Combining Semantic Web and Machine Learning for Auditable Legal Key Element Extraction

Anna Breit[1], Laura Waltersdorfer[2], Fajar J. Ekaputra[3,2], Sotirios Karampatakis[1], Tomasz Miksa[2], and Gregor Käfer[2]

[1] Semantic Web Company, Vienna, Austria first.last@semantic-web.com
[2] TU Wien, Vienna, Austria first.last@tuwien.ac.at
[3] Vienna University of Economics and Business, Vienna, Austria first.last@wu.ac.at

Abstract. Based on a real world use case, we developed and evaluated a hybrid AI system that aims to extract key elements from legal permits by combining methods from the Semantic Web and Machine Learning. Specifically, we modelled the available background knowledge in a custom Knowledge Graph, which we exploited together with the usage of different language- and text-embedding-models in order to extract different information from official Austrian permits, including the Issuing Authority, the Operator of the facility in question, the Reference Number, and the Issuing Date. Additionally, we implemented mechanisms to capture automatically auditable traces of the system to ensure the transparency of the processes. Our quantitative evaluation showed overall promising results, while the in-depth qualitative analysis revealed concrete error types, providing guidance on how to improve the current prototype.

Keywords: legal permits · information extraction · semantic web · machine learning · auditability

1 Introduction

Given its manifold and distributed nature combined with a large number of associated exceptions and exemptions, the legal domain can be considered one of the most complex areas. In addition, most of the knowledge is stored (e.g., in laws) and distributed (e.g., in permits) in unstructured, textual form, that makes use of highly convoluted and composite language. In order to facilitate the searchability and processability of such legal documents, specific *key elements* – representing the most important actors and aspects of the document – are added as metadata. If not added at creation time of the document, these key elements need to be extracted manually in a later point in time. Due to the high cost of jurists, this extraction is often performed by laypersons for whom this task can be demanding and resource-intense.

In this work, we introduce a system that aims at assisting laypersons in their task of extracting key elements from official permits. For doing so, we collect requirements from and perform an evaluation along a real-world use case situated

in Austria. Specifically, we build a hybrid AI system, which combines methods from Machine Learning and Semantic Web technologies to provide suggestions for specific key elements, while providing auditable traces of the extraction procedure, which increase the transparency of the system.

The rest of the paper is structured as follows. Section 2 describes the use case and the associated requirements in more detail, while Section 3 introduces the developed system. Section 4 explains methods and evaluation setup, results of our quantitative and qualitative analysis are shown in Section 5. Finally, we report our conclusions and future work in Section 6.

2 The Use Case

The use case concerns the operation of an electronic permit management system (EPMS) which facilitates the organisational and bureaucratic processes around official permits in Austria, including application, decision, and amendments, as it provides a common platform for all involved stakeholders. Alongside the digitized version of the permit document, the EPMS also provides a summary of *key elements* characterizing the most important aspects of the permit.

The task of filling these structured summaries is currently conducted by data management staff. However, as they are usually not specifically trained for this highly complex exercise, the task completion requires a lot of efforts and can end in poor data quality. Therefore, our goal is to provide a system that supports the administrators for extracting information by providing suggestions for the key elements that need to be extracted.

Specifically, for this use case, we are focusing on the following key elements (cf. Figure 1): (1) the *Operator* of the installation a permit is targeting, (this could either be a legal or natural person), (2) the *Issuing Authority* in charge of the content of the permit, (3) the *Reference Number* of the permit, being its unique identifier, and (4) the *Issuing Date*. Along these key elements, which are mentioned in the permit content, we are further interested in extracting additional meta-information concerning the *Permit Types* including (5) the *Object Type* describing whom the permit is for (e.g., for an Installation Site), (6) the *Processing Type* describing the type of request and outcome of the permit (e.g., application, amendment, withdrawal), and (7) the *Procedure Type* describing the legal procedure under which the permit was issued (e.g., simplified procedure).

Requirements: Different requirements arise from the presented use case. First, the extracted key elements should be matched to pre-defined entities to ensure data quality. Second, the system must be able incorporate symbolic expert knowledge. Specifically, expert-created mappings of legal regulations to the different Permit Types should be exploitable by the system. Finally, to ensure the transparency of the provided suggestions and thus the acceptance of the users, auditability capabilities to conduct regular internal audits for error detection of past system executions should be included in the system. To this end the following audit requirements should be fulfilled: (A1) availability of audit traces logging complete system executions, (A2) automation of audit trace collection,

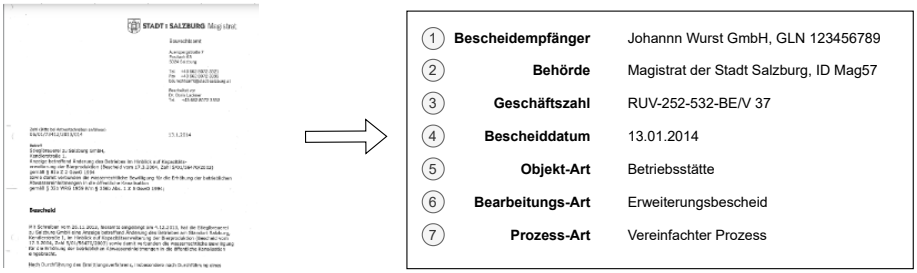


Fig. 1: Example of a structured summary of permit key elements.

transformation and management and (A3) the capability and ease of use to ask and answer to audit questions based on stakeholder input.

