Semi-automatic Ontology Builder Based on Relation Extraction from Textual Data

Anjali Thukral, Ayush Jain, Mudit Aggarwal and Mehul Sharma

Abstract This paper proposes a semi-automated tool to build ontology from text. The tool consists of an analyzer to parse the given text and a mapper that maps NLP triple to RDF triple under user supervision. The resulted RDF triple is then validated through "triple validator" for its existence in the ontology. The triple is augmented to the ontology if it does not exist. System learns during this process and provides better mapping suggestions with time, making ontology building faster.

Keywords Ontology builder · Text to ontology · Concept ontology
RDF triple

1 Introduction

Ontology is a description (like a formal specification of a program) of the concepts and relationships that can formally exist for an agent or a community of agents [1]. A domain ontology (or concept ontology) represents the conceptual model underlying a certain domain, describing it in a declarative way [2]. In the present time, majority of the data is in the textual form such as journals, documents, and website. Converting these unstructured data into ontologies requires a lot of time and manual work. So, there is high requirement of some kind of automation to make the task of conversion easier. To fulfill this requirement, we developed a tool that converts text into concept ontology with minimum human intervention. The tool uses three main methods relation extraction, rule-based mapping, and validator and storage which are described in later sections. This paper is organized as follows: Sect. 2 presents the related work on building ontologies. Details of the proposed system are provided in Sect. 3. Section 4 provides performance evaluation of the system, followed by the conclusion in Sect. 5.

A. Thukral (✉) · A. Jain · M. Aggarwal · M. Sharma
Keshav Mahavidyalaya University of Delhi, Delhi, India
e-mail: athukral@keshav.du.ac.in

© Springer Nature Singapore Pte Ltd. 2018

S. Bhattacharyya et al. (eds.), *Advanced Computational and Communication Paradigms*, Advances in Intelligent Systems and Computing 706,
https://doi.org/10.1007/978-981-10-8237-5_33

2 Related Work

A lot of attention has been given toward the building of ontology over the years. A huge set of efforts have been put in the field of automatic building of ontologies, mainly focusing on domain (concept) ontology. Maddi et al. [3] presented the method of extraction of ontologies for text documents using linear algebraic method called as single-value decomposition for obtaining the concepts from terms and used bipartite ontology graphs to represent the results. Yeh and Yang [4] proposed a method of automatic ontology creation for historical documents from digital library using latent topic extraction and topic clustering. Bedini and Nguyen [5] presented a framework that evaluates the automation of ontology generation. The work also provides a comprehensive analysis of existing software. The work of Moreno and Sanchez [6] provides a methodology to build ontologies automatically and extracting information from web documents. The work of Gantayat and Iyer [7] for automatic building of ontology from lecture notes is available at several courseware repositories. It uses NLP for keyword extraction, term frequency inverse document frequency (tf-idf) for extracting the concepts from keywords, and "apriori algorithm" to determine associations among concepts. Gillam and Ahmad [8] have proposed statistical methods for extracting the concepts from the text. Most of the work done for building of ontology is for a specific domain, whereas the proposed tool is a generic system that can build any ontology in less time under the domain expert (user) supervision, irrespective of its domain. Moreover, final decision is in the hands of domain expert, so it prevents the insertion of ambiguous and incorrect information into the ontology. This tool generates standard OWL ontology which can be accessed and manipulated in ontology editors such as protégé [9].

3 Proposed Approach

The proposed tool is a user-friendly generic semi-automated ontology builder which uses relation extraction and rule-based mapping procedure to build concept ontologies from text (Please refer Fig. 1). The ontology builder creates the concept ontology, besides domain learning during the building process. With time and enough learning, the system is able to produce concept ontology in less time with minimal human intervention. It prunes the input texts and converts it into basic form, i.e., triples (subject, predicate, and object) using relation extraction and adds it to ontology after validation from user. If builder is not able to recognize the semantics of the information, user will provide the semantics, which will intern add to the learning of the software, so that it becomes capable to recognize the similar semantics of the information in future and is able to add the correct information in the ontology.

