ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)
Vol 4, Issue 11, November 2017

Unstructured Text to DBPEDIA RDF Triples – Entity Extraction

[1] Monika S G, [2] Chiranjeevi S, [3] Harshitha A, [4] Harshitha M, [5] V K Tivari, [6] Raghevendra Rao
[1 – 5] Department of Electronics and Communication Engineering, Sri Sairam College of Engineering, Anekal, Bengaluru.
[5] Assistant Professor, Department of Electronic and Communication Engineering, Sri Sairam College of Engineering, Anekal, Bengaluru.
[6].Assistant Professor, Department of Computer Science and Engineering, Sri Sairam College of Engineering, Anekal, Bengaluru

Abstract:- **In the means of current technologies Use of data, information has grown significantly over the last few years. The information processing facing an issue like where the data is originating from multiple sources in an uncontrolled environment. The reason for the uncontrolled environment is the data gathered beyond the organization and generated by many people working outside the organization. The intent of this paper is delving into this unformatted information and build the framework in such a way that the information becomes more managed and used in the organization. Case and point for resume submitted for particular positions should become searchable. In this framework, we try and solve the problem and provide suggestions on how to solve other similar problem. In this paper, we describe an end-to-end system that automatically extracts RDF triples describing entity relations and properties from unstructured text. This system is based on a pipeline of text processing modules that includes an asemantic parser and a co-reference solver. By using co-reference chains, we group entity actions and properties described in different sentences and convert them into entity triples. We applied our system to over 114,000 Wikipedia articles and we could extract more than 1,000,000 triples. Using an ontology-mapping system that we bootstrapped using existing DBpedia triples, we mapped 189,000extracted triples onto the DBpedia namespace. These extracted entities are available online in the N-Triple format.**

Index Terms — **Framework, Knowledge base, TST, Inverted Index.**

I. INTRODUCTION

By using the structured and semi-structured information from Wikipedia, DBpedia [1] has created very large amounts of linked data and is one the most significant achievements of the Semantic Web initiative. Datasets from DBpedia are used in a wide range of applications such as faceted search, model training for information extraction, etc. DBpedia focuses on extracting structured information from Wikipedia articles, such as info box templates and categorization information. However, the unstructured text of the articles is left unprocessed. Some recent projects have attempted to use this text content to extend the DBpedia triple base. Examples include iPopulator [2] that populates in complete info boxes with attribute values it identifies from the article text, while two recent systems, LODifier [3] and Knowledge Store [4], extract semantic information from the text. LODifier creates RDF triples based on Word Net URIs while Knowledge-Store uses its own ontology. Nonetheless, these systems show limitations in the form of preexisting info box templates or data structures that are not fully compliant with the DBpedia name space.In this paper, we introduce a frame work to carry out an end-to-end extraction ofDBpedia triples from unstructured text. Similarly to LODifier and Knowledge Store, our framework is based on entities and identifies predicate–argument structures usinga generic semantic processing pipeline. However, instead of recreating new semantic structures, we integrate the DBpedia property ontology and therefore make the reuse and extension of the DBpedia dataset much easier. Starting from the DBpedia dataset, we link the triples we extract from the text to the exis ting DBpedia ontology, while going beyond the existing info box templates. Applications already using DBpedia would then benefit from a richer triple store. Related Work. The extraction of relational facts from plain text has long been of interest in information extraction research. The key issue in relation extraction is to balance the trade-off between high precision, recall, and scalability. With the emergence of the Semantic Web and numerous ontologies, data integration has become an additional challenge. There has been a considerable amount of research on semi-supervised [5–7] methods using bootstrapping techniques together with initial seed relations to create extraction patterns. Unsupervised approaches [8, 9] have contributed further improvements by not requiring hand-labeled data.

These approaches have successfully answered scalability and precision factors, when applied on web-scale corpora. The challenge of ontology and data integration has been addressed by [10].Due to concerns on scaling, the use of syntactic or semantic relation extraction techniques in relation extraction has been relatively sparse. Few systems carry out a complete analysis of the source documents using co-reference resolution or discourse analysis to extract all statements. Exceptions include LODifier [3] and Knowledge-Store [4], that have extracted semantic information and applied co reference resolution. However, the entities extracted by these systems have not been integrated to a single homogenous ontology. In contrast to these approaches, we suggest an end-to-end system, that extracts all the entity relations from plain text and attempts to map the entities onto the DBpedia name space. We balance precision and recall by employing a combination of NLP tools, including semantic parsing, co reference resolution, and named entity linking. Scalability issues are handled by parallelizing the tasks on a cluster of computers. Furthermore, we propose an ontology mapping method that bootstraps learning from existing triples from the DBpedia dataset.