3 The System

The implemented solution consists of three main components, being (1) a knowledge graph (KG) containing the background knowledge and corresponding entities, (2) the Key Element Extraction Module, comprising of a main pipeline that orchestrates a set of extraction services, and (3) the AuditBox, responsible for collecting and providing audit traces of the system. To facilitate the accessibility of the developed system, a simple web interface was added through which user interaction can take place. An overview over the system is shown in Figure 2.

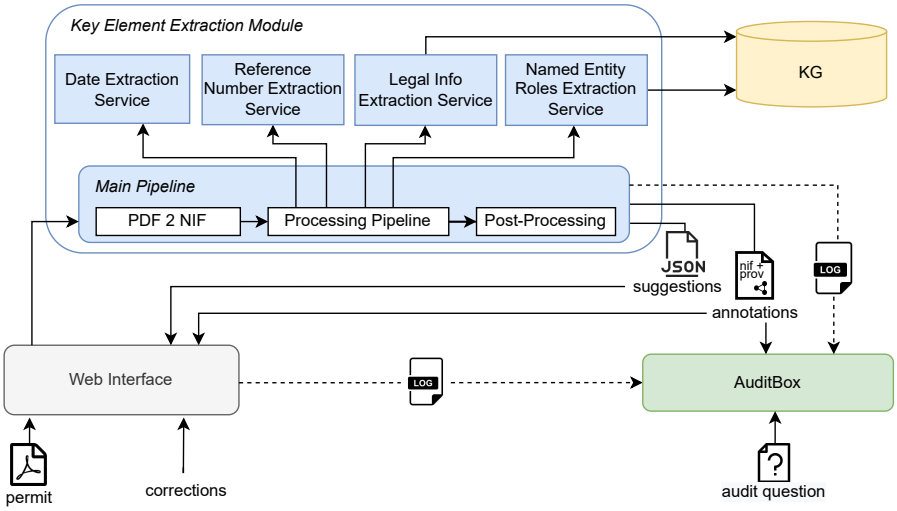


Fig. 2: Overview over the developed Auditable Key Element Extraction System

3.1 Knowledge Graph

We developed a knowledge graph (KG) as a central component to store and provide information both about the involved entities (e.g., legal persons) as well as background knowledge about these entities (e.g., associations of legal regulations to Permit Types) within legal permits.

To build such a KG, we first identified relevant datasets. For this use case, we collected (i) a *geo-location dataset* structured according to Geonames ontology[4], (ii) a *legal person dataset*, representing our operator entities, (iii) *authorities dataset*, (iv) *regulations dataset* structuring relevant Austrian law, and (v) the *Permit Types dataset* mapping the regulations to the different types. Afterwards, we develop an ontology which described the data model of the data from these datasets and the links between these data (e.g., the location of a legal person from legal person databases should be in the location registered in the geo-location dataset). The ontology is summarized in Figure 3 and online available[5]. With the ontology in place, we transformed the source data into its KG representation, by integrating the transformed data from individual data sources in a triplestore. Finally, we enriched the generated KG with additional information, e.g., SKOS hierarchies based on the inputs from domain experts.

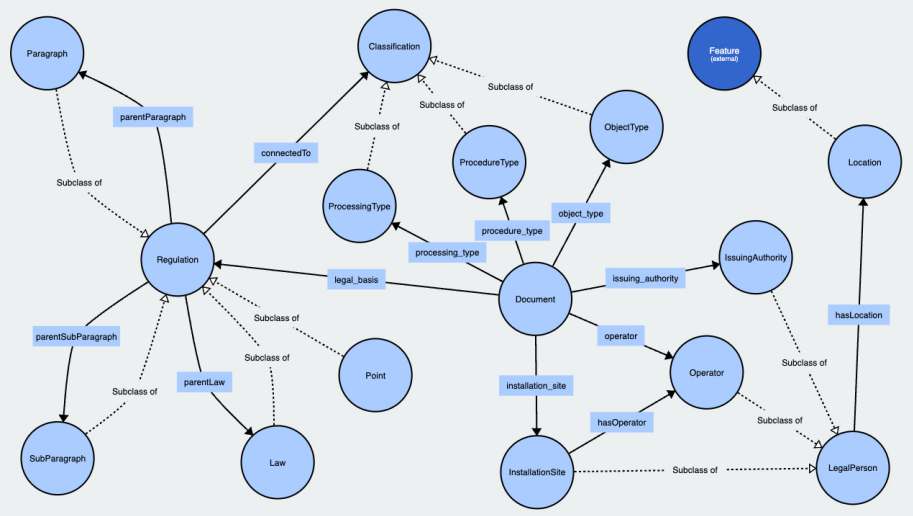


Fig. 3: Overview of the ontology classes for legal permit extraction. We modelled the class Location as a subclass of GeoNames Feature.

[4] http://www.geonames.org/ontology/documentation.html
[5] The link to the online version will be made available upon acceptance

3.2 Key Element Extraction Module

The Key Element Extraction Module of the developed system follows the paradigm of a microservice architecture, where the module is structured as a collection of decoupled services. Each microservice can be developed and deployed independently, providing flexibility on using different programming languages for their implementation. Microservices are by design minimal and autonomous, in contrast to monolithic integrated systems, thus can be developed more efficiently [12]. Furthermore, deployment can be automated to a large extent, making the solution easy to scale based on demand.

