RegMiner: Taming the Complexity of Regulatory Documents for Digitalized Compliance Management

Karolin Winter[1], Manuel Gall[1], Stefanie Rinderle-Ma[1,2]

[1]Research Group Workflow Systems and Technology, Faculty of Computer Science,
[2]Data Science@Uni Vienna, University of Vienna, Vienna, Austria
{firstname.lastname}@univie.ac.at

Abstract. Business process compliance has become a crucial aspect for companies due to severe fines that can be imposed if constraints and rules emerging from regulatory documents are violated. Regulatory documents are often written in natural language and analyzing them is mainly done manually since only limited tool support is available. Therefore, we present *RegMiner*, a web service for discovering and visualizing constraints from regulatory documents. By employing NLP and data mining techniques, compliance constraints can be automatically extracted, grouped, and visualized leading to a separation of relevant and non-relevant document parts and insights into, e.g., duties of stakeholders. A case study based on a current document from the European parliament regarding the financial domain demonstrates RegMiner's maturity.

Keywords: Business Process Compliance, Natural Language Processing, Regulatory Documents

1 Introduction and Significance for the BPM Field

Analyzing regulatory documents as well as assessing the compliance of business processes with regulations poses a tremendous challenge on companies and is still mostly done manually though the amount of regulatory documents is constantly increasing. Compliance violations, in turn, can cause severe problems, e.g., more than 20 million euros in case of violating the General Data Protection Regulation (GDPR)[1]. Hence, providing support for digitalized compliance management is crucial. Yet, only few tools for extracting, analyzing and visualizing compliance constraints have been proposed. Most tools rather focus on a one or bidirectional mapping of natural language texts to imperative or declarative process models [2,3,5]. Hence, we propose *RegMiner*, a web service for extracting, processing and visualizing constraints from regulatory documents, based on previous publications [7,9]. Section 2 describes the innovations and architecture

[1] https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32016R0679

of RegMiner, Sect. 3 elaborates on a case study based on recent document on *macro-financial assistance* provided by the European parliament and council as well as an outlook on future work. Potential users of RegMiner are legal and compliance experts as well as researchers.

2 Innovation and Architecture

RegMiner can be accessed via

`http://regminer.wst.univie.ac.at/`

A tutorial as well as example data sets are available at

`http://gruppe.wst.univie.ac.at/projects/RegMiner/index.php?t=prototypes`

A screencast is available at

`http://gruppe.wst.univie.ac.at/~gallm6/RegMiner/Video/`

RegMiner is based on the command line prototypes of previous publications [7,9] and consist of a three tier architecture as depicted in Fig. 1. By offering RegMiner as web service we aim at providing a low-threshold entry for analyzing regulatory documents in an automated way for non-technical experts. But also technical experts and researches will benefit since RegMiner can be quickly tested as no installation of libraries is necessary. RegMiner's visualization allows for easily separating relevant from non-relevant document parts as well as gaining insights on, e.g., duties of stakeholders in an aggregated form.

2.1 Presentation Tier

The presentation tier consists of a single page application based on HTML markup and JavaScript components. In particular, we use the Bootstrap toolkit[2] and d3.js[3]. The following information must be provided by the user.

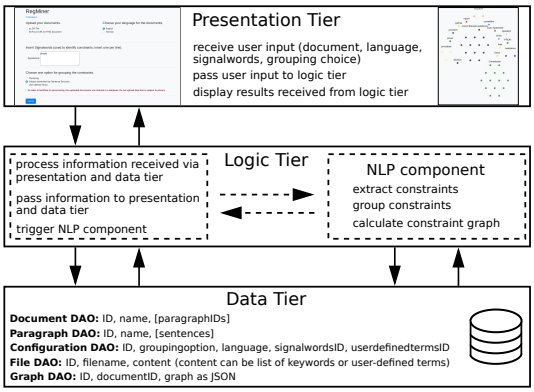

Fig. 1: RegMiner – Architecture

Document. The input can be provided in two ways. Either as ZIP file which can contain a) one document, b) a set of documents, or c) a set of paragraphs. Option c) represents a partitioning of one or several documents, e.g., based on the document structure into its sections (cf. [9]). Though this is not mandatory, the partitioning into paragraphs facilitates the visual inspection of results afterwards. In addition, this way, a restriction to a selection of document parts can be enforced by the user.