II. SYSTEM ARCHITECTURE

The architecture of our system is a pipeline that takes the Wikipedia articles as input and produces entities in the form of DBpedia RDF triples. As main features, the system includes a generic semantic processing component base on a semantic role labeler (SRL) to discover relations in text, an automatic learning of ontology mappings to link the extracted triples to the DBpedia namespace, and an algorithm to rank named entity links (NEL) found in coreference chains in order to discover representative mentions.

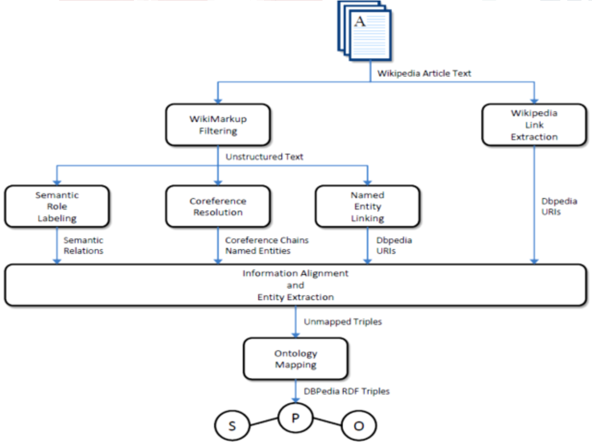


Fig. 1. Overview of the entity extraction pipeline.

In total, the end-to-end processing of Wikipedia article text consists of seven modules
(Figure 1):

1. A WikiMarkup filtering module that removes theWikimedia markup, providing the plain text of the articles to the subsequent modules;
2. A Wikipedia link extractor that extracts Wikipedia links from the articles;
3. A semantic parsing module, Athena [11], a framework for large-scale semantic parsing of text written in natural language;
4. A coreference resolution module that detects and links coreferring mentions in text;
5. A mention-to-entity linking module that links mentions to a corresponding DBpediaURI;
6. An information aligning and entity extracting module that aligns the output from top-level modules and extracted entities in the form of triples.
7. An ontology mapping module that carries out the final mapping of predicates from thePropbank nomenclature onto the DBpedia namespace.

4 Processing of Wikipedia Article Text WikiMarkup Filtering. Prior to any analysis, the text must be filtered. This is an essential step that seeks to remove annotations and markups without affecting the running text.

Without this step, subsequent modules would fail in their analysis and lead to erroneous extractions. Wikipedia articles are composed of text written in natural language annotated with a special markup called wikitext or wiki markup. It is a simple markup language that allows among other things the annotation of categories, templates, and hyper linking to other Wikipedia articles. Wikipedia also allows the use of common HTML tags. By filteringWikipedia text, weaim at removing all annotations, sections that contain only links and references, and keeping only the running text. This process is difficult since the HTML syntax is often invalid. The most common errors are tags that are leftunclosed or are incorrectly nested.

Wikipedia Link Extraction:
During the Wikipedia link extraction, we extract and preserve the original links along with their corresponding mentions in the article. In addition to extracting the links annotated by the article authors, we make the assumption that the first noun phrase in the first sentence corresponds to the article link. The rationale behind it is that the longest coreference chain in the article often starts with this first mention.The direct correspondence between Wikipedia articles and DBpedia resources allowsus to map Wikipedia links onto their corresponding DBpedia URI by simply adding the DBpedia name space. Semantic Parsing.Frame

semantics [12] is a linguistic theory that assumes that the meaning of a sentence is represented by a set of predicates and arguments. The Proposition Bank [13] is a project that applied this theory to annotate corpora with predicate argument structures. For each predicate, Propbank identifies up to six possible core arguments denoted A0, A1, ..., and A5 that go beyond the traditional annotation of subjects and objects. Propbank also includes modifiers of predicates, such as temporal and location adjuncts. These roles are instrumental in performing the extraction of entities as they allow the identification of properties containing temporal and locational data with high precision. We use the Athena framework created for parallel semantic parsing of unstructuredtext. At its core, the system uses a high-performance multilingual semantic role labeler that obtained top scores in the CONLL-2009 shared task [14, 15]. Coreference Resolution. A coreference resolver creates chains of coreferring mentions by discovering and linking anaphoric phrases to their antecedents. We used a co reference solver, included in the Stanford CoreNLP package [16, 17], to link mentions of entities in the different parts of text. This allows us to group entity actions and properties described in different sentences. CoreNLP uses a pipeline of tokenizers, partof-speech tagger, named entity recognizer, syntactic parser, and coreference solver to annotate unstructured text. In addition to co reference annotation, we store the named entity classification created by the pipeline. The named entity classes are used to filter named entity links having a conflicting ontology classification.

