner ontological structure creates a powerful tool for knowledge discovery (by exploring entities, relationships between entities or new facts). This provides a competitive edge to enterprises using large and complex data volumes for a variety of applications: recommendation engines, fraud

Manuscript submitted to ACM

detection, regulatory compliance or investigative journalism. The advantages are that users have quick access to the network of facts across large collections of unstructured text and these are easy to "grasp" from the visual representation of the graph (when applying relevant filters).

The fast increase in unstructured text data volume leads to information overload on market participants. Such data can be web-generated (as web articles, blogs, surveys, chats, customer support or correspondence) and represents an invaluable pool of information - providing business leads, in-depth knowledge on customers, on markets, companies, or technologies. From these unstructured texts users need to find the relevant *Facts*, the *Entities* and the *Links* between *Entities* and their evolution.

The purpose of the KnowText is to respond the above demand performing automated extraction a Knowledge Graph from text (a collection of text documents) and enable users to query the graph and its background ontology. KnowText enables visualization of Named Entities and their relations, provides fast access to cumulative data references for each entity, inference based on local relations, in-depth understanding of facts extracted. The KnowText application described here serves as a proof-of concept, proving the capabilities of the proposed approach to provide fast and easy to use solutions for text data management in business applications (ie customer support, HR, banking, journalism).

2 RELATED WORK

Knowledge Graphs (KG) organize raw information by capturing relationships (edges) between entities (nodes) being structured across two layers: a Knowledge Base (KB) – which stores the domain's data as Entities and Relationships using a domain ontology schema) – and a Reasoning Engine, which supports: data integration from multiple sources; data query from the graph and data inference [4].

Knowledge Bases are usually built as relational schema using expert knowledge from existing ontologies [5], or by ontology extraction from semi-structured ([8, 23]) or unstructured text [7, 13]. Automatic ontology learning from unstructured text involves extracting concepts, entity linking and ontology population, using methods based on NLP, information retrieval, machine learning and data mining [11, 12] or dedicated platforms (OntoLearn [24], Text2Onto [2], OntoText [18]).

Some of the most known "industrial" KG as Google Knowledge Graph, YAGO [8] and ConceptNet [22] have mapped billions of entities and relations from web-based ontologies like Wikipedia, WordNet, Wikidata, Freebase or Facebook [20]. The KB data models for such industrial KGs are structured and strongly typed. Existing open-access knowledge graphs have been built by either: (i) a group of experts (WordNet), (ii) collaboratively by volunteers (Freebase, Wikidata), (iii) auto-populated using semi-structured resources as Wikipedia info-boxes and regular expressions (YAGO and DBpedia), or (iv) by facts extraction from unstructured resources using Natural Language Processing and Machine Learning (i.e. NELL [17] and KnowledgeVault [19]).

Such large-scale KG do not allow for easy customization. Also, to query domain specific information from large silos of data can be complex, time consuming and requires expertise.

To enable customization, various frameworks for KG generation have been proposed (SemTK [3], WhyIs [16]) usually based on triple stores and data mapping to external resources. KG frameworks enable customization providing pre-designed KB and reasoners as graph data bases or triple stores with external data mapping which require the use of specific query languages. To replace ontologies, reasoner systems are designed as graph data bases (Grakn.AI and Neo4j [26]). The majority of such platforms use SPARQL or platform specific query languages. Nevertheless, such KG frameworks also require user expertise for set-up in a specific domain and for query and their use can be time and computationally expensive – as very large KGs become difficult to query.

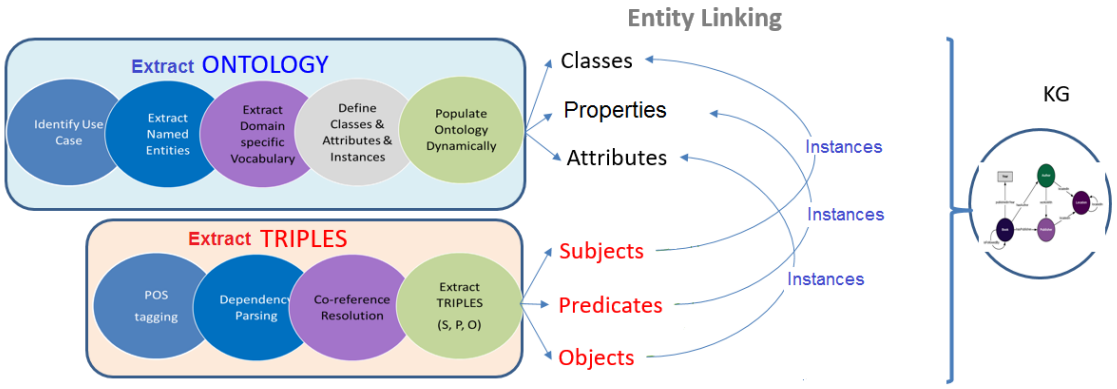

Fig. 1. Steps and Methods used for dynamic extraction of triples and ontology in automated KG generation.

