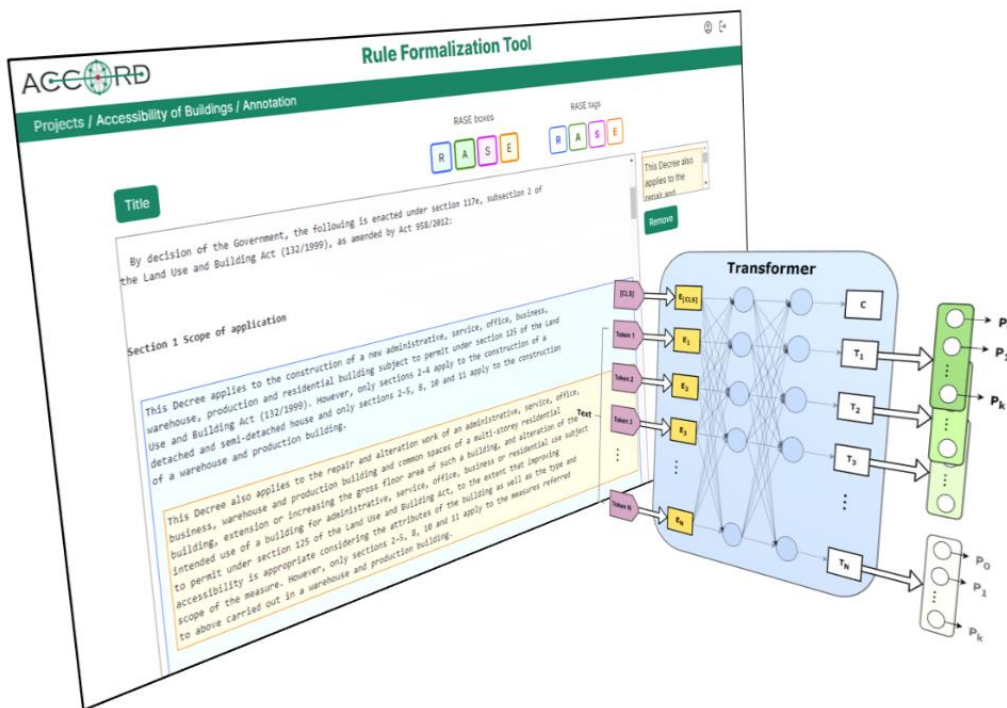


D2.3 Rules Toolset

August 28, 2024



This project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement no. 101056973.



**Funded by
the European Union**

UK Participants in Horizon Europe Project [ACCORD] are supported by UKRI grant numbers [10040207] (Cardiff University), [10038999] (Birmingham City University and [10049977] (Building Smart International).



**Innovate
UK**

Funded by the European Union. The views and opinions expressed are those of the author(s) only and do not necessarily reflect those of the European Union or European Health and Digital Executive Agency (HaDEA). Neither the European Union nor the granting authority can be held responsible for them.

Project Title	ACCORD - Automated Compliance Checks for Construction, Renovation or Demolition Works, grant agreement No: 101056973.
Deliverable:	D2.3 Rules Toolset
Type:	Other
Dissemination level:	Public
Work package:	2
Lead Beneficiary:	FUNITEC
Deliverable leader:	FUNITEC
Contributing partners:	BCU, CU, Ontotext.
Due date:	31 August 2024
Date:	28 August 2024
Status – version, date:	V1.2 – 28-08-2024
Authors:	Gonçal Costa, Álvaro Sicilia (FUNITEC)
	Hansi Hettiarachchi, Edlira Vakaj, Dhoyazan Al-Turki, Mohamed Gaber (BCU)
	Thomas Beach (CU)
Reviewer 1:	Maxime Lefrançois (IMT)
Reviewer 2:	Rick Makkinga (FUI)

DOCUMENT HISTORY

Version	Date	% Complete	Comments	Main Authors (organisation)
0.1	13/05/2024	5	Initial Deliverable Structure and Table of Contents	Edlira Vakaj (BCU), Gonçal Costa (FUNITEC)
0.2	13/05/2024	40	Section 0, Annex A, B and C	Hansi Hettiarachchi, Mohamed Gaber(BCU)
0.3	21/06/2024	45	Introduction, partial contribution in Sections 1 and 0	Gonçal Costa (FUNITEC)
0.6	02/07/2024	75	Sections 1 and 0	Gonçal Costa (FUNITEC)
0.7	03/07/2024	90	Section 3.3 RASE Automation	Dhoyazan Al-Turki (BCU), Edlira Vakaj (BCU)
0.8	10/07/2024	93	Edits and Internal Review	Edlira Vakaj (BCU), Thomas Beach (CU)
0.9	16/07/2024	95	Integration of the results of the focus group activity in section 2.5	Gonçal Costa (FUNITEC), Álvaro Sicilia (FUNITEC)
1.0	26.8.2024	97	Internal review	Maxime Lefrançois (IMT)
1.1	28/08/2024	99	Corrections according to review	Edlira Vakaj (BCU)
1.2 FINAL	28/08/2024	100	Submitted	Rita Lavikka (VTT)

Statement of originality:

This deliverable contains original, unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

The information in this document is provided "as is", and no guarantee or warranty is given that the information is fit for any particular purpose. The above-referenced consortium members shall have no liability for damages of any kind, including without limitation direct, special, indirect, or consequential damages that may result from the use of these materials subject to any liability which is mandatory due to applicable law.

Executive summary

This deliverable presents the results of Tasks 2.4, “Artificial Intelligence for Natural Language Processing of Building Codes”, and 2.5 “, Design and Implementation of Rule Formalisation Tool”, of the ACCORD project.

The aim of the ACCORD project is to digitalise building permitting and compliance procedures to improve the quality and productivity of design and construction processes and support the development of a sustainable built environment. This is achieved by adopting a semantic approach where different individual software components are combined to create flexible solutions that eliminate the need for expensive centralized systems that are difficult to establish and manage.

Building on the results achieved in Tasks 2.1, “Technical Review of Existing Standards”, 2.2, “Technical Review of Existing Standards”, and 2.3, “Machine-executable Regulations”, where existing ontologies, standards and data models in the construction data domain have been analysed, a methodology has been proposed to digitize and formalize regulations. The results of its application are instances of the Architecture Engineering and Construction Compliance Checking Ontology (AEC3PO) – an ontology created to represent the building compliance data domain – together with Building Compliance Rule Language (BCRL) – a domain-specific rules language to express checking rules – this deliverable will report the tasks related to the development of a Rule Formalisation Tool (RTF) which encompasses and integrates all these concepts, included the Artificial Intelligence (AI)-powered Rule Formalisation for Building Codes.

More specifically, this deliverable:

1. Provides a description of the design, architecture, and development of the RFT.
2. Introduces an AI-driven approach to formalising building code standards.

Publishable summary

This deliverable presents the results of Tasks 2.4, “Artificial Intelligence for Natural Language Processing of Building Codes”, and 2.5, “Design and Implementation of Rule Formalisation Tool” of the ACCORD project.

The aim of the ACCORD project is to digitalise building permitting and compliance procedures to improve the quality and productivity of design and construction processes and support the development of a sustainable built environment. This is achieved by adopting a semantic approach where different individual software components are combined to create flexible solutions that eliminate the need for expensive centralized systems that are difficult to establish and manage.

Building on the results achieved in Tasks 2.1, “Technical Review of Existing Standards”, 2.2 “Technical Review of Existing Standards” and 2.3 “Machine-executable Regulations”, where existing ontologies, standards and data models in the construction data domain have been analysed, a methodology has been proposed to digitize and formalize regulations. The results of its application are instances of the Architecture Engineering and Construction Compliance Checking Ontology (AEC3PO) – an ontology created to represent the building compliance data domain – together with Building Compliance Rule Language (BCRL) – a domain-specific rules language to express checking rules – this deliverable will report the tasks related to the development of a Rule Formalisation Tool (RTF) which encompasses and integrates all these concepts, included the Artificial Intelligence (AI)-powered Rule Formalisation for Building Codes.

More specifically, this deliverable:

1. Provides a description of the design, architecture, and development of the RFT.
2. Introduces an AI-driven approach to formalising building code standards.

Contents

Executive summary.....	4
Publishable summary.....	5
Contents	6
List of Figures	7
List of Tables	8
1. Introduction	9
1.1 The ACCORD Project	9
1.2 Aims and Objectives.....	9
1.3 Structure of the document	10
2. Rule formalisation tool	11
2.1 Overview	11
2.2 Process to formalise regulations in the rule formalisation tool	12
2.3 Functionalities to assist in the generation of rules	19
2.4 Architecture and components.....	23
2.5 Testing and results.....	24
2.5.1 Description	24
2.5.2 Results	25
2.6 Future improvements	26
3. AI-powered Rule Formalisation for Building Codes.....	27
3.1 Rule Formalisation	27
3.2 NLP Background	28
3.2.1 Language Models/Transformers	28
3.2.2 Large Language Models.....	30
3.3 ACCORD-NLP Data, AI Models and Workflows	31
3.3.1 CODE-ACCORD: A Corpus of Building Regulatory Data for Rule Generation towards Automatic Compliance Checking.....	31
3.3.2 SNOWTEC: Synthetic Natural Language Oversampling with Transformer-based Information Extraction for Automated Compliance Checking	42
3.3.3 RASE-LLM: RASE Automation Leveraging Large Language Models.....	55
3.3.4 Resource Index	67
3.3.5 Future Improvements	69
4. Conclusions.....	70
5. References.....	71
Annex A. SNOWTEC: Model Hyper-parameters	76
Annex B. SNOWTEC: Error Analysis of Entity Classifier.....	76
Annex C: SNOWTEC: Error Analysis of Relation Classifier.....	78
Annex D: RASE-LLM Tool Interface.....	81

List of Figures

Figure 1. Diagram showing the rule formalisation process elaborated in Task 2.3.	11
Figure 2. Screenshots of the login and register interfaces.....	12
Figure 3. Screenshot of the interface to upload a pdf document.....	13
Figure 4. Screenshot of the interface to select the manual or automatic annotation.	13
Figure 5. Screenshot of how the PDF document looks in a regular PDF viewer.	14
Figure 6. Screenshot of the interface to apply the RASE method manually where the annotation has not yet been applied.....	15
Figure 7. Screenshot of the interface to apply the RASE method manually after the annotation has been applied showing RASE boxes (filled colour) and RASE tags (only a coloured outline, no fill).	15
Figure 8. Screenshot of the interface to apply the RASE method manually, highlighting the selection and remove button for the selected content.	16
Figure 9. Screenshot of the interface to validate the BCRL expressions.	17
Figure 10. Screenshot of the interface to create or select a project.....	18
Figure 11. Screenshot of the interface showing the list of projects already created by the user. ...	19
Figure 12. Screenshot of the interface with the RASE TEXT tagging panel in the right side: defining an object.	20
Figure 13. Screenshot of the interface with the RASE TEXT tagging panel in the right side: defining a property.....	21
Figure 14. Screenshot of the interface with the RASE TEXT tagging panel in the right side: defining a reference.....	22
Figure 15. Diagram showing the architecture of the Rule Formalisation Tool.	23
Figure 16. Transformer encoder architecture	29
Figure 17. CODE-ACCORD semi-automatic data preparation approach.....	34
Figure 18. Sequence length distribution of annotated sentences in the CODE-ACCORD dataset.	39
Figure 19. Distribution of entity categories	40
Figure 20. Distribution of the number of entities per sentence.....	40
Figure 21. Sequence length distribution of annotated text spans as entities.....	41
Figure 22. Distribution of relation categories	41
Figure 23. Distribution of the number of relations per sentence.....	41
Figure 24. Overview of the SNOWTEC pipeline	43
Figure 25. Entity classification architecture	44
Figure 26. Relation classification architecture	45
Figure 27. Distribution of the relation categories in original data	47
Figure 28. Distribution of the relation categories in augmented data	47
Figure 29. Sequence length distribution of text elements in the regulatory text corpus.....	49
Figure 30. Training and validation/evaluation learning curves of entity classifiers built using the RoBERTa-Large model on different learning rates.....	51
Figure 31. Training and validation/evaluation learning curves of relation classifiers built using the RoBERTa-Large model on different learning rates.....	53
Figure 32. RASE automation tool architecture.	65
Figure 33. Confusion matrix of the best-performed relation classifier (RoBERTa-Large) on the test dataset.....	79
Figure 33. Login Screen.....	81
Figure 34. Sign Up Screen.....	81
Figure 35. Change Password.....	82
Figure 36. Projects List	82
Figure 37. Add Project	83

Figure 38. Project output (YAML Editor).....	83
--	----

List of Tables

Table 1. Building codes of England and Finland	33
Table 2. Statistical summary of data collection.....	35
Table 3. Entity categories. A colour theme is used to enhance the clarity of the sample annotations given in this document.	36
Table 4. Sample entity annotations	36
Table 5. Relation categories	37
Table 6. Sample relation annotations	38
Table 7. Format of entity data file.....	39
Table 8. Format of relation data file.....	40
Table 9. Original sentence samples and their corresponding synthetic samples. Columns e1 and e2 represent the categories of each entity. A colour scheme representing categories is involved to highlight the entities in original and synthetic samples.	46
Table 10. Statistics of original and augmented relation-annotated training data	48
Table 11. Performance evaluation of the entity classifier across validation and test datasets using various transformer models. The best F1 score is marked in bold.....	50
Table 12. Impact on best-performed entity classifier's (RoBERTa-Large) results by different optimisation techniques. The best F1 score is marked in bold.....	51
Table 13. Performance evaluation of the relation classifier across validation and test datasets using various transformer models, with and without the integration of data augmentation. The best F1 score is marked in bold. The improvement in F1 score on the test data after applying data augmentation is indicated within brackets.	52
Table 14. Impact on best-performed relation classifier's (RoBERTa-Large) results by different optimisation techniques. The best F1 score is marked in bold.....	53
Table 15. Knowledge graphs generated by SNOWTEC for a set of sample sentences. In. and Out. denotes the input to and output from the pipeline.....	54
Table 16. Hyper-parameter specifications.....	76
Table 17. Entity categories with a few samples.....	77
Table 18. Results of the best-performed entity classifier (RoBERTa-Large) on validation and test datasets across different entity categories	77
Table 19. Results of the best-performed entity classifier (RoBERTa-Large) on test datasets depending on the entity count per sentence.....	78
Table 20. Results of the best-performed relation classifier (RoBERTa-Large) on validation and test datasets across different relation categories	78
Table 21. Frequently misclassified relations with identified causes and samples	80

1. Introduction

1.1 The ACCORD Project

The ACCORD project aims to provide a framework for the digitalisation of building permitting and compliance processes, using Building Information Modelling (BIM), Geographic information systems (GIS), and other data sources. The end goal is to improve the productivity and quality of design and construction processes. ACCORD is based on the principle that these digitised processes must be human-centred, transparent, and cost-effective for the permit applicants and authorities and, above all, relevant to the industry within which they are to be employed.

To address this challenge, ACCORD proposes developing a semantic framework for European digital building permitting processes, regulations, data, and tools. This framework will drive the formalisation of regulations into a set of rules and the integration of existing tools to check compliance with building codes and regulations as microservices in a dynamic ecosystem. Software solutions will be developed, providing consistency, interoperability and reliability with municipal, regional, national, and international regulatory frameworks, processes, and standards.

Building codes and regulations often involve complex language and technical jargon that can be difficult to understand and apply in practice. This ambiguity can lead to different possible interpretations. This reality represents a challenge for the development of automation processes to check compliance with regulations. One way to address this challenge may be through a semantisation carried out through a process of formalisation of regulations described in plain text into machine-readable documents using a domain-specific rule language.

Semantic Web technologies, ontologies, and semantic rule languages provide a foundation for creating a solution aligned with this semantisation approach. Also, Artificial Intelligence (AI) methods, such as Natural Language Processing (NLP), can be used to extract and analyse compliance requirements from natural language text in an automated way. This process of transforming natural language into machine-readable data with explicit meaning can also help address this challenge by creating structured representations of regulations that can be exchanged and processed by computers.

1.2 Aims and Objectives

This deliverable reports the results of tasks involved in the implementation of the rule formalisation process that takes place in WP2. These tasks include the implementation of developed concepts, such as the methodology to formalise regulations as instances of the AEC3PO together with the BCRL language. The generated graphs, according to the AEC3PO ontology, are represented in Resource Description Framework (RDF) format and are aimed to be used in automated code compliance (ACC) processes. The knowledge graph includes rules extracted from the textual data, such as building codes, standards, and regulations.

This deliverable reports the outcomes of Tasks 2.4 “Artificial Intelligence for Natural Language Processing of Building Codes” and 2.5 “Design and Implementation of Rule Formalisation Tool” of the ACCORD project. The overall aim of this work is to provide the implementation of the method to perform the rule formalisation process, elaborated in the previous tasks of WP2, to formalise building codes and regulations by converting them into RDF graphs representing instances of the AEC3PO ontology.

The objectives of these tasks are:

1. **Task 2.4:** Implementation of the AI-driven rule formalisation elements for building codes.
2. **Task 2.5:** Implementation of the Rule Formalisation tool.
3. **Tasks 2.4 and 2.5:** Evaluation of the outcome of the tool testing, including the AI-driven rule formalisation part for building codes.

1.3 Structure of the document

The remainder of this document is organised into two sections:

- **Section 2** introduces the rule formalisation tool developed in ACCORD. This starts with a general description of the tool in Section 2.1, followed by the steps to formalise regulations through the different interfaces implemented in the tool in Section 2.2. Section 2.3 contains a description with specific functionalities to help the user define the rules. The tool's architecture and components are described in Section 2.4. Finally, the process and results of testing and a focus group activity to evaluate the tool are documented in Section 2.5, and possible future improvements and additional developments are presented in Section 2.6.
- **Section 3** introduces an AI-powered rule formalisation suite, leveraging recent advancements in NLP. The section provides a comprehensive overview of the background, design and implementation approaches, and findings of this development. Section 3.1 introduces the concept of rule formalisation, presenting a summary of previous approaches and outlining ACCORD's objectives within the context of state-of-the-art technologies. Subsequently, Section 3.2 elaborates further on the recent trends in NLP that this development has followed. Finally, Section 3.3 details various sub-components developed by ACCORD to facilitate automatic rule formalisation.

2. Rule formalisation tool

2.1 Overview

The purpose of the rule formalisation tool is to provide a means for technicians working in public administrations, such as city councils and other governmental bodies responsible for generating building codes and regulations in European countries, to be able to formalise the regulations provided in plain text documents into a machine-processable format according to a semantic data schema, an ontology, with the purpose of it becoming a standard.

This rule formalisation process has already been defined in Task 2.3 “Machine-executable Regulations” and reported in Deliverable D2.2 “BC Ontology and Rule Format” under the name Regulation Digitalisation Methodology (Figure 1). The tool integrates components already defined in this Task 2.3 and developed in Task 2.2 “Development of the Building Compliance Ontology”. These components are the AEC3PO ontology and the BCRL language. The tool also integrates an automated annotation process using NLP techniques, which is described in section 3. This process has been implemented within an autonomous and separate service that is invoked from the tool.

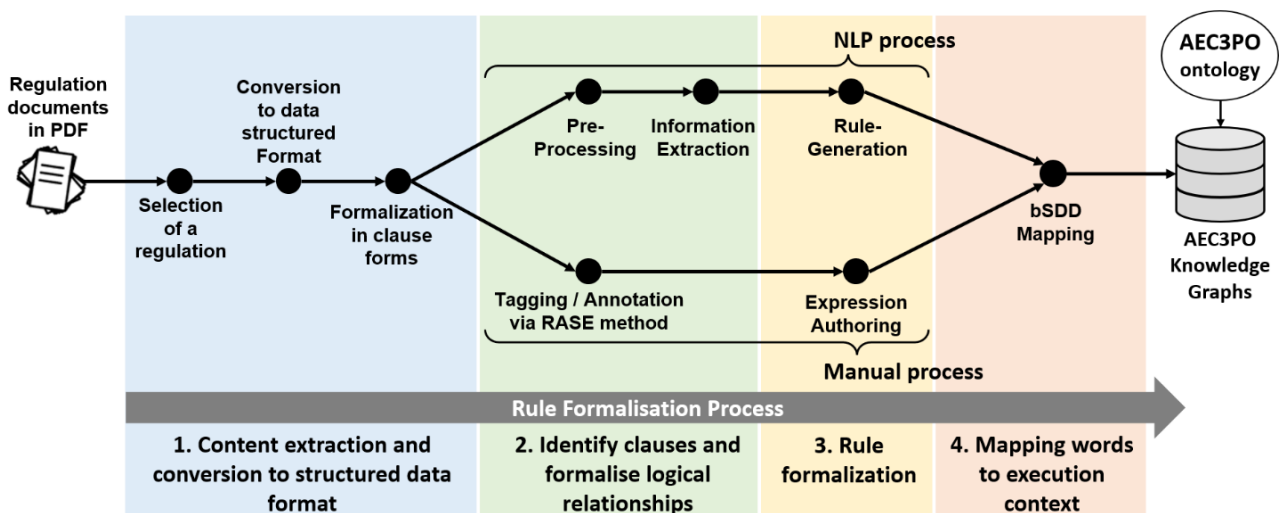


Figure 1. Diagram showing the rule formalisation process elaborated in Task 2.3.

As can be seen in Figure 1, the tool covers the entire formalisation process, from uploading a PDF file of a regulation by the user, to obtaining the result: an RDF graph generated as an instance of the AEC3PO ontology, and which is published to an external rules database.

The rule formalisation tool has been implemented as a web application where users can register and create projects for each regulation to be transformed into machine-processable format. The usual data such as name, surname, email address, etc. are requested during registration (see Figure 2). Once registered, the user can begin to perform several actions to carry out rule formalisation process for one or more regulations. The following section 2.2 details how this process is carried out in the tool from the user's perspective and, in parallel, the processes that are carried out internally.