Named Entity Linking:

An important step in entity extraction is the grounding of named entities to unique identifiers. In most articles, only the first mention of a named entity is annotated with a corresponding Wikipedia link; subsequent mentions are often left unannotated. Wikifier [18] is a named entity linking system that annotates unstructured text with Wikipedia links. By applying Wikifier, we can link unannotated namedentities in the Wikipedia articles to a corresponding DBpedia URI. Ontology Mapping. During semantic parsing, the sentences are annotated with predicate–argument structures called rolesets. As dictionary, the parser uses PropBank that defines more than 7,000 rolesets. Propbank associates each predicate with a set of senses, for instance bear has six senses denoted bear.01, bear.02, ..., bear.06. Finally, each predicate-sense has a set of core arguments that differ with each roleset. For example, bear.02 has two core arguments: A0, the mother, and A1, the child. Considering only the core roles, this amounts to more than 20,000 roles. The objective of ontology mapping is to map the predicate and argument roles from PropBank onto DBpedia properties. We perform this final step to create the DBpedia RDF triples. Figure 2 shows an example of end-to-end processing to DBpedia RDF triples of the sentences: Luc Besson (born 18 March 1959) is a French film director, writer and producer. Besson was born in Paris to parents who were both Club Medscuba diving instructors.

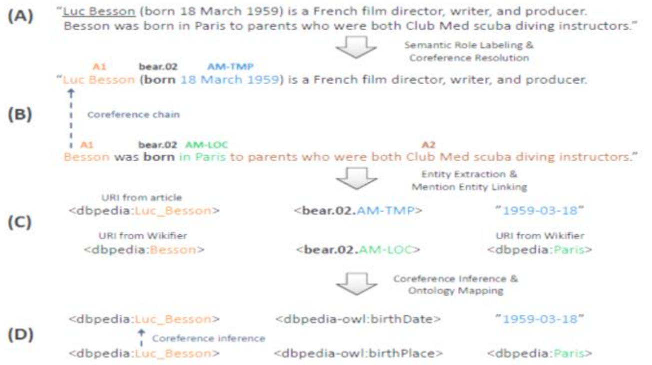

(A)
"Luc Besson (born 18 March 1959) is a French film director, writer, and producer.
Besson was born in Paris to parents who were both Club Med scuba diving instructors."
Semantic Role Labeling & Coreference Resolution
A1
bear.02
AM-TMP
"Luc Besson (born 18 March 1959) is a French film director, writer, and producer.
(B)
Coreference chain
A1
bear.02
AM-LOC
A2
Besson was born in Paris to parents who were both Club Med scuba diving instructors."
Entity Extraction & Mention Entity Linking
URI from article
<dbpedia:Luc_Besson>
<bear.02.AM-TMP>
"1959-03-18"
(C)
URI from Wikifier
<dbpedia:Besson>
<bear.02.AM-LOC>
URI from Wikifier
<dbpedia:Paris>
Coreference Inference & Ontology Mapping
<dbpedia:Luc_Besson>
<dbpedia-owl:birthDate>
"1959-03-18"
(D)
Coreference inference
<dbpedia:Luc_Besson>
<dbpedia-owl:birthPlace>
<dbpedia:Paris>

Fig. 2. An ideal conversion from text to the DBpedia RDF triples: (A) The input sentences. (B) The sentences after semantic parsing and coreference resolution. (C) Entity extraction. (D) Ontology mapping.

Entity Extraction

The arguments created during semantic parsing are searched in order to find namedentity links corresponding to RDF subjects and objects. This process uses the mentions discovered by the co reference solver, Wikipedia links predicted by Wikifier, and Wikipedia links extracted from the article. In order to keep the task tractable, we have limited the entities to those found in DBpedia and we do not introduce new named entities to the DBpedia ontology. RDF Subjects. PropBank uses the A0 label as the argument describing agents, causers, or experiencers, while arguments labeled as A1 describe entities undergoing a state of change or being affected by an action. In both cases, arguments labeled A0 or A1 can be considered containing RDF subjects and are consequently searched for named entity links. Arguments labeled A0 are searched first, arguments labeled A1 are only searched if a named entity link wasn't discovered in the preceding arguments. RDF Objects. Following the subject extraction, the remaining arguments are examined to discover potential objects. The core arguments and two auxiliary arguments, temporal AM-TMP and location AM-LOC, are searched. The extracted data types can be categorized as following: Named entity links expressed as DBpedia URIs, dates andyears, integers, and strings. We search date expressions in the temporal arguments AMTMP using regular expressions. By using seven common date patterns, we are able to extract a large amount of date and year expressions. We associate the location arguments AM-LOC to named entity links representing places. These links are extracted only if they are classified as dbpedia-owl:Placeby the DBpedia ontology.