There has been a long term interest in automated KG extraction from unstructured text [14] and a need to provide a pre-defined interactivity in KG, enable easy customization for use-case and domain and easy query methods based on Natural Language or Visualizations, which are approached in KnowText.

3 THE KNOWTEXT SYSTEM

We present KnowText, an easy-to-use web application dedicated to business applications (as banking, HR, CRM, customer support) that performs automated Knowledge Graphs extraction from a collection of unstructured texts, and enables users to visualize and query the KG and its background ontology.

The aim of KnowText is to enable non-expert users to generate a specialised KG based on a specific collection of documents and to enable queries by free text and KG visualization filtering – while accessing facts from data stored in a local ontology.

The Knowledge graph consists of a knowledge base (an OWL ontology) that correlates to and includes in its schema the *Facts* extracted from the text collection as ⟨Subject, Predicate, Object⟩.

The triples are represented as a graph and linked to an automatically extracted OWL Ontology featuring 16 Classes (15 Named Entities and a domain vocabulary) within a basic schema. As the ontology and KG are designed for use in business applications domains as customer support, HR, banking, journalism, the given (pre-defined) Classes based on NER extraction cover the required Classes of interest.

The two processes of triples extraction and ontology generation are performed in parallel using a succession of NLP techniques. Both the extracted triples and the OWL ontology files can be downloaded and further explored.

3.1 Methods

While dynamic extraction of triples and ontology from unstructured text are still domains of active research, a number of powerful platforms and methods were available and used for the development of KnowText. The main methods used here for dynamic extraction of triples from unstructured text and the generation of the ontology are shown in Figure 1. The extraction of triples from unstructured text is performed in several steps:

Table 1. The set of rules for filtering Triples

Filter	Description
max_word_count	S, P and O should be made up of max 3 words.
min_char_count	S, P and O should be made up of min 2 characters.
days_of_week	Triples should not contain days of the week.
nn_subject	S should be a noun.
pronouns	Triples should not contain pronouns.
duplicate_subject_words	S should not contain duplicate words.
duplicate_object_words	O should not contain duplicate words.
suppositions	Triples should not contain suppositions.
special_chars	Triples should not contain special characters.
subject_verbs	S should not contain verb phrases.
object_verbs	O should not contain verb phrases.
subset_phrases_subject	When 2 triples exist with similar S, longer S is chosen.
subset_phrases_predicate	When 2 triples exist with similar P, longer S is chosen.
subset_phrases_object	When 2 triples exist with similar O, longer S is chosen.
same_subject_object	S and O should not be the same.

- Using NLP platforms (**Spacy** [10] or **NLTK**) to perform POS tagging;
- Perform co-reference resolution using Spacy's **neuralcoref** library [9] which identifies the named entities (persons, locations, organizations) when these appear referenced indirectly in the text using pronouns;
- Perform dependency parsing to annotate words in the text with their syntactic role using python platforms (here we used Stanford coreNLP's, **StanfordOpenIE** library [21] although SpaCy's library for dependency parsing has also been considered.
- Extract (S, P, O) triples as Subject (the subject in a phrase), Predicate (the predicate defining the action in a phrase) and Object (the entity to which the action is applied). There are many platforms with dedicated libraries performing automatic triple extraction from text among which: Spacy, Stanford coreNLP, StanfordOpenIE, the latter being preferred here for the rich triples' constructs and the large number of outputs. Various conditions based on heuristic rules for the POS type of each (S, P, O) in triples had been made in order to remove the least relevant triples and improve the validity of extracted facts. Several examples of the heuristic rules used for filtering triples are shown below (a complete list is shown in Table 1):
  - the word(s) in Subject can only be of type Noun (NN, NNP, NNs, NNPs, etc);
  - the word(s) in Predicate can only be of type Verb;
  - the maximum number of words in any of the (S, P, O) in triples is three;

These filters are effective, showing a significant decrease in the number of triples and increasing their accuracy (example in Figure 2, using the "US-Economic-News" dataset[1], Kaggle).

