1st International Conference on Advances in Science, Engineering and Robotics Technology 2019 (ICASERT 2019)

Extracting RDF Triples from Raw Text

Yeasmin Ara Akter
Dept. of Computer Science and Engineering
East Delta University
Chittagong, Bangladesh
yeasmin.a@eastdelta.edu.bd

Md. Ataur Rahman
Dept. of Language Science and Technology
University of Saarland
Saarbrücken, Germany
arahman@coli.uni-saarland.de

***Abstract*—This manuscript manifests the results of our work on extracting RDF triples from raw text data. We took a corpus of news articles and applied several methods for extracting "subject - verb - object" relationships from texts. The first method that we used is syntactic parsing and constructing triples from *nsubj* and *obj* syntactic relations. The second approach incorporates extracting semantic roles with the help of frame parser and constructing triples from *agent*, *patient* and *predicate* relationships. After applying the aforementioned methods, a manual evaluation was done over a small number of hand labeled samples. Our system achieved a Precision score of *0.34*, Recall of *0.46* and a combined score of *0.39* for F-measure as the final result.**

***Keywords*—Knowledge and Data Engineering, Semantic Information Processing, RDF, Text Analysis and Mining, Natural Language Processing.**

I. Introduction

Dealing with unstructured data is a common problem in the field of Semantic Web and Linked Data. Probably the most prominent project in the field – DBpedia – has mostly processed data in Wikipedia tables, thus discarding the huge amount of information available in the article texts. The system we developed is thus dedicated to extracting *'Resource Description Framework'* (RDF) triples from raw texts. We aim at extracting specifically RDF data and not just some named entities because the extracted RDF triples then can be used for constructing knowledge graph and creating intelligent *Question Answering* (QA) systems and chatbots.

The goal of our project is to test simple methods that can be used to extract RDF data from natural language texts. The existing research has shown that carefully extracting triples from the text is a very complex task, but we can explore what we can achieve using shallow parsing and simple rules. Even a small amount of extracted data can be useful in various applications.

The full paper has the following structure: in Section II, we will describe the background and existing research, in Section III we will be describing our methodology and in Section IV, the results of our system will be depicted. Section V will present the evaluation of our results. In Section VI we will report what other methods apart from syntactic and semantic parsing we also tried to use. Finally, in Section VII we will conclude our work.

II. Background And Motivation

Several previous works have dealt with the topic of extracting RDF triples from unstructured text. In particular, we are inspired by the work by [1]. They have used a pipeline of semantic processing tools to extract DBpedia RDF triples from Wikipedia articles. Among others, they extracted existing Wikipedia links from the articles, perform semantic parsing based on PropBank [2] and coreference resolution on the article texts and used the Wikifier [3] tool to recognize further entities for which DBpedia entries exist. After the above steps, they obtained triples of the form *subject - predicate - object*, which they subsequently mapped onto the DBpedia namespace. Thus, the central contribution of their work is the extension of the DBpedia knowledge base with the facts which they extracted from Wikipedia articles in the form of RDF triples.

Other applications in the same vein include LODiffier by Augenstein et al. [4] and KnowledgeStore by Cattoni et al. [5]. The former also incorporated Wikifier and used deep semantic analysis and discourse representation structures to obtain relations between the entities recognized. The latter creates a knowledge base from multimedia resources with the aid of an existing background knowledge base; among others, they resolved coreference and used clustering to detect entities about which attributes can be extracted using so-called *triggers*. Gerber and Ngomo [6], on the other hand, used a pattern-based approach; instead of performing semantic parsing, they obtained triples by extracting recurrent patterns which represent natural language predicate-argument constellations.

In our project, we largely proceed along the lines of [1]. From the raw text, we intend to extract facts in the form of *subject-predicate-object* RDF triples with the help of semantic processing which will be compatible to be mapped into the DBpedia namespace where possible. In contrast to [1], we will not be using Wikipedia articles but rather plain running texts. This has the advantage that we will not need to clean our data of annotations in Wikipedia's markup language. On the other hand, Wikipedia articles will already contain useful links for entities which are covered by DBpedia, while our plain text data will not.