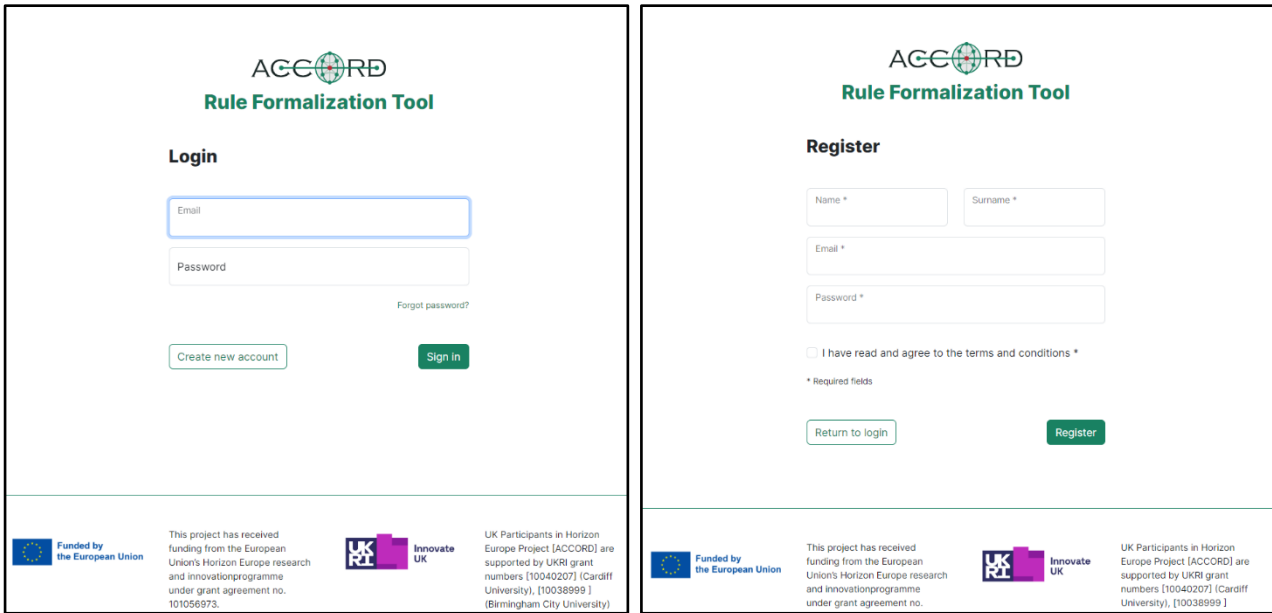


Figure 2. Screenshots of the login and register interfaces.

2.2 Process to formalise regulations in the rule formalisation tool

The process of formalizing a regulation through the tool is carried out through 4 stages: creation, annotation, validation, and publication, where a specific interface is provided for each of them. The steps carried out in each of these stages are described below:

Step 1: Project definition

The first step is to create a project. Within the tool, a user can create multiple projects. The idea is to create a project for each regulation that is going to be formalized.

When creating a project, the tool asks the user to define the following fields:

- The name of the project.
- The URL of the regulation (for example, the URL of the original public official website).
- The publication date (this may be the date of its official publication or can also be the date of a new version of the graph due to errors or other amendments for example).
- A short code (can be some kind of abbreviation, code name of the regulation, or a classification category for example).
- The scope (International, National, Regional, and Municipal).
- The Country where the regulation applies (in case of National, Regional, or Municipal).
- Regulation language.
- Measure system (Metric or Imperial).
- A summary or description of the regulation.

Step 2: Uploading the regulatory document

This is a simple step that involves uploading the regulation document in PDF format (Figure 3).

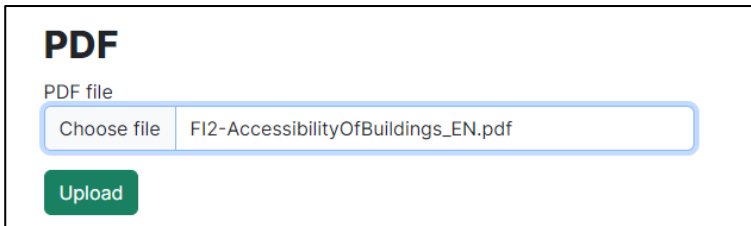


Figure 3. Screenshot of the interface to upload a pdf document.

Step 3: Choosing manual or automatic annotation

Once the regulatory document has been uploaded in PDF format, the tool offers the user two options to annotate the text using the Requirement Application Selection Exception (RASE) [1, 2] method: (1) from scratch, or (2) obtain a proposal with the annotation already made on the text (Figure 4). The second option involves utilising AI to automatically apply the RASE method. Details about this service, which includes other additional functionalities, are provided in section 3.

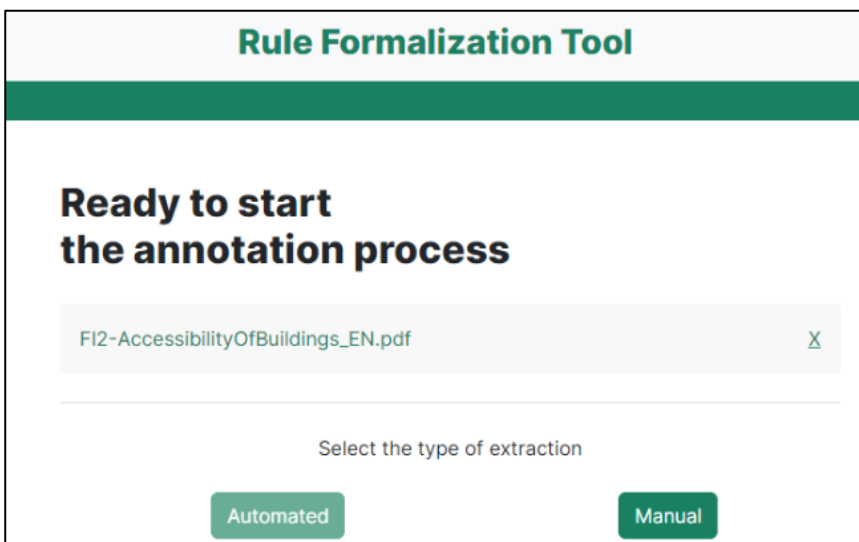


Figure 4. Screenshot of the interface to select the manual or automatic annotation.

Step 4: Annotation

If the user selects the manual option, the tool provides a view of the document like those shown by most PDF viewers (see Figure 5). The intention is for the user to start with a familiar view of the document, where text justification and the content per line are maintained to provide the same view as in the PDF viewer as much as possible. However, as the user begins to select parts of the text, applying the RASE method, the original formatting is replaced with the new RASE formatting.

Unofficial translation.
Legally binding only in Finnish and Swedish

Government Decree

on Accessibility of Buildings

By decision of the Government, the following is enacted under section 117e, subsection 2 of the Land Use and Building Act (132/1999), as amended by Act 958/2012:

Section 1

Scope of application

This Decree applies to the construction of a new administrative, service, office, business, warehouse, production and residential building subject to permit under section 125 of the Land Use and Building Act (132/1999). However, only sections 2–4 apply to the construction of a detached and semi-detached house and only sections 2–5, 8, 10 and 11 apply to the construction of a warehouse and production building.

This Decree also applies to the repair and alteration work of an administrative, service, office, business, warehouse and production building and common spaces of a multi-storey residential building, extension or increasing the gross floor area of such a building, and alteration of the intended use of a building for administrative, service, office, business or residential use subject to permit under section 125 of the Land Use and Building Act, to the extent that improving accessibility is appropriate considering the attributes of the building as well as the type and scope of the measure. However, only sections 2–5, 8, 10 and 11 apply to the measures referred to above carried out in a warehouse and production building.

The provisions of this Decree concerning buildings intended for a specific use also apply to a space intended for a similar use in another building.

Section 2

Passageway leading to a building

There shall be an easily noticeable passageway with a width of at least 1,200 millimetres and

Figure 5. Screenshot of how the PDF document looks in a regular PDF viewer.

The interface provides eight buttons to annotate the regulatory text using the RASE method. There are four buttons to define clauses and subclauses according to whether they refer to a Requirement, Application, Selection, or Exception. Then, there are four more buttons to select parts of the text based on these same categories to define objects, properties, or references to other parts of the text of the regulation or to another document (see Figure 6). The interface also provides another button to create sections. Therefore, when the user selects, for example, the title of a section of the regulation, the tool automatically creates a section from this point in the text to the next defined section. If none has been defined, the section includes all the text that follows the title.

Figure 6 shows an example of how the user views the regulatory text before starting the annotation process while Figure 7 shows the result after applying the RASE method to the same text shown in the current part of the interface. Tagging options through the interface are explained in section 2.3.

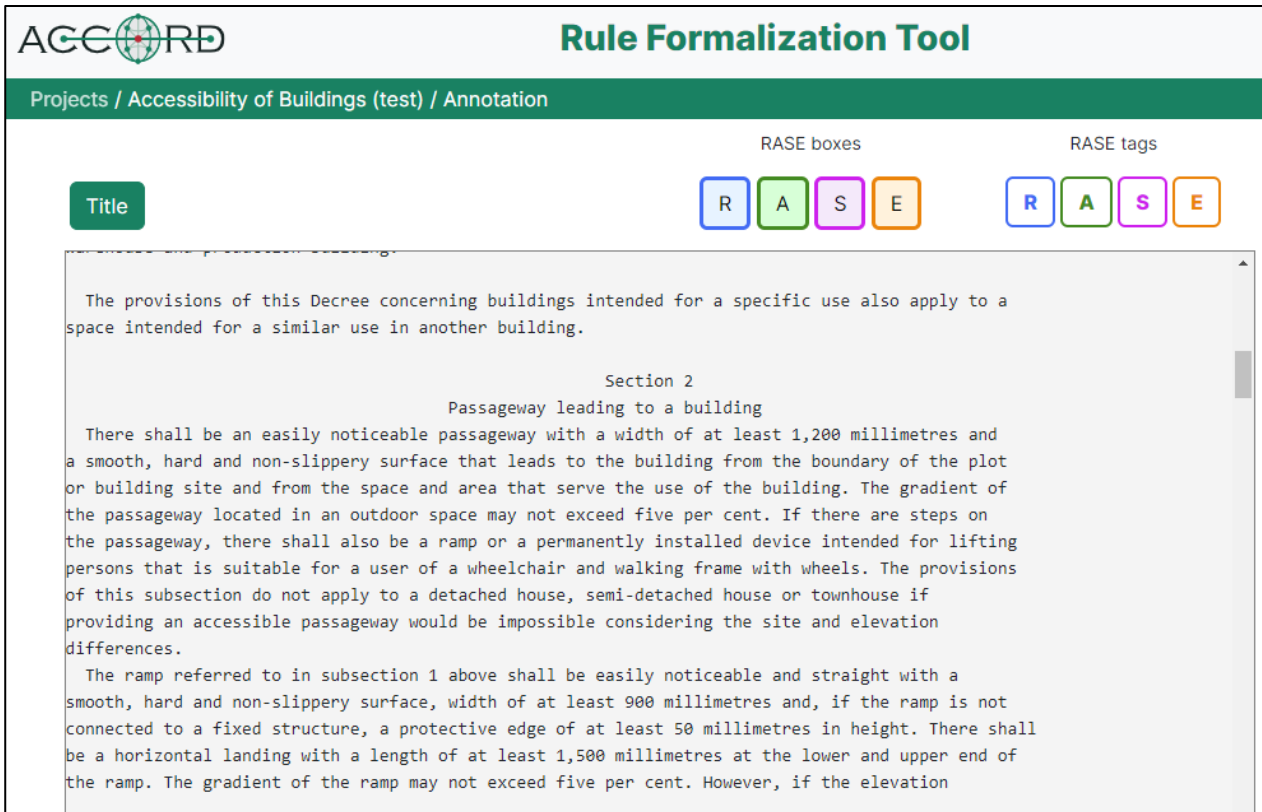


Figure 6. Screenshot of the interface to apply the RASE method manually where the annotation has not yet been applied.

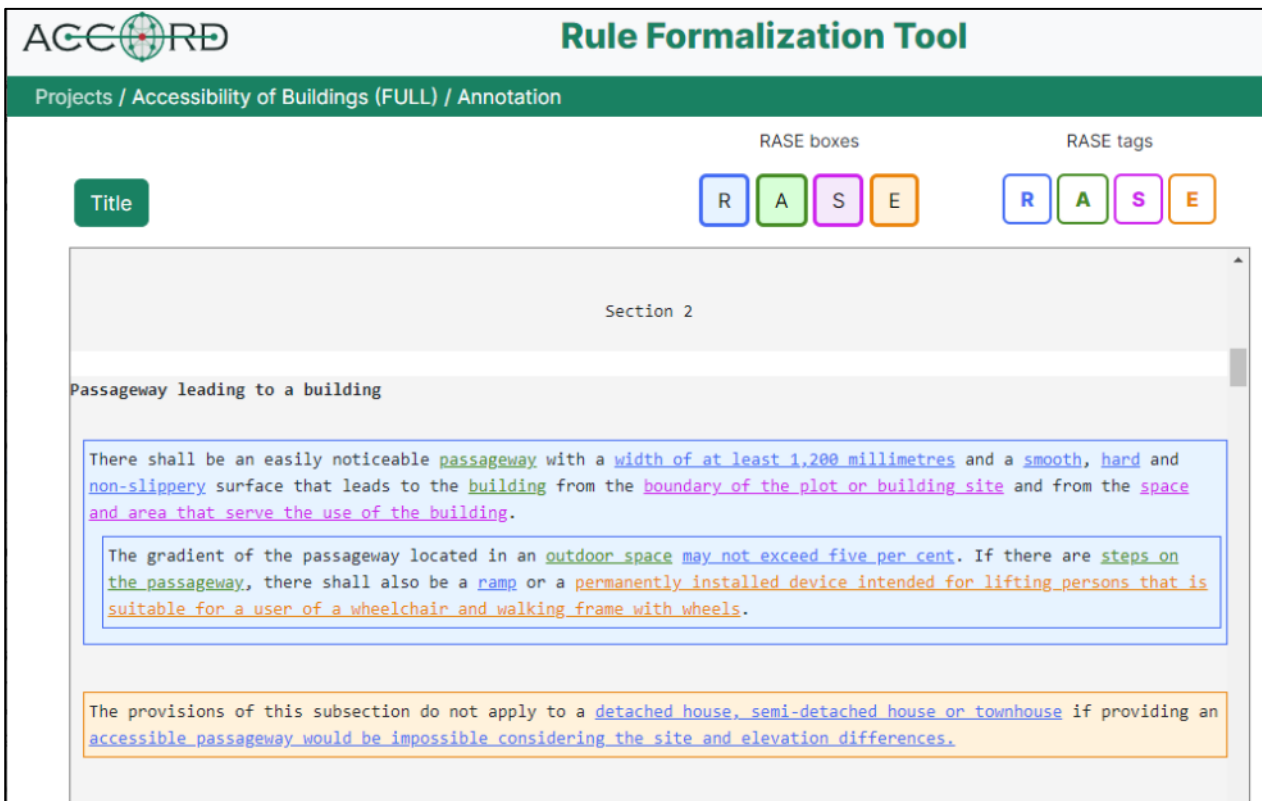


Figure 7. Screenshot of the interface to apply the RASE method manually after the annotation has been applied showing RASE boxes (filled colour) and RASE tags (only a coloured outline, no fill).

The annotation interface includes a text box just to the right of the regulatory text, which displays the text that is selected for tagging, or which has already been tagged before but the user has selected it by clicking on it. Linked to this aspect, the interface also has a “Remove” button that allows the user to remove the current RASE tag from the selected text (see red boxes in Figure 8). Since there are three levels of RASE tagging (text, box, section), the user must assume that deleting a higher level will also delete all included tagging. For example, deleting a section (by selecting the title) also means deleting all existing tagging in the text belonging to that section.

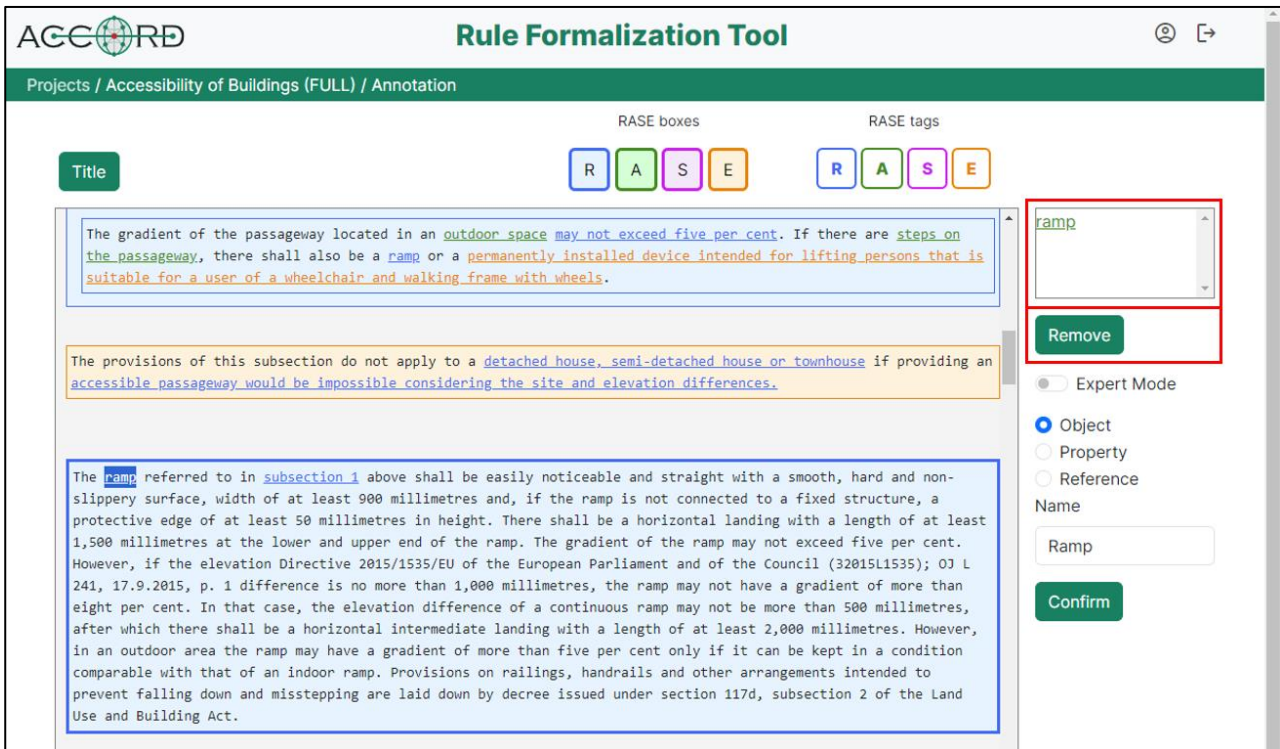


Figure 8. Screenshot of the interface to apply the RASE method manually, highlighting the selection and remove button for the selected content.

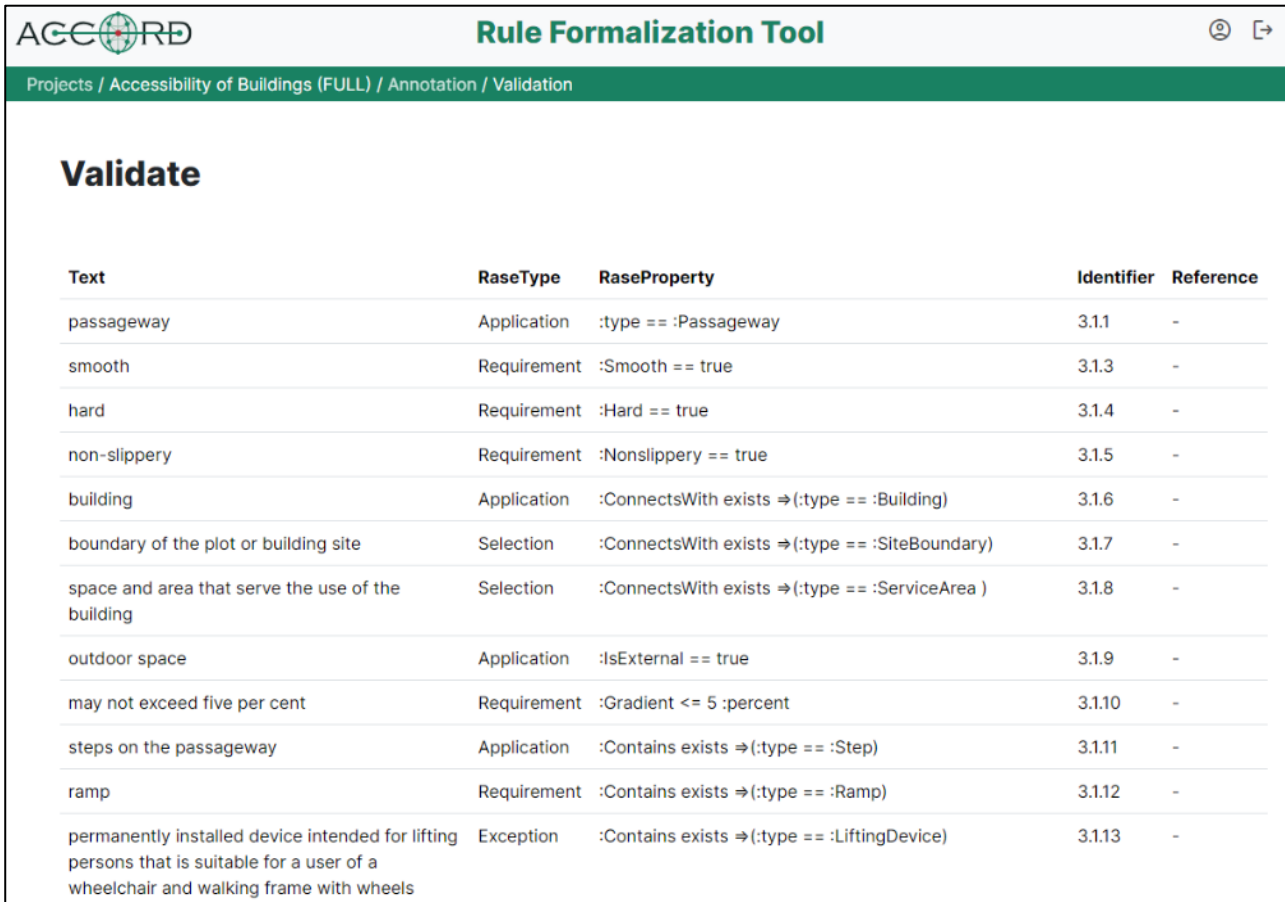
If, in step 3, the user selects the option to automatically apply the RASE method using AI tools, then instead of performing all the annotation manually, they will only need to correct/extend the annotation generated by the AI.

Step 5: BCRL validation

Once the user is sure that the text is correctly tagged according to the RASE method, the next step is to validate the BCRL expressions in terms of correctness in the definition. BCRL is a language developed within the ACCORD project introduced in [3] with a grammar generated with Another Tool for Language Recognition (ANTLR)¹ for building compliance checking purposes. The tool has been designed to accept definitions in BCRL that comply with its grammar. However, it may happen that authors provide an expression incorrectly. For example, the expression of a rule may indicate that a property value must be higher than a specific reference value when in reality, the text is indicating that this value should be lower (e.g., “:Width < 1500” instead of “:Width > 1500”).

¹ <https://www.antlr.org/>

To facilitate this review process in a more visual way, the user will see a table with rows that will include the text involved in the assignment and the corresponding BCRL expression (Figure 9). This provides users with a view that allows them to view only the rules and the text from which they originate to ease the process of review.



