RExtractor: a Robust Information Extractor

Vincent Kríž and Barbora Hladká
Charles University in Prague
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics
{kriz, hladka}@ufal.mff.cuni.cz

Abstract

The RExtractor system is an information extractor that processes input documents by natural language processing tools and consequently queries the parsed sentences to extract a knowledge base of entities and their relations. The extraction queries are designed manually using a tool that enables natural graphical representation of queries over dependency trees. A workflow of the system is designed to be language and domain independent. We demonstrate RExtractor on Czech and English legal documents.

1 Introduction

In many domains, large collections of semi/unstructured documents form main sources of information. Their efficient browsing and querying present key aspects in many areas of human activities.

We have implemented an information extraction system, `RExtractor`, that extracts information from texts enriched with linguistic structures, namely syntactic dependency trees. This structure is represented as a rooted ordered tree with nodes and edges and the dependency relation between two nodes is captured by an edge between them. Namely, we work with the annotation framework designed in the Prague Dependency Treebank project.[1]

RExtractor forms an extraction unit of a complex system performing both information extraction and data publication according to the Linked Data Principles. More theoretical and practical details on the system are provided in (Kríž et al., 2014). The system focuses on processing Czech legal documents and has been implemented in an applied research project addressed by research and business partners.[2]

The extraction systems known from literature were evaluated against gold standard data, e.g. DKPro Keyphrases (Erbs et al., 2014), RelationFactory (Roth et al., 2014), KELVIN (McNamee et al., 2013), Propminer (Akbik et al., 2013), OLLIE (Mausam et al., 2012). We name this type of evaluation as *academic* one. According to the statistics provided by International Data Corporation (Gantz and Reinsel, 2010), 90% of all available digital data is unstructured and its amount currently grows twice as fast as structured data. Naturally, there is no capacity to prepare gold standard data of statistically significant amount for each domain. When exploring domains without gold standard data, a developer can prepare a small set of gold standard data and do academic evaluation. He gets a rough idea about his extractor performance. But he builds a system that will be used by users/customers, not researchers serving as evaluators. So it is user/customer feedback what provides evidence of performance. This particular feature of information extraction systems is discussed in (Chiticariu et al., 2013) together with techniques they use academic systems and commercial systems.

We decided to do a very first RExtractor testing by experts in accountancy. It has not done yet so we have no evidence about its quality from their perspective. However, we know what performance the

[1] `http://ufal.mff.cuni.cz/pdt3.0`

[2] `http://ufal.mff.cuni.cz/intlib`

Proceedings of NAACL-HLT 2015, pages 21–25,
Denver, Colorado, May 31 – June 5, 2015.

system achieves on the gold standard data that we prepared in the given domain. We list it separately for entity extraction, where Precision = 57.4%, Recall = 91.7%, and relation extraction, where P = 80.6%, R = 63.2%. Details are provided in (Kríž et al., 2014).

2 RExtractor Description

RExtractor is an information extractor that processes input documents by natural language processing tools and consequently queries the parsed sentences to extract a knowledge base of entities and their relations. The parsed sentences are represented as dependency trees with nodes bearing morphological and syntactic attributes. The knowledge base has the form of *(subject, predicate, object)* triples where *subject* and *object* are entities and *predicate* represents their relation. One has to carefully distinguish subjects, predicates and objects in dependency trees from subjects, predicates and objects in entity-relation triples.

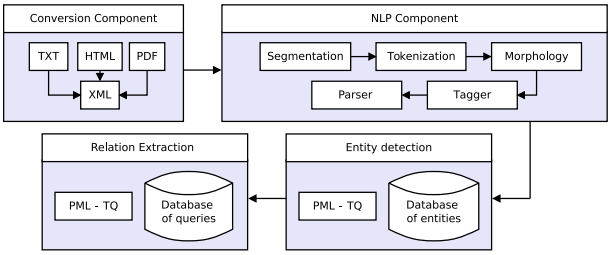

Figure 1: RExtractor workflow

RExtractor is designed as a four-component system displayed in Figure 1. The NLP component outputs a syntactic dependency tree for each sentence from the input documents using tools available in the Treex framework.[3] Then the dependency trees are queried in the Entity Detection and Relation Extraction components using the PML-TQ search tool (Pajas and Štěpánek, 2009). The Entity Detection component detects entities stored in Database of Entities (DBE). Usually, this database is built manually by a domain expert. The Relation Extraction component exploits dependency trees with detected entities using queries stored in Database of queries (DBQ). This database is built manually by a domain expert