For the dynamic generation of the ontology, the **owlready2** library has been used [15]. This library enables creating dynamically Classes and DataProperties as well as as well as populating these with entities and properties and finally assigning property values to generate an OWL ontology. As shown in Figure 1, the steps in generating the ontology involve:

- Identification of use case, then careful and selective choice of a collection of texts which is representative for the use case. This collection of texts will create the base of the ontology;

[1]https://www.kaggle.com/heeraldedhia/us-economic-news-articles/version/1

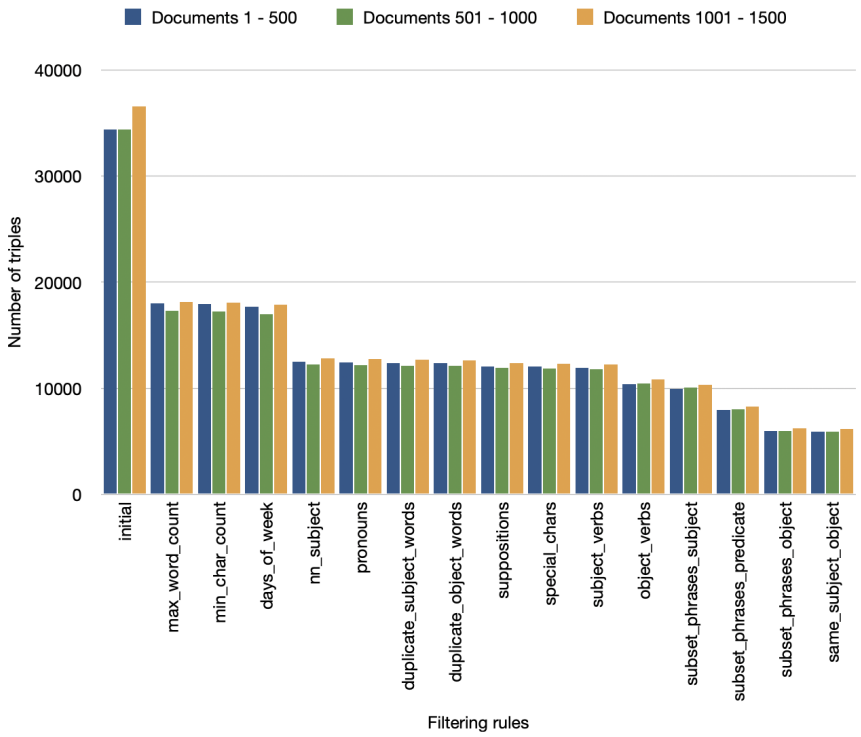

Fig. 2. Evolution of number of Triples after applying each filtering rule. Examples use three samples containing 500 documents from the "US-Economic-News" dataset, Kaggle.

- Extraction of Named Entities (i.e.: Persons, Politics, Organizations, Geographic Locations, etc.) using the NER package from SpaCy (the complete list of types of NE extracted are presented in Table 2 along with their original definitions);
- An ad-hoc, domain vocabulary is dynamically extracted from the collection of texts as words that occur as Subject in one of the triples but do not occur in the NE categories. Various conditions on the type and structure of such words can be added in order to select the most meaningful domain-vocabulary words. Although such words are dynamically extracted and expert domain-knowledge is not applied for this task, the majority of words extracted into the domain-vocabulary category appear relevant for the domain when the initial collection of texts is relevant and selectively chosen to belong to a specific domain.
- The extracted NE and domain vocabulary categories are defined as classes in the ontology defining the ontology schema;
- The ontology schema is dynamically populated with instances from extracted triples.

Nevertheless, when the Subject is a proper Noun (as somebody's name) the various constructs involving "Mr.", "Mrs.", or person's function ("Senator", "President") are not always captured in the same construct as the NE extracted by spaCy. This can lead to the same person being seen under various representations (i.e.: Mr. Obama, "President Obama", "Barack

Table 2. Description of Named Entity types as defined by SpaCy

Named Entity Type	Description
PERSON	People, including fictional.
NORP	Nationalities or religious or political groups.
FAC	Buildings, airports, highways, bridges, etc.
ORG	Companies, agencies, institutions, etc.
GPE	Countries, cities, states.
LOC	Non-GPE locations, mountain ranges, bodies of water.
PRODUCT	Objects, vehicles, foods, etc. (Not services.)
EVENT	Named hurricanes, battles, wars, sports events, etc.
WORK_OF_ART	Titles of books, songs, etc.
LAW	Named documents made into laws.
LANGUAGE	Any named language.
DATE	Absolute or relative dates or periods.
QUANTITY	Measurements, as of weight or distance.
ORDINAL	“first”, “second”, etc.
CARDINAL	Numerals that do not fall under another type.