[2] `https://getbootstrap.com/`
[3] `https://d3js.org/`

As a second option, we decided to integrate support for the EUR-Lex platform[4] which contains an extensive collection of legal documents such as the General Data Protection Legislation affecting stakeholder of various domains. The user can provide an URL referencing the HTML markup of the desired EUR-Lex document. The document is downloaded and automatically split into sections based on HTML tags and attributes.

Language. English and German are supported as document languages.

Signal Words. At least one signal word to identify constraints, e.g., "shall", "should" or "must" (cf. [7]), has to be selected.

Grouping Options. Three options for grouping constraints are available

Clustering Constraints are clustered based on term frequency using k-means++; in this case the user needs to specify the number of clusters.

Subject determined by Sentence Structure The subject of each constraint is identified based on NLP tags determined by a NLP parser. Per subject one group is created.[5] This option is fully automatic, i.e., requires no further user input.

User-defined Terms Constraints are grouped based on terms defined by the user which need to be uploaded as .txt file. Each term defines one group, e.g., if "authority" and "user" are contained in the .txt file, all constraints containing "authority" are assigned to one group, all constraints containing "user" are assigned to another group. If neither of them is present, the constraint is shifted to group "undefined", if both of them are present, the constraint will be contained in the "authority" as well as "user" group.

On submission, the information is passed to the logic tier. As soon as the results were retrieved, a graph consisting of several dots accumulated in clusters is displayed. Each dot represents one constraint and the color indicates the corresponding cluster. By hovering over the dots the constraint is shown. By double-clicking onto a dot the paragraph containing the constraint is displayed.

2.2 Logic Tier

The logic tier is written in Python 3 and consists of two components. The first one processes incoming and outgoing requests from and to the presentation tier, i.e., delivering the single page application to the client's browser, handling user input and returning constraint graphs using the JSON file format and is based on the web application framework Flask[6]. The second component, called NLP component, conducts the actual constraint extraction and grouping procedure. It is based on the NLP framework SpaCy [4] for linguistic analysis and scikit-learn [6] for cluster analysis. Choosing SpaCy rests on three pillars, i.e., its accuracy in determining NLP tags, its built-in similarity function relying on pre-trained word vectors and its speed (cf. [1] for a detailed comparison of NLP parsers). The information retrieved from the presentation tier as well as the result returned by the NLP component are passed to the data tier.

[4] `https://eur-lex.europa.eu/homepage.html?locale=en`

[5] For further details see [7].

[6] `https://palletsprojects.com/p/flask/`

2.3 Data Tier

For storing and retrieving data RegMiner uses the MongoDB[7] database. Each document uploaded by the user, consists of a name and a list referencing the corresponding paragraphs, stored in separate files. Each paragraph consists of a name and content, i.e., the sentences within the paragraph. Signal words as well as the user-defined terms are stored in separate files. Furthermore, a configuration file holding the grouping choice, the language and a reference to the signal words resp. user-defined terms file is created. Information stored within the database becomes identifiable via a unique hash-value. This has two benefits. First, if a document having the same hash value already exists, it is not stored in the database multiple times. Secondly, as the results of the NLP component are also stored in the database, recomputing can be avoided. This enhances the usability as results can be displayed immediately. The latter is important when analyzing extensive documents, which can take up to several minutes.