Main Pipeline The use of a microservice architecture mandates the use of a management tool in order to orchestrate the execution of the different services and steps needed to accomplish more complex tasks. To this end, we used UnifiedViews ETL[6][6] – an Extract-Transform-Load (ETL) framework and platform that allows users to define, execute, monitor, debug, schedule, and share data processing tasks as a pipeline – as it has the advantage of natively supporting the processing of RDF data. There, we defined a main pipeline which orchestrates the processing step needed to perform 1) the parsing and pre-processing of the legal document, 2) the execution of the extraction services, and 3) a number of post-processing steps needed to produce the final outcome of the main pipeline. Additionally, for each extraction service, we define a secondary annotation pipeline, in order to decouple configuration needs of each service from the main pipeline and enable parallel execution of the services. Fault tolerance is achieved through this design, as in case an error occurs on any of the extraction services, the main pipeline will continue to process.

By design, all extraction services use the NIF ontology as I/O format (see below). We developed a Data Processing Unit (DPU), a pluggable processing component for UnifiedViews, with the task to parse a document in PDF format and convert it into NIF. The DPU uses the DKPro-core Library [2] to achieve the conversion. Then, the NIF document is sent to next DPU in the pipeline, which triggers the secondary pipeline for each of the extraction services. Upon completion of all secondary pipelines, a set of post-processing steps is executed. First, the results are merged, then we filter the annotation units produced by the annotation services based on a confidence threshold. Finally, we serialize the results in JSON, by utilizing JSON-LD framing [9]. Thus the final results can be presented on the user interface of the system.

The NLP Interchange Format The NLP Interchange Format (NIF) [5] is an RDF/OWL-based format that aims to achieve interoperability between Natural Language Processing (NLP) tools, language resources and annotations. The NIF 2.0 Core Ontology[7] provides classes and properties to describe the relations

[6] https://www.poolparty.biz/agile-data-integration/
[7] http://persistence.uni-leipzig.org/nlp2rdf/ontologies/nif-core#

between substrings, text, documents by assigning URIs to strings. These URIs can then be used as subjects in RDF triples and therefore enable easy annotation.

In the course of the presented use case, NIF 2.1[8] is used as the base data model for all annotation services. Each service should be able to parse NIF data as input for process. Additionally, output of each annotation service is expected as NIF. Thus, interoperability between all annotation services of the developed system is achieved through the NIF data model. Through the extensive usage of nif:AnnotationUnit, results from each service can be merged effortless to produce the final results of the annotation pipelines. Moreover, the use of provenance metadata on each of nif:AnnotationUnit ensures the auditability of the results produced by the annotation pipelines. An example of the output of an annotation service is shown in Listing 1.1.

```
@prefix nif:
    <http://persistence.uni-leipzig.org/nlp2rdf/ontologies/nif-core#> .
@prefix alkees-nif: <https://alkees.org/ontology/nif-alkees#> .
@prefix its-rdf: <http://www.w3.org/2005/11/its/rdf#> .
@prefix alkees-permit: <https://w3id.org/alkees/ns/permit#> .
@prefix alkees-authority: <http://w3id.org/alkees/id/authority/> .
@prefix alkees-service: <http://alkees.org/ns/service/>

<file:///data/CONTENT17_71708645.pdf#offset_0_9672>
  a nif:Context, nif:OffsetBasedString, nif:String ;
  nif:beginIndex 0;
  nif:endIndex 9672;
  nif:isString "Lots of text...";
  alkees-nif:annotations
      <file:///data/CONTENT17_71708645.pdf#offset_177_223> .

<file:///data/CONTENT17_71708645.pdf#offset_177_223>
  a alkees-nif:MatchedResourceOccurrence, nif:Annotation,
      nif:OffsetBasedString;
  nif:beginIndex 177;
  nif:endIndex 223;
  nif:referenceContext <file:///data/CONTENT17_71708645.pdf#offset_0_9672> ;
  nif:anchorOf "Bezirkshauptmannschaft Baden";
  nif:annotationUnit [ a nif:AnnotationUnit;
                       its-rdf:taAnnotatorRef
                            alkees-service:pp-concept-extraction-annotator-v1;
                       its-rdf:taClassRef alkees-permit:ConceptAnnotation;
                       its-rdf:taConfidence 1.0E0;
                       its-rdf:taIdentRef alkees-authority:308 .
                     ] .
```

Listing 1.1: Example NIF output of an annotation service.

Extraction Services In total, we developed four microservices to extract the five types of key elements, which we describe below in more detail. The communication of the services is based on well defined REST interfaces, simplifying the communication and allowing decoupling the client from the server.

Date Extraction: There is a variety of tools and libraries available that target the recognition and parsing of dates from textual data. After analyzing a

[8] Please note, that version 2.1 has not yet officially been released yet, but is the latest develop branch of the ontology

selection of these libraries, (including *dateutil*[9], *dateparser*[10], and *datefinder*[11]) we decided for heideltime[12] as it provided the best range of functions and performance for the intended use case. This Java tool developed by Heidelberg University supports a wide variety of languages and date formats, captures the type of annotation (full date, relative date, ...), and directly annotates these mentions within the text. This gives the advantage of being able to easily filter out irrelevant dates, e.g., for our use case, where the goal is to extract the issuing date of the legal permit, we would only take into consideration full dates.