III. Methodology And Experimental Setup

A. Dataset

For the purpose of our experiment, we decided to use a corpus consisting of *News Articles*[1] provided by Signal Media [7]. It consists of texts from several news agencies, websites and blogs. The overall size of the dataset is over 100 million tokens, but we decided to use a portion of it that consists of 500000 tokens.

[1] http://research.signalmedia.co/newsir16/signal-dataset.html

978-1-7281-3445-1/19/$31.00 ©2019 IEEE

B. Pipeline

We intended for our system to be a pipeline of the following component:

a. The Plain raw text has been obtained from the corpus data.
b. Tokenization and POS-tagging is performed on the data.
c. Both the dependency parsing and semantic role-labeling techniques have been explored.
d. From the processing output of step c, triples are extracted in the form: *subject argument - verb - object argument*
e. Finally, triples of the aforementioned structure are converted into RDF-triple (JSON-LD format).

Both the extraction of triples based on dependency parsing (hence syntactic processing) and on semantic role-labeling (hence semantic processing) were explored. At a later stage, we chose the latter.

C. Triple Extraction Based on Dependency Parsing

We performed syntactic parsing using UDPipe [8], which tokenizes the input sentences and returns for each sentence, among others, the word lemmas and the universal dependency relations which hold between the individual words in the sentence. We then extracted the following three lemmas to form a triple:

- The lemma with the dependency relation *root* (i.e. the root word of the sentence).
- The lemma with the dependency relation *nsubj* to the sentence root word.
- The lemma with the dependency relation *obj*, to the sentence root word.

The above three lemmas would respectively correspond to the *verb*, the *subject argument* and the *object argument* of a triple. To illustrate, Fig. 1 shows the relevant parts of the UDPipe output for the sentence '*The Texas State football team scored 63 points in its home opener*', from which we extracted the triple *(team - score - point)* (in red).

1	The	the	DET	DT	Definite=Def|PronTy	5	det
2	Texas	Texas	PROPN	NNP	Number=Sing	3	compound
3	State	State	PROPN	NNP	Number=Sing	4	compound
4	football	football	NOUN	NN	Number=Sing	5	compound
5	team	team	NOUN	NN	Number=Sing	6	nsubj
6	scored	score	VERB	VBD	Mood=Ind|Tense=Pa	0	root
7	63	63	NUM	CD	NumType=Card	8	nummod
8	points	point	NOUN	NNS	Number=Plur	6	obj
9	in	in	ADP	IN	_	12	case
10	its	its	PRON	PRP$	Gender=Neut|Numb	12	nmod:poss
11	home	home	NOUN	NN	Number=Sing	12	compound
12	opener	opener	NOUN	NN	Number=Sing	8	nmod
13	.	.	PUNCT	.	_	6	punct

Fig.1. Example UDPipe output with tokens forming the triple marked in red

This approach, based on dependency parsing, was characterized by the fact that each triple element would consist of exactly one single token. But this easily gave rise to errors of the following kind: The alleged triple *(Williams - get - America)* was falsely extracted from the sentence *Vanessa Williams Finally Got Her Miss America Apology*, which is at least in part due to the fact that the object argument, *(Her) Miss America Apology* consists of multiple tokens, while the dependency parser will only assign to one of the dependency relation *obj*. For this reason, we chose not to base further processing steps on this approach but turn instead to triple extraction on the basis of semantic processing described in the next section.

D. Triple Extraction Based on Semantic Role-labeling

Semantic role-labeling was conducted using the Senna system [9], which performs, among others, tokenization, chunking and semantic role-labeling on plain texts. A sample output of Senna for the sentence '*VETERANs saluted Worcester's first ever breakfast club for ex-soldiers which won over hearts, minds and bellies*' is given in Fig. 2 (with the columns with information on POS-tags and semantic roles marked in red).

VETERANS saluted Worcester's first ever breakfast club for ex-soldiers which won over hearts, minds and bellies.