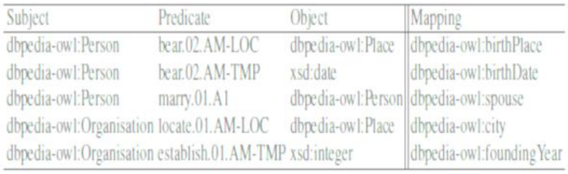
Named Entity Link Ranking and Selection. During the search of RDF subject and
objects, we search and select candidate named entity links in the following order:
1. Wikipedia links, converted to DBpedia URIs. We consider named entity links extracted
from the article as being most trustworthy.
2. Wikifier-predicted Wikipedia links, converted to DBpedia URIs, and having a DBpedia ontology class matching the predicted named entity class. A predicted named entity link is chosen only in the case when an extracted Wikipedia link isn't given. Furthermore, predicted links are pruned if their DBpedia ontology class doesn't match the named entity class predicted by the Stanford co reference solver.
3. Co reference mentions; the most representative named entity link (according to the score described in section Using Co reference Chains) in the co reference chain is selected. We consider named entities inferred through co reference chains as the least trustworthy and select them only if an extracted or predicted named entity link is not given. A mention placed in the wrong co reference chain will be considered as an incorrect named entity link; a situation which Wikifier can rectify with higher precision.
Using Co reference Chains. Co reference chains are used to propagate named entity links to arguments having neither an extracted nor a predicted named entity link. This situation arises most commonly for arguments consisting of a single pronoun. Before propagation takes place, we determine the most representative named entity link in the
Co reference chain using a ranking and scoring system: – Extracted named entity links are always selected over predicted links. – A score of +2 is given to a named entity link if it has a DBpedia ontology class matching the predicted named entity class.
– The score is increased by the number of tokens of the named entity minus 1.
– If a tie is given between equally scoring named entity links, the link closest to the top of the chain is selected.
We derived the set of scoring rules by performing an empirical examination of coreference
chains. We observed that coreference chains representing people, often started with a mention containing the full name, followed by single-token mentions having only the first or last name. The named entity links of single-token mentions, as predicted by Wikifier, often incorrectly pointed to either a place or a family. By rewarding named entity links having multiple tokens and matching ontology classes, we filtered these incorrect links. Table 1 shows an example, where the mention Robert Alton, a person name, is given the highest score due to matching entity classes and token length. Although the mention Alton refers to the same entity and belongs to the co reference chain, an incorrect named entity link to a city (Alton, Illinois) has been predicted. Given our previous rules, the predicted named entity link is discarded due to a mismatch with the predicted named entity class. The correct named entity link is thus resolved by propagating the link through the co reference chain. Unmapped Triple Generation. Given a set of extracted RDF subjects and objects, we create binary relations from n-ary predicate–argument relations by a combinatorial generation. We discover negative relations by searching the argument roles for AMNEG; these are then discarded.

III. CONCLUSIONS

The framework for unstructured data processing and example case study of resume management system yielded some learning's. We attempt to list some of them here, not in order of importance.
A. Generic Pre-Observations
•Intelligent classification of unstructured data types yields to better processing techniques.
•Common techniques can be exploited for greater benefit if we know in advance as to what kind of information we would be looking at.
•Application building around unstructured data is complex and time consuming task
B. Specific Post Conclusions
•The understanding of data set improves the design quite significantly and yields to better database design. In this particular resume processing was better done because of indexing technique.
•Solutions that work on unstructured data fit in today's web architecture and in very rare cases may need modification to n-tier mode.
•The custom needs to such application can be folded into business logic of web applications.
•XML formatting is important tool that aids for unstructured custom reporting.
•The maximum task in entire work was sent inpresentation logic, which will remain pain point and will need custom work every time.