The screenshot shows the 'Rule Formalization Tool' interface. At the top, there is a navigation bar with the AGCORD logo, the title 'Rule Formalization Tool', and a help icon. Below the navigation bar, a breadcrumb trail reads 'Projects / Accessibility of Buildings (FULL) / Annotation / Validation'. The main content area is titled 'Validate' and contains a table with the following data:

Text	RaseType	RaseProperty	Identifier	Reference
passageway	Application	:type == :Passageway	3.1.1	-
smooth	Requirement	:Smooth == true	3.1.3	-
hard	Requirement	:Hard == true	3.1.4	-
non-slippery	Requirement	:Nonslippery == true	3.1.5	-
building	Application	:ConnectsWith exists ⇒ (:type == :Building)	3.1.6	-
boundary of the plot or building site	Selection	:ConnectsWith exists ⇒ (:type == :SiteBoundary)	3.1.7	-
space and area that serve the use of the building	Selection	:ConnectsWith exists ⇒ (:type == :ServiceArea)	3.1.8	-
outdoor space	Application	:IsExternal == true	3.1.9	-
may not exceed five per cent	Requirement	:Gradient <= 5 :percent	3.1.10	-
steps on the passageway	Application	:Contains exists ⇒ (:type == :Step)	3.1.11	-
ramp	Requirement	:Contains exists ⇒ (:type == :Ramp)	3.1.12	-
permanently installed device intended for lifting persons that is suitable for a user of a wheelchair and walking frame with wheels	Exception	:Contains exists ⇒ (:type == :LiftingDevice)	3.1.13	-

Figure 9. Screenshot of the interface to validate the BCRL expressions.

Step 6: Publishing

Once the user has validated the result, the next step is to publish it in the rule database, a component developed outside the scope of this tool and, therefore, provided externally. This way, when the user pushes the publish button, the graph will be stored as an RDF graph in this rule database which is a semantic triple store (Ontotext GraphDB).

When the user clicks on the validate button, the tool transforms the tagged RASE file into an AEC3PO instance that is represented as a graph described in the JSON-LD format [4].

The graphs are uniquely identified in the database by their URI. This URI is composed of different parts, according to its characteristics, as described below:

- **Base URI:** This part is common for all the graphs to be uploaded to the rule database (e.g., <https://graphdb.accordproject.eu/resource/aec3po/>). However, this part can be modified in the user settings interface if a user wants to connect to another possible alternative rule database.
- **Country Code:** It involves the country code where the regulation applies (e.g., FI), or it may include the continent code (e.g., EU if it is a regulation that applies at a European level).

- **Classification:** This field may contain a short text that identifies the regulation in some way. Its purpose is to differentiate one regulation from another in the same country or even when they are of the same type.
- **Language:** It involves the language code in which the regulation is specified (e.g., en-GB) since the same regulation can be provided in different languages.
- **Publication date:** In principle, this is the date the graph is published. However, this date can be selected by the user due to certain reasons described below. The date is expressed according to ISO 8601 standard.

This way, the URL is configured like this:

[https://graphdb.accordproject.eu/resource/aec3po/\\${country}/\\${classifier}/\\${language}/\\${date}](https://graphdb.accordproject.eu/resource/aec3po/${country}/${classifier}/${language}/${date})

The following example URL could be one for a published graph:

<https://graphdb.accordproject.eu/resource/aec3po/FI/ACC/en-GB/2024-09-08>

The fields that form the URI cannot be edited directly by the end user in this interface except for the publication date. The reason for this is that commonly, a graph of the regulation is published at a different time than when the regulatory text has been officially published. Thus, the user needs to be able to enter the official publication date.

The interface provided by the tool to perform this step also offers the user the option to download the graph in JSON-LD format² (see Figure 10).

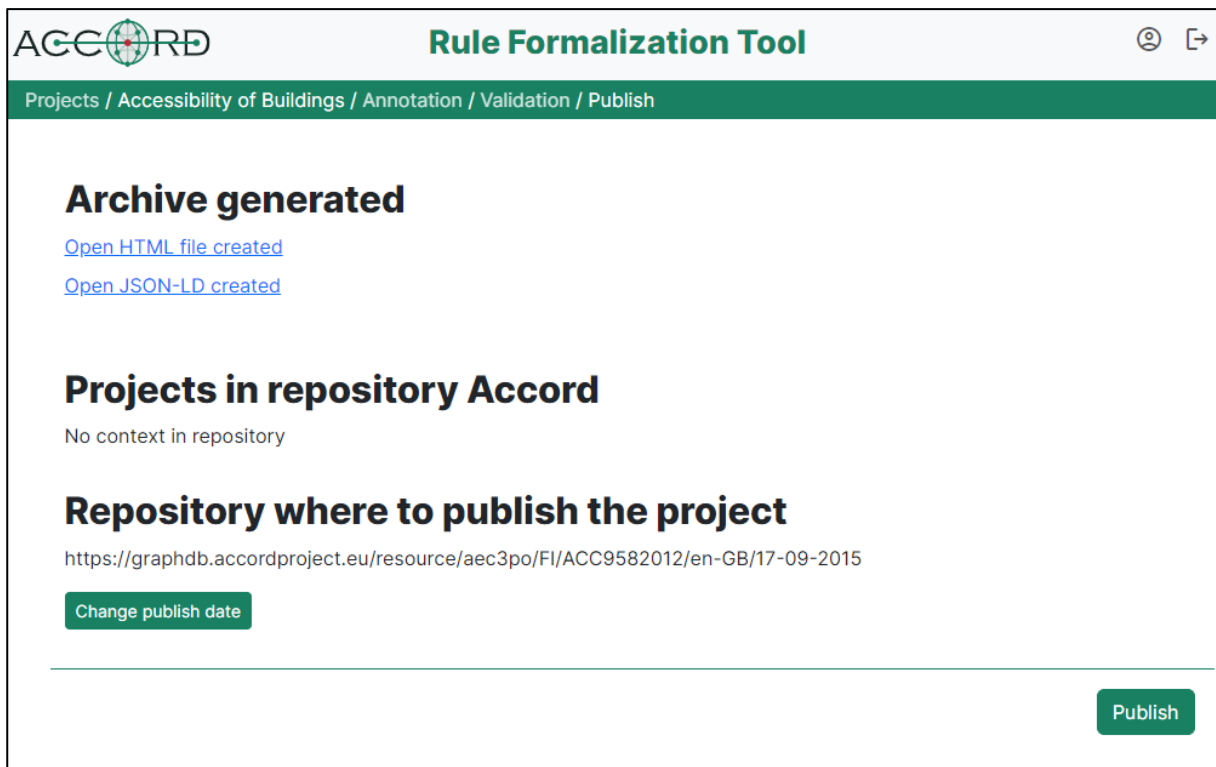


Figure 10. Screenshot of the interface to create or select a project.

² <https://json-ld.org/>

The tool has another interface where the user can create or select one of the projects previously created, being able to resume the process of formalising the regulation from the last stage (creation, annotation, validation, or publication) where this was left (Figure 11). Apart from resuming the process where the user left off by clicking on the “arrow” button, the parameters of a project can be edited by clicking on the configuration button, as well as deleted by clicking the “Delete” button.

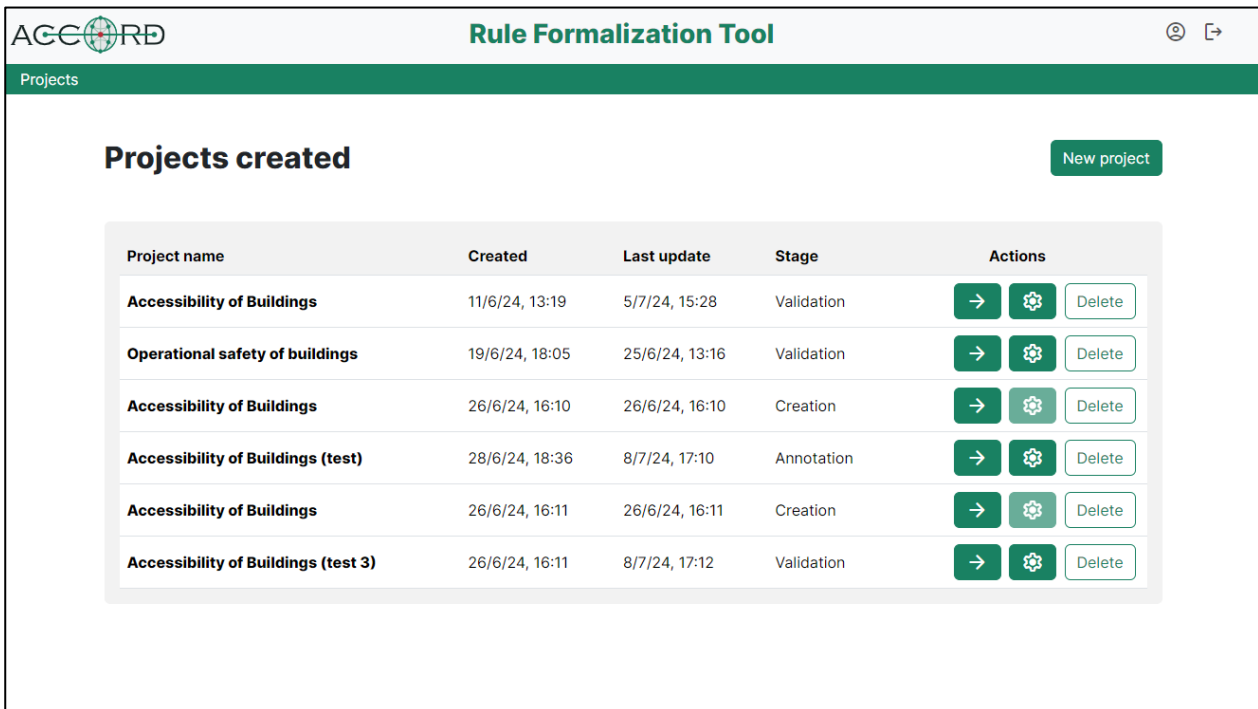


Figure 11. Screenshot of the interface showing the list of projects already created by the user.

2.3 Functionalities to assist in the generation of rules

The rule formalisation tool has been designed as a web interface tool that combines some typical functionalities, such as user registration and configuration, while others are specific. Once registered, the user can follow the process described above in the previous section 2.2. However, to carry out the specific tagging part of the RASE process, some additional functionalities are provided to assist in rule generation.

The challenge tackled by the development of the interface has been to try to simplify the extraction of rules that must be specified in BCRL expressions. This extraction is complex since it implies that the user is familiar with this language, an aspect that is not common and that implies that an RASE/BCRL expert assists the user in carrying out this task. To try to avoid this to a certain extent, an additional form is provided on the right side of the interface aimed at simplifying these actions and trying to abstract the user from the BCRL expressions in most cases. Below is a detail of how these functionalities have been implemented through a panel that appears on the right side of the interface.

After the user has selected a text and clicked on one of the Requirement, Application, Selection, or Exception buttons, a panel appears offering three main options for the user to choose from “object”, “property”, or “reference”, which are described below:

1. **Object.** The user can select this option when they wish to indicate that the regulatory text selected refers to the definition of an object (Figure 12). In BCRL expressions, objects are defined through the following expression: “type == :Name_of_the_object.” in the screenshot above. The name of the object can correspond to the word selected in the text or the user can provide a different object name if required .

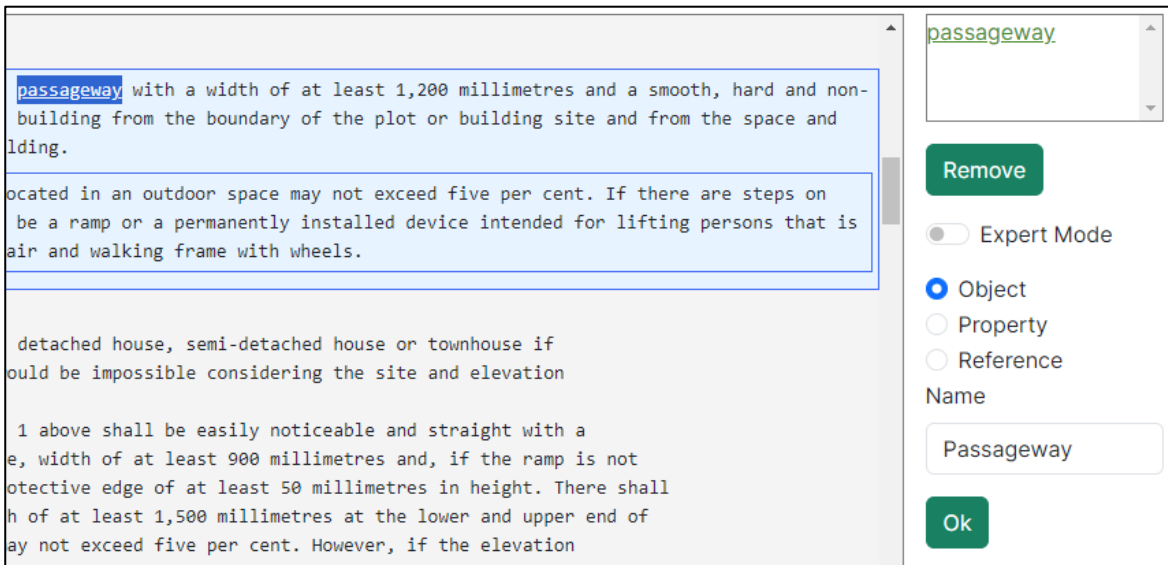


Figure 12. Screenshot of the interface with the RASE TEXT tagging panel in the right side: defining an object.

2. **Property.** The user can select this option when they wish to indicate that the selected regulatory text selected refers to a property (Figure 13). In this case, the tool tries to populate the input automatically, for the user to review. To do this, the tool has three internal lists:
 - A list containing typical properties used in the AEC domain (e.g., height, width, weight). This way, if a property name is contained in the selected text, this will be proposed as the default property name.
 - A list containing comparison operators, for example “more than, >” or “should not be less, >=”. If the selected text contains any expression from the list, then the associated operator is selected by default.
 - A list containing names of units. If the selected text contains any expression from the list, then the associated unit name is selected by default.

The name of the units is not restricted nor defined in BCRL expression. However, the tool makes sure that the name of the unit is valid by checking it against a list of accepted names. Also, the tool distinguishes whether a property is Boolean and, in this case, removes the part corresponding to the units from the panel.

by noticeable **passageway** with a **width of at least 1,200 millimetres** and a smooth, hard and non-slippery surface that leads to the building from the boundary of the plot or building site and from the space and area of the building.

A passageway located in an outdoor space may not exceed five per cent. If there are steps on the passageway there shall also be a ramp or a permanently installed device intended for lifting persons that is suitable for a wheelchair and walking frame with wheels.

This rule shall not apply to a detached house, semi-detached house or townhouse if the installation of a passageway would be impossible considering the site and elevation.

In subsection 1 above shall be easily noticeable and straight with a smooth and slippery surface, width of at least 900 millimetres and, if the ramp is not straight, a protective edge of at least 50 millimetres in height. There shall be a landing with a length of at least 1,500 millimetres at the lower and upper end of the ramp and the gradient of the ramp may not exceed five per cent. However, if the elevation difference of the ramp is more than 1,000 millimetres, the ramp may not have a gradient of more than 1:12. In any case, the elevation difference of a continuous ramp may not be more than 1,500 millimetres. In any case, there shall be a horizontal intermediate landing with a length of at least 1,500 millimetres. However, in an outdoor area the ramp may have a gradient of more than 1:12. The ramp shall be kept in a condition comparable with that of an indoor ramp. The ramp shall be equipped with handrails and other arrangements intended to prevent falling down and to ensure safety. In accordance with the provisions of section 117d, subsection 2 of the Land Use Act, provided for a building, an adequate number of them, but at least one,

width of at least 1,200 millimetres

Remove

Expert Mode

Object

Property

Reference

Name

Width

Operator

>=

Value

1200

Units

millimetres

Ok

Figure 13. Screenshot of the interface with the RASE TEXT tagging panel in the right side: defining a property.

- References.** There are situations in regulatory text where some articles or clauses refer to other parts of the text, whether it is a specific piece of information, a clause, or an entire section. It may also happen that these references refer to an external document. To make this relationship explicit, the user can select this option (Figure 14). For the first case, the tool provides a mechanism to allow the user to create an internal link to other content at the RASE text, RASE box or Section level. The mechanism includes an option so that when the reference has been made, the user can identify it by displaying the referred content framed within a rectangle with a red frame. For the second case, the tool provides a text field for the user to provide a URL or the name of the regulation being referenced.

The screenshot displays a software interface for tagging text. At the top, a red banner reads "Select a clause or section title to which the selected text refers to". Below this, the main content area shows "Section 2" with the heading "Passageway leading to a building". The text is annotated with several colored boxes: a blue box highlights a sentence about passageway requirements; an orange box highlights a sentence about gradient limits; a yellow box highlights an exemption clause; and another blue box at the bottom highlights a detailed ramp specification. On the right side, a control panel includes a dropdown menu with "subsection 1" selected, a "Remove" button, an "Expert Mode" toggle, radio buttons for "Object", "Property", and "Reference" (with "Reference" selected), a "Select type of reference" dropdown set to "Internal", and a "Select" button.

Figure 14. Screenshot of the interface with the RASE TEXT tagging panel in the right side: defining a reference.

RASE visualization and annotation actions are performed on an HTML file that is stored within the application. All modifications are automatically saved in this file. This involves any tagging action, whether creating RASE BOX clauses or tagging words or parts of text and confirming them. Therefore, it is not necessary to perform the entire tagging process at the same time. The user, for example, can perform tagging of one section one day and continue with other sections another day.

2.4 Architecture and components

As it is shown in Figure 15, the architecture of the rule formalisation tool consists of three main layers: (1) Data layer, (2) Business Logic Layer, and (3) Presentation Layer. These layers and their components are explained below.

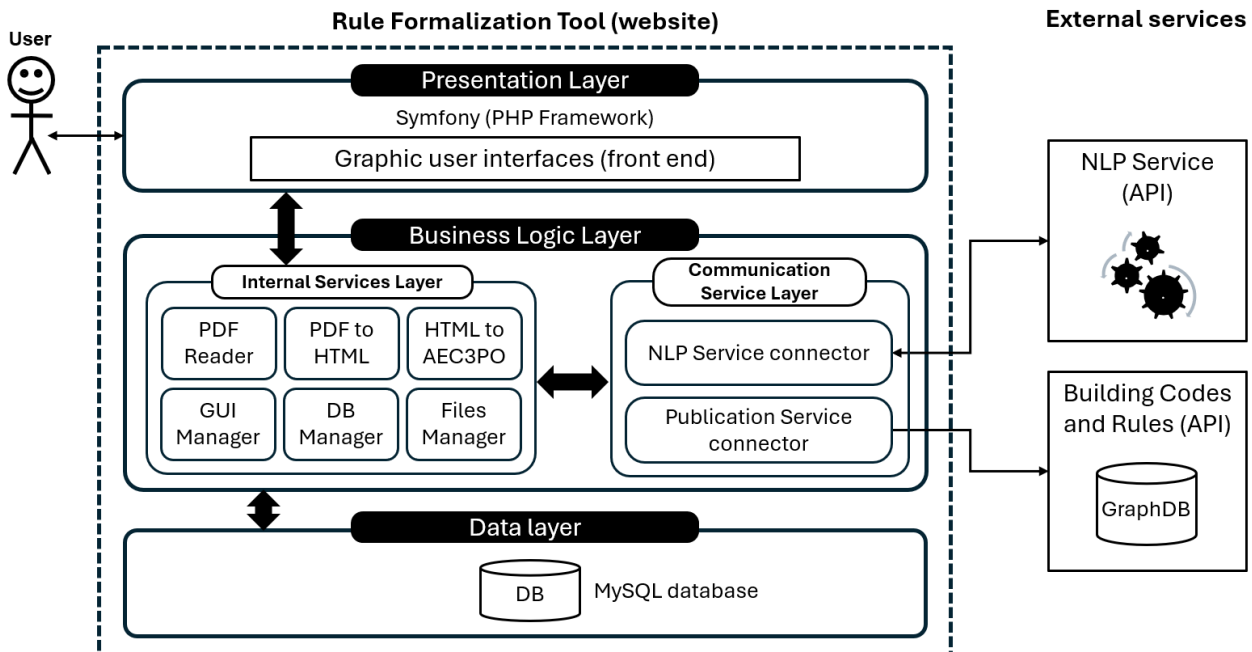


Figure 15. Diagram showing the architecture of the Rule Formalisation Tool.

Layers:

- **Data Layer.** It includes all the data involved and necessary for the management of projects for the formalisation of regulatory texts. This includes the management of the different files that must be created internally during the formalisation process depending on the stage for each user. The database has been implemented using the MySQL software package.
- **Business Logic Layer.** This is divided into two parts or sublayers:
 1. **Internal Services Layer.** It contains the modules with major functionalities of the tool. Three of them provide file management functionality, database connection, and GUI connection. The other three provide functionalities to transform one content into another depending on the stage of the formalisation process as follows:
 - a. **PDF Reader:** It converts the content of a PDF document provided by a user into plain text, maintaining a content layout similar to the original view by using whitespace characters. The output is a file in text format.
 - b. **TEXT to HTML:** It converts a plain text into an HTML document to enable the conduct of the RASE tagging process.
 - c. **HTML to AEC3PO:** Converts an HTML document tagged with the RASE method into instances of the AEC3PO ontology that include the corresponding relationships with the rules in BCRL language.

2. **Communications Service Layer.** It provides the link to the external modules of the tool and provides the communication between the data layer and them. The communication logic layer provides the mechanisms for obtaining data from the data layer and for introducing data into it from such external modules, acting as a middleware. External services are: (1) NLP Service and (2) Building Codes and Regulations API. This last service provides the connection with the semantic triple store (graphDB) where the graphs are stored. This communication is provided via API REST [5].
- **Presentation Layer.** It works as a point of interaction between the tool and the user. It provides the necessary functionality for the user to perform all the actions indicated in the formalisation process described in the previous section 2.2, through the corresponding interfaces.

2.5 Testing and results

2.5.1 Description

To evaluate the operation and user experience in the current prototype of the tool, the focus group method was chosen. A small group composed of four participants from four different partners of the project consortium was selected. All of these were familiar with or had some knowledge, experience, or connection with the area of elaboration, definition and/or application of regulations in building projects. However, to provide a “blind” test of the tool, no prior training was given about the use of the rule formalisation tool. This method was chosen to enable us to gather a true first impression of the use of the tool by reasonably qualified users.

The activity was performed in three stages:

1. **Testing the tool:** this exercise was designed to last approximately 30 to 45 minutes maximum, and it consisted of asking participants to carry out the process of formalising a regulation through the rule formalisation tool. To do this, each participant had to register in the tool and follow a series of steps from uploading a PDF of the regulation to publishing the graph in the rules database. The same PDF document was provided to all participants to be able to compare the results.