VETERANS	NNP	S-NP	0	-	S-A0	0	(S1(S(NP*)
saluted	VBD	S-VP	0	saluted	S-V	0	(VP*
Worcester	NNP	S-NP	S-ORG	-	B-A1	0	(NP(NP*
's	POS	B-NP	0	-	I-A1	0	*)
first	JJ	I-NP	0	-	I-A1	0	*
ever	RB	I-NP	0	-	I-A1	0	*
breakfast	NN	I-NP	0	-	I-A1	0	*
club	NN	E-NP	0	-	E-A1	0	*)
for	IN	S-PP	0	-	B-A2	0	(PP*
ex-soldiers	NNS	S-NP	0	-	I-A2	S-A0	(NP(NP*)
which	WDT	S-NP	0	-	I-A2	S-R-A0	(SBAR(WHNP*)
won	VBD	S-VP	0	won	I-A2	B-V	(S(VP*
over	IN	S-PP	0	-	I-A2	E-V	(PP*
hearts	NNS	S-NP	0	-	I-A2	B-A1	(NP*
,	,	0	0	-	I-A2	I-A1	*
minds	NNS	B-NP	0	-	I-A2	I-A1	*
and	CC	I-NP	0	-	I-A2	I-A1	*
bellies	NNS	E-NP	0	-	E-A2	E-A1	*))))))))
.	.	0	0	-	0	0	*))

Fig.2. Example Senna output with POS-tags and semantic roles marked in red

In accordance with the PropBank framework [2], adopted by [9] for Senna - *V*, *A(rgument)0* and *A(rgument)1* respectively denote the predicate, the agent and the agent arguments. For each sentence, we therefore, extracted from the Senna output those words tagged as *A0* as the subject argument, those tagged as *V* as the verb, and those tagged as *A1* as the object argument. In the example shown in Fig. 2, we would output the triples *(VETERANS - saluted - Worcester's first ever breakfast club)* and *(ex-soldiers which - won over - hearts, minds and bellies)*.

However, without further cleaning, our approach would include alleged triples in which the subject is a personal pronoun, or in which some elements (e.g. the verb) length is longer than usual length for example the chunk '*obtain amid an expected record turnout, in what would be a big boost to a secession campaign which has been losing support over the last two years*' as Senna would tag the entire phrase as *V*. Therefore, we included the following constraints into the extraction process, which relate to the length of each element of a triple as well as the POS categories of the words:

- Words forming the verb in the triple must have as their POS-category one of the following tags: VB, VBD, VBG, VBN, VBP, VBZ or IN. These are the verbal POS-categories with the addition of IN for prepositions, which is intended to include phrasal verbs such as *win over*.

- Words forming the subject and object arguments must *not* be of the POS-categories PRP, PRP$, WDT or WP, which represent personal and possessive pronouns as well as WH-determiners and pronouns.

- Alleged triple is discarded if the *verb* consists of more than three words.

- Alleged triple is discarded if the *subject* or *object* argument consists of more than eight words.

The above constraints ensured that, for instance, we would in fact extract *(ex-soldiers - won over - hearts, minds and bellies)* from the example in Fig. 2, excluding the word *which* from the subject argument.

E. Conversion of Triples to RDF

After extracting the triples from the Senna output we needed to convert it to a suitable format so that these can be read as RDF. We used JSON-LD format as an intermediate form because it is easy to convert text into JSON programmatically. Since JSON-LD is a *Linked Data* type, it is easily readable by any program such a Protégé. We thus used Protégé to convert the JSON-LD file to RDF/XML.

The entire pipeline of our implementations is given in Fig.3., with the UDPipe module greyed out as we did not continue onwards with its output.