Reference Number Extraction For extracting the reference numbers, we chose a two step-approach, being (1) candidate identification and (2) classification.

First, candidates are identified based on a set of RegEx patterns. Specifically, for the presented use case, we used a recall-optimized pattern to collect candidates, which we further refined by automatically filtering out groups of false positives such as all-caps words, or gendered terms (containing a "Binnen-I") with corresponding patterns to improve the quality.

In the second step, the reference number candidates are ingested into a string classifier. We decided for a 1-dimensional-CNN-based architecture to generate character embeddings for the candidates, which are then further fed into a binary classification layer.

Legal Info Extraction For mapping the legal documents to the Permit Types, we chose a two-step approach: (1) we extracted and normalized the mentions of laws from the text, and then (2) mapped these mentions to the specific types.

Unfortunately, existing legal annotation tools (e.g., [14]) did not show sufficient quality in preliminary experiments. Therefore, we decided to develop our own reference parsing tool based on a context-free grammar. The basis for the grammar is different legislature elements such as article, paragraph, subparagraph, point, and sentence. These elements were organized in a transitive hierarchical way, so that different levels can be skipped to still form a valid overall mention. We also included known separators including commas, but also phrases such as "as well as", or "in combination with".

To automatically map the annotated and normalized legislature mentions to the different Permit Types, we queried the corresponding information provided in the KG.

Named Entity Role Extraction We chose a two-step approach to extract the Operator and Issuing Authority from a legal permit: (1) we annotated all known entities in the text as candidates, and then (2) classified the candidates whether or not they appear in the role of interest.

[9] https://github.com/dateutil/dateutil
[10] https://github.com/scrapinghub/dateparser
[11] https://github.com/akoumjian/datefinder
[12] https://github.com/HeidelTime/heideltime

In order to annotate the entity candidates, we used PoolParty Extractor[13] to identify all mentions of concepts from the relevant sub-branch of the taxonomy from our KG. For being able to also identify surface forms that are unknown to the KG, we further deployed a BERT-based Named Entity Recognition (NER) model (`bert-base-german-cased` model with token classification head) which we fine-tuned on a German Legal NER dataset [7]. The candidates are then matched to their corresponding entity in the KG, using a fuzzy-string matching algorithm based on n-gram tf-idf-scores, with a empirically determined negative dot product cutoff of -0.8.

The extracted candidates are then used as fine-tuning examples for a disambiguation classifier, which decides whether the concept is used in the target role or not. The classifier consists of a `bert-base-german-cased` model with a binary classification layer. Specifically, the task of role disambiguation is re-formulated in as Target Sense Verification task [1], so that –given a context containing the target entity, as well as a label and definition of the target role– the task is to decide whether the entity in the context is used in the target role or not. This task formulation leads to flexibility regarding the target roles, but also allows to show ambiguities, e.g., when an entity embodies multiple roles in one sentence.

Finally, to overcome the problem of deprecated names used in the permits due to e.g., re-branding of companies, we automatically extracted the installation sites from the permits using RegEx queries parsing the real-estate numbers and used them to retrieve corresponding operators by querying the KG.

3.3 AuditBox

AuditBox offers a flexible and adaptable implementation to collect and transform audit traces from heterogeneous sources into a unified representation. This representation is built upon a workflow model in RDF-format which defines main activities with relevant in- and outputs (e.g. file uploads, suggestions for key elements etc.) and their data sources (e.g. extraction services, user interface). For our use case, traces are collected for each system execution ranging from permit upload to users, permit transformation, information extraction and users being able to correct extracted key elements.

Core components of the AuditBox include:

Audit Collection supports the automatic generation of endpoints (APIs) where traces from different sources and applications can be sent to. AuditBox collectors are REST APIs allowing a flexible and standardised coupling of services. A custom service ontology aides their automatic generation.

Audit Transformation: is responsible for the integration of the heterogeneous audit traces. A set of RML mappings is used to transform data from different sources into RDF format aligned to the workflow model. The transformed data is stored in a graph-based repository, for which we use GraphDB.

Audit Management supports the querying of transformed data from the repository and provides pre-defined queries for users without SPARQL expertise.

[13] https://www.poolparty.biz/poolparty-extractor

Non-functional (security) capabilities include: User authentication and authorization to ensure that only authorized entities can send audit traces and query the stored data.

4 Methods

4.1 Evaluation Setup of Extraction Performance

The basis for this use case were 4612 historic permits that had been entered and annotated in the EPMS system. For evaluation purposes, we reserved 777 permits which were not used in the training of the models. For each of these permits, we performed the extraction of the relevant key elements, using the meta-data from EPMS as silver labels to compare against. As all services – with exception to the Legal Info Extraction – provide a confidence-sorted list of the extracted entities, we chose hits@k as an appropriate evaluation metric, i.e., the proportion of times the correct instance was within the top k ranked predictions. We chose k to be in $\{1, 3, all\}$.