To carry out the exercise, a set of files were provided to carry out the test:

- a) A document with the instructions.
 - b) A PDF document with the regulations that participants had to use.
 - c) An HTML document with the RASE markup already applied that they had to use as a reference to try to reproduce the same tagging through the tool, since the test was not aimed at assessing their knowledge of the RASE method but rather the ability of the tool to apply it.
2. **Evaluation of results:** It was carried out through a questionnaire via Microsoft Forms, which all participants had to complete immediately after completing the exercise with the tool. The questions focused on collecting data of interest for evaluating the experience with the tool and for the focus group session.
 3. **Focus group session:** This was the final step of the activity and the one most closely linked to the concept of focus group. This exercise consisted of bringing together all the participants in a session where the most relevant aspects for the evaluation of the experience in carrying out the exercise through the tool were discussed in a common and guided manner.

2.5.2 Results

The testing and focus group activity was carried out by four participants, a woman and three men, with different profiles: an architect (participant 1), a researcher (participant 2), and two software developers (participants 3 and 4). The questionnaire consisted of two different types of questions: (1) assessment by scoring and (2) text description.

Evaluation questions were proposed to assess the tool in various aspects (previous user experience, functionality, usability, etc.) with a score between 1 and 5 (Table 1).

Table 1. Result of the questionnaire for rating questions.

Question	Rating par. 1	Rating par. 2	Rating par. 3	Rating par. 4	Rating Average
What is your knowledge and expertise in the field of definition and specification of building codes and regulations on a scale from 1 (beginner) to 5 (very expert)?	4	2	4	1	2.75
What is your expertise and knowledge in the field of application of building codes and regulations on a scale from 1 (beginner) to 5 (very expert)?	4	3	4	3	3.50
What is your knowledge and expertise in the RASE method on a scale from 1 (neophyte) to 5 (very expert)?	4	2	1	1	2.00
What is your knowledge about the BCRL language on a scale from 1 (no knowledge) to 5 (full knowledge)?	3	2	1	3	2.25
To what extent would you be able to perform the RASE tagging in a new regulation autonomously, without instructions, on a scale of 1 (impossible) to 5 (affordable)?	4	4	1	2	2.00
Please indicate whether you have had problems for tagging a text or word indicated in the instructions on a scale of 1 (never) to 5 (frequently)?	1	1	5	3	2.50
To what extent would you be able to perform the RASE tagging in a new regulation autonomously, without instructions, on a scale of 1 (impossible) to 5 (affordable)?	4	2	1	1	2.00
Based on the experience gained from this exercise, what do you consider to be the tool's ability to provide the formalization of a regulation on a scale from 1 (very cumbersome process) to 5 (easy to do)?	4	3	1	2	2.50
To what extent do you think this tool could be useful for law and policy makers on a scale from 1 (useless,) to 5 (totally useful)?	5	3	1	2	2.75

Key findings that can be summarised from the questions are:

1. Several users encountered minor bugs and issues in tagging the text using the RASE method.
2. Most participants who had no previous experience with the RASE method would have struggled to apply it independently or would find its application easy to do.

3. The level of previous experience in the application of the RASE method is directly linked to how useful participants feel the tool will be for law and policymakers.

The questionnaire results were discussed in the focus group session, which, in large part, focused on how to reduce the need to carry out the rule formalisation process with the assistance of a RASE expert.

Based on the feedback gathered from the questionnaire and the focus group, the following actions have been agreed:

1. All bugs and issues identified will be corrected – this has already been completed.
2. Training material on the RASE method will be created to guide new users of the tool in completing the digitisation process.

A final piece of feedback from the focus group was that using AI to automatically apply the RASE method and review the results will be key to easing the tool's adoption. However, it is important to note that knowledge of the RASE method will still be required to ensure users are able to accurately and confidently check the results generated by the AI process.

2.6 Future improvements

Although the tool meets all the functional requirements for which it was designed, fulfilling its purpose, several improvement actions can be implemented in the future. The most relevant ones are described below according to each category.

- **Connection to bSDD:** The buildingSMART Data Dictionary (bSDD) is a service for sharing definitions for describing the built environment to help agents of the AEC industry use agreed and consistent terms³. These definitions and relations are necessary to facilitate automation in the application of microservices developed within the ACCORD framework. Currently, existing tools allow the definition of the dictionaries required by the rule formalisation process in bSDD. Thus, to avoid re-creating existing tools, this has not been implemented within the RFT. In the future, it may be desirable to provide an interface within the tool for convenience. However, it should be noted that this task should probably be performed by a user with different knowledge and more familiar with the data structure of the IFC model.
- **Multiple languages:** Although the tool allows a user to upload files in different languages, the language of the interface is only in English. Therefore, one improvement would be to provide the interfaces in different languages.
- **Larger evaluations with actual potential users:** Regardless of whether the tool continues to develop throughout the project, new tests will need to be carried out aimed at a broader audience to give greater validity to the usefulness of the tool in each of the aspects to be assessed.

³ <https://www.buildingsmart.org/users/services/buildingsmart-data-dictionary/>

3. AI-powered Rule Formalisation for Building Codes

Compliance checking is integral within the Architecture, Engineering, and Construction (AEC) sector to ensure the safety, stability, reliability, and usability of building designs. Conventionally, compliance checking relied on manual methods, which proved to be labour-intensive, time-consuming, expensive, and error-prone [6]. Thus, there has been a significant shift towards Automated Compliance Checking (ACC), which has been extensively studied over the past 50 years [1]. The aim of ACC is to enhance the accuracy and productivity of compliance-checking processes while addressing the shortcomings of manual approaches and meeting the evolving demands of the industry.

Primarily, compliance checking is a two-fold process. Initially, it requires identifying the relevant building codes or regulatory requirements applicable to a building design, followed by verifying compliance against these requirements. Comparatively, the second phase is less challenging for automation, given that modern building designs are typically generated using computer software, resulting in digital or machine-processable formats. However, a significant challenge arises from the fact that building codes are mainly written in textual documents intended for human consumption, such as domain experts. Natural languages are hard for computers to process automatically due to their unstructured nature and inherent complexities arising from human-centred design [7]. These challenges are further amplified in the regulatory texts by their complex structures, ambiguities, and domain-specific characteristics. Thus, in enabling ACC, formalising, or interpreting the information conveyed in regulatory text into machine-processable formats emerged as a pivotal and complex phase [8].

To address this critical need, an AI-powered rule formalisation suite, leveraging recent advancements in NLP, has been developed as part of the ACCORD project. This section provides a comprehensive overview of the background, design and implementation approaches, and findings of this development. Section 3.1 introduces the concept of rule formalisation, presenting a summary of previous approaches and outlining ACCORD's objectives within the context of state-of-the-art technologies. Subsequently, Section 3.2 elaborates further on the recent trends in NLP that this development has followed. Finally, Section 3.3 details various sub-components (i.e., data, AI models and workflows) developed by ACCORD to facilitate automatic rule formalisation. The outputs presented in Section 3 can be used as stand alone models and tools or integrated in other tools as exemplified by the RFT of ACCORD.

3.1 Rule Formalisation

Formalisation is a process of refining something into a more precise and structured format. In the context of Architecture, Engineering, and Construction (AEC), rule formalisation particularly involves converting regulatory information, typically written in text, into a structured or standardised representation that computers or machines can readily understand and process. This conversion is essential to automate the compliance checking processes. Moreover, it improves the clarity and consistency of information, thereby facilitating more effective knowledge base generation.

Previous research within AEC has proposed various approaches for rule formalisation from text, aiming to enhance the effectiveness of ACC. In early work, manual methods were popularly used for this task due to text complexities and domain-specific demands [2, 9]. However, the labour-intensive nature of manual approaches and their limitations in efficiently supporting ACC prompted a shift towards automating information extraction utilising NLP and Machine Learning (ML) techniques. This shift mainly introduced rule-based approaches for text formalisation [10, 11, 12]. Despite their

performance, rule-based methods inherently lacked adaptability and flexibility, heavily relying on domain-specific characteristics [13]. Their accuracy highly depended on manually crafted rules, requiring extensive domain expertise. These limitations have encouraged a transition within rule formalisation methods towards ML techniques, leveraging predictive models to capture textual information rather than relying on laborious handcrafted rules.

According to recent ML-based research, adopting deep-learning techniques has emerged as a promising avenue for rule formalisation. Bidirectional Long Short-Term Memory (Bi-LSTM) and Convolutional Neural Network (CNN) architectures have been commonly used in extracting information from regulatory text due to their proficiency in learning long-term dependencies and contextual features [14, 15]. However, the approaches proposed within the AEC sector for rule interpretation exhibit notable gaps, particularly in light of recent advancements in the NLP domain. In NLP, Language Models (LMs) have gained significant recognition, showcasing state-of-the-art performance across various tasks, including information extraction from text [16, 17, 18, 19]. Overall, LMs represent a significant advancement in text processing with their sophisticated ability to capture text's contextual details and transfer pre-trained knowledge to downstream tasks, outperforming the models based on Recurrent Neural Network (RNN) architectures [20, 21]. Within the ACCORD project, our primary focus has been on bridging the gap between the AEC and NLP domains by proposing and exploring LM-based approaches for rule formalisation.

3.2 NLP Background

Within the Natural Language Processing (NLP) domain, there is a clear shift towards deep learning and language modelling-based text processing approaches across various applications. The improved text encoding and decoding using the advanced embedding techniques capable of automatically capturing underlying linguistics stands as a major driving force behind this transformation. During this transformation, both pre-trained Language Models (which are also known as transformers), and Large Language Models (LLMs) attracted wide attention from the NLP community recently following their more robust language understanding capabilities [22]. ACCORD is a pioneer in the use of LM-based approaches applied to AEC ACC rule formalisation.

3.2.1 Language Models/Transformers

Transformers are deep neural networks which utilise the attention mechanism to capture text's contextual details and long-range dependencies [23]. Their invention is a remarkable milestone in neural language modelling, following their capability to capture the influence on each word by another, notably enhancing natural language understanding. There are three main types of transformer architectures, namely encoder-only, decoder-only and encoder-decoder models. However, this work only utilised encoder-only models following its requirements in rule formalisation, and the following text will refer to such models as transformers for simplicity.

The encoder-only models primarily target encoding text into numerical representations that hold the underlying linguistics and contextual information. These models are originally built for language understanding tasks, such as text classification, in which the model requires predicting a label given the input text [22]. Following the general transformer encoder architecture, different model variants such as BERT [20], RoBERTa [24] and ALBERT [25] were proposed with the ability to generate contextual text representations and fine-tune for downstream tasks by transferring the pre-trained knowledge. These capabilities allowed the transformer-based architectures to achieve state-of-the-art results in many complex NLP tasks such as information extraction [18, 17], machine translation [26] and question answering [27].

3.2.1.1 Transformer Architecture

The transformer architecture consists of multi-layer bidirectional encoders that utilise the self-attention mechanism [23] to produce language representations which capture underlying linguistics and the context, as illustrated in Figure 16. It processes a sequence of text (e.g., sentence) and outputs representations/embeddings that correspond to the entire sequence and its tokens, which can be used to learn downstream tasks while conserving the original text's linguistic features.

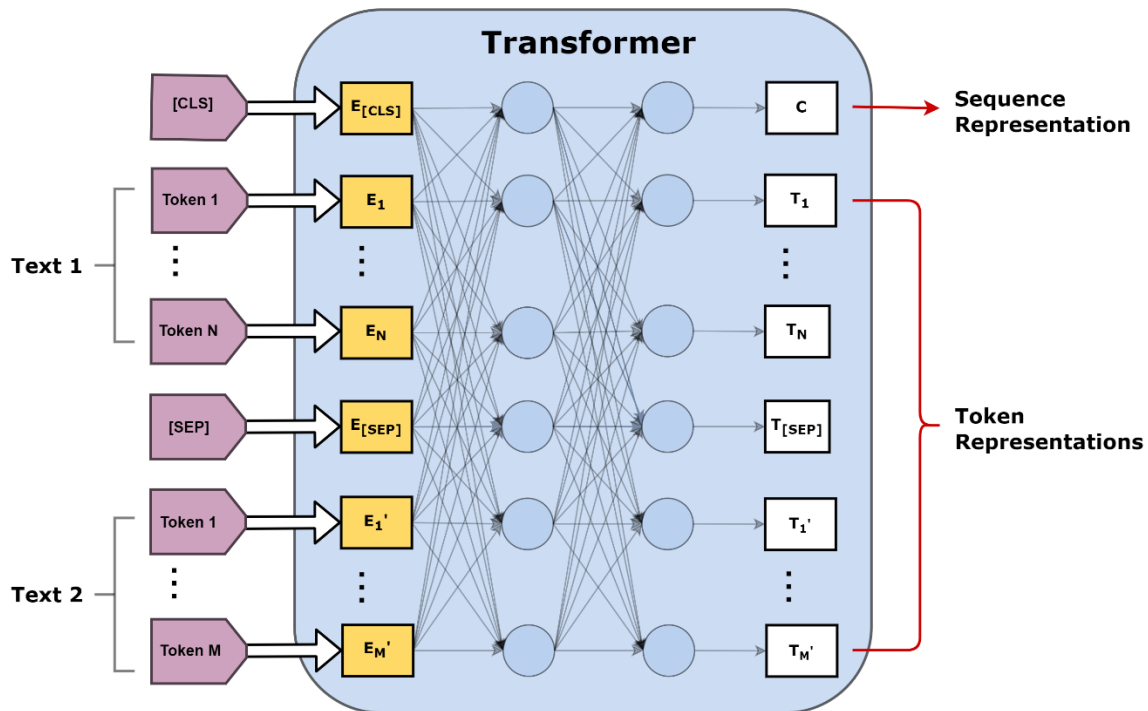


Figure 16. Transformer encoder architecture

Transformer Input Format: The transformer encoder takes a text sequence or a text sequence pair as its input, enabling it to handle diverse downstream tasks. It uses special tokens, including [CLS] and [SEP], to organise the input text. [CLS] is the initial token, which indicates the beginning of the sequence. [SEP] is the separator to be placed in between if the input has two sequences. After this raw text formatting, a tokeniser converts the sequence into a token embedding. In addition to the token embedding, a segment and a position embedding are required to format the input. The sum of these three embeddings constitutes the transformer's input. The segment embedding contains Boolean values (i.e., 0 and 1) indicating the separation of sequences/segments. The position embedding contains sequential numbers starting from 0, which specify each token's position within the sequence.

Transformer Output Format: The transformer encoder generates representations/embeddings per token in the input sequence. The output of the initial token ([CLS]) is an embedding representing the complete input sequence. It can be used for sequence-based predictions/classifications. Similarly, other outputs are token embeddings corresponding to each input token, which form contextual word/token embeddings that can be used with token-based predictions/classifications. The transformer architecture needs to be modified by including an additional layer, like a classification head suitable for the targeted task, on top of the output layer to fine-tune the model for a downstream task.

3.2.1.2 Learning Methods

Unlike other deep neural networks, transformers undergo two training steps according to their design: (1) pre-training/language modelling and (2) fine-tuning.

1. **Pre-training/language Modelling:** In the pre-training phase, a language model is built by training the transformer architecture on unlabelled textual data or free text. Therefore, pre-training is also known as language modelling. There are two commonly used pre-training approaches: (1) masked language modelling (MLM) and (2) next sentence prediction (NSP) [20]. The MLM approach randomly masks a portion (e.g., 15%) of the input tokens and then trains the model to predict the masked tokens. This task helps to learn bidirectional representations by training the model to predict masked tokens in a multi-layered context while focusing on both directions. NSP is a binary prediction task that focuses on recognising whether a sentence pair appears consecutively in a monolingual corpus. This task mainly helps the model to understand the relationships/interconnections between sentences. Among these two techniques, MLM is commonly used by various transformer architectures, such as RoBERTa [24] and ALBERT [25], as it was found to be competitive. Thus, we also involve MLM in language modelling within this work.
2. **Fine-tuning:** Fine-tuning is usually conducted by targeting a downstream task. To train a transformer for a downstream task, an appropriate layer(s), such as a classification head, needs to be added to the top of the transformer's output layer, depending on the targeted task. During this process, the model initialises using its pre-trained parameters and then gets fine-tuned for the targeted task utilising the task-specific labelled data. The initialisation with pre-trained parameters commences the learning process with the model's original knowledge and facilitates the effective learning of the downstream task. Fine-tuning can be applied to all model parameters (i.e., parameters of the transformer and its additional layers) or only to the additional layers. According to previous research, fine-tuning the transformer together with its newly-added layers obtained the best results for downstream tasks, as this approach adjusts the weights in the entire architecture, including the generic language model focusing on a particular task or domain-specific data rather than only fine-tuning the task-specific layers [28]. Thus, our approach also follows the entire fine-tuning process. More details about the modifications we made to the transformer architecture to support different text classification tasks within this effort are described in Section 3.3.

3.2.2 Large Language Models

Large Language Models (LLMs) are transformer-based language models with a vast number of parameters pre-trained on extensive text corpora [22]. In contrast to conventional language models or transformers, LLMs not only have larger model sizes but also demonstrate superior natural language understanding and generation capabilities, largely attributed to their extensive pre-training on massive free text datasets. LLMs also exhibit emergent capabilities that are not available in smaller-scale models or transformers, including (1) instruction following, (2) in-context learning, and (3) multi-step reasoning [22].

1. **Instruction Following:** Instruction following enables the model to perform a new task solely based on the given instructions without seeing any explicit examples [29]. This process is also referred to as zero-shot learning. In this setup, the quality of outputs heavily relies on the model's pre-trained knowledge and the clarity of instructions.
2. **In-context Learning:** In-context learning allows the LLM to learn a new task by only seeing a small set of examples provided through the prompt during inferencing [30]. This technique is also known as few-shot learning. This approach is particularly useful for complex tasks that

are challenging to explain without examples. Also, this helps mitigate data scarcity issues by requiring only a few samples for task learning.

3. **Multi-step Reasoning:** Multi-step reasoning provides LLM with the ability to solve a complex problem or task by breaking it down into a series of intermediate reasoning steps, also known as the chain of thoughts [31]. This strategy allows the model to solve simplified versions of the original task at each step, leading to the final output.

Moreover, LLMs are also capable of augmenting their knowledge using external sources [32] and improving themselves using reinforcement learning from human feedback [33], enabling them to effectively handle new tasks.

With the rapid advancement of LLM development, several model families have emerged as prominent players in the field. Among them, GPT, LLaMA and PaLM represent three popular families. Generative Pre-trained Transformer (GPT) models, developed by OpenAI, are decoder-only language models [21]. This family consists of various model versions, with the latest being GPT-4, released in 2023. Currently, GPT-4 is widely regarded as the most powerful model within this family, featuring the ability to process multi-modal data. LLaMA comprises a set of foundational language models developed by Meta [34]. Unlike GPT models, LLaMA models are openly available, making them accessible to a wider community. Even though the first LLaMA model was released in 2023, the family has now expanded to include several model variants, such as LLaMA, Alpaca, Koala, and more, showcasing significant growth. Pathway Language Models (PaLM), developed by Google, represent another notable family of LLMs [35]. Similar to the LLaMA models, PaLM models are free to use; however, they remain closed-source like GPT. Leveraging pretraining on extensive high-quality text corpora and a vast number of parameters, PaLM models exhibit robust multilingual and reasoning capabilities.

Even though LLMs are still in their early stages of development, they have gained tremendous popularity within the NLP community and beyond, emerging as general task resolvers, as showcased by Microsoft Co-Pilot. The rapid pace of advancement in this field introduces new models, strategies and findings within weeks or months. As a result of it, there is no established optimal approach for leveraging LLMs to perform a particular task. Also, researchers face challenges in identifying the most effective setups following this rapid growth. Therefore, our work also focuses on exploring various strategies to develop the most effective LLM-powered systems, as detailed in Section 3.3.

3.3 ACCORD-NLP Data, AI Models and Workflows

3.3.1 CODE-ACCORD: A Corpus of Building Regulatory Data for Rule Generation towards Automatic Compliance Checking

Useful Links:

- [GitHub Repository](#)
- [Hugging Face](#) (Datasets)
- [Annotation Manual](#)

Since building regulations are written in textual documents within the AEC sector, extracting information from textual rules to facilitate rule formalisation has been a challenge due to the complexities associated with natural languages. Recent NLP developments, especially with deep neural networks and language models, have overcome most existing challenges, surpassing traditional information extraction approaches. However, having annotated data as ground truth is crucial to train and evaluate such advanced models. Yet, to the best of our knowledge, there are no

readily available datasets within the AEC sector to support complete information extraction from regulatory text. We compiled CODE-ACCORD, following this requirement to empower the capacity to involve recent trends in NLP for ACC.

CODE-ACCORD comprises 862 self-contained sentences extracted from the building regulations of England and Finland. As a self-contained sentence, we refer to a regulatory sentence that expresses a rule and contains all the details itself without any linguistic co-references that are unresolvable within the sentence, references to external sources or incomplete/ambiguous concepts. Such sentences are essential for ACC as they express rules that can be directly extracted and interpreted without extensive cross-referencing or additional context. Aligned with our core objective of facilitating information extraction from text for machine-processable rule generation, each sentence was annotated with entities and relations. Entities represent specific components such as '*window*' and '*smoke detectors*', while relations denote semantic associations between these entities, collectively capturing the conveyed ideas in natural language. We manually annotated all the sentences using a group of 12 annotators. Each sentence underwent annotations by multiple annotators and subsequent careful data curation to finalise annotations, ensuring their accuracy and reliability, thereby establishing the dataset as a solid ground truth.

In summary, CODE-ACCORD's main contributions are as follows:

1. A novel data annotation approach to extract regulatory information from text covering entities and their semantic relations, which are integral for understanding the ideas conveyed in natural language.
2. An Entity-annotated dataset in English, sourced from a subset of England's and Finland's building regulations, which underwent a rigorous data annotation and curation process.
3. A Relation-annotated dataset in English, sourced from a subset of England's and Finland's building regulations, which underwent a rigorous data annotation and curation process.

More details about the CODE-ACCORD's data collection methodology are described in Section 3.3.1.1. Subsequently, Section 3.3.1.2 details the data annotation methodology. Finally, the information of datasets compiled by this effort together with data statistics, are included in Section 3.3.1.3. Also, the CODE-ACCORD approach and its findings have been submitted to the Data in Brief Journal, which is currently under review⁴.

3.3.1.1 Data Collection Methodology

CODE-ACCORD consists of the English Building Regulations and the English translation of the Finnish National Building Code, as we aimed to build a corpus in English⁵. In both countries, text regulations are published in PDF documents and available online to the public.

Table 2 presents a statistical overview of the approved documents. However, CODE-ACCORD excluded England's approved documents C, D, H and J from further processing due to some complex formatting associated with them.

⁴ CODE-ACCORD journal preprint is available on <https://arxiv.org/pdf/2403.02231>

⁵ The primary data were collected from the official websites of the [UK Department for Levelling Up, Housing and Communities and the Ministry of Housing, Communities & Local Government](#), and the [National Building Code of Finland from the Ministry of Environment](#).