Obama"). These various constructs can later be collated by the user under the same name in the visualisation of the graph and the changes will update the instances in the ontology.

3.2 Workflow and Functionality

A schematic of the workflow in the KnowText demonstrator is shown in Figure **??**. Extracted triples are linked to elements of the ontology (classes, relations, attributes) and used to populate the ontology. The workflow includes the following steps:

- The file upload is followed by the extraction of triples and dynamic generation of the ontology processes which take place in the background.
- The Knowledge Graph is visualised as an interactive directed graph which allows to observe entities (Subjects and Objects) as nodes and relationships (Predicates) as edges.
- Drop-down lists of entities and relationships in alphabetical order enable users to select a specific entity or relationship to visualize its close neighbours in the graph. This enables visual exploration of the entities and relationships inside triples as well as observing various details otherwise "hidden" inside text.
- Entities can also be queried by clicking on the specific node - which will open a dialog box with data stored inside the ontology.
- The generated ontology is a RDF/XML (OWL compatible) file and can also be downloaded for further development outside the demonstrator.

3.3 The User Interface

A preview of the user interface is shown in Figure 3. To support non-expert users, the interface is user-friendly and minimal. The dashboard contains several tabs ("Upload data file", "Visualisation", "Help", "Logout"). Several drop down lists can be used to see which is the data uploaded last (file name), a list of Subjects and a list of Relations as extracted from text.

The graph is interactive and represents the triples into an acyclic graph where the nodes are Subjects or Objects and the edges arrows indicates the Objects. The graph can be filtered choosing one Subject from the drop-down list of

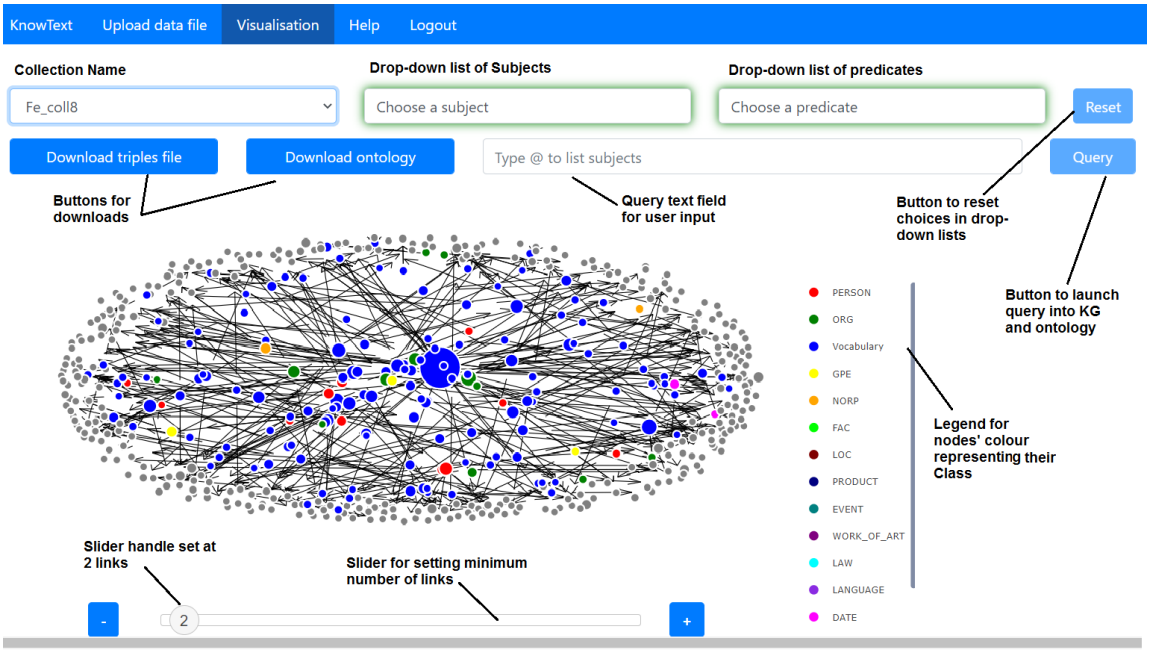

Fig. 3. KowText user interface showing main functionalities.

Subjects (which will draw the graph based on that Subject and its relations only) or by choosing one relation from the drop-down relation list, which will re-draw the graph showing all the subjects connected by that connection only.

The dashboard of the demonstrator (Figure 3) includes elements that allow to query the ontology (a text-box with data from the ontology will pop-up when requested by the user) as well as a text box for free text queries across the KG.