3 Maturity

The underlying concepts of the NLP component have proven to yield valuable insights into regulatory documents from various domains, such as security, the medical domain, financial domain and generally applicable regulatory documents like the GDPR [7,8,9]. Within this paper we demonstrate how to gain insights of regulatory documents that just came into effect. In such a situation a user would not be able to carry out a purposive search for specific terms without having read the document. RegMiner's benefit is that a user must not have previous knowledge about a regulatory document but can still gain insights into a regulatory document quickly at low effort. For the case study, we selected the *macro-financial assistance to enlargement and neighbourhood partners in the context of the COVID-19 pandemic*[8]. RegMiner took around 8 seconds to discover the graph depicted in Fig. 2 using "shall", "should" and "must" as signal words and sentence structure as grouping option. It can be recognized at first glance that the terms *macro-financial*

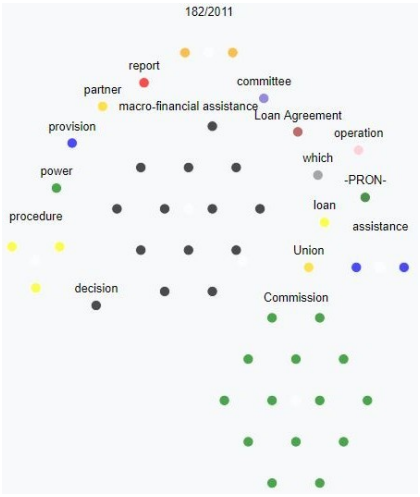

Fig. 2: RegMiner – Result for Case Study on Macro-financial Assistance

[7] https://www.mongodb.com/

[8] https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32020D0701&qid=1591613404669&from=DE

assistance and *Commission* form the largest groups (black and green). By further examining the underlying constraints by hovering over the dots, the actions and duties for the commission are immediately apparent and can be easily summarized. A user interested under which conditions *macro-financial assistance* applies or which aims it has, can directly gain this information based on the graph. If a user needs the context of the constraint, the corresponding paragraph can be displayed by double-clicking onto the constraint. Thereby, a user can directly retrieve the right position within the document. The ability to retrieve such a graph out-of-the-box for an arbitrary and recent document proves the applicability of RegMiner. Finally, RegMiner is tested and discussed with stakeholders from the financial domain through transfer project RegMiner[9].

As future work we plan to improve the pre-processing of documents, provide support for document formats such das PDF, integrate further visualization concepts, especially for complex and extensive documents and investigate domain specific language models as well as the integration of ontologies.

Acknowledgment

This work has been funded by the Vienna Science and Technology Fund (WWTF) through project NXT19-003.

References

1. Al Omran, F.N.A., Treude, C.: Choosing an NLP library for analyzing software documentation: a systematic literature review and a series of experiments. In: Mining Software Repositories. pp. 187–197 (2017)
2. Delicado Alcántara, L., Sánchez Ferreres, J., Carmona Vargas, J., Padró, L.: Nlp4bpm: Natural language processing tools for business process management. In: BPM Demo and Industrial Track 2017 Proceedings. pp. 1–5 (2017)
3. Freytag, T., Allgaier, P.: Woped goes nlp: Conversion between workflow nets and natural language. In: BPM (Dissertation/Demos/Industry). pp. 101–105 (2018)
4. Honnibal, M., Montani, I.: spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing (2017)
5. López, H.A., Debois, S., Hildebrandt, T.T., Marquard, M.: The process highlighter: From texts to declarative processes and back. BPM (Dissertation/Demos/Industry) **2196**, 66–70 (2018)
6. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
7. Winter, K., Rinderle-Ma, S.: Detecting constraints and their relations from regulatory documents using NLP techniques. In: CoopIS. pp. 261–278 (2018)
8. Winter, K., Rinderle-Ma, S.: Untangling the GDPR using conrelminer. Tech. rep. (2018), `http://arxiv.org/abs/1811.03399`
9. Winter, K., Rinderle-Ma, S., Grossmann, W., Feinerer, I., Ma, Z.: Characterizing regulatory documents and guidelines based on text mining. In: CoopIS. pp. 3–20 (2017)

[9] `https://www.wwtf.at/programmes/new_exciting_transfer_projects/NXT19-003`