For the extracted Reference Number, we decided to perform a two evaluation settings. a While the *strict* setting required an exact string match, the *lenient* setting allowed sub-string matches. Finally, for errors that occur in services with a two-step approach, we analyse whether the correct annotation was in the candidates or not, indicating if the error occurred in the candidate annotation step, or in the classification step.

4.2 Training of ML models

While the Date and Legal Info Extraction services only require configuration, the models behind Reference Number and Named Entity Roles Extraction need to go through a training phase before they can be applied to the presented use case. As the legal permits provided for training only consisted of the documents themselves and their associated labels, but no annotations, we created the training data in an distant supervised fashion, where we used the corresponding candidate creation modules of the services to create the instances, and annotated all instances that target the correct entity as true, while others as false. Finally, we balanced the created instance sets to contain an equal number of positive and negative examples, and split into training and validation. It shall be noted that the outcomes of this procedure can contain noisy instances, as a context containing the target might be using this entity in a completely different role, but still serve as a positive example. However, we assumed that these noisy instances are mitigated during the training process.

For the CNN-based classifier, we were able to generate over 30.000 training instances. After 10 epochs of training, the model was able to reach an accuracy of 98%, and F_1 score of 92% (precision of 94%, recall of 90%) on the validation set. For the BERT-based classifier, due to hardware restrictions (Intel Xeon CPU E5-2640, 6x, 2.5GHz) and the size of the model, we only used a training-set of

2000 instances and a validation set of 500 instances to optimize the model. After 10 epochs, the model achieved an accuracy and F_1 of 94%, (precision of 91%, recall of 97%) on the validation data.

4.3 Evaluation Setup of Auditability

Several methods were proposed to scope the context information to be collected for a given system and environment: A generic provenance-focused methodology e.g., [10], more specialised approaches for AI-supported systems, such as from an accountability perspective e.g. [13], or artefact-focused e.g., [15]. However, they do not focus on automatic collection and management of such context information. To this end, we adapted and refined the method presented in [4]. The three main phases are (cf. Figure 4):

During the *Scoping Phase*, goal and aims of the audit are defined to identify main activities and entities to be included in the workflow. In the *Preparation Phase*, the workflow model and mappings to transform collected traces are created, to support the automatic management of audit traces. These two outputs are also needed to orchestrate the AuditBox. The *Execution Phase* is concerned with the execution of audits: audit traces are automatically collected and sent to the generated endpoints by the AuditBox. The traces are then transformed according to the mappings and stored in a graph-based format. Audit questions can be answered in the form of SPARQL queries prepared in the AuditBox or also by writing custom ones.

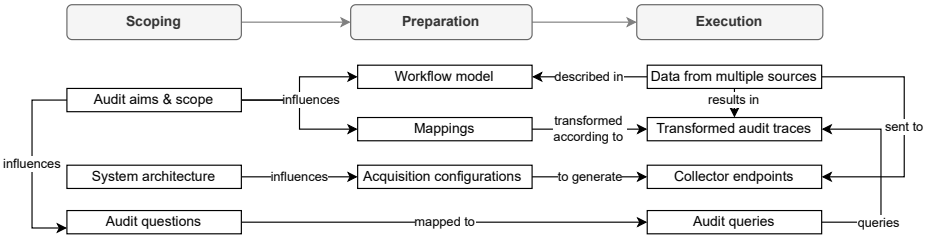


Fig. 4: Auditability method

Such audit questions (and the requirements from Section 2) form the main basis for our evaluation. Similar to the scenario-based and question-driven approaches to achieve explainable AI [3, 8], we collected a set of audit questions (similar to competency questions) from relevant stakeholders involved in the operation of the EPMS to guide the audit scope. The structure of such a question:

As a <stakeholder>, I want <some goal>so that <some reason>.

One concrete example:

As a technical support staff, I want to know which suggestions of metadata has been corrected by a user so that either the system can ideally learn from errors or areas for enhancement can be detected.

5 Results

5.1 Quantitative Analysis

The results from the quantitative analysis are summarized in Table 1. The extraction of the Issuing Date is the most successful across all key elements and yielded in 79% of the cases the correct result on the first rank of the suggestions, while the consideration of later ranks would not bring too much benefit to the overall performance. In contrast, for other ranked key elements, a longer list of suggestions would result in an increased recall of the correct entity.

The Reference Number Extraction Service, despite the good performance on the training examples, was only able to achieve hits@all of 0.76 for the lenient and 0.55 for the strict setting. The performance difference in the two evaluation settings is quite large, with a total of 165 cases where the extracted reference number was only correct under the lenient scenario, indicating that the boundary detection capabilities of the Reference Number Extraction Service can still be improved. Another interesting insight is that if only incorrect reference numbers are suggested, the probability that the correct one is annotated as a candidate is much higher than compared to cases where no suggestions were provided.

For the key elements originating from the NE-Roles service, i.e., Issuing Authority and Operator, we reach a performance of 0.62 and 0.77 hits@all rate, respectively. Interestingly, the ranking for the extracted Operator suggestions seems to be significantly worse than for the Issuing Authority, as the comparison between the corresponding hits@3 and hits@all rates show. This could be due to the fact that the number of operator candidates contained in a permit is considerably larger than the number of authorities, therefore, the possibility for false positives is increased. For both Issuing Authority and Operator, the vast majority of errors can be traced back to the failed annotation of the corresponding candidate entities in the permit.