The proposed system contains three modules analyzer subsystem, mapper subsystem, and triple validator and storage. Home screen of the tool provides the

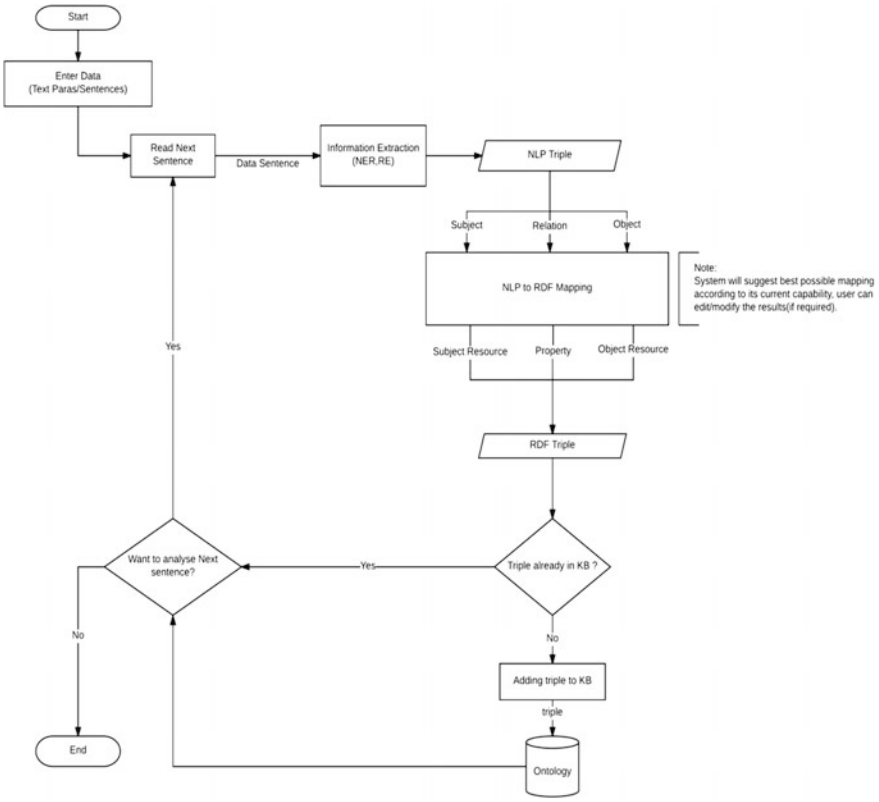

Fig. 1 Proposed system flow

interface for the features like starting a new ontology building session, viewing the current ontology file, importing and exporting of OWL file, rule file, and performance CSV file which are explained below.

3.1 Analyzer Subsystem

Analyzer subsystem performs the task of relation extraction. It processes the input text and presents all the possible relations that can be extracted from the given sentence by passing the text through OpenIE component [10] of "Stanford CoreNLP API [11]". User then selects the best relation among the system suggestions, which is then passed to the mapper subsystem for further processing.

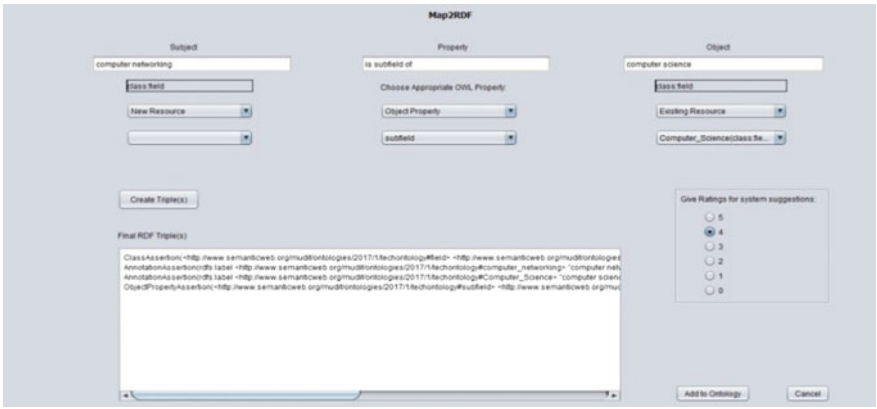

Fig. 2 Mapper screen

3.2 Mapper Subsystem

The mapper uses a relation that was extracted using analyzer, schema of the concept ontology, and the rule file which defines certain mapping rules to map predicate text to an appropriate OWL property. Mapping of subject/object text is done using label annotations of existing individual in the ontology. If the predicate text is mapped to a data property, then object and subject text are mapped to a literal and an individual, respectively. But, both subject and object texts are mapped to an individual if predicate is mapped to an object property.

Rule File

The rule file contains rule of format "Predicate text" → "OWL property IRI". Rules are arranged in the lexical order of predicate text for efficient searching. During the mapping process whenever user explicitly maps predicate text to an OWL property, then corresponding rule is added to the rule file.

Working of Mapper

The mapper first tries to map subject, predicate, and object itself (Ref. Fig. 2). In case the mapper does not find an appropriate solution, it recommends the possible options to be selected by a user manually. After mapping all the components of relation, it creates appropriate OWL triples using OWL API [12]. User is asked to give a feedback from scale of 0–5 as on what level tool was helpful. User feedback is saved in a CSV file with the help of Java CSV API.[1] The resulted triples are then passed to the next module, i.e., Triple validator and storage system.

[1] https://www.csvreader.com/java_csv.php.