4 IMPLEMENTATION

The KnowText demonstrator performs automated Knowledge Graph extraction from unstructured text collections in the field of banking, HR, economic news, customer reviews. It is a web application accessible through a web-link, built using the Django 3.0 - Python 3.6 framework. This is a portable, flexible and scalable approach and it is also easy to use, even by the non-expert users.

KnowText performs the following automated tasks: (i) extracts triples (Subject, Predicate, Object) from the text collection; (ii) Generates automatically a basic OWL type ontology which can distinguish between 16 Classes (Named entities and Vocabulary) which are populated with facts (data type properties) as extracted from the text collection; (iii) dynamic ontology extraction from unstructured text generates a basic OWL type ontology which can distinguish between 16 Classes (including Named Entity and Domain Vocabulary extraction); (iv) dynamic linking and population of the ontology with Triples extracted from text. We describe next the set of design considerations (C1 - C4) and approaches for KnowText.

C1: Address the Domain Vocabulary extraction

An ad-hoc, domain vocabulary is dynamically extracted from the collection of texts as words that occur as Subject in one of the triples but do not occur in the NE categories. Various conditions on the type and structure of such words can be added in order to select the most meaningful domain-vocabulary words. Although such words are dynamically extracted and expert domain-knowledge is not applied for this task, the majority of words extracted into the domain-vocabulary category appear relevant for the domain when the initial collection of texts belongs to a specific domain.

C2: Dynamic generation of Ontology

The automated generation of the KG and its knowledge base (ontology) are obvious advantages since these do not require prior data processing or prior ontology development. The generated ontology can be downloaded as an OWL file and can further developed if necessary outside the application.

C3: Address the temporal evolution of the KG

KnowText uses a logical construct called a Collection, which facilitates the grouping together of related documents. Different Collections can be created with one KG being built and maintained for each Collection. When the information from new documents need to be added to the knowledge base, the user can take a decision on which Collection to add the new documents to, and the respective KG (and ontology) are refreshed using the newly added documents. This feature allows the KG to evolve organically over time, as the number of documents increase. Collections are initially created by the administrator of the KnowText system.

C4: Enable user-friendly querying

User-friendly querying (no need for a specific query language) is provided by using the following methods:
(i) text based queries from user input (natural language typed-in text). Such queries enable query on entity or relation (and its synonyms) present in the typed-in text (which does not have to follow grammar, but should contain the desired words). The text input for search should be concise – as long phrases give a very focused result or may take in uninteresting and un-related topics.

The output shows the KG filtered upon the words located in the input text (and synonyms), an example being shown in Figure 4 showing a free text query (i.e."Ann said").

(ii) filtering the KG by entity(node) or by relationship (link) by selecting a word from the drop-down list of Subjects or Predicates, the output being a filtered representation of the graph (nodes and links) containing the selected word either as node or relationship (Figure 5).
(iii) from the ontology: display ontology records as Class and properties (attributes) of the entity (when querying a node by selecting it from the drop-down list of Subjects).
(iv) Several interactive visualizations of KG can be used also for filtering the KG: (a) based on the number of nodes to be

shown (using a slider provided); (b) based on the ontology Class for nodes (by choosing the colour from the Classes colour legend).