When analysing the extraction performance of the installation site in isolation, we can see that the overall performance, while staying below 60%, is considerably high in precision, as only 2% of the suggestions only contained incorrect entities, and if the correct entity was with the suggestions, it was within the first three ranks. However, in 40% of the cases, no suggestion for the installation site was provided, indicating that the identification and parsing of the real-estate number alone is not sufficient to reliably extract this kind of information.

The coverage of Permit Types that we were able to achieve with the tested strategy of extracting and mapping law mentions was considerably low with 23%-39% for hits@all rate for the different types. In a large amount of cases, no suggestion for the different types could be extracted. The model has produced a maximum of 3 suggestions for Object Type and Processing Type, and a maximum of 6 suggestions for Procedure Type across all test permits. Interestingly, the error rate, i.e., when only incorrect types were suggested, varies a lot among the different types: 11% for Object Type to 30% for Processing Type. Concluding, it can be said that the hypothesis that the extraction of Permit Types could be achieved by solely the parsing of law mentions could not be verified.

Table 1: Extraction performance for the different key elements. For Reference Number, we report both the performance for strict (extract string match) as well as lenient (sub-string) evaluation. Incorrect denotes the percentage of cases, where only incorrect suggestions were provided, while Nothing denotes the cases where no suggestions were provided. For those key elements that are extracted in a two-step approach, we report the percentage of cases where the correct entity was not contained in the candidates in parenthesis.

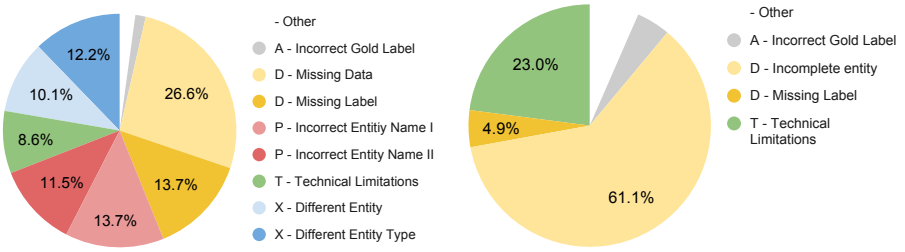
Key Element	hits@1	hits@3	hits@all	Incorrect	Nothing
Issuing Date	0.79	0.82	0.82	0.18	0.00
Ref. Number (strict)	0.41	0.49	0.56	-	-
Ref. Number (lenient)	0.56	0.67	0.77	0.10 (0.09)	0.12 (0.07)
Issuing Authority	0.51	0.63	0.65	0.09 (0.08)	0.25 (0.24)
Operator	0.47	0.59	0.77	0.14 (0.13)	0.08 (0.04)
Installation Site	0.51	0.57	0.57	0.02	0.40
Object Type	-	-	0.25	0.11	0.64
Processing Type	-	-	0.23	0.30	0.47
Procedure Type	-	-	0.39	0.19	0.42

5.2 Qualitative Analysis

To better understand the details of when the different services failed, we performed a qualitative error analysis for the Reference Number, Issuing Authority, and Operator.

One interesting aspect of the Reference Number outcomes is the rather large performance difference of the strict and lenient evaluation setting. Analysing the 165 cases where the lenient setting brings a positive and the strict setting a negative hits@all result, we could find that in 65% of the cases, the matched Reference Number extracted from the permit seemed to be an addendum to the one entered into the EPMS system (e.g, `RKU-2123987` vs `RKU-2123987-v2`). Adding this addendum Reference Number to the original entry would probably be of benefit regarding searchablility. In 14% of the cases, the Reference Number entered into the EPMS could not be used directly to compare against, as they were followed by a describing string Finally, in 21% of the cases, the extractor indeed did not correctly identify the boundaries when extracting the Reference Number. More specifically, the Reference Number followed by the a date was extracted. This type of error could easily be mitigated by adjusting the candidate annotation algorithm. In addition to the errors produced during the extraction of the Reference Number, it is further noteworthy that the 9% and 7% of cases, where the correct entity was not among the candidates for only incorrect and no suggestions, respectively, way more than half of the times (i.e., 5% and 4%), this could be traced back to the fact that the string entered as Reference Number into the EPMS system was not present at all in the permit document.

For the key elements extracted by the NE-Roles Extraction service, we analysed why the correct entities were not among the candidates, since this scenario is extremely common in cases of errors. Together with domain experts, we cat-



(a) Operator (b) Issuing Authority

Fig. 5: Analysis of NE-Roles Extraction errors when the correct entity is not among the candidates.

egorised the different reasons into five different super classes: deficiencies in the metadata stored in the EPMS (A), deficiencies in the data contained in the KG (D), deficiencies contained in the permit itself (P), technical limitations of the developed system (T), and cases where it is not clear why the extraction failed (Other). Finally, we defined a sixth group (X) of errors, which we could identify, but not properly evaluate as crucial background information was missing, e.g., when the EPMS metadata contained a completely different entity type than mentioned in the permit.

For the Operator, over 40% of the cases of missing correct candidates could be traced back to missing data in the KG. On the one hand, these were missing labels, especially when the Operator represented a natural person. On the other hand, the current version of the KG lacked some deeper insight, e.g., the renaming of companies.