[3] http://ufal.mff.cuni.cz/treex

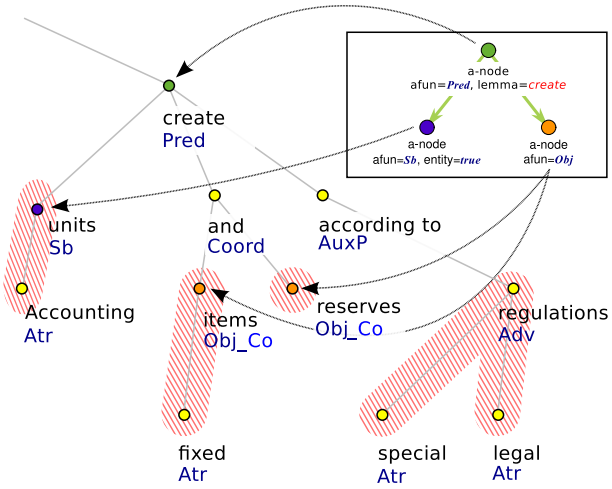

Figure 2: Extraction of *who creates what*

Subject	Predicate	Object
accounting unit	create	fixed item
accounting unit	create	reserve

Table 1: Data extracted by the query displayed in Figure 2

in cooperation with an NLP expert. Typically, domain experts describe what kind of information they are interested in and their requests are transformed into tree queries by NLP experts.

Illustration Let's assume this situation. A domain expert is browsing a law collection and is interested in the *to create something* responsibility of any body. In other words, he wants to learn *who creates what* as is specified in the collection. We illustrate the RExtractor approach for extracting such information using the sentence *Accounting units create fixed items and reserves according to special legal regulations*.

Firstly, the NLP component generates a dependency tree of the sentence, see Figure 2. Secondly, the Entity Detection component detects the entities from DBE in the tree: *accounting unit*, *fixed item*, *reserve*, *special legal regulation* (see the highlighted subtrees in Figure 2). Then an NLP expert formulates a tree query matching the domain expert's issue *who creates what*. See the query at the top-right corner of Figure 2: (1) he is searching for *creates*, i.e. for the predicate having lemma *create* (see the root node), (2) he is searching for *who*, i.e. the subject

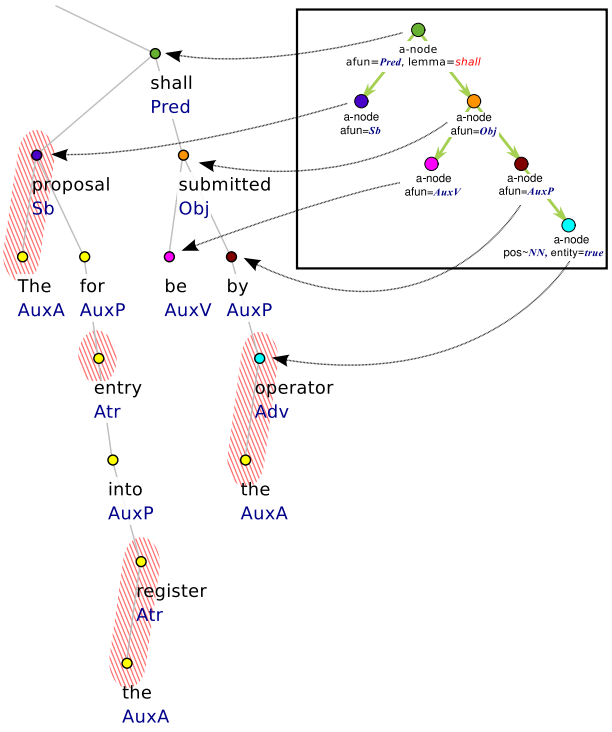

Figure 3: Extraction of *who should do what*

Subject	Predicate	Object
operator	submit	proposal

Table 2: Data extracted by the query displayed in Figure 3

(see the left son of the root and its syntactic function afun=`Sb`), and *what*, i.e. the object (see the right son of the root and its syntactic function afun=`Obj`). Even more, he restricts the subjects to those that are pre-specified in DBE (see the left son of the root and its restriction entity=`true`). Finally, the Relation Extraction component matches the query with the sentence and outputs the data presented in Table 1.