Table 2. Building codes of England and Finland

Country	Approved Document/Decree	# Volumes	# Pages
England	A: Structure	1	54
	B: Fire Safety	2	384
	C: Site preparation and resistance to contaminants and moisture	1	52
	D: Toxic Substances	1	10
	E: Resistance to Sound	1	86
	F: Ventilation	2	110
	G: sanitation, hot water safety and water efficiency	1	55
	H: drainage and waste disposal	1	64
	J: Combustion appliances and fuel storage systems	1	89
	K: Protection from falling, collision and impact	1	68
	L: Conservation of fuel and power	2	220
	M: Access to and use of buildings	2	143
	O: Overheating	1	44
	P: Electrical safety	1	22
	Q: Security in dwellings	1	20
	R: Infrastructure for electronic communications	2	56
	S: Infrastructure for charging electric vehicles	1	47
Material and workmanship: Approved Document 7	1	24	
Finland	Accessibility	1	6
	Fire Safety	1	25
	Energy Efficiency	1	18
	Planning and Supervision	1	7
	Strength and Stability of Structures	1	55
	Safety of Use	1	9
	Health (Indoor Climate; Water and sewerage; and Humidity)	3	16
	Acoustic Environment	1	4
Total		33	1688

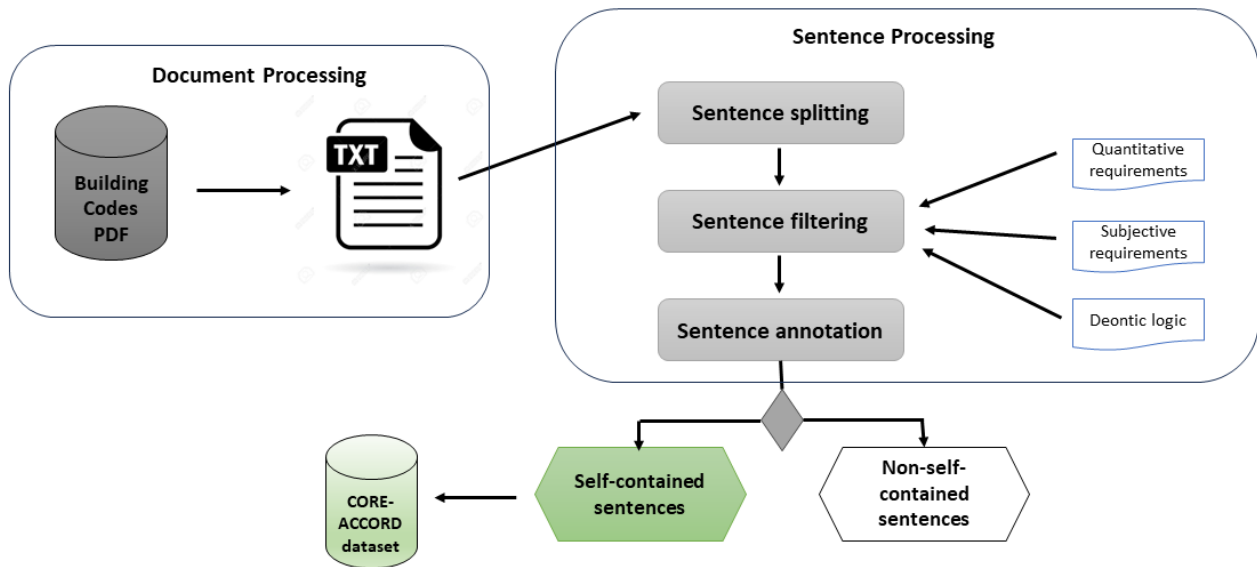


Figure 17. CODE-ACCORD semi-automatic data preparation approach

To extract the text data from the documents and prepare them for data annotation, a semi-automatic approach, as shown in Figure 17, is followed. It mainly contains two phases: (1) document processing and (2) sentence processing. Document processing primarily focused on converting PDF files to TXT files while preserving the textual structure and information. Initially, this step converted PDF files to TXT files using the PDFMiner library⁶. Then, the TXT files were further processed by removing line breaks, footnotes, tables, figures, and unnecessary sections to have the cleaned versions ready for sentence processing. Sentence processing is mainly aimed at extracting and filtering sentences for the entity and relation annotations, following the three steps below.

1. **Sentence Splitting:** This step splits the regulatory text into individual sentences, allowing the next steps to process sentences.
2. **Sentence Filtering:** During this step, the sentences were automatically filtered to extract sentences containing rules based on three distinct features.
 - a. **Qualitative Requirements:** Qualitative requirements are specific conditions expressed numerically or with quantitative terms. These requirements often specify precise values, measurements, or numerical criteria that must be met to ensure compliance with the regulations. Examples of quantitative requirements may include keywords such as 'less than', 'greater than', 'equal', 'at least', 'higher than', 'more than' and 'lower than', followed by numerical values or thresholds. The quantitative requirements are considered since they are mostly used in building codes for describing requirements [8].
 - b. **Subjective Requirements:** Subjective requirements are stipulations that involve the use of subjective language or expressions. These requirements are not defined by precise numerical values or measurements but rather by language that conveys recommendations, preferences, or suggestions. Subjective requirements often include terms like 'should be', 'recommended', 'preferred' or 'advisable'. While subjective in nature, these requirements are important in building regulations as they allow for flexibility and adaptation to different situations while still providing a framework for best practices and quality standards. To the best of our knowledge,

⁶ PDFMiner documentation is available on <https://pypi.org/project/pdfminer/>

existing research in the field of applying NLP for the automation of building regulations has not addressed subjective requirements in their analyses, methodologies, or datasets [8].

- c. **Deontic Logic:** Deontic logic refers to the logic that deals with the expression of permissions, obligations, prohibitions, and other normative statements. It is used to represent rules and requirements that are binding or mandatory, such as rules that specify what *'must'*, *'shall'*, *'could'* or *'prohibit'* within building regulations. Deontic logic plays a crucial role in modelling the normative aspects of these regulations, providing a formal framework to represent and reason about mandatory and discretionary requirements. Similar to subjective requirements, deontic logic has not been extensively considered in previous research efforts. This is primarily due to the focus of most research on quantitative requirements, given their higher frequency within building regulations [8].
3. **Sentence classification:** The final step classified each filtered sentence as a self-contained or non-self-contained sentence. This was performed manually to remove any false positive sentences identified during the sentence filtering step. A self-contained sentence is a regulatory sentence that expresses a rule and contains all the details itself without any linguistic co-references that are unresolvable within the sentence, references to external sources or incomplete/ambiguous concepts. Following this step, all non-self-contained sentences were excluded from the sentence collection.

Completing the data preparation approach resulted in a self-contained sentence collection ready for the manual entity and relation annotation process, as summarised in Table 3.

Table 3. Statistical summary of data collection

Country	Sentence Count	Self-contained Sentence Count
England	19201	963
Finland	1473	283
Total	20674	1246

3.3.1.2 Data Annotation Methodology

Since CODE-ACCORD's primary focus is facilitating information extraction from regulatory text required for rule formalisation, two key types of information: (1) entities and (2) relations, which are essential for comprehending the ideas conveyed in natural language [36], were aimed during data annotation. Our annotation group contained 12 annotators with either a computer science or a civil engineering/construction background. Since this work targets the automation of compliance checking using machine learning concepts, we believe it is important to involve experts from both areas in the annotation process. To collect human annotations, we used the LightTag text annotation platform [37], considering its text annotation coverage, including entities and relations, project management support and user-friendly interfaces.

We conducted the entity and relation annotations in rounds. Before moving into the actual rounds, there were two test rounds using a team of two annotators to refine the annotation strategy. Once the strategy was finalised, there were seven rounds for actual annotations. Each sentence was annotated by two to three annotators through these rounds. Given the complexity associated with annotations due to the multi-step annotation process, which is further explained below, each

annotation round is followed by a curation round to determine the final annotations. Three members joined as data curators, and their curation jobs were assigned without overlaps with the annotation jobs. During curation, the curator decided on the final annotation for all entities and relations with disagreements between annotators, considering the proposed annotations. More details about entity and relation annotation approaches are described below.

Entity Annotation: Entities are specific pieces of information or concepts that can be categorised. Or, simply, they are anything that can be referred to using a proper name [36]. Following the idea proposed in [8] [13], we picked four named entity categories, described in Table 4, for entity annotation. However, deviating from previous work, CODE-ACCORD adopted a simple category structure, mainly aiming at the generalisability of our annotation approach across different subdomains, such as structure, fire safety and accessibility, and information coverage, when defining the named entities. A two-step annotation process: (1) mark entity text spans and (2) assign entity categories was used for entity annotations, and annotations are conducted in rounds as described above. A few annotated samples are shown in Table 5. As can be seen, the selected categories are versatile enough to capture all entities in different sentence structures. Also, these samples are from Accessibility and Fire Safety regulations to indicate the general applicability of our annotation strategy in different subdomains.

Table 4. Entity categories. A colour theme is used to enhance the clarity of the sample annotations given in this document.

Category	Description
object	An ontological concept which represents a thing that is subject to a particular requirement (e.g., window, fire door)
property	Property of an object (e.g., width, height)
quality	Quality or uncountable characteristic of an object/property (e.g., horizontal, self-closing)
value	A standard or a numerical value that defines a quantity (e.g., 1,500 millimetres, five per cent)

Table 5. Sample entity annotations

Original Sentence	Annotated Sentence
The gradient of the passageway located in an outdoor space may not exceed five per cent.	The <property>gradient</property> of the <object>passageway</object> located in an <object>outdoor space</object> may not exceed <value>five per cent</value> .
There shall be a horizontal landing with a length of at least 1,500 millimetres at the lower and upper end of the ramp.	There shall be a <quality>horizontal</quality> <object>landing</object> with a <property>length</property> of at least <value>1,500 millimetres</value> at the <property>lower and upper end</property> of the <object>ramp</object> .
A fire door must be self-closing and self-bolting.	A <object>fire door</object> must be <quality>self-closing</quality> and <quality>self-bolting</quality> .

Relation Annotation: Relations are semantic connections/associations among entities in the text [36]. Extraction of relations together with entities is a crucial process to transform information embedded in unstructured texts into structured data formats such as knowledge graphs. Altogether, we picked ten relation categories described in Table 6 after carefully analysing the possible relations in the regulatory text. Similar to our approach in defining entity labels, we mainly focused on the generalisability across different subdomains and coverage of semantic information when identifying these relation categories. The final category, '*none*', is added considering the potential model requirements for identifying instances without relation between entity pairs. Comparatively, relation annotation requires a more complex process than the entity annotation process. As a result, a four-step process: (1) mark entity text spans, (2) assign entity categories, (3) identify entity pairs which form relations, and (4) assign relation categories, was used. This task has proven more challenging than entity annotations, primarily due to its multiple intricate steps and the potential for error propagation. However, we instructed the annotators to adhere to the entire flow, allowing them the flexibility to highlight all relevant content simultaneously. Furthermore, this approach enabled them to review the provided annotations by examining the final entity-relation representation. We only applied the manual annotation process targeting the first nine entity categories without the '*none*' category because once all the available relations are known, the remaining possible entity pairs form the no relations. A few annotated samples are shown in Table 6.

Table 6. Relation categories

Category	Description
selection	A limit to the scope of an object/property based on another object, a quality or a user
necessity	A qualitative/subjective or existential necessity of an object/property (e.g., should, should have, shall be, etc.)
part-of	Being a part of an object/property
not-part-of	Not being a part of an object/property
greater	A value that should be greater than to
greater-equal	A value that should be greater than or equal to
equal	A value that should be equal to
less-equal	A value that should be less than or equal to
less	A value that should be less than to
none	No relation

Table 7. Sample relation annotations

Original Sentence	Entity Pairs	Relation
The gradient of the passageway located in an outdoor space may not exceed five per cent.	The <property>gradient</property> of the <object>passageway</object> located in an outdoor space may not exceed five per cent.	part-of
	The gradient of the <object>passageway</object> located in an <object>outdoor space</object> may not exceed five per cent.	Part-of
	The <property>gradient</property> of the passageway located in an outdoor space may not exceed <value>five per cent</value> .	less-equal
Entity-relation Representation		
There shall be a horizontal landing with a length of at least 1,500 millimetres at the lower and upper end of the ramp.	There shall be a <quality>horizontal</quality> <object>landing</object> with a length of at least 1,500 millimetres at the lower and upper end of the ramp.	selection
	There shall be a horizontal <object>landing</object> with a <property>length</property> of at least 1,500 millimetres at the lower and upper end of the ramp.	part-of
	There shall be a horizontal landing with a <property>length</property> of at least <value>1,500 millimetres</value> at the lower and upper end of the ramp.	greater-equal
	There shall be a horizontal <object>landing</object> with a length of at least 1,500 millimetres at the <property>lower and upper end</property> of the ramp.	necessity
	There shall be a horizontal landing with a length of at least 1,500 millimetres at the <property>lower and upper end</property> of the <object>ramp</object> .	part-of
Entity-relation Representation		

3.3.1.3 CODE-ACCORD Datasets

CODE-ACCORD consists of two main annotated datasets, including entities and relations, which are publicly available⁷. These annotations were applied to 862 self-contained sentences extracted from the building regulations of England and the English translation of the National Building Code of Finland (Section 3.3.1.1). Figure 18 illustrates the sequence length distribution of the selected sentences, revealing that a majority consist of fewer than 40 tokens.

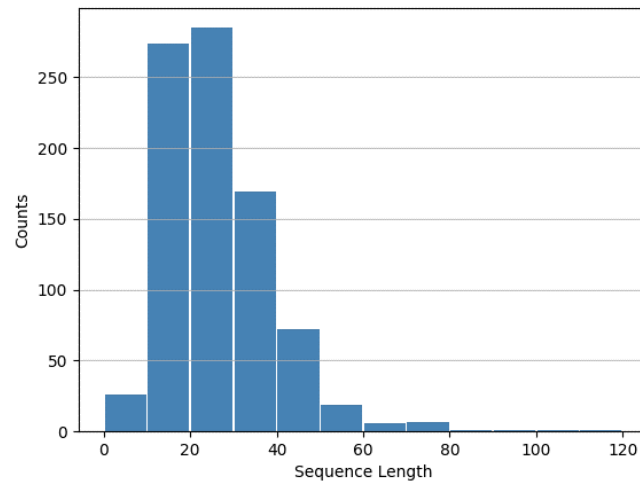


Figure 18. Sequence length distribution of annotated sentences in the CODE-ACCORD dataset

The format of an entity-annotated CSV data file is summarised in Table 8. The entity annotations are available in the BIO (Beginning, Inside, Outside) format, which is considered the standard for information extraction tasks [38]. Since an entity can span over multiple words/tokens, ‘B’ marks the beginning word, and ‘I’ marks the other words which belong to the entity. All the remaining words that do not represent any entity will be marked with ‘O’. A sentence together with its BIO-tagged entities is shown below.

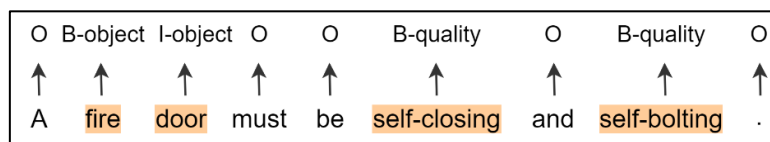


Table 8. Format of entity data file

Attribute	Description
example_id	Unique ID assigned for each sentence/data sample
content	Original textual content of the sentence
processed_content	Tokenised (using NLTK’s word_tokenize package) textual content of the sentence
label	Entity labelled sequence in BIO format
metadata	Additional information of sentence (i.e. original approved document from which the sentence is extracted)

⁷ CODE-ACCORD datasets are accessible through Hugging Face on <https://huggingface.co/ACCORD-NLP>

The format of a relation-annotated CSV data file is summarised in Table 9. The following format was adopted to tag the entity pairs during relation data formatting in accordance with formats utilised in recent studies [17] [39]. The special tags <e1> and </e1> represent the start and end of the first entity that appeared in the sentence. Similarly, <e2> and </e2> represent the second entity.

The <e1>gradient</e1> of the <e2>passageway</e2> located in an outdoor space may not exceed five per cent.

Table 9. Format of relation data file

Attribute	Description
example_id	Unique ID assigned for each sentence/data sample
content	Original textual content of the sentence
metadata	Additional information of sentence (i.e. original approved document from which the sentence is extracted)
tagged_sentence	Sentence with tagged entity pair
relation_type	Category of the relation between the tagged entity pair

More details about the final statistics of entity and relation data are described below.

Entity Statistics: The CODE-ACCORD entities dataset has 4,297 entities distributed over four categories, as shown in Figure 19. As can be seen in Figure 20, which illustrates the distribution of the number of entities per sentence, most sentences consist of up to five entities. Figure 21 presents the sequence length distribution of text spans from each category. Accordingly, most entity spans are composed of one or two words/tokens. However, overall, there are more lengthy text spans under 'quality' than in the other categories.

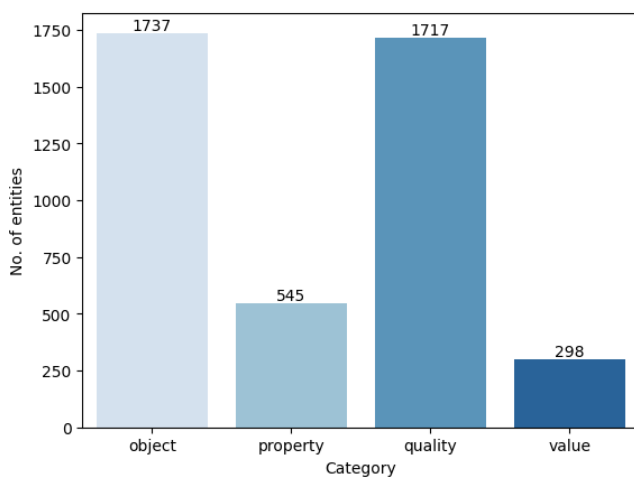


Figure 19. Distribution of entity categories

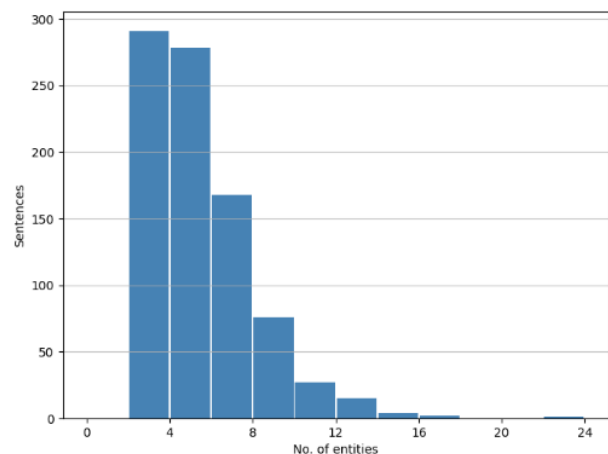


Figure 20. Distribution of the number of entities per sentence

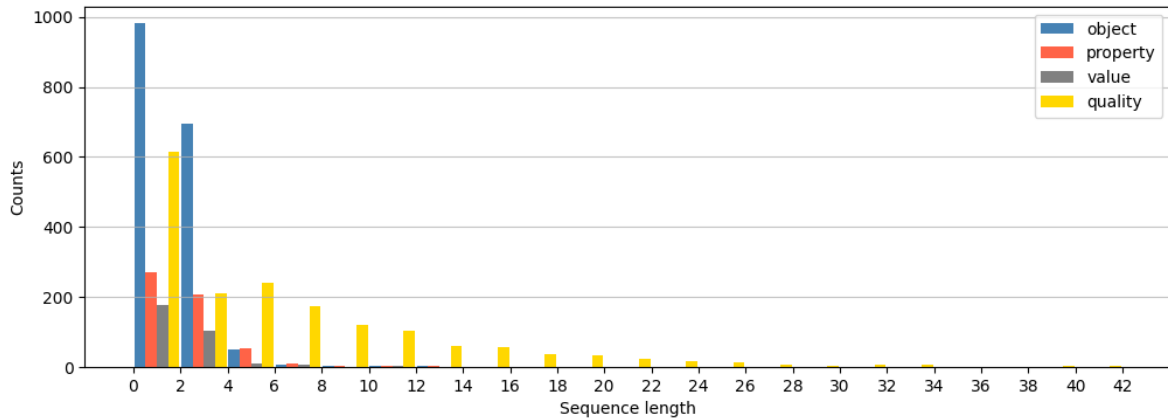


Figure 21. Sequence length distribution of annotated text spans as entities

Relation Statistics: The CODE-ACCORD relations dataset has 3,329 human-annotated relations over nine categories. We automatically identified the unannotated entity pairs within sentences as unrelated entity pairs which belong to the tenth category of 'none'. Out of 8,104 samples categorised as 'none', a random subset of 1,000 is included in the final dataset to ensure a balanced distribution with other relations. The breakdown of a total of 4,329 relations across ten categories is shown in Figure 22. Additionally, Figure 23 illustrates the distribution of the number of relations per sentence. As can be seen, most sentences contained two or three relations, although a minority had over ten relations.

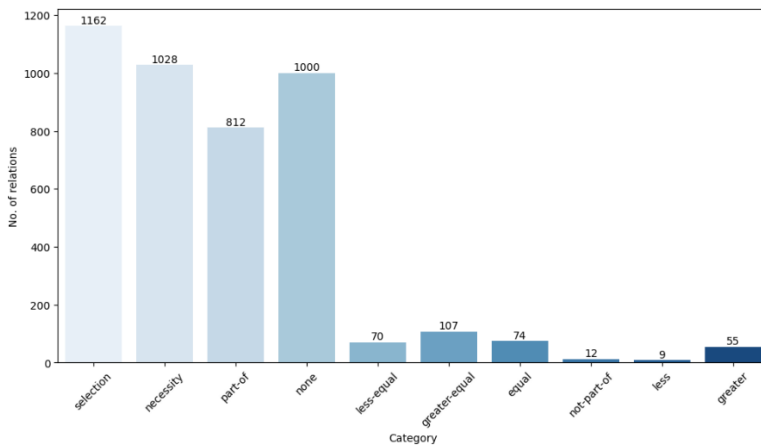


Figure 22. Distribution of relation categories

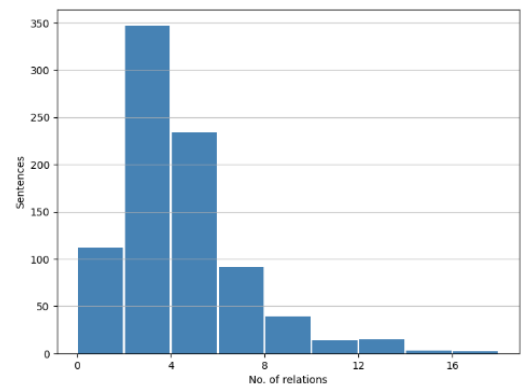


Figure 23. Distribution of the number of relations per sentence

3.3.2 SNOWTEC: Synthetic Natural Language Oversampling with Transformer-based Information Extraction for Automated Compliance Checking

Useful Links:

- [GitHub Repository](#)
- [Hugging Face](#) (Datasets, Models and Demo)
- [Python Package](#)
- [Live Demo](#)

Building codes are primarily written in textual form, requiring extracting information from text to decode these data to support rule formalisation. This requirement encouraged the development of various Information Extraction (IE)/text formalisation techniques spanning manual, rule-based and machine-learning methodologies over the past decades. Recent research has shown promise in adopting deep learning; however, as far as we know, within Architecture, Engineering, and Construction (AEC), the transformers/language models' potential remains untapped/unexplored for this task, yet they hold state-of-the-art performance across various Natural Language Processing (NLP) tasks. To address this gap, we proposed SNOWTEC, harnessing transformer-based architectures to extract information from regulatory sentences and transform it into standardised formats (i.e., knowledge graphs). We particularly focused on self-contained sentences within this development, which express rules while containing all the details themselves without any linguistic co-references that are unresolvable within the sentence, references to external sources or incomplete/ambiguous concepts. Such sentences form an essential component within Automated Compliance Checking (ACC), following their direct expression of rules that can be extracted straightforwardly.