Another fourth of the missed annotations could be attributed to the inaccurate usage of entity names in the permits. For companies, the usage of the exact wording is crucial to qualify their reference as correct, and therefore the permit as valid. In about 14% of the analysed cases , the permit authors would use slightly different wordings (e.g., `Umweltservice` vs `Umwelt Service`, `Umwelt Service` vs `Umwelt-Service`, or `&` vs `and`) to refer to the operating company. In these cases, despite the errors, it was rather obvious to the human which operator the permit was attributed to. However, in another 12% of the cases, the naming used in the permit diverged to a larger extent to the metadata annotated in the EPMS. For these instances, certain aspects (like first or last names, or business forms) are added, swapped, or removed from the registered company name. (e.g. `Johann Wurst GmbH` vs `Wurst Johann GmbH` vs `Wurst GmbH`, or `Wurst GmbH` vs `Wurst BaugmbH` vs `Wurst AG`). In these cases, it is hard to predict whether the true origin of these differences, i.e., from incorrect citation, or incorrectly annotated EPMS metadata. In a few cases, we could indeed verify that –despite the large similarity of the company names– the entity stored in the EPMS was a different from the one referred to in the permit.

In 8.6% of the cases, the missing annotation originated from technical limitations, e.g., when the entity was separated by a page break.

The reasons of missing candidate annotations for the Issuing Authority were quite different and less diverse then for the Operator. First, we could not identify any issues originating from deficiencies contained in the permit itself. Second, most of the errors can be attributed to the incomplete mention of entities, for example, the permit would only refer to the `Governor` instead of the `Governor of Lower Austria`, with the corresponding county only being unidentifiable e.g., by the address in the letter head. Additional data modelling or extraction strategies would be needed to correctly process this distributed information. Interestingly, the vast majority of errors that fall in this category are connected to the usage of a single template by one Austrian state, meaning, that the targeted adaptation to this type of permits could significantly increase the performance.

5.3 Auditability Analysis

We analyse our approach and report on lessons learned based on the audit requirements (A1-A3) (cf. Section 2):

Completeness of audit trace capturing: We analyse completeness on two levels: the use case level and the lifecycle level. For the concrete use case, the baseline for completeness is the set of audit questions (cf. Section 4.3). These audit questions were targeted with different SPARQL queries which can be divided into two different types: *overview queries* provide general logging information of past executions, including login, extraction, or error events. One example being Listing 1.2 showcasing all metadata elements suggestions that were changed by users. With this information specific executions can be inspected in further detail for the model id, occurring errors etc. The second type being *metric queries* calculating numeric summaries of past executions, e.g., the rate of successful extractions over a specific time period. With this set of queries, we were able to answer all audit questions of the stakeholders, resulting in the fulfilment of the completeness requirement for this use case.

Analysing the completeness of the developed system in a broader context, i.e., lifecycle view, we identified that the capturing of audit traces is focused on the operation phase, while auditability aspects of the development phase are partly neglected. Concretely, information about different ML model versions, associated high-level characteristics and hyperparameter are already captured, however, more detailed information (e.g. Model Cards [11] or data retrieved from MLOPs tools such as Weights & Biases[14] or MLflow[15]) are yet to be integrated.

Automatic Collection, Transformation and Management of Audit Traces: This requirement is completely fulfilled through the usage of AuditBox. After an initial set-up phase, all relevant information and traces are collected in a fully automatic way, validated according to the defined workflow model and stored in a graph format. *Ability to access results and Ease of Use:* In order to provide

[14] https://wandb.ai/site
[15] https://mlflow.org

easy access to the audit results, we provided the SPARQL queries necessary to answer the audit questions as a set of over 20 templates. However, for complex queries and future work, a user interface to create custom audit dashboards leaves room for improvement to also enable users without SPARQL expertise to conduct regular audits with custom queries.

```
select ?uuid ?metadata_name ?metadata_extracted ?metadata_refined where {
    ?provenance rdf:type ep-plan:Bundle .
    ?provenance :uuid ?uuid .
        ?provenance ep-plan:hasTraceElement ?postProcessing .
        ?provenance ep-plan:hasTraceElement ?refineMetadata .
        ?postProcessing rdf:type :PostProcess .
        ?postProcessing prov:used ?metadata .
        ?metadata rdf:type ?metadata_name .
        ?metadata :value ?metadata_extracted .
        ?refineMetadata rdf:type :RefineMetadata .
    ?refinedMetadata rdf:type ?metadata_name .
    ?refinedMetadata prov:wasGeneratedBy ?refineMetadata .
    ?refinedMetadata :value ?metadata_refined .
    FILTER(?metadata_name != prov:Entity && ?metadata_name != ep-plan:Entity
&& ?metadata_name != prov:Metadata && ?metadata_name != prov:Metadata)
}
```

Listing 1.2: Overview query on corrected suggestions by user

6 Conclusion and Outlook

The extraction of key elements from legal documents is a complex task which is time-consuming and error-prone for non-trained experts. The heterogeneity of permits in information and format make it complex on multiple levels. The combination of symbolic and machine learning techniques for this use case allows to leverage strengths of both approaches: the ability to extract data through language and embedding models while incorporating background knowledge in the form of a knowledge graph can improve the overall detection of entities. Further improvements could be achieved by extending the ontology to e.g., cover historic changes in entity types or the complete set of labels. Another area for improvement could be expanding usable background knowledge in the form of explicit rules for issuing authorities or for specific responsibilities.