A domain expert could be interested in more general responsibility, namely he wants to learn *who should do what* where *who* is an entity in DBE. A tree query matching this issue is displayed in Figure 3. The query is designed to extract (*subject*, *predicate*, *object*) relations where the *subject* is the object in a sentence. We extract the data listed in Table 2 using this query for entity-relation extraction from the sentence *The proposal for entry into the register shall be submitted by the operator*.

Technical details RExtractor is conceptualized as a modular framework. It is implemented in Perl programming language and its code and technical details are available on Github:

`http://github.com/VincTheSecond/rextractor`

Each RExtractor component is implemented as a standalone server. The servers regularly check new documents waiting for processing. A document processing progress is characterized by a document processing status in the extraction pipeline, e.g. *520 – Entity detection finished*.

The system is designed to be domain independent. However, to achieve better performance, one would like to adapt the default components for a given domain. Modularity of the system allows adding, modifying or removing functionalities of existing components and creating new components. Each component has a configuration file to enable various settings of document processing.

A scenario with all settings for the whole extraction pipeline (set up in a configuration file) is called an *extraction strategy*. An extraction strategy sets a particular configuration for the extraction pipeline, e.g. paths to language models for NLP tools, paths to DBE and DBQ.

The RExtractor API enables easy integration into more complex systems, like search engines.

3 RExtractor Demonstration

The RExtractor architecture comprises two core components: (a) a background server processing submitted documents, and (b) a Web application to view a dynamic display of submitted document processing.

Web interface enables users to submit documents to be processed by RExtractor. In the submission window, users are asked to select one of the extraction strategies. Users can browse extraction strategies and view their detailed description. After successful document submission, the document waits in a queue to be processed according to the specified extraction strategy. Users can view a display of submitted document processing that is automatically updated, see Figure 4.

In Figure 5, the following information is visualized: (1) **Details** section contains metadata about document processing. (2) **Entities** section shows an

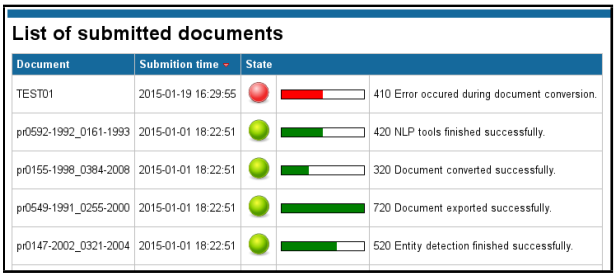

Figure 4: REXtractor web interface, part 1

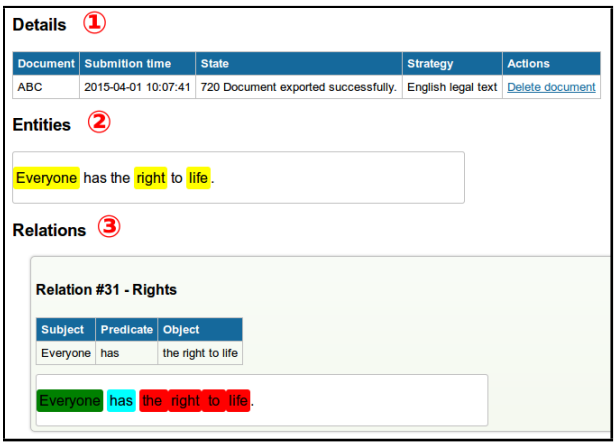

Figure 5: REXtractor web interface, part 2

input document with the highlighted entities that can be viewed interactively: by clicking on the entity, an additional information about the entity is uploaded and presented. (3) **Relations** section consists of tables where *(subject, predicate, object)* triples are listed. In addition, the relevant part of the document with the highlighted triples is presented as well.

Our demonstration enables users to submit texts from legal domain and process them according to two currently available extraction strategies, Czech and English. Once the document processing is finished, users can browse extracted entity-relation triples.