Experimental Results

The aim of the evaluation is to answer the question of how much information in the form of entity relation triples can be extracted from sentences.We also wish to evaluate the quality of the extracted triples. Since there is no gold standard annotation of entities found in the main text of Wikipedia articles, we performed the evaluation by manually analyzing 200 randomly sampled sentences from different articles. Sampled sentences are examined for relevant subject-predicate-object triples and compared to the correspondingretrieved triples. We computed the precision, recall, and F1 scores, and in the occurrence of an extraction error, we made a note of the originating source. We evaluated the attributes of each triple in a strict sense: Each extracted attribute must exactly match the corresponding attribute in the sentence. For instance, in evaluating the birthplace of a person, if a sentence states a city as the location, we only consideran extracted DBpedia link to the city as correct. In contrast, if the extracted link refers to a more generalized toponym, such as region or country, we mark the extracted object as erroneous. In total, we processed 114,895 randomly selected articles amounting to 2,156,574 sentences. The articles were processed in approximately 5 days on a cluster of 10 machines.



Subject	Predicate	Object	Mapping
dbpedia-owl:Person	bear.02.AM-LOC	dbpedia-owl:Place	dbpedia-owl:birthPlace
dbpedia-owl:Person	bear.02.AM-TMP	xsd:date	dbpedia-owl:birthDate
dbpedia-owl:Person	marry.01.A1	dbpedia-owl:Person	dbpedia-owl:spouse
dbpedia-owl:Organisation	locate.01.AM-LOC	dbpedia-owl:Place	dbpedia-owl:city
dbpedia-owl:Organisation	establish.01.AM-TMP	xsd:integer	dbpedia-owl:foundingYear

Table 2. Five of the most frequent ontology mappings learned through bootstrapping.

Table 3, left, shows the number of processed articles categorized by DBpedia

Ontology classes. From the processed articles, we extracted a total of 1,023,316 triples, of which 189,610 triples were mapped to the DBpedia ontology. The unmapped triples differ in having the predicate localized to the Propbank namespace. In Table 3, right, we can see that from the 189,610 extracted triples, 15,067 triples already exist in the DBpedia dataset. This means that our framework introduced 174,543 new triples to the DBpedia namespace. Almost 3% of the extracted triples are duplicates, the majority of these are triples repeated only once. Since a statement with the same meaning can occur in more than one article, we consider these occurrences natural. In comparing the number of extracted triples to the number of processed sentences, we find that roughly every second sentence yields one extracted triple. In comparison to the number of processed articles, we extracted nearly 9 triples per article. The extracted mapped triples reached a F1 score of 66.3%, a precision of 74.3%, and a recall of 59.9%. The largest source of errors came from predicates, 46.4%, followed by subjects, 27.2%, and objects, 26.4%. Based on post-mortem analysis of the evaluated triples, we find that reasons for the extraction errors can be attributed to the following causes:

– An incorrect mapping from the Propbank predicate-argument roles to the DBpedia ontology properties.

– A new entity is detected, that has previously not been introduced to the DBpedia datasets and therefore lacks a corresponding DBpedia URI.

– The wrong URI is predicted for an entity and cannot be resolved or corrected by the scoring algorithm.

– A mention is placed in the incorrect coreference chain by the coreference solver.

REFERENCES

[1] Robert Sadgewick “Algorithms IN C” 3rd ed 1976

[2] Brian W. Kernighan & Dennis M. Ritchie “The C Programming Language”, Second Edition Prentice Hall 1988

[3] K. N. King W.W. Norton & Company “C Programming: A Modern Approach” 1996

[4] David Megginson “Structuring XML Documents”Publisher: Prentice
Hall April, 1998

[5]. Agichtein, E., Gravano, L.: Snowball: extracting relations from large plain-text collections. In: Proceedings of DL ’00, New York, ACM (2000) 85–94

[6]. Etzioni, O., Cafarella, M., Downey, D., Kok, S., Popescu, A.M., Shaked, T., Soderland, S., Weld, D.S., Yates, A.: Web-scale information extraction in knowitall. In: Proceedings of WWW ’04, New York, ACM (2004) 100–110

[7]. Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka, E.R., Mitchell, T.M.: Toward an architecture for never-ending language learning. In: Proceedings of AAAI-10. (2010) 1306–1313

[8]. Banko, M., Etzioni, O.: Strategies for lifelong knowledge extraction from the web. In: Proceedings of K-CAP ’07, New York, ACM (2007) 95–102

[9]. Fader, A., Soderland, S., Etzioni, O.: Identifying relations for open information extraction. In: Proc. of EMNLP ’11. (2011) 1535–1545

[10]. Suchanek, F.M., Sozio, M.,Weikum, G.: Sofie: A self-organizing framework for information extraction. In: Proceedings of WWW ’2009, New York (2009) 631–640

[11]. Exner, P., Nugues, P.: Constructing large proposition databases. In: Proc. of LREC’12, Istanbul (2012)