The auditability of the described system is a key feature to increase the user acceptance towards this AI-assisted solution approach. AuditBox, provides a generic tool for collecting and managing audit traces from multiple sources. Overall usability could be improved by providing a dashboard complementing the query-based approach. Further work in extending the audit traces to other system lifecycle phases (ML training) is planned.

Acknowledgements

This work has been supported by OBARIS (https://www.obaris.org/), a project funded by the Austrian Research Promotion Agency (FFG) under grant 877389.

References

1. Breit, A., Revenko, A., Rezaee, K., Pilehvar, M.T., Camacho-Collados, J.: WiC-TSV: An evaluation benchmark for target sense verification of words in context. In: Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume. pp. 1635–1645. Association for Computational Linguistics, Online (Apr 2021). https://doi.org/10.18653/v1/2021.eacl-main.140, https://aclanthology.org/2021.eacl-main.140
2. Eckart de Castilho, R., Gurevych, I.: A broad-coverage collection of portable NLP components for building shareable analysis pipelines. In: Proceedings of the Workshop on Open Infrastructures and Analysis Frameworks for HLT. pp. 1–11. Association for Computational Linguistics and Dublin City University, Dublin, Ireland (Aug 2014). https://doi.org/10.3115/v1/W14-5201, https://aclanthology.org/W14-5201
3. Eiband, M., Schneider, H., Bilandzic, M., Fazekas-Con, J., Haug, M., Hussmann, H.: Bringing transparency design into practice. In: 23rd international conference on intelligent user interfaces. pp. 211–223 (2018)
4. Ekaputra, F.J., Ekelhart, A., Mayer, R., Miksa, T., Šarčević, T., Tsepelakis, S., Waltersdorfer, L.: Semantic-enabled architecture for auditable privacy-preserving data analysis. Semantic Web (Preprint), 1–34 (2021)
5. Hellmann, S., Lehmann, J., Auer, S., Brümmer, M.: Integrating nlp using linked data. In: International semantic web conference. pp. 98–113. Springer (2013)
6. Janowicz, K., Knap, T., Hanečák, P., Klímek, J., Mader, C., Nečaský, M., Van Nuffelen, B., Škoda, P.: Unifiedviews: An etl tool for rdf data management. Semant. Web **9**(5), 661–676 (jan 2018). https://doi.org/10.3233/SW-180291, https://doi.org/10.3233/SW-180291
7. Leitner, E., Rehm, G., Moreno-Schneider, J.: Fine-grained Named Entity Recognition in Legal Documents. In: Acosta, M., Cudré-Mauroux, P., Maleshkova, M., Pellegrini, T., Sack, H., Sure-Vetter, Y. (eds.) Semantic Systems. The Power of AI and Knowledge Graphs. Proceedings of the 15th International Conference (SEMANTiCS 2019). pp. 272–287. No. 11702 in Lecture Notes in Computer Science, Springer, Karlsruhe, Germany (9 2019), 10/11 September 2019
8. Liao, Q.V., Gruen, D., Miller, S.: Questioning the ai: informing design practices for explainable ai user experiences. In: Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems. pp. 1–15 (2020)
9. Longley, D., Sporny, M., Kellogg, G., Lanthaler, M., Lindström, N.: Json-ld 1.1 framing (Jul 2020), https://www.w3.org/TR/json-ld-framing/
10. Miles, S., Groth, P., Munroe, S., Moreau, L.: Prime: A methodology for developing provenance-aware applications. ACM Transactions on Software Engineering and Methodology (TOSEM) **20**(3), 1–42 (2011)
11. Mitchell, M., Wu, S., Zaldivar, A., Barnes, P., Vasserman, L., Hutchinson, B., Spitzer, E., Raji, I.D., Gebru, T.: Model cards for model reporting. In: Proceedings of the conference on fairness, accountability, and transparency. pp. 220–229 (2019)
12. Moreno-Schneider, J., Rehm, G., Montiel-Ponsoda, E., Rodriguez-Doncel, V., Revenko, A., Karampatakis, S., Khvalchik, M., Sageder, C., Gracia, J., Maganza, F.: Orchestrating NLP services for the legal domain. In: Proceedings of the Twelfth Language Resources and Evaluation Conference. pp. 2332–2340. European Language Resources Association, Marseille, France (May 2020), https://aclanthology.org/2020.lrec-1.284

13. Naja, I., Markovic, M., Edwards, P., Cottrill, C.: A semantic framework to support ai system accountability and audit. In: European Semantic Web Conference. pp. 160–176. Springer (2021)
14. Ostendorff, M., Blume, T., Ostendorff, S.: Towards an open platform for legal information. In: Proceedings of the ACM/IEEE Joint Conference on Digital Libraries in 2020. p. 385–388. JCDL '20, Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3383583.3398616, https://doi.org/10.1145/3383583.3398616
15. Raji, I.D., Smart, A., White, R.N., Mitchell, M., Gebru, T., Hutchinson, B., Smith-Loud, J., Theron, D., Barnes, P.: Closing the ai accountability gap: Defining an end-to-end framework for internal algorithmic auditing. In: Proceedings of the 2020 conference on fairness, accountability, and transparency. pp. 33–44 (2020)