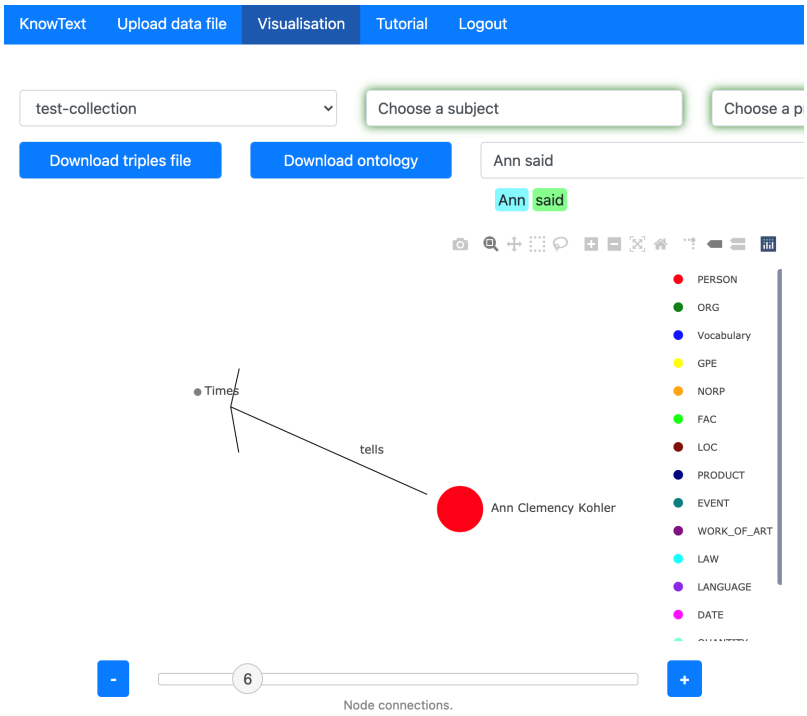

Fig. 4. KnowText user interface showing a free text query (i.e."Ann said")

5 EVALUATION

Traditional Knowledge Graphs (generated through pre-defined ontology schema and manual assignment and definition of relation types) can be characterized by evaluating a number of quality metrics (Syntactic validity, Semantic accuracy, Consistency, Conciseness and Completeness) [6].

Dynamically built KG (as in KnowText) that use a small, restricted and domain specific data (unstructured text) can answer some of the above requirements by default:

- The RDF/URI availability and inter-linking as well as accessibility are ensured by default through the dynamic generation of the KG (and its underlining OWL ontology);
- Security is established by default since company documents are usually signed, verified and validated prior to addition into the company's database.
- Performance (in terms of throughput) has less relevance since it is intended mainly for one user at any one time. The performance as latency can vary between 0.1 s to 5 s depending on the size of the dataset (20kB -200kB).
- Scalability can be described in terms of time required to generate the KG and is affected by data size, since triples, the ontology and the KG visualization are generated dynamically. Typical generation times are about 5 s/kB. The

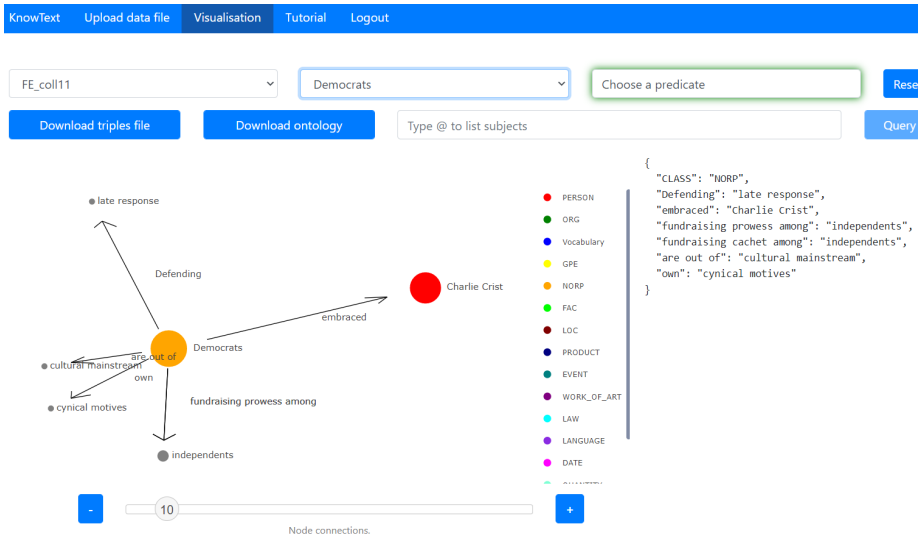

Fig. 5. KnowText user interface showing a query based on a NE (NORP).

maximum data size per batch is here set at 200kB due to memory limitations imposed by the **spaCy** algorithm for NE extraction. Nevertheless, many batches can be uploaded to the same Collection, which will dynamically update with the new data both the ontology and the KG. There is no limitation in total data size.

As regarding the “intrinsic dimensions” [6], KnowText can be further improved for:

- Syntactic validity: so far KnowText is not designed for explicit definition of the allowed values for a certain data type or to verify syntax or spelling;
- Semantic accuracy rules were not yet implemented although rules for dependencies between the values of two or more different properties can be described in the definition of the ontology. Such rules can be numerous and quite domain specific, and they can be added as ”on-demand” parameters.

The "Consistency" properties are described in Table 3 where each property is discussed in terms of relevance and/or implementation in the system and the "Result" column indicates conformity or not to property indicated. At the moment only one third of requirements are satisfied and up two thirds can be achieved with further development, while another third is not applicable to our case. The "Conciseness" – refers to the uniqueness of Properties and Classes and they are ensured by the dynamic generation and entity linking of the OWL ontology. The "Completeness" (referring to the degree of representation of all Classes in schema and Properties) is not applicable here as completeness cannot be ensured by a limited amount of information contained in a given (small) dataset.