4 REXtractor Online

`http://odcs.xrg.cz/demo-rextractor`

5 Conclusion

We presented the REXtractor system with the following features:

- Our ambition is to provide users with an interactive and user-friendly information extraction system that enables submitting documents and browsing extracted data without spending time with understanding technical details.
- A workflow of REXtractor is language independent. Currently, two extraction strategies are available, for Czech and English. Creating strategies for other languages requires NLP tools, Database of entities (DBE) and Database of queries (DBQ) for a given language.
- A workflow of REXtractor is domain independent. Currently, the domain of legislation is covered. Creating strategies for other domains requires building DBE and DBQ. It is a joint work of domain and NLP experts.
- REXtractor extracts information from syntactic dependency trees. This linguistic structure enables to extract information even from complex sentences. Also, it enables to extract even complex relations.
- REXtractor has both user-friendly interface and API to address large-scale tasks. The system has already processed a collection of Czech legal documents consisting of almost 10,000 documents.
- REXtractor is an open source system but some language models used by NLP tools can be applied under a special license.

Our future plans concern the following tasks:

- experimenting with syntactic parsing procedures in the NLP component that are of a crucial importance for extraction
- evaluating REXtractor against the data that are available for various shared tasks and conferences on information retrieval, e.g. TAC[4], TRAC[5]

[4] `http://www.nist.gov/tac/`
[5] `http://trec.nist.gov/`

- making tree query design more user-friendly for domain experts
- getting feedback from customers
- incorporating automatic procedures for extraction of both entities and relations that are not pre-specified in Database of Entities and Database of Queries, resp.
- creating strategies for other languages and other domains

Through this system demonstration we hope to receive feedback on the general approach, explore its application to other domains and languages, and attract new users and possibly developers.

Acknowledgments

We gratefully acknowledge support from the Technology Agency of the Czech Republic (grant no. TA02010182), The Bernard Bolzano Foundation and SVV project no. 260 224. This work has been using language resources developed and/or stored and/or distributed by the LINDAT/CLARIN project. We highly appreciate RExtractor-related discussions with Martin Nečaský and colleagues from Sysnet, Ltd.

References

Alan Akbik, Oresti Konomi, and Michail Melnikov. 2013. Propminer: A workflow for interactive information extraction and exploration using dependency trees. In *Proceedings of the 51st Annual Meeting of the ACL: System Demonstrations*, pages 157–162. ACL.

Laura Chiticariu, Yunyao Li, and Frederick R. Reiss. 2013. Rule-based information extraction is dead! long live rule-based information extraction systems! In *EMNLP*, pages 827–832. ACL.

Nicolai Erbs, Bispo Pedro Santos, Iryna Gurevych, and Torsten Zesch. 2014. Dkpro keyphrases: Flexible and reusable keyphrase extraction experiments. In *Proceedings of 52nd Annual Meeting of the ACL: System Demonstrations*, pages 31–36. ACL.

John Gantz and David Reinsel. 2010. The digital universe decade – Are you ready?

Vincent Kríž, Barbora Hladká, Martin Nečaský, and Tomáš Knap. 2014. Data extraction using NLP techniques and its transformation to linked data. In *Human-Inspired Computing and Its Applications - 13th Mexican International Conference on Artificial Intelligence, MICAI 2014, Tuxtla Gutiérrez, Mexico, November 16-22, 2014. Proceedings, Part I*, pages 113–124.

Mausam, Michael Schmitz, Robert Bart, Stephen Soderland, and Oren Etzioni. 2012. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 523–534, Stroudsburg, PA, USA. ACL.

Paul McNamee, James Mayfield, Tim Finin, Tim Oates, Dawn Lawrie, Tan Xu, and Douglas Oard. 2013. Kelvin: a tool for automated knowledge base construction. In *Proceedings of the 2013 NAACL HLT Demonstration Session*, pages 32–35. ACL.

Petr Pajas and Jan Štěpánek. 2009. System for querying syntactically annotated corpora. In Gary Lee and Sabine Schulte im Walde, editors, *Proceedings of the ACL-IJCNLP 2009 Software Demonstrations*, pages 33–36, Suntec, Singapore. Association for Computational Linguistics.

Benjamin Roth, Tassilo Barth, Grzegorz Chrupała, Martin Gropp, and Dietrich Klakow. 2014. Relationfactory: A fast, modular and effective system for knowledge base population. In *Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the ACL*, pages 89–92, Gothenburg, Sweden, April. ACL.