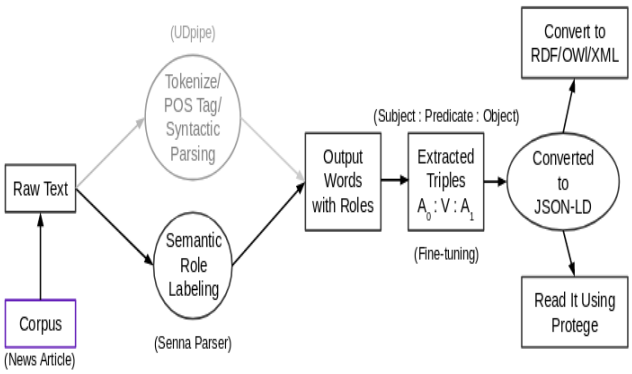

Fig.3. The Pipeline of our system with UDPipe module greyed out

IV. Result

As the result of our project, we extracted 2361 triple from our data[2]. The triples look in the following way:

Subject	Predicate	Object
A migrant	holds	A banner

The extracted triples were organized in an ontology with different subjects and objects serving as instances of individual classes, and predicates serving as object properties. The overall organization of the ontology can be seen in Fig. 4.

The ideal organization of the ontology would be to create different general classes like *Person*, *Organization* etc, but as we do not have a gold standard which we could use as a training data for classifying our instances, this step remains to be added in the future.

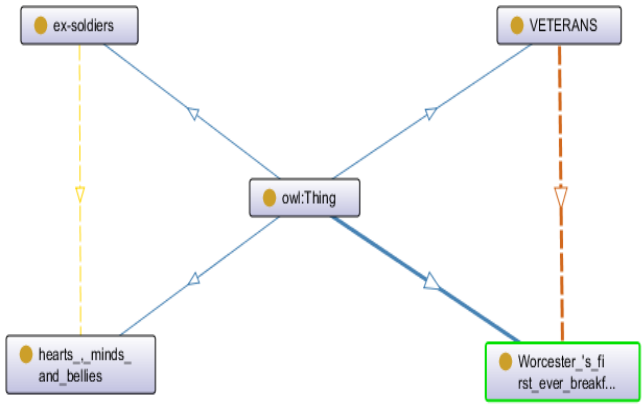

Fig.4. Organization of triples in the ontology.

V. Evaluation

Evaluation poses a challenge to our proposed project since we cannot compare the triples extracted by our system to an existing gold-standard. One possibility is simply to select a small sample of data and to manually extract triples from these sentences. The results of this manual extraction can function as gold-standard against which the system's performance could be evaluated in terms of precision, recall and F-measure.

To perform this kind of evaluation, we manually annotated a sample of 100 sentences. The main difficulty is to define what exactly we want to count as a part of a triple. It is obvious that triples should not include only named entities. At the same time, many cases are not interesting or useful for constructing a knowledge graph. For example, in a sentence '*mother talks with her daughter*' we wouldn't want to extract anything, though the triple can be extracted with the means of semantic parsing. In the end, we decided to analyze as triples those entities which are not generic, for which a Wikipedia page exists or might exist, or which can be considered named entities.

In the selected sample our system has found 81 triples. Based on the annotated sample, we calculated recall and precision measures. They can be found in Table I. The most common problem that can be noticed while annotating the triples is that borders of an extracted and annotated triple do not coincide. The system tends to take larger chunks, whereas only two or three words are enough for manual annotation. Also, our system has clear rules by which it selects certain phrases as parts of a triple: the chunks should have such semantic roles as agent, patient or predicate. Meanwhile, the data that can be used in RDF triples is much more varied, and many other semantic roles might be included in the triple. For example, time and place such as in the phrase '*The Worcester Breakfast Club ... met at the Postal Order in Foregate Street*

[2] Results of our system can be downloaded from the following link under the folder *Final RDF* - https://goo.gl/z4Yuiz

at 10 am on Saturday'. are not detected by our system, whereas they should be included in a proper knowledge graph.

TABLE I. RESULTS FOR THE EXTRACTION OF RDF TRIPLES

Precision	*Recall*	*F-measure*
0.34	0.46	0.39

VI. DISCUSSIONS AND FUTURE WORK

At present, our triples have been mapped to our own example URL and are therefore not linked to any existing knowledge base. Ideally, we would like to enrich an existing knowledge graph with the facts we express in our extracted triples to the effect that these facts can be queried in a question answering system. This would resemble the final step in Exner and Nugues's work [1], in which they map their extracted triples to the DBpedia namespace.