6 DISCUSSION AND SUMMARY

KnowText enables users to visualise a graph representation of facts extracted as triples. Unlike industrial KG based on large web-based ontologies (i.e. dbpedia, wikipedia, etc.) KnowText restricts the information pool to the representative data uploaded by the user, allowing for high specificity – within a company domain of interest (based on banking, CRM,

Table 3. Consistency metrics discussed for KnowText according to metrics types defined in [6].

Metrics type	Result	Discussion
1: detection of use of NE as members of disjoint classes	Yes	Ratio of: # of all NE in disjoint classes by # all words is equal to 1. as we create the Domain Vocabulary.
2: detection of misplaced classes or properties	Yes	By default, in (S,P,O) triples all S are classified into Classes. Missclassifcation of S or O can occur while extracting triples due to unclear syntax. These will be visible in the visualisation interface.
3: detection of misuse of owl:DatatypeProperty or owl:ObjectProperty through the ontology maintainer	Yes	Only owl:DatatypeProperty are implemented so far. The owl:ObjectProperty can be implemented by domain experts, using existing backend code or Protégé. The Reasoner in the OWL ontology will show the error.
4: detection of use of members of owl:DeprecatedClass or owl:DeprecatedProperty by specifying manual mappings from deprecated terms to compatible terms.	NA	The ontology is not mapped manually. The S, P when updated by the user are updated across the ontology avoiding further contradictions
5: detection of bogus owl:InverseFunctionalProperty values by checking the uniqueness and validity of the inverse-functional values	No	Not implemented
6: detection of the re-definition by third parties of external classes/properties (ontology hijacking)	N.A.	Not applicable as Classes cannot be re-defined. Number and type of Classes is fixed for the user.
7: detection of negative dependencies or correlation among properties using association rules	No	Such properties have not been defined.
8: detection of inconsistencies in spatial data through semantic and geometric constraints	Yes, partially	Spatial data referring to geographic NE is extracted automatically. Other constraints were not implemented.
9: the attribution of a resource's property to be of a certain type detected by use of SPARQL queries as a constraint.	N.A	Properties are not described through limitations, SPARQL is not used.
10: detection of inconsistent values by the generation of a particular set of schema axioms for all properties in a dataset and the manual verification of these axioms	No	Not implemented

HR, investigative journalism). Here the specificity refers to evaluating only the entities and relations present in the existing company documentation.

At the end of computation, two files are generated: one containing extracted facts (triples) and another containing the OWL ontology extracted from text, which can be downloaded.

There are numerous directions for further improvement by adding Inverse and Functional DataProperties which can enrich the strength of the ontology reasoner. The KnowText application presents a number of advantages for data management and information extraction from unstructured text as it offers several interactive options to the user: node based and edge (relationship) based searches through filtering of the graph, or node based searches into the ontology, along with user-friendly, natural language text-based querying (no need for a specific query language). The dynamic generation of the KG design for a custom and domain-specific document collection can prove useful for managing information in medium size companies.

More information about the KnowText application can be found in the "Demonstrators" area at: https://www.ceadar.ie/our-work/technology-demonstrators/.

REFERENCES

[1] T. Berners-Lee and O. Hendler, J. & Lassila. 2001. The semantic web. *Scientific american* 284, 5 (2001), 34–43.

[2] Philipp Cimiano and Johanna Völker. 2005. Text2Onto A framework for ontology learning and data-driven change discovery. *Lecture Notes in Computer Science* 3513 (2005), 227–238. https://doi.org/10.1007/11428817_21

[3] Paul Cuddihy, Justin McHugh, Jenny Weisenberg Williams, Varish Mulwad, and Kareem Aggour. 2018. SemTK: A semantics toolkit for user-friendly SPARQL generation and semantic data management. In *CEUR Workshop Proceedings*, Vol. 2180. 1–16.

[4] Lisa Ehrlinger and Wolfram Wöß. 2016. Towards a definition of knowledge graphs. *CEUR Workshop Proceedings* 1695 (2016).