3.3 Triple Validator and Storage

Validator module checks for the existence of all the triples received from mapper, in the ontology either as it is or in the form of inference from the existing information in ontology using HermiT Reasoner [13]. "Already Exists" status depicts that triple information is already present in the ontology, and "New" status depicts that the triple is added to the ontology as it was not there earlier. After adding all the triples, ontology is checked for the consistency using HermiT Reasoner [13]. In case of inconsistency, changes are not reflected in the original ontology file.

4 Experimental Results

We created ontology on "Technology Classification" for experimental purpose. The ontology classifies various fields and subfields of science and technology. The ontology and relevant text data (collected from various sources) were fed into the system. Experiment was performed over four sessions under supervision of domain expert to display the building process. Growth of ontology, generated rule file, and performance of the system were captured. Ontology graph was created using SOVA plug-in[2] for Protégé ontology editor [9].

4.1 Technology Classification Ontology

Ontology is illustrated in Fig. 3. Ontology schema "Technology Classification" fed to the application was created using protégé ontology editor [9] and was fed to the system as ontology schema.

Figure 4 illustrates the populated "Technology Classification" ontology which was obtained after passing significant amount of text data into the system.

4.2 Rule File Generation

All rules formed during the mapping procedure for the predicate text part of the relation are stored in rule file. The system refers to this rule file for mapping purpose.

[2] https://protegewiki.stanford.edu/wiki/SOVA_1.0.0.

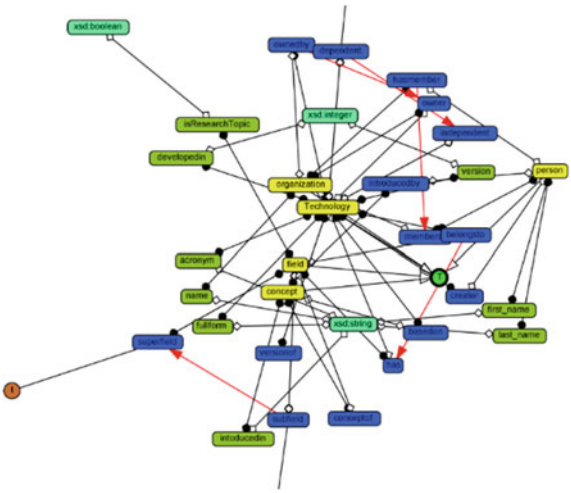

Fig. 3 Ontology schema "Technology Classification" fed to the application

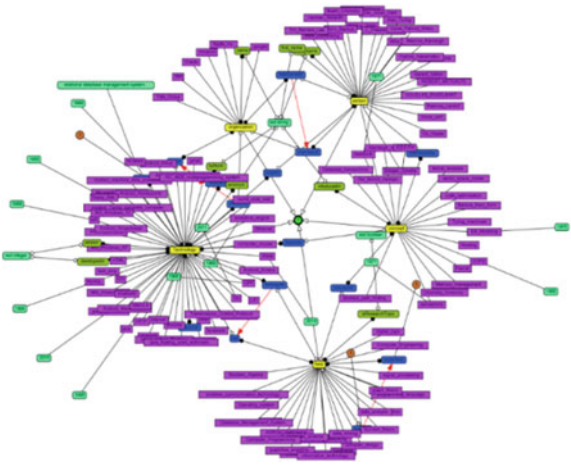

Fig. 4 Ontology obtained after processing through application

4.3 Performance

A user may follow the guidelines (tabulated in Fig. 5) while providing feedback regarding the mapping accuracy.

For subject/object, 0, 1, and 2 stand for new resource, existing resource but not mapped by the system, and existing resource which is mapped by the system, respectively. Similarly, for predicate, 0 and 1 stand for predicate which was not mapped to a property and mapped to a property, respectively.

Figure 6 shows the performance of the tool during four sessions. It shows that as the process of ontology building continues, the system is able to efficiently convert text to RDF triples, making the ontology building process faster.

Fig. 5 Feedback guidelines regarding mapping

Subject	Predicate	Object	Rating
0	0	0	0
0	0	1	0
0	0	2	1
0	1	0	1
0	1	1	1
0	1	2	3
1	0	0	0
1	0	1	1
1	0	2	1
1	1	0	1
1	1	1	2
1	1	2	4
2	0	0	1
2	0	1	1
2	0	2	2
2	1	0	3
2	1	1	4
2	1	2	5