There are two main types of information found in the text: (1) entities (also known as named entities) and (2) relations, which are crucial for understanding the ideas expressed in natural language [36]. An entity is a concept or a specific piece of information that can be categorised or, simply, anything that can be identified using a proper name [36]. For instance, given the sentence *'The gradient should not exceed five per cent'*, *'gradient'* and *'five per cent'* represent its entities. A relation is a semantic connection/association among entities in the text [36]. For example, the sentence mentioned above illustrates a *'less equal'* relation between the entities: *'gradient'* and *'five per cent'*. Collectively, these entities and their relations capture the rule(s) expressed in natural language.

Altogether, SNOWTEC included the development of a comprehensive pipeline capable of converting regulatory sentences into knowledge graphs of entities and relations, streamlining ACC. Concerning rule formalisation, unlike the many approaches which focus on converting regulatory text into rule languages like SPARQL [40] [41], our focus lies on an intermediate, yet descriptive, representation. Specifically, we leverage knowledge graphs to depict textual information in formats readable by both humans and machines, facilitating the easy expansion of knowledge. Also, the human-readable aspect enhances human-in-the-loop reviewing processes, ensuring the correctness and trustworthiness of the outputs. This approach fosters a collaborative environment, promoting confidence in the accuracy of the extracted information.

The primary aim of SNOWTEC was to utilise the transformers' potential for IE from regulatory sentences within the AEC sector, covering both entities and relations, to facilitate effective rule formalisation. However, besides the modelling approach, the quality and quantity of annotated data used to train the models significantly influence the final performance in predictive modelling. There is a noticeable scarcity of annotated data in the AEC sector specifically targeting IE from the regulatory text, which also can be seen as one of the main reasons for the limited adaptation of

recent NLP techniques. Data augmentation/oversampling has recently emerged as a popular solution in the NLP domain to combat data scarcity issues. However, there is a notable absence of data augmentation applications within the AEC sector focusing on textual data, and our efforts within SNOWTEC also target addressing this gap by proposing a novel data augmentation method. Moreover, our experimental studies compare the performance of several recent transformer models, including BERT [20], RoBERTa [24] and ALBERT [25], employing diverse learning techniques. Also, data from multiple domains such as accessibility, fire safety, structural integrity, energy efficiency and more are involved in evaluating the general applicability of the proposed methodology for IE.

In summary, SNOWTEC’s main contributions are as follows.

1. A novel information extraction pipeline utilising state-of-the-art transformer/language models to automatically extract information from the regulatory sentences in a domain-independent manner.
2. A novel data oversampling/augmentation method designed to address data scarcity concerns that impede the predictive capabilities of language models.
3. Formulation of the information extraction process from the regulatory text as a knowledge graph generation task, introducing a new direction to facilitate effective rule formalisation for ACC.

More information about the design and development of the SNOWTEC pipeline is described in Section 3.3.2.1. Section 3.3.2.2 summarises the experimental studies conducted on SNOWTEC, including model performances and error analyses. Finally, Section 3.3.2.3 demonstrates the functionality of the entire pipeline with sample inputs and outputs. Also, a paper written based on the SNOWTEC approach, and its findings have been submitted to the Computers in Industry Journal, which is currently under review.

3.3.2.1 SNOWTEC Information Extraction Pipeline

SNOWTEC, the information extraction pipeline developed as a part of the ACCORD project, primarily aimed at producing a machine-processable output (i.e., an entity-relation graph) that encapsulates the information expressed in natural language, given a regulatory sentence as the input. This graph can be readily processed to automate compliance checks while supporting human-in-the-loop review processes.

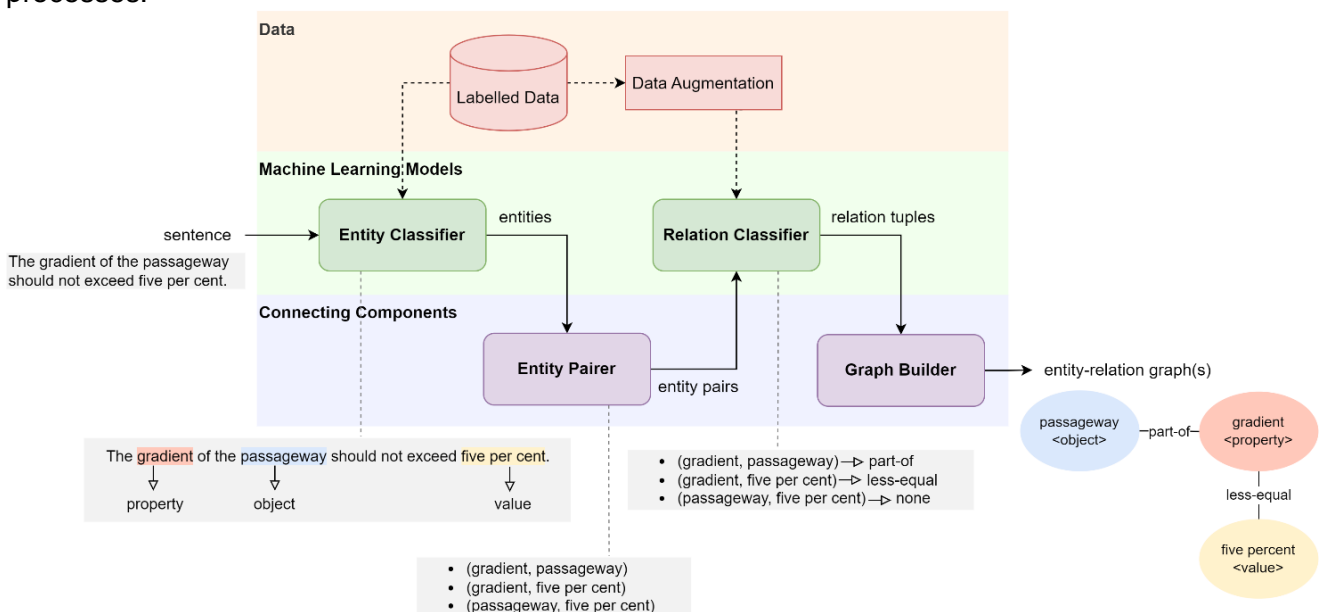


Figure 24. Overview of the SNOWTEC pipeline

Overall, the SNOWTEC pipeline comprises four integral components: (1) entity classifier, (2) entity pairer, (3) relation classifier and (4) graph builder. The overall pipeline, which includes input, output, and intermediate data samples, is illustrated in Figure 24. Entity and relation classifiers are machine learning models that utilise state-of-the-art language model/transformer-based architectures to extract entities and relations from textual data effectively. Also, the proposed relation classification approach is coupled with a novel data augmentation method to synthetically generate data to train language models effectively. These components play a pivotal role within this pipeline, and they are further described below. The entity pairer and graph builder act as connecting components that transform data to support the data flow within the pipeline. Specifically, the entity pairer translates the outputs generated by the entity classifier into the input format required by the relation classifier. Graph builder transforms the predictions of the relation classifier into the final knowledge graph.

Entity Classifier: For entity classification, SNOWTEC involves a modified version of the general transformer architecture shown in Figure 16 with softmax layers for each output token, as illustrated in Figure 25. This architecture was commonly used within the NLP domain for sequence labelling following its remarkable results [20] [42]. We formulated the entity classification task as a sequence labelling problem to fit this architecture, as all the input tokens can be associated with a corresponding entity label. The same format (i.e. BIO - Beginning, Inside, Outside format) used with CODE-ACCORD entity formatting was involved here as the model's data format (Section 393.3.1.3). The raw text was passed to the model as input, allowing it to predict the BIO label with entity category as the output. Since the model requires processing a single sentence per instance, we only use the [CLS] token for input formatting without including the [SEP] token. Each softmax layer consists of k number of neurons equal to the class/category count aimed at by the classifier (i.e., entity types). Each neuron adopts the softmax activation function following Equation (1), which returns P_i , the probability per class i . The input and output vectors are represented by z_i and z_j . Once all probabilities are calculated, the class with the highest probability is identified as the final prediction.

$$P_i = \frac{e^{z_i}}{\sum_{j=1}^k e^{z_j}} \tag{1}$$

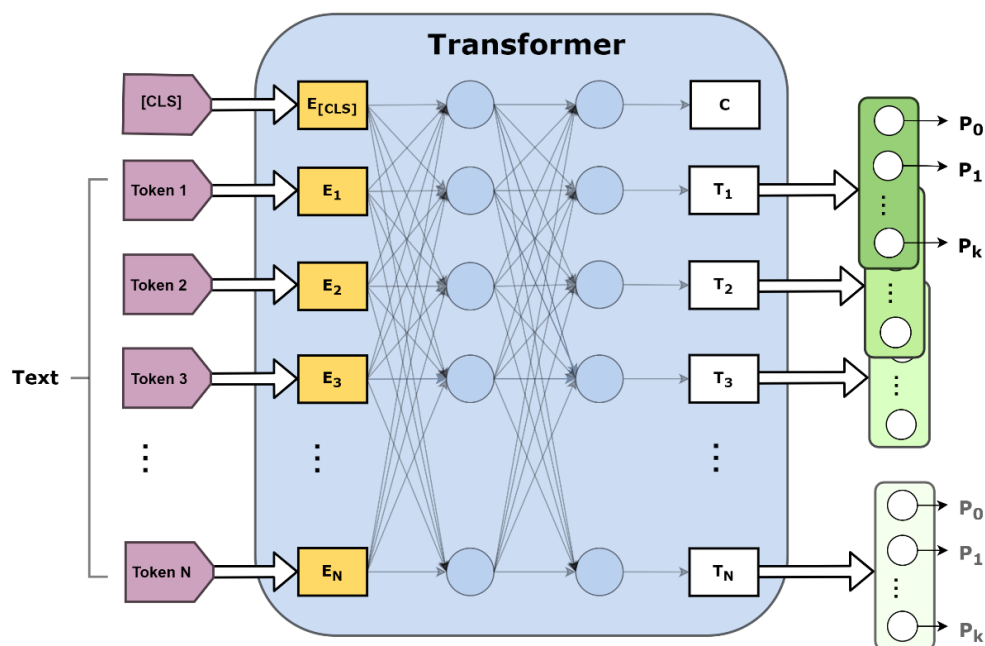


Figure 25. Entity classification architecture

Relation Classifier: For relation extraction, SNOWTEC adapts the transformer-based architecture shown in Figure 26, which was proposed by recent studies in the NLP domain [17] [43]. Since the aim is to identify the semantic connection between an entity pair, we involved four additional special tokens: $\langle e1 \rangle$, $\langle /e1 \rangle$, $\langle e2 \rangle$ and $\langle /e2 \rangle$ to format the model's input. $\langle e1 \rangle$ and $\langle /e1 \rangle$ denote the start and end of the first entity that appeared in the text sequence from the selected entity pair. Similarly, $\langle e2 \rangle$ and $\langle /e2 \rangle$ denote the start and end of the second entity. Among the transformer's special tokens, we only use [CLS] with the relation classifier, as it requires processing a single sentence per instance, similar to the entity classifier. From the output layer, we take the embeddings corresponding to $\langle e1 \rangle$ and $\langle e2 \rangle$ as linguistic representations for the given entities. Then, their concatenation is considered the final output representation, which is fed into a softmax layer to predict the associated relationship. As described under the entity classifier above, a softmax layer calculates the probability per class (i.e., relation types) following Equation (1). We pick the relation type with the highest predicted probability as the final prediction.

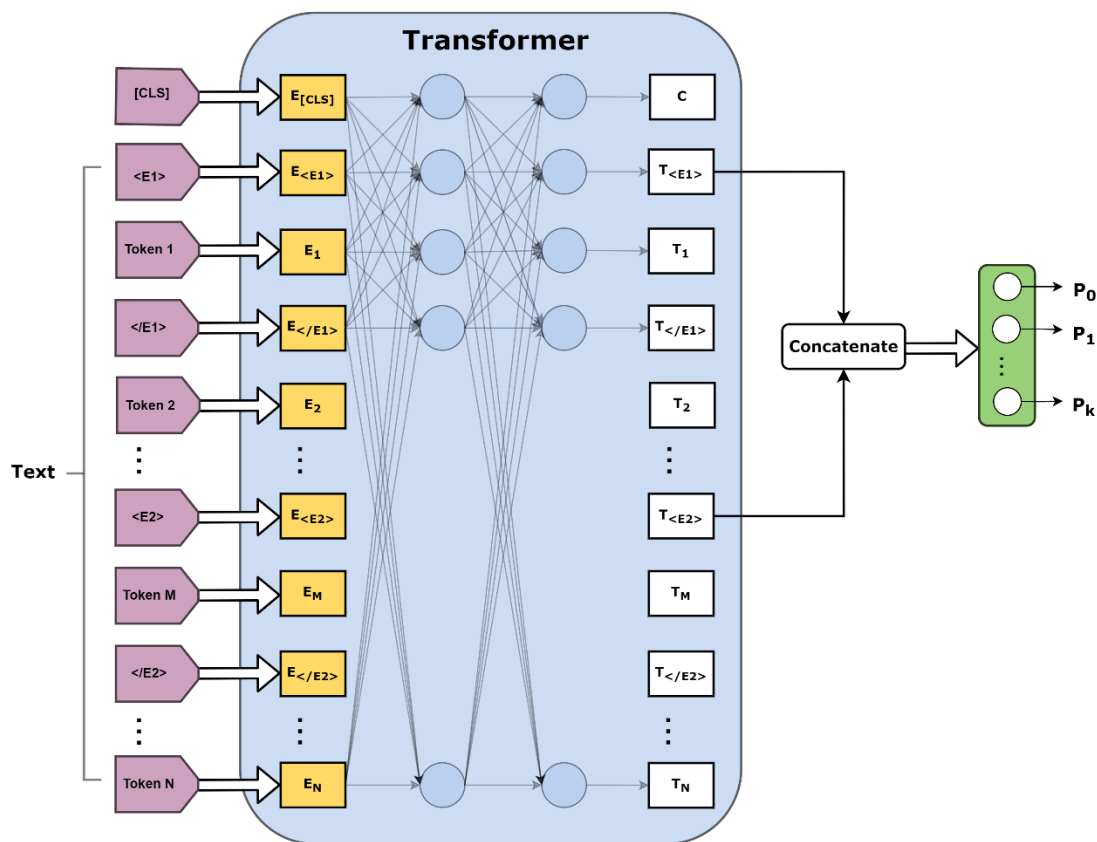


Figure 26. Relation classification architecture

Data Augmentation: Data augmentation is the process that constructs synthetic data from an available dataset. Such a process proves advantageous in scenarios with limited or unbalanced data, mitigating the risk of overfitting in model training [44]. Even though data augmentation has been widely used with image data, its utilisation in NLP is still at an early stage [45]. One of the primary challenges in augmenting textual data lies in preserving the intricate linguistic structure available in original data with synthetic data. Back-translation and random synonym replacement are the two popularly used data augmentation techniques within NLP, considering their ability to generate data without losing underlying linguistics [46] [47]. However, back-translation cannot be used when the original data includes token-level labels, as back-translated samples could have a different number of tokens with no direct mapping with the original tokens. Similarly, random

synonym replacement is not applicable for regulatory data, as the majority of the entities in regulatory data appear as multi-word instances (e.g., `fire door`, `cross-section area`, etc.) and replacing some words with their synonym could result in text phrases which are unknown to the domain or no longer entities. After carefully analysing the limitations in available data augmentation techniques, we developed a novel method to synthetically generate relation-annotated data utilising domain-specific entities, which is demonstrated in Table 10.

Table 10. Original sentence samples and their corresponding synthetic samples. Columns e1 and e2 represent the categories of each entity. A colour scheme representing categories is involved to highlight the entities in original and synthetic samples.

Original Sample	e1	e2	Relation	Synthetic Samples
Between the locking points for the mortice lock and surface-mounted rim lock, the <e1>distance</e1> should be <e2>400-600mm</e2>.	property	value	equal	Between the locking points for the mortice lock and surface-mounted rim lock, the <e1>cross-sectional area</e1> should be <e2>1m ³ </e2>.
				Between the locking points for the mortice lock and surface-mounted rim lock, the <e1>emission rate</e1> should be <e2>25W/m ² </e2>.
				Between the locking points for the mortice lock and surface-mounted rim lock, the <e1>power input from controls</e1> should be <e2>50</e2>.
<e1>Extinguishing routes</e1> of basement storeys must not be connected to fire and smoke-proof <e2>exits</e2>.	object	object	not-part-of	<e1>dwelling units</e1> of basement storeys must not be connected to fire and smoke-proof <e2>space heating or air-conditioning system</e2>.
				<e1>room thermostat</e1> of basement storeys must not be connected to fire and smoke-proof <e2>tears</e2>.
				<e1>kitchens</e1> of basement storeys must not be connected to fire and smoke-proof <e2>motor</e2>.

Each relation-annotated instance includes an entity pair marked in a sentence or textual context with corresponding entity categories and an associated relation, as can be seen in Table 10 (columns Original Sample, e1, e2 and Relation). In such a setting, we can create more samples by replacing one or both entities with other entities of the same category. Table 10 contains three augmented samples per original sample to illustrate the idea further. On some occasions, the generated samples could hold false information. For instance, the first synthetic sample in Table 10 states an area (i.e., cross-sectional area) should be equal to a volume measure (i.e., $1m^3$). However, it illustrates the text pattern and semantic relationship between two such entities within the given context accurately, allowing a model to learn associations among entities given a textual context. This fact is also reinforced by the results reported in Section 3.3.2.2. Also, this approach is widely applicable to any entity-relation dataset as there are no data-specific constraints.

3.3.2.2 Model Performance Overview

As mentioned above, SNOWTEC consists of two machine learning models that extract entities and their semantic relations during the information extraction process. We primarily utilised the human-annotated data available with the CODE-ACCORD datasets to train the models and evaluate their performance.

Human-annotated Data: CODE-ACCORD datasets (Section 3.3.1) were used to evaluate the transformer-based models within SNOWTEC. As of now, to the best of our knowledge, CODE-ACCORD stands as the sole publicly available dataset offering comprehensive entity and relation annotations specifically designed for building regulatory data. In addition to incorporating building regulations from two countries (i.e., England and Finland), this dataset also spans different subdomains, enabling the assessment of the general applicability of proposed machine learning models. More details about the CODE-ACCORD entity and relation datasets, including the used categories, their distribution and sample count, are described in Section 3.3.1.3. For both proposed classifiers, we considered 80% of the samples as training data and the remaining 20% as testing data during the evaluations.

Considering the high imbalance nature within relation distribution, primarily characterised by underrepresented categories, we applied the augmentation technique outlined in Section 3.3.2.1 for the CODE-ACCORD's relation-annotated data.

Augmented Data: Data augmentation was only applied for the relation categories: 'equal', 'greater', 'greater-equal', 'less', 'less-equal' and 'not-part-of', which exhibited a scarcity in representation. Importantly, the test set remained unaltered while we exclusively augmented the training data. For each sample in the training set, we generated 12 synthetic samples, considering the overall categorical distribution. Altogether, data augmentation increased the original training dataset of 3,463 samples by 2,912 samples, as summarised in Table 11. Moreover, Figure 27 illustrates the distribution of the final training dataset, following the original distribution in Figure 28. The augmented dataset is also publicly available for utilisation by future research⁸.

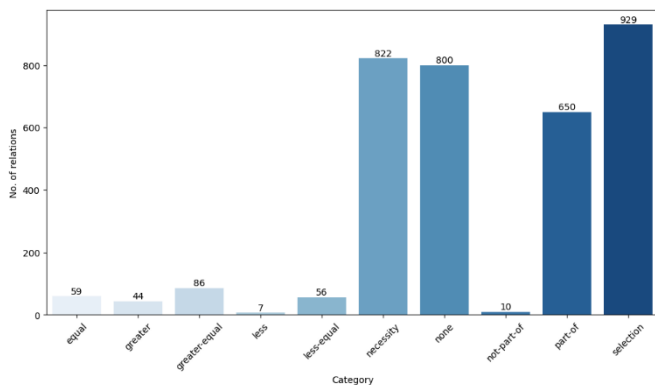


Figure 27. Distribution of the relation categories in original data

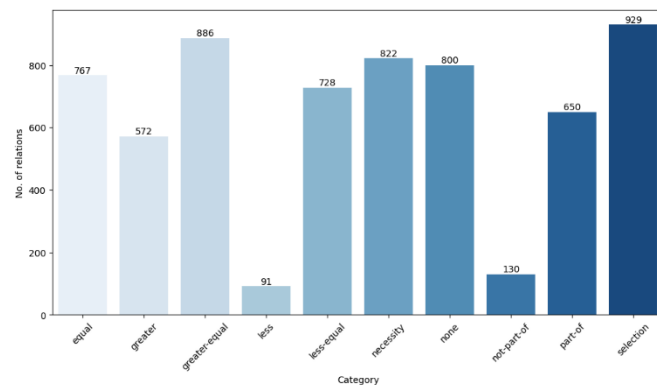


Figure 28. Distribution of the relation categories in augmented data

⁸ More information on how to access the CODE-ACCORD augmented datasets are available on <https://github.com/Accord-Project/accord-nlp>

Table 11. Statistics of original and augmented relation-annotated training data

Category	Original Data	Synthetic Data	Total
equal	59	708	767
greater	44	528	572
greater-equal	86	800	886
less	7	84	91
less-equal	56	672	728
necessity	822	-	822
none	800	-	800
not-part-of	10	120	130
part-of	650	-	650
selection	929	-	929
Training data	3463	2912	6375

Furthermore, we developed a text corpus by leveraging England's approved document collection⁹ to enrich transformer models with domain-specific knowledge by utilising the language modelling approach described in Section 3.2.1.2.