Hence, we also attempted to link our triples to DBpedia entities using DBpedia Spotlight [10] or Wikifier [3], the latter of which have been used both by Exner and Nugues [1] and by Augenstein et al. [4]. The idea was to apply either of the tools to our raw text data. Ideally, some of the entities recognized (i.e. entities covered by DBPedia) would correspond to the subject or object arguments in our triples, and the fact expressed by our triples would be added to those entities as novel knowledge.

However, we ran into a similar problem as mentioned in the previous section with regard to evaluation: Senna outputs large chunks of text as subject and object arguments. They often have a complex internal structure and, for our purpose, ought to be further broken down into more "atomic" units. For instance, the object argument '*Worcester's first ever breakfast club*' of our aforementioned triple (see section III.D) is in fact internally complex, from which we would ideally extract, at a minimum, the triples *(unique_id – isA - breakfast_club)* and *(unique_id - hasLocation - Worcester)*. Both DBpedia Spotlight and Wikifier were able to recognise and link *Worcester* and at times also *breakfast_club* (depending on the pre-selected confidence value); however, '*Worcester's first ever breakfast club*' as a whole, which functions as the object argument in our triple, could, plausibly, not be recognized by either tool. Therefore, in order to connect our triples to the DBpedia knowledge base in the future, more fine-grained processing of the elements of each triple is needed, possibly using further semantic tools.

VII. CONCLUSION

In our system, we attempted to apply simple methods for extracting RDF triples from raw text data. It turned out that this task can be solved to some extent using semantic parsing and converting the extracted semantic roles to RDF triples. The resulting F-measure was 0.39, which can be improved in many ways, but still is decent for the beginning.

The process of extracting RDF triples can be improved both on the side of semantic parsing and on the side of conversion. In terms of semantic parsing, we can try to extract more complex semantic relations like date and time and also do anaphora resolution. Apart from that, our data needs more linguistic preprocessing like filtering out non-English words and special symbols, lemmatizing all the triple instances and removing possible duplicates. While in the case of conversion, we need to better work through the structure of our ontology and develop macro-classes for our individual instances so that the ontology becomes less sparse and more concise. For doing that, we need to collect gold standard data which can be used for applying machine learning to classify individual instances.

REFERENCES

[1] P. Exner and P. Nugues, "Entity extraction: From unstructured text to dbpedia rdf triples," in *The Web of Linked Entities Workshop (WoLE 2012)*, 2012.

[2] M. Palmer, D. Gildea and P. Kingsbury, "The proposition bank: An annotated corpus of semantic roles," *Computational linguistics,* vol. 31, pp. 71-106, 2005.

[3] L. Ratinov, D. Roth, D. Downey and M. Anderson, "Local and global algorithms for disambiguation to wikipedia," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, 2011.

[4] I. Augenstein, S. Padó and S. Rudolph, "Lodifier: Generating linked data from unstructured text," in *Extended Semantic Web Conference*, 2012.

[5] R. Cattoni, F. Corcoglioniti, C. Girardi, B. Magnini, L. Serafini and R. Zanoli, "The KnowledgeStore: an Entity-Based Storage System.," in *LREC*, 2012.

[6] D. Gerber and A.-C. N. Ngomo, "Extracting multilingual natural-language patterns for rdf predicates," in *International Conference on Knowledge Engineering and Knowledge Management*, 2012.

[7] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of machine learning research,* vol. 12, pp. 2493-2537, 2011.

[8] D. Corney, D. Albakour, M. Martinez-Alvarez and S. Moussa, "What do a million news articles look like?," in *NewsIR@ ECIR*, 2016.

[9] J. Daiber, M. Jakob, C. Hokamp and P. N. Mendes, "Improving efficiency and accuracy in multilingual entity extraction," in *Proceedings of the 9th International Conference on Semantic Systems*, 2013.

[10] M. Straka, J. Hajic and J. Straková, "UDPipe: Trainable Pipeline for Processing CoNLL-U Files Performing Tokenization, Morphological Analysis, POS Tagging and Parsing.," in *LREC*, 2016.