[5] Michael Färber and Achim Rettinger. 2018. Which Knowledge Graph Is Best for Me? arXiv:1809.11099 http://arxiv.org/abs/1809.11099

[6] Junyang Gao, Xian Li, Yifan Ethan Xu, Bunyamin Sisman, Xin Luna Dong, and Jun Yang. 2019. Efficient knowledge graph accuracy evaluation. In *Proceedings of the VLDB Endowment*, Vol. 12. 1679–1691. https://doi.org/10.14778/3342263.3342642 arXiv:1907.09657

[7] Toader Gherasim, Mounira Harzallah, Giuseppe Berio, and Pascale Kuntz. 2013. Methods and tools for automatic construction of ontologies from textual resources: A framework for comparison and its application. *Studies in Computational Intelligence* 471 (2013), 177–201. https://doi.org/10.1007/978-3-642-35855-5_9

[8] Johannes Hoffart, Fabian M. Suchanek, Klaus Berberich, and Gerhard Weikum. 2013. YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia. *Artificial Intelligence* 194 (2013), 28–61. https://doi.org/10.1016/j.artint.2012.06.001

[9] Matthew Honnibal. [n. d.]. SpaCy library. https://spacy.io/

[10] Matthew Honnibal. [n. d.]. Spacy 'neuralcoref' library. downloadablefrom{https://spacy.io/universe/project/neuralcoref}

[11] Agnieszka Konys. [n. d.]. Knowledge repository of ontology learning tools from text. *Procedia Computer Science* ([n. d.]), 1614–1628. https://doi.org/10.1016/j.procs.2019.09.332

[12] Agnieszka Konys. [n. d.]. Knowledge Systematization for ontology learning methods. *Procedia Computer Science* ([n. d.]), 2194–2207. https://doi.org/10.1016/j.procS.2018.07.229

[13] Alfred Krzywicki, Wayne Wobcke, Michael Bain, John Calvo Martinez, and Paul Compton. 2016. Data mining for building knowledge bases: Techniques, architectures and applications. *Knowledge Engineering Review* 31, 2 (2016), 97–123. https://doi.org/10.1017/S0269888916000047

[14] Kundan Kumar and Amitabh Mukherjee. 1237. Constructing knowledge graph from unstructured text. *Siddhant Manocha* 12375 (1237). http://home.iitk.ac.in/{~}kundan/report{_}365.pdf

[15] Jean Baptiste Lamy. 2017. Owlready: Ontology-oriented programming in Python with automatic classification and high level constructs for biomedical ontologies. *Artificial Intelligence in Medicine* 80 (2017), 11–28. https://doi.org/10.1016/j.artmed.2017.07.002

[16] James P. McCusker, Sabbir M. Rashid, Nkechinyere Agu, Kristin P. Bennett, and Deborah L. McGuinness. 2018. The Whyis knowledge graph framework in action. *CEUR Workshop Proceedings* 2180 (2018).

[17] T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel., J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. 2015. Never-Ending Learning. *Proceedings of the National Conference on Artificial Intelligence* 3 (2015), 2302–2310. https://doi.org/10.1007/978-3-642-83740-1_24

[18] Ontotext. 2017. GraphDB Free. (2017). https://ontotext.com/

[19] Heiko Paulheim. 2017. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic Web* 8, 3 (2017), 489–508. https://doi.org/10.3233/SW-160218

[20] Heiko Paulheim. 2018. *Machine Learning with and for Semantic*. Vol. 11078. 110–141 pages. https://doi.org/10.1007/978-3-030-00338-8

[21] Philippe Remy. [n. d.]. Stanford OpenIE library. https://pypi.org/project/stanford-openie/

[22] Robyn Speer, Joshua Chin, and Catherine Havasi. 2016. ConceptNet 5.5: An Open Multilingual Graph of General Knowledge. Singh 2002 (2016). arXiv:1612.03975 http://arxiv.org/abs/1612.03975

[23] Fabian Suchanek and Gerhard Weikum. 2013. Knowledge harvesting from text and web sources. In *Proceedings - International Conference on Data Engineering*. IEEE, 1250–1253. https://doi.org/10.1109/ICDE.2013.6544916

[24] Paola Velardi, Roberto Navigli, Alessandro Cucchiarelli, and Francesca Neri. 2005. Evaluation of OntoLearn, a methodology for automatic learning of domain ontologies. *Ontology Learning from Text: Methods, evaluation and applications* 123 (2005), 92.

[25] Jihong Yan, Chengyu Wang, Wenliang Cheng, Ming Gao, and Aoying Zhou. 2018. A retrospective of knowledge graphs. *Frontiers of Computer Science* 12, 1 (2018), 55–74. https://doi.org/10.1007/s11704-016-5228-9

[26] Zhanfang Zhao, Sung-Kook Han, and In-Mi So. 2018. Architecture of Knowledge Graph Construction Techniques. *International Journal of Pure and Applied Mathematics* 118, 19 (2018), 1869–1883.