Regulatory Text Corpus: In developing the regulatory text corpus, we first converted all PDF files into TXT format using the PDFMiner library¹⁰. Subsequently, we conducted a series of data-cleaning procedures, including the rectification of formatting inconsistencies that occurred during the conversion process and the elimination of non-semantic elements such as pointers (e.g., G1, 1.2, a., etc.). Additionally, we filtered out section headings, incomplete text segments and those containing fewer than three words, exclusively retaining complete sentences and bullet-pointed text for the refined corpus. The final dataset comprised 14,336 text segments totalling 302,469 tokens. Figure 29 illustrates the sequence length distribution within this refined corpus. We split the data using an 80-20% split to have training and testing data. This text corpus is publicly available for utilisation by future research¹¹.

⁹ England's approved document collection is available on <https://www.gov.uk/government/collections/approved-documents>

¹⁰ PDFMiner documentation is available on <https://pypi.org/project/pdfminer/>

¹¹ The regulatory text corpus is available on <https://github.com/Accord-Project/accord-nlp/tree/main/data/lm>

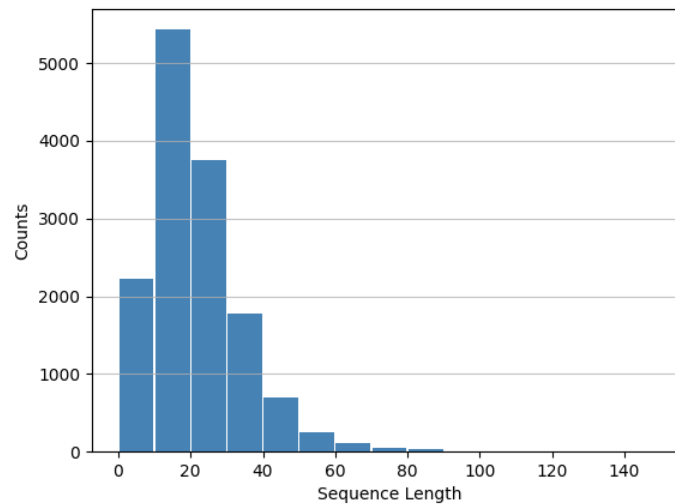


Figure 29. Sequence length distribution of text elements in the regulatory text corpus

Our experiments utilised several pre-trained transformer/language models in conjunction with the proposed architectures (Figure 25 and Figure 26). The selected models have gained significant popularity in recent research due to their outstanding performance [48] [49].

Pre-trained Transformers: Altogether, we involved four transformer models: (1) BERT-Large (bert-large-cased) [20], (2) RoBERTa-Large [24], (3) ALBERT-Large (albert-large-v2) [25], and (4) ELECTRA-Large (electra-large-discriminator) [50], pre-trained on English text for our experiments. These models were pre-trained in different settings, showcasing distinct characteristics. RoBERTa is a BERT variant that is trained on more data with longer training sentences. ALBERT is a lite version of BERT with parameter reduction to speed up the training process. ELECTRA follows a token discrimination-based pre-training process to facilitate more efficient training. All these pre-trained transformers were obtained from HuggingFace's model repository [51] to conduct the experiments. Also, we used a common hyper-parameter setup for all model architectures, which is further described in Annex A.

To evaluate the model's performance, we used precision (P), recall (R) and F1 scores, considering their coverage and common usage across various machine learning applications.

Evaluation Metrics: Equations (2)-(4) were used for precision, recall and F1 score calculations. In these equations, TP, FP and FN refer to the true positive, false positive and false negative counts, respectively. Precision measures the accuracy of the predictions made by the model, while recall measures the portion of actual instances that the model correctly predicted. F1 is the weighted harmonic mean of precision and recall, combining their properties. For entity evaluation, we used the seqeval framework with strict mode on BIO format [52]. This compares the text span, category and BIO label to compute TP, FP and FN values and marks a span correct (TP) only if all these elements match the ground truth. For relation evaluation, TP, FP, and FN values are computed solely by comparing predicted and actual categories. A prediction is marked correct (TP) only if it matches the ground truth category perfectly.

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

$$Recall = \frac{TP}{TP + F} \quad (3)$$

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (4)$$

Given the equal importance of multiple categories in both entity and relation classification, we employed the macro-averaging method to derive the final metrics. This involved initially computing each category's P, R and F1 values individually. Subsequently, we calculated their unweighted mean to obtain the final metrics, following Equation (5). In this equation, X represents a metric (i.e., P, R or F1), n represents the total number of categories, and X_i represents a per-class metric value.

$$Macro X = \frac{\sum_{i=1}^n X_i}{n} \quad (5)$$

SNOWTEC implementations were conducted in Python, leveraging PyTorch [53] and Huggingface [51] libraries to build machine learning models¹². The Graphviz library¹³ was used to visualise the graph outputs. All experiments were performed using an Intel Xeon Gold 6240 GPU. More details on entity and relation classifiers' performance are available below.

Entity Classifier's Performance: For entity classification, we evaluated the performance of the proposed architecture (Figure 25) using four pre-trained transformers. The results obtained on validation and test datasets are summarised in Table 12. As the model is fine-tuned based on the results of validation data, which comprises a smaller subset (10% of the training data) compared to the test data, it is expected to yield more accurate predictions on the validation set. However, the notable discrepancy indicates the complexity of the task and suggests potential challenges arising from data scarcity. Among the transformer models, RoBERTa-Large showcased superior performance, exhibiting an average increase of 5% in validation F1 scores and 4% in test F1 scores compared to all other models. This also suggests that if the language model is pre-trained on a large corpus, it can be fine-tuned effectively for a downstream task.

Table 12. Performance evaluation of the entity classifier across validation and test datasets using various transformer models. The best F1 score is marked in bold.

Transformer	Validation			Test		
	P	R	F1	P	R	F1
BERT-Large	0.6182	0.5448	0.5686	0.3990	0.3511	0.3649
ELECTRA-Large	0.6838	0.5486	0.5964	0.3630	0.2882	0.3059
ALBERT-Large	0.6612	0.5825	0.6176	0.4172	0.3542	0.3791
RoBERTa-Large	0.6485	0.6335	0.6400	0.3985	0.3902	0.3922

¹² SNOWTEC implementation is publicly available on <https://github.com/accord-project/accord-nlp>

¹³ The documentation of Graphviz is available on <https://graphviz.readthedocs.io/en/stable/>

We further optimised our best-performing entity classifier, and our findings are summarised in Table 13. We primarily focused on optimising the learning rate during the hyper-parameter tuning (Annex A). We experimented with $1e^{-3}$, $1e^{-4}$, and $1e^{-6}$, following the initial setup of $1e^{-5}$. The corresponding training and validation curves are visualised in Figure 30. Notably, $1e^{-4}$ performed on par with $1e^{-5}$ but showcased better convergence. Overall, fine-tuning the learning rate resulted in a significant 8.9% increase in validation and a 4.8% increase in test F1 scores.

Table 13. Impact on best-performed entity classifier's (RoBERTa-Large) results by different optimisation techniques. The best F1 score is marked in bold.

Optimisation Technique	Validation			Test		
	P	R	F1	P	R	F1
-	0.6485	0.6335	0.6400	0.3985	0.3902	0.3922
Hyper-parameter tuning	0.7674	0.6956	0.7287	0.4827	0.4103	0.4399
Language modelling	0.7655	0.7100	0.7351	0.4635	0.4279	0.4444

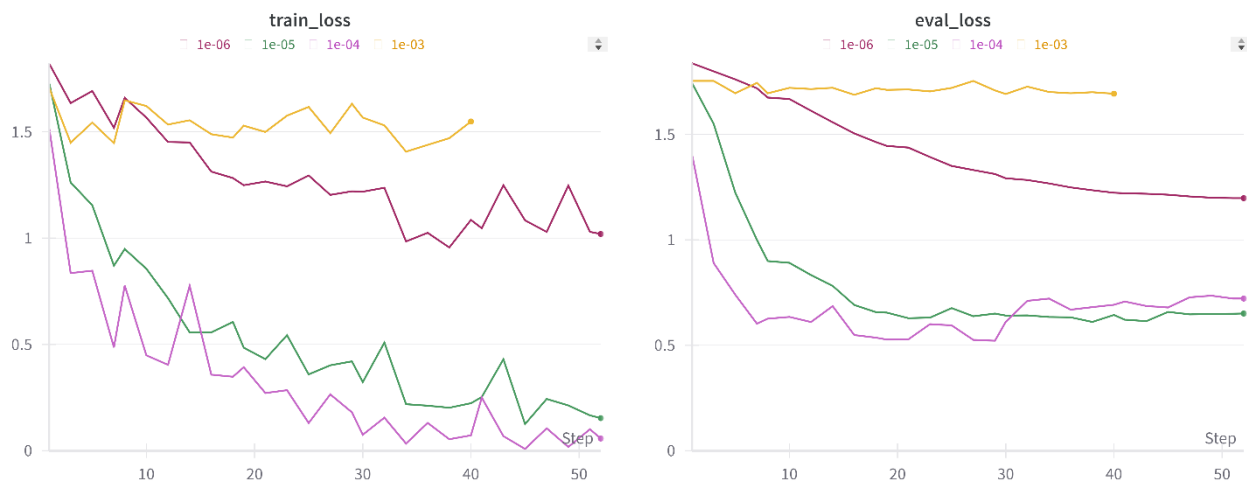


Figure 30. Training and validation/evaluation learning curves of entity classifiers built using the RoBERTa-Large model on different learning rates.

Furthermore, we applied language modelling to the RoBERTa-Large model using the regulatory text corpus and then fine-tuned it for entity classification. As shown in Table 13, this contributed to an additional improvement of 0.6% in validation and 0.4% in test F1 scores. These limited gains suggest that leveraging larger text corpora for language modelling could lead to more substantial enhancements, aligning with other recent studies. Nonetheless, our findings indicate that imparting domain knowledge to language models helps in enhancing entity classification performance. For an in-depth analysis, we delve into the performance of the best entity classifier in Annex B.

Relation Classifier's Performance: For relation classification, we evaluated the performance of the proposed architecture (Figure 26) using three pre-trained transformers. Considering the limited performance observed during entity classification experiments, the ELECTRA-Large model was

excluded from relation classification. The summary of results obtained from both validation and test datasets is presented in Table 14. Similar to the trends observed in the entity classifier results (Table 12), the validation set reflects more accurate predictions compared to the test set. The standard practices of model fine-tuning based on validation outcomes and the smaller size of the validation set compared to the test set mainly result in such trends. However, unlike the entity classifier outcomes, no significant disparities exist between the validation and test results of the relation classifier. This suggests the model underwent a comprehensive learning process utilising a substantial dataset.

Table 14. Performance evaluation of the relation classifier across validation and test datasets using various transformer models, with and without the integration of data augmentation. The best F1 score is marked in bold. The improvement in F1 score on the test data after applying data augmentation is indicated within brackets.

Transformer	Validation			Test		
	P	R	F1	P	R	F1
BERT-Large	0.6129	0.4887	0.5144	0.5714	0.5029	0.5243
ALBERT-Large	0.6332	0.5393	0.5590	0.5592	0.4978	0.5122
RoBERTa-Large	0.6099	0.5895	0.5903	0.5577	0.5481	0.5498
<i>Data Augmentation</i>						
BERT-Large	0.9215	0.9239	0.9207	0.8280	0.8161	0.8009 (+28%)
ALBERT-Large	0.9098	0.9128	0.9109	0.8060	0.7511	0.7556 (+24%)
RoBERTa-Large	0.9448	0.9459	0.9450	0.7997	0.8352	0.8011 (+25%)

Overall, Table 14 demonstrates a significant enhancement in the performance of the relation classifier achieved through data augmentation. On average, all models experienced a noteworthy 26% increase in F1 scores after integrating augmented data. Similar to observations in the entity classifier scenario, among the transformer models, RoBERTa-Large exhibited superior performance. Initially, using the original dataset, RoBERTa-Large showcased an average increase of 5.4% in validation F1 scores and 3.4% in test F1 scores compared to other models. Even after incorporating augmented data, RoBERTa-Large maintained its lead, exhibiting an average increase of 2.9% in validation F1 scores and 2.3% in test F1 scores. These outcomes reinforce the notion that pre-training a language model on a substantial corpus enables effective fine-tuning for downstream tasks, consistent with our observations in entity classifiers. Additionally, the performance of the BERT-Large model approached that of RoBERTa-Large following data augmentation. This highlights that an increased volume of data for a downstream task (i.e., relation classification) facilitates the model in learning task-specific details, minimising reliance on the original knowledge of the language model.

We further optimised our best-performing relation classifier using the same techniques we used with the entity classifiers (i.e., hyper-parameter tuning and language modelling). Table 15 provides an overview of the obtained results. Under hyper-parameter tuning, we explored the model's performance by experimenting with three additional learning rates (i.e., $1e^{-3}$, $1e^{-4}$, and $1e^{-6}$) subsequent to the initial setup of $1e^{-5}$. However, upon reviewing the corresponding training and validation curves depicted in Figure 31, $1e^{-5}$ turned out as the optimal learning rate.

Table 15. Impact on best-performed relation classifier's (RoBERTa-Large) results by different optimisation techniques. The best F1 score is marked in bold.

Optimisation Technique	Validation			Test		
	P	R	F1	P	R	F1
-	0.9448	0.9459	0.9450	0.7997	0.8352	0.8011
Hyper-parameter tuning	0.9448	0.9459	0.9450	0.7997	0.8352	0.8011
Language modelling	0.9441	0.9461	0.9447	0.8507	0.7722	0.7922

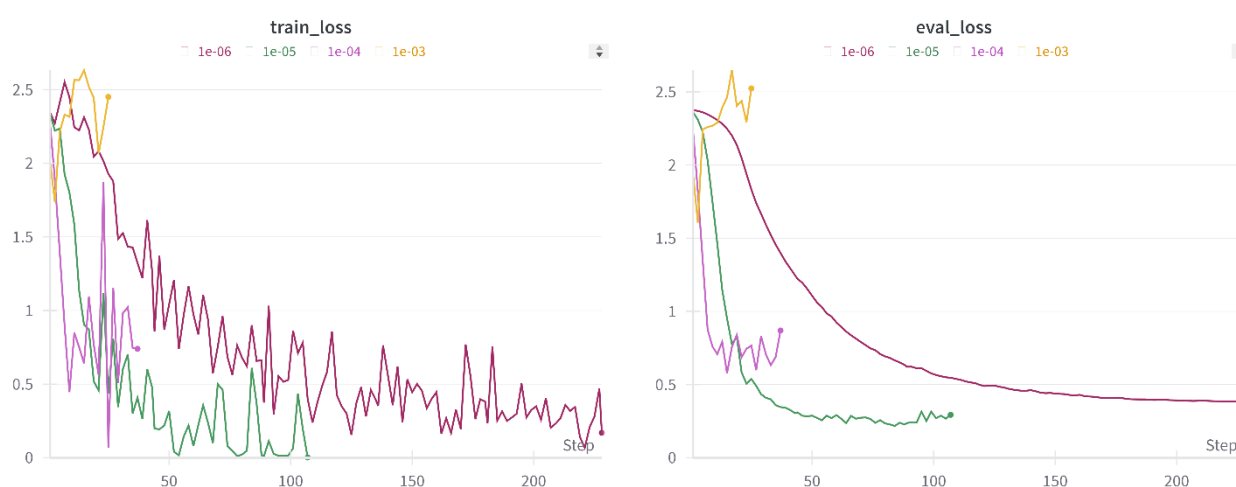


Figure 31. Training and validation/evaluation learning curves of relation classifiers built using the RoBERTa-Large model on different learning rates.

In language modelling, we employed the same RoBERTa-Large model trained on the regulatory text corpus that was utilised for the entity classification experiments. Surprisingly, unlike the entity classification results, the relation classification outcomes exhibited a slight decrease following language modelling. This suggests a potential risk of confusion when applying language modelling using a limited domain-specific corpus alongside a rich training dataset for the downstream task. Overall, the application of optimisation techniques did not improve the relation classifier's performance. We also conducted an in-depth analysis of the best relation classifier, and our findings are discussed in Annex C.

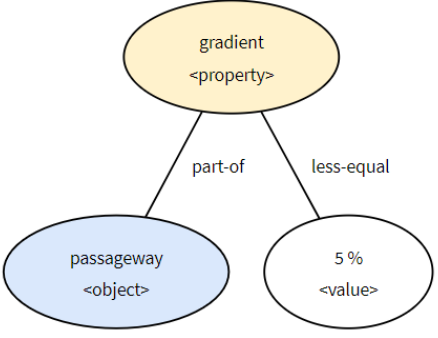
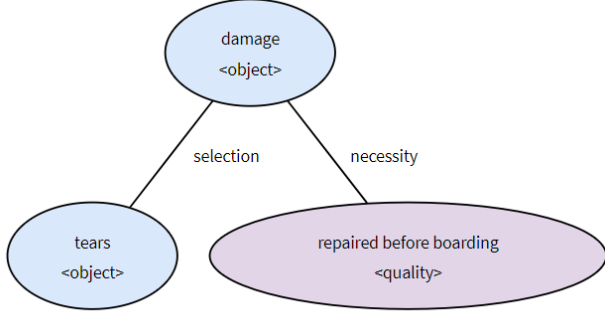
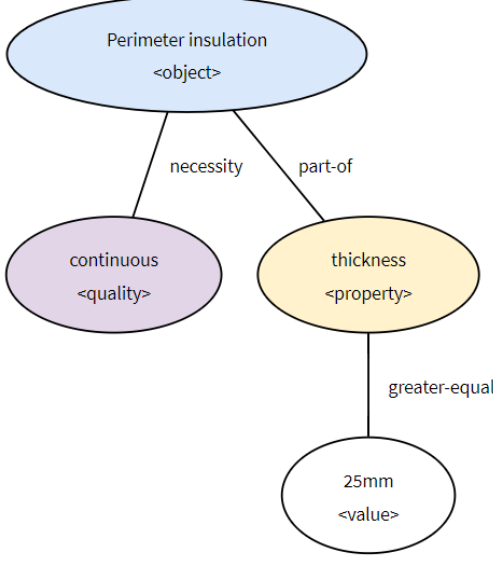
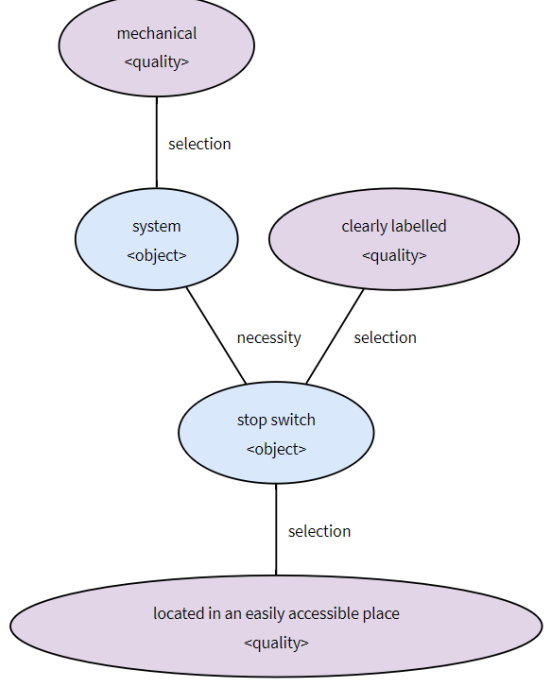
3.3.2.3 SNOWTEC Demonstration

The final SNOWTEC pipeline (Figure 24)¹⁴ was built leveraging the best-performing entity and relation classifiers described above (i.e., RoBERTa-Large entity and relation classifiers). It automatically identifies entities and their relations within regulatory sentences to generate knowledge graphs that encapsulate the conveyed information, performing rule formalisation. Table 16 presents a selection of tested samples processed through the finalised pipeline, each accompanied by its

¹⁴ SNOWTEC live demo is available on <https://huggingface.co/spaces/ACCORD-NLP/information-extractor>

generated graph. As can be seen, these graphs adeptly encapsulate the essence of the original sentences in a structured manner. Also, these graphs serve as a machine-processable output, meticulously capturing and processing the linguistic complexities inherent in regulatory sentences. Ultimately, this transformation will facilitate effective Automated Compliance Checking (ACC) by rendering complex regulatory data into a simpler format.

Table 16. Knowledge graphs generated by SNOWTEC for a set of sample sentences. In. and Out. denotes the input to and output from the pipeline.

<p>In.</p>	<p>The gradient of the passageway should not exceed 5%.</p>	<p>Any damage, such as tears, should be repaired before boarding.</p>
<p>Out.</p>		
<p>In.</p>	<p>Perimeter insulation should be continuous and have a minimum thickness of 25mm.</p>	<p>In a mechanical system, there shall be a clearly labelled stop switch, which shall be located in an easily accessible place.</p>
<p>Out.</p>		

3.3.3 RASE-LLM: RASE Automation Leveraging Large Language Models

Useful Links:

- [Text to HTML Few Shots](#)
- [Text to YAML Few Shots](#)
- [Fine-tuned Model](#)
- [RASE Automation Tool](#)

Building codes often exhibit a complex clausal structure, spanning multiple sentences, textual elements (e.g., bullet points), and even paragraphs. Such a design is particularly established because building codes are written in natural language targeting domain experts. Thus, achieving complete rule formalisation from textual sources has posed a challenge within the Architecture, Engineering, and Construction (AEC) domain. RASE, a widely recognised approach in AEC, addresses this challenge by capturing regulatory information from text blocks along with their underlying clausal structure and logic [2]. It mainly transforms regulatory/normative text into well-defined logical rules while capturing the structure and semantics embedded in the text, facilitating Automated Compliance Checking (ACC) processes. However, to the best of our knowledge, RASE has primarily been employed by domain experts so far as a markup language to manually annotate regulatory text, aiming to uncover the logic implied within text documents and resolve linguistic ambiguities to convert textual data into machine-processable formats [54]. To bridge this gap, we proposed <rase-method>, leveraging the capabilities of Large Language Models (LLMs) to automate the RASE annotation process.

The RASE scheme is based on four operators: (1) R - requirement, (2) A - applicability, (3) S - selection and (4) E - exception. Requirements are the checks that need to be satisfied. They usually appear together with imperatives such as *'must'* and *'shall'*. The applicability identifies to which or under which circumstance the check should be applied. Simply, it restricts the scope of the check. A check is associated with at least one applicability. Unlike applicability, selection defines more scope for a check. Exceptions describe when the check should be excluded. Overall, requirements and applicabilities are common in the regulatory text, while selections and exceptions are relatively rare. The RASE methodology for rule formalisation has three main steps. The first step is adding markup to text based on four RASE operators, aligning with the text semantics. This step mainly focuses on extracting the structure/arrangement of rules expressed in text, often resulting in nested tags. Secondly, each fine-granular annotation is paired with corresponding metadata to facilitate the generation of logical rules. This involves identifying the object, target/property, comparator, value, and unit associated with each annotated text phrase. The final step is to transform the operators into rules in Boolean logic, adhering to the pre-defined logical structure of RASE [2].