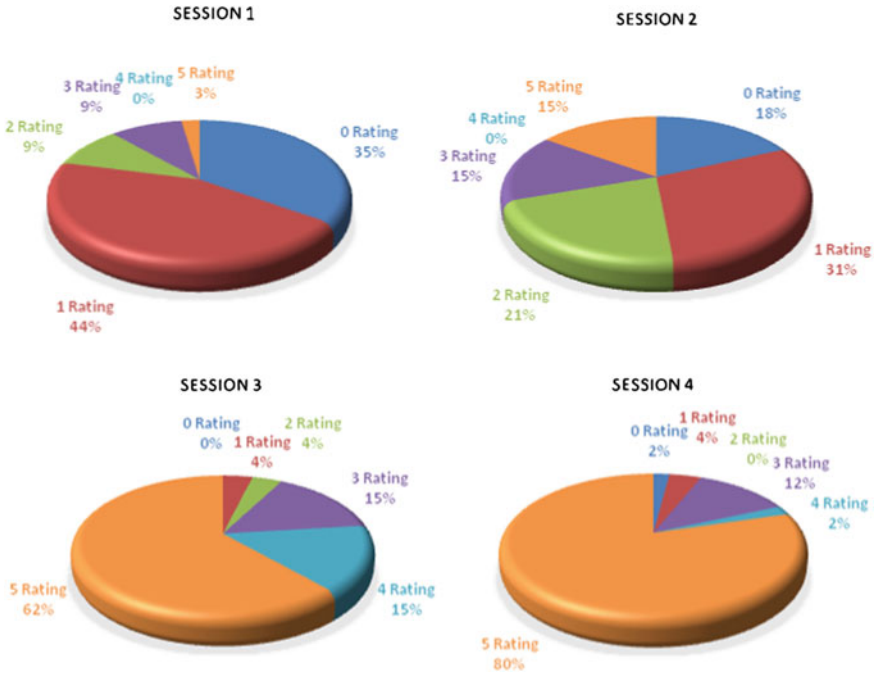

Fig. 6 The tool performance (rating based)

5 Conclusion

A tool for building ontology from text was proposed in the paper. The design of the proposed tool is modular with three components: analyzer subsystem, mapper subsystem, and triple validator and storage. The experiment was performed on technology classification ontology. It was noted that system's performance and results are greatly dependent upon the ontology schema and the input text fed to the tool. The ontology building is a learning process, and the perceived knowledge is a result of user's interaction with the system. It was also noted that whatever be the ontology, after converting a certain amount of text (which covers good amount of domain's knowledge) into triples, the system was able to get trained and was able to make correct recommendations. As the future work, the analyzer subsystem module can be improved to handle complex sentences.

References

1. Gruber, T.R.: Toward principles for the design of ontologies used for knowledge sharing. Int. J. Hum. Comput Stud. **43**(5–6), 907–928 (1995)
2. Bedi, P., Thukral, A., Banati, H., Behl, A., Mendiratta, V.: A multi-threaded semantic focused crawler. J. Comput. Sci. Technol. **27**(6), 1233–1242 (2012)
3. Maddi, G.R., Velvadapu, C.S., De Lamadrid, J.G.: Ontology extraction from text documents by singular value decomposition (2001)
4. Yeh, J.H., Yang, N.: Ontology construction based on latent topic extraction in a digital library. In: International Conference on Asian Digital Libraries, pp. 93–103. Springer, Berlin, Heidelberg (2008)
5. Bedini, I., Nguyen, B.: Automatic ontology generation: state of the art. PRiSM Laboratory Technical Report. University of Versailles (2007)
6. Sanchez, D., Moreno, A.: Creating ontologies from web documents. In: Recent Advances in Artificial Intelligence Research and Development, vol. 113, pp. 11–18. IOS Press (2004)
7. Gantayat, N., Iyer, S.: Automated building of domain ontologies from lecture notes in courseware. In: 2011 IEEE International Conference on Technology for Education (T4E), pp. 89–95. IEEE (2011)
8. Ahmad, K., Gillam, L.: Automatic ontology extraction from unstructured texts. In: On the Move to Meaningful Internet Systems 2005: Coopis, Doa, and Odbase, pp. 1330–1346 (2005)
9. Musen, M.A.: The protégé project: a look back and a look forward. AI Matters **1**(4), 4–12 (2015)
10. Angeli, G., Premkumar, M.J., Manning, C.D.: Leveraging linguistic structure for open domain information extraction. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL 2015) (2015)
11. Manning, C.D., Surdeanu, M., Bauer, J., Finkel, J.R., Bethard, S., McClosky, D.: The Stanford CoreNLP natural language processing toolkit. In: ACL (System Demonstrations), pp. 55–60 (2014)
12. Horridge, M., Bechhofer, S.: The OWL API: a Java API for OWL ontologies. Semant. Web **2** (1), 11–21 (2011)
13. Glimm, B., Horrocks, I., Motik, B., Stoilos, G., Wang, Z.: HermiT: an OWL 2 reasoner. J. Autom. Reason. **53**(3), 245–269 (2014)