It is pivotal to encapsulate all the RASE tags into a structured data format to automate the annotation process using predictive modelling. However, given that RASE was originally designed as a manual approach for domain experts, it lacks an established machine-processable format capable of capturing all annotated information. In response to this limitation, a novel YAML-based data format named Building Compliance Rule Language (BCRL) has been developed as part of the ACCORD project. More information on BCRL is available in the ACCORD deliverable 2.2¹⁵. BCRL effectively captures the document structure, context, and comprehensive RASE annotations while following a

¹⁵ ACCORD deliverable 2.2 is available on https://accordproject.eu/wp-content/uploads/2024/02/ACCORD_D2.2_BCO_Ontology_and_Rules_Format.pdf

simple yet informative schema that can facilitate automated rule formalisation. Also, the format's simplicity and consistent organisation, aligned with the original text's structure, emphasise its appropriateness for human verification, which is a crucial aspect of ACC. Given all these advantages, BCRL has been selected as the output data format for the RASE automation process, while the raw text is used as the input data.

RASE-LLM included the development of an automated RASE annotation approach harnessing advanced natural language understanding and generation techniques offered by state-of-the-art Large Language Models (LLMs). This methodology primarily operates by taking raw text blocks (e.g., sections, chapters, or documents) as input and producing fully annotated RASE outputs in the BCRL format, solely leveraging the capabilities of LLMs without any human intervention.

In summary, RASE-LLM's main contributions are as follows.

1. A novel automated RASE annotation approach utilising the predictive capabilities of the state-of-the-art Large Language Models (LLMs) to streamline rule formalisation within AEC.
2. A comprehensive evaluation procedure to assess the model's prediction accuracy, focusing on both structural and semantical information crucial for rule extraction.
3. A detailed experimental study to explore the LLMs' capacity to recognise intricate information in regulatory documents, including document structure and nested rules, involving different learning techniques.

More information about the design and development of RASE-LLM are described in Section 3.3.3.1. Section 3.3.3.2 summarises the model performance overview and evaluation procedure. Finally in Section 3.3.3.3 demonstrates UI and functionality of the proposed approach with sample inputs and outputs. The interfaces presented in Section 3.3.3.3 are part of the RFT tool presentation in Section 1 and 2 representin the automatic approach of RASE tagging and regulation interpretation.

The contect of this section has been for consideration in the Nature Scientific Reports Journal.

3.3.3.1 RASE- LLM Method

This section outlines the methodologies used to automate the annotation of regulatory texts through the RASE method. By leveraging the capabilities of LLMs, this study aimed to transform textual descriptions of building regulations into structured YAML formats described in more details in [3], which facilitate ACC processes.

The adopted methodology primarily operates by taking both the raw text blocks of the building regulations and their RASE-annotated YAML counterpart data as input along with prompt instructions to the LLMs to create similar RASE-annotated YAMLS from unseen examples of text regulations. Two types of prompt engineering methods have been implemented: fine-tuning and fewshot prompting. Few-shot learning is a technique whereby we prompt the LLM with several concrete examples of task performance, in our use case examples of RASE-annotated YAML files created from unstructured text regulations. Fine-tuning, on the other hand, is a technique whereby we take an off-the-shelf open-source or proprietary model, re-train it on a variety of concrete examples, and save the updated weights as a new model checkpoint. The fine-tuned model can be deployed to create new YAML files from text regulations.

Experiment 1: Few-Shot Learning with GPT-4o (Text to YAML)

The first experiment utilized the GPT-4o model, a variant of the third-generation language models developed by OpenAI. This model offers enhanced processing speed and efficiency due to its optimized architecture and employs a few-shot learning approach. This approach allows the model to generalize from a few examples to new inputs without extensive traditional training. For the setup,

we selected three examples from the dataset as few-shot prompts, providing the model with clear input-output pairs to establish the context for the task. The input consisted of text file content, and the output was the corresponding YAML file. The model was then tested on new, unseen text files, converting these texts into YAML format using the context learned from the few-shot examples. The evaluation criteria focused on the model's ability to recognize and convert intricate information in regulatory documents, including document structure and nested rules, ensuring both structural and semantic accuracy in the YAML conversion.

#Example 1

Input: Text file1

Output: YAML file1

#Example 2

Input: Text file2

Output: YAML file2

#Example 3

Input: Text file3

Output: YAML file3

The following are snapshots of both input and output samples:

Horizontal Escape Route Design

Principles of Escape Routes

Means of escape should be provided from any point on a floor to an exit. The general principle is that any person confronted by a fire can turn away from it and escape safely.

In certain conditions, typically classrooms, a single direction of escape (a dead-end condition) can be accepted as providing reasonable safety, provided that the recommendations of Table 1 on travel distances in a single direction are met and the occupancy of the space is limited to 60.

Number of Escape Routes and Exits, and Limits on Travel Distance

The number of escape routes and exits depends on the number of occupants and the limits on travel distance to the nearest exit, as given in Table 1. It is only the distance to the nearest exit that needs to meet the recommendations; other exits may be further away. In multi-storey buildings, more than one stair will be needed for escape.

```

|
$type: Document
subject: PM_35_30_30
coverage: GB-ENG
temporal: 2016-05-27
title: "Building Bulletin: BB100"
modified: 2021-05-27
issued: 2021-05-27
identifier: Building_Bulletin:_BB100
$id: Building_Bulletin:_BB100
hasPart:
- $id: Building_Bulletin:_BB100/4.1
  identifier: 4.1
  $type: DocumentSubdivision
  title: Horizontal escape route design
  hasPart:
  - $type: Statement
    $id: Building_Bulletin:_BB100/4.1.1
    identifier: 4.1.1
    asText: Means of escape should be provided from any point on a floor to an exit from the floor.
  - relation: Table/1
    $type: Statement
    $id: Building_Bulletin:_BB100/4.1.2
    identifier: 4.1.2
    hasInlinePart:

```

Algorithm 1 Few-Shot Learning with GPT-4o for YAML Conversion

```

1: procedure FEWSHOTLEARNING(Examples, Model)
2:   FewShotPrompts ← SELECTEXAMPLES(Examples)
3:   TestData ← SELECTUNSEENTEXTFILES
4:   EvalResults ← TESTMODEL(Model, FewShotPrompts, TestData)
5:   return EvalResults
6: end procedure
7: function SELECTEXAMPLES(Examples)
8:   FewShotPrompts ← []
9:   for each example in Examples do
10:    FewShotPrompt ← format example as input-output pair
11:    append FewShotPrompt to FewShotPrompts
12:   end for
13:   return FewShotPrompts
14: end function
15: function TESTMODEL(Model, FewShotPrompts, TestData)
16:   EvaluationResults ← []
17:   for each testItem in TestData do
18:    prediction ← GENERATEOUTPUT(Model, FewShotPrompts, testItem.input)
19:    evaluation ← EVALUATEPREDICTION(prediction, testItem.output)
20:    append evaluation to EvaluationResults
21:   end for
22:   return EvaluationResults
23: end function
24: function EVALUATEPREDICTION(prediction, expectedOutput)
25:   return comparison of prediction with expectedOutput
26: end function

```

Experiment 2: Few-Shot Learning with GPT-4o (Text to HTML)

The second experiment utilized the GPT-4o model, an advanced variant of OpenAI's third-generation language models. This model boasts improved processing speed and efficiency thanks to its optimized architecture and employs a few-shot learning approach. This approach enables the model to generalize from a few examples to new inputs without requiring extensive traditional training. For the setup, we selected three examples from the dataset as few-shot prompts, providing the model with clear input-output pairs to establish the task's context. The input consisted of text file content, and the output was the corresponding HTML file. The model was then tested on new, unseen text files, converting these texts into HTML format based on the context learned from the few-shot examples. The evaluation criteria focused on the model's ability to recognize and convert intricate information in regulatory documents, including document structure and nested rules, ensuring both structural and semantic accuracy in the HTML conversion.

#Example 1

Input: Text file1

Output: HTML file1

#Example 2

Input: Text file2

Output: HTML file2

#Example 3

Input: Text file3

Output: HTML file3

Horizontal Escape Route Design

****Principles of Escape Routes****

Means of escape should be provided from any point on a floor to an exit. The general principle is that any person confronted by a fire can turn away from it and escape safely.

In certain conditions, typically classrooms, a single direction of escape (a dead-end condition) can be accepted as providing reasonable safety, provided that the recommendations of Table 1 on travel distances in a single direction are met and the occupancy of the space is limited to 60.

****Number of Escape Routes and Exits, and Limits on Travel Distance****

The number of escape routes and exits depends on the number of occupants and the limits on travel distance to the nearest exit, as given in Table 1. It is only the distance to the nearest exit that needs to meet the recommendations; other exits may be further away. In multi-storey buildings, more than one stair will be needed for escape.

Horizontal escape route design
Means of escape should be provided from any point on a floor to an exit from the floor. The general principle is that any person confronted by a fire within a building can turn away from it and escape safely.
In certain conditions, typically classrooms, a single direction of escape (a dead end condition) can be accepted as providing reasonable safety providing that the recommendations of Table 1 on travel distances in a single direction is met and the occupancy of the space is limited to 60.
Number of escape routes and exits, and limits on travel distance
The number of escape routes and exits to be provided depends on the number of occupants in the room, tier or storey in question and the limits on travel distance to the nearest exit given in Table 1.
It is only the distance to the nearest exit that needs to meet the recommendations. The other exits may be further away and in multi-storey buildings, more than one stair will be needed for escape.
In many cases, there will not be an alternative at the beginning of the route. For example, there may be only one exit from a room to a corridor, from which point escape is possible in two directions. A single route is acceptable for parts of a floor from which a storey exit or escape in two directions can be reached within the travel distance limit for travel in one direction set in Table 1. Figure 1 shows an example of a dead-end condition in an open plan layout.
Very young children (nursery, reception and infant class) will move more slowly than older children or adults and require constant supervision and direction during egress. Having direct access to an external place of safety from their classrooms is an advantage.
Number of occupants and exits
The value used for the number of occupants will normally be that specified as the basis for the design.
When the number of occupants likely to use a room, tier or storey is not known, the capacity should be calculated according to the appropriate floor space factors (see Table 2).
Table 3 gives the minimum number of escape routes and exits from a room or storey according to the number of occupants. The number of exits may have to be increased to comply with the limits on travel distances given in Table 1.
Alternative escape routes
A choice of escape routes is of little value if two or more are likely to be disabled simultaneously. Alternative escape routes should therefore satisfy the following criteria:
a. they are in directions 45° or more apart (see Figure 1); or

Experiment 3: Fine-Tuning with GPT-4o

The second experiment focused on fine-tuning the GPT-4o model, which supports customization for specific tasks by training on a targeted dataset. For data preparation, we used 25 paired text and YAML files derived from building regulations, selecting 15 pairs for training, 5 pairs for validation, and reserving 5 pairs for testing. Two JSONL files, `training.jsonl` and `validation.jsonl`, were created for the fine-tuning process, with each entry formatted to include the system's role, user input (text file content), and assistant output (YAML file content).

The fine-tuning process involved initiating a fine-tuning job on OpenAI's server using the prepared JSONL files, training the model on the `training.jsonl` file. The fine-tuned model was then tested on the five reserved text-YAML file pairs to evaluate its performance. Detailed results and analysis of this testing phase are discussed in the "Model Evaluation" section.

Algorithm 2 Fine-Tuning GPT-4o for YAML Conversion

```

1: procedure FINETUNEMODEL(Data, Model)
2:   TrainingData ← select 15 pairs from Data
3:   ValidationData ← select 5 pairs from Data
4:   TestData ← select 5 pairs from Data
5:   TrainingJSONL ← CONVERTTOJSONL(TrainingData)
6:   ValidationJSONL ← CONVERTTOJSONL(ValidationData)
7:   upload TrainingJSONL and ValidationJSONL to OpenAI API
8:   FineTunedModel ← CREATEFINETUNINGJOB(Model, TrainingJSONL, ValidationJSONL)
9:   train FineTunedModel using OpenAI API
10:  EvalResults ← EVALUATEMODEL(FineTunedModel, TestData)
11:  return EvalResults
12: end procedure
13: function CONVERTTOJSONL(Data)
14:  for each item in Data do
15:    JSONLItem ← {"messages" : [{"role" : "system", "content" : "RASEContext"}, {"role" :
      "user", "content" : item.text}, {"role": "assistant", "content": item.yaml}]}
16:    append JSONLItem to JSONLData
17:  end for
18:  return JSONLData
19: end function
20: function EVALUATEMODEL(Model, TestData)
21:  for each testItem in TestData do
22:    prediction ← GENERATEOUTPUT(Model, testItem.text)
23:    compare prediction with testItem.yaml
24:  end for
25:  return evaluation results
26: end function

```

Incremental Prediction Strategy

All experiments implemented an incremental prediction strategy to overcome the token generation limits of GPT models, which are restricted to 4096 tokens per generation. This strategy was essential for processing lengthy regulatory documents that exceeded the model's output capacity. The incremental prediction method is described as follows:

1. Initial Input Processing:

- The model begins by processing the initial segment of the regulatory text. This segment is within the token limit, ensuring that the model can generate an output without truncation.

2. Partial YAML Generation:

- The model generates a portion of the YAML output based on the initial input. This partial YAML represents a segment of the complete conversion that fits within the token limits.

3. Context Reinsertion:

- The generated YAML segment is then reinserted into the model's context. This updated context includes both the initial regulatory text segment and the corresponding generated YAML.

4. Subsequent Input Segmentation:

- The following segment of the regulatory text is appended to the updated context. Care is taken to ensure that the combined context (previous context plus the next segment) stays within the token limit.

5. Iterative Generation:

- The model is prompted again to continue the YAML generation from where it left off. This iterative process involves repeatedly generating new segments of the YAML output, reinserting them into the context, and appending subsequent text segments until the entire document is processed.

6. Handling Long Documents:

- For particularly lengthy documents, this incremental approach ensures that the entire text is eventually converted into YAML format, with each segment processed and reintegrated methodically. This avoids token overflow issues and ensures continuity in the generated output.

7. Evaluation of Continuity and Accuracy:

- At each step, the generated YAML segments are evaluated for continuity and accuracy to ensure that the incremental predictions align seamlessly with the previously generated segments. This evaluation helps maintain the structural and semantic integrity of the output.

By utilizing this incremental prediction strategy, the experiments effectively managed the token generation limits of GPT models, enabling the successful conversion of extensive regulatory texts into YAML format without loss of information or context. This approach was particularly crucial for ensuring the fidelity of rule extraction and the overall reliability of the automated compliance checking processes.

Algorithm 4 Incremental Prediction Strategy for YAML Conversion

```

1: procedure INCREMENTALPREDICTION(Text, Model)
2:   Context  $\leftarrow \emptyset$ 
3:   Segments  $\leftarrow$  SEGMENTTEXT(Text)
4:   YAML  $\leftarrow \emptyset$ 
5:   for each segment in Segments do
6:     Context  $\leftarrow$  Context + segment
7:     if TOKENCOUNT(Context)  $\geq$  4096 then
8:       Context  $\leftarrow$  TRUNCATE(Context)
9:     end if
10:    PartialYAML  $\leftarrow$  GENERATEYAML(Model, Context)
11:    YAML  $\leftarrow$  YAML + PartialYAML
12:    Context  $\leftarrow$  Context + PartialYAML
13:  end for
14:  return YAML
15: end procedure
16: function SEGMENTTEXT(Text)
17:   Segments  $\leftarrow$  SPLIT(Text, MaxSegmentLength)
18:   return Segments
19: end function
20: function TOKENCOUNT(Context)
21:   return COUNTTOKENS(Context)
22: end function
23: function TRUNCATE(Context)
24:   return REMOVEOLDESTSEGMENTS(Context, MaxTokenLimit)
25: end function
26: function GENERATEYAML(Model, Context)
27:   PartialYAML  $\leftarrow$  MODELGENERATE(Model, Context)
28:   return PartialYAML
29: end function

```

3.3.3.2 Model Performance Overview

In this section, we describe the evaluation metrics used to assess the performance of our models in generating YAML files from regulatory texts. The evaluation concentrates on two main aspects: structural similarity and text similarity. These metrics provide a thorough assessment of both the structure and content of the generated YAML files.

Structure Evaluation (Graph Similarity):

The structural accuracy of the generated YAML files is assessed using graph similarity metrics. Each YAML file is converted into a graph representation where nodes represent keys and values, and edges depict the hierarchical relationships between these elements. To compute the similarity between the generated and reference graphs, we utilize the SimGNN model, a neural network-based approach. The process begins by converting both the generated and reference YAML files into graph structures. Initial node embeddings are computed using Graph Convolutional Networks (GCNs), which aggregate information from neighbouring nodes to create a detailed representation of the graph's structure. These node embeddings are then aggregated into a single graph-level embedding via a global context-aware attention mechanism, which calculates a global context vector for the graph and uses it to weigh the importance of each node's embedding.

Interaction scores between graph-level embeddings are computed using a Neural Tensor Network (NTN). Additionally, pairwise similarity scores between nodes of the two graphs are calculated, and histogram features are extracted from these scores. Finally, graph-level interaction scores and histogram features are combined using fully connected layers to produce a final similarity score.

To quantify the structural differences between the generated and reference graphs, we compute the Graph Edit Distance (GED), which measures the minimum number of operations (node/edge insertions, deletions, and relabelling) required to transform one graph into the other. The GED is normalized and transformed into a similarity score ranging from 0 to 1 using an exponential function.

$$(\lambda(x) = e^{\{-x\}}, \text{ where } (x) \text{ is the normalized GED.}$$

Content Evaluation (Text Similarity):

The text similarity between the contents of the generated and reference YAML files is evaluated using accuracy and adjusted accuracy metrics. The accuracy metric measures the overlap of words between the two texts, while the adjusted accuracy accounts for differences in chunk sizes to penalize discrepancies in text segmentation. First, the number of overlapping words between the generated and reference texts is calculated to determine the common words. Then, the total number of unique words in both texts combined is computed. The basic accuracy is calculated as the ratio of common words to total unique words, defined as $(\text{accuracy}(x, y) = \frac{\text{common}(x, y)}{\text{total}(x, y)})$

To account for text segmentation differences, we determine the number of chunks in both the generated $((c_x))$ and reference $((c_y))$ texts. The chunk penalty is then computed using the logarithmic difference in chunk counts, normalized by the total number of chunks in the larger text $((t))$. This is defined as $(\text{chunk penalty}(c_x, c_y, t) = \frac{\log_2(\text{chunk difference}(c_x, c_y) + 1)}{\log_2(t + 1)})$

Finally, the adjusted accuracy is calculated to incorporate the chunk penalty, reducing the impact of segmentation differences. The formula for adjusted accuracy is

$$(\text{adjusted accuracy}(x, y, c_x, c_y, t, w) = \text{accuracy}(x, y) \times (1 - w \times \text{chunk penalty}(c_x, c_y, t))),$$

Where (w) is the chunk penalty weight.

Results and Discussion

This section presents and analyzes the results from the three experiments conducted to assess the performance of the GPT-4o model in generating YAML files from regulatory text. The evaluation focuses on two main metrics: structural similarity and text similarity.

In the first experiment, we employed a few-shot learning approach with GPT-4o, supplying the model with three sample pairs as prompts. The model was then tasked with predicting YAML structures for 22 test samples. The structural similarity scores for these predictions are detailed in the following table, with an average structural similarity score of **0.69**, indicating a moderate level of structural accuracy.

0.75	0.64	0.63	0.69	0.77	0.79
0.7	0.75	0.67	0.76	0.79	0.7
0.64	0.63	0.68	0.76	0.74	0.64
0.61	0.6	0.57	0.65		

The text similarity scores, also shown in the table, averaged **0.76**. This higher average compared to the structural similarity score suggests that while the model was relatively effective at capturing the content, it had more difficulty maintaining the correct hierarchical structure.

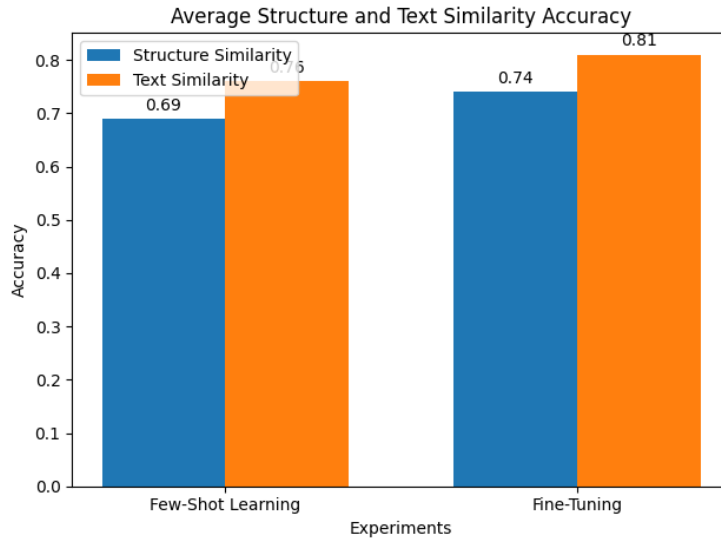
0.81	0.79	0.77	0.75	0.74	0.74
0.85	0.7	0.75	0.84	0.73	0.79
0.69	0.73	0.72	0.75	0.83	0.68
0.8	0.69	0.79	0.78		

The second experiment focused on fine-tuning the GPT-4o model using a dataset of 25 paired text and YAML files, divided into 15 pairs for training, 5 pairs for validation, and 5 pairs for testing. The structural similarity results for this experiment are detailed in the following table, with an average score of **0.74**. This improvement over the few-shot learning approach suggests that fine-tuning enables the model to more effectively learn and replicate the hierarchical structures in the YAML files.

0.78	0.67	0.78	0.70	0.78
-------------	-------------	-------------	-------------	-------------

The text similarity scores for this experiment, also shown in the following table, averaged **0.81**. The higher scores in both structural and text similarity metrics compared to the few-shot learning approach highlight the effectiveness of fine-tuning in enhancing model performance.

0.82	0.78	0.80	0.84	0.81
-------------	-------------	-------------	-------------	-------------



3.3.3.3 Rase Automation Tool

3.3.3.1.1 Introduction

The main target of this tool is to streamline the process of converting regulatory text files into YAML format using an AI model. The web application integrates several components to provide real-time, accurate conversion of regulatory texts into YAML files. The architecture of the application (Figure 32) is designed to handle user inputs, process them using an AI model, and provide the results back to the user efficiently. The system includes a web application, a REST API, an AI model, a database, and a Real-Time Notification Provider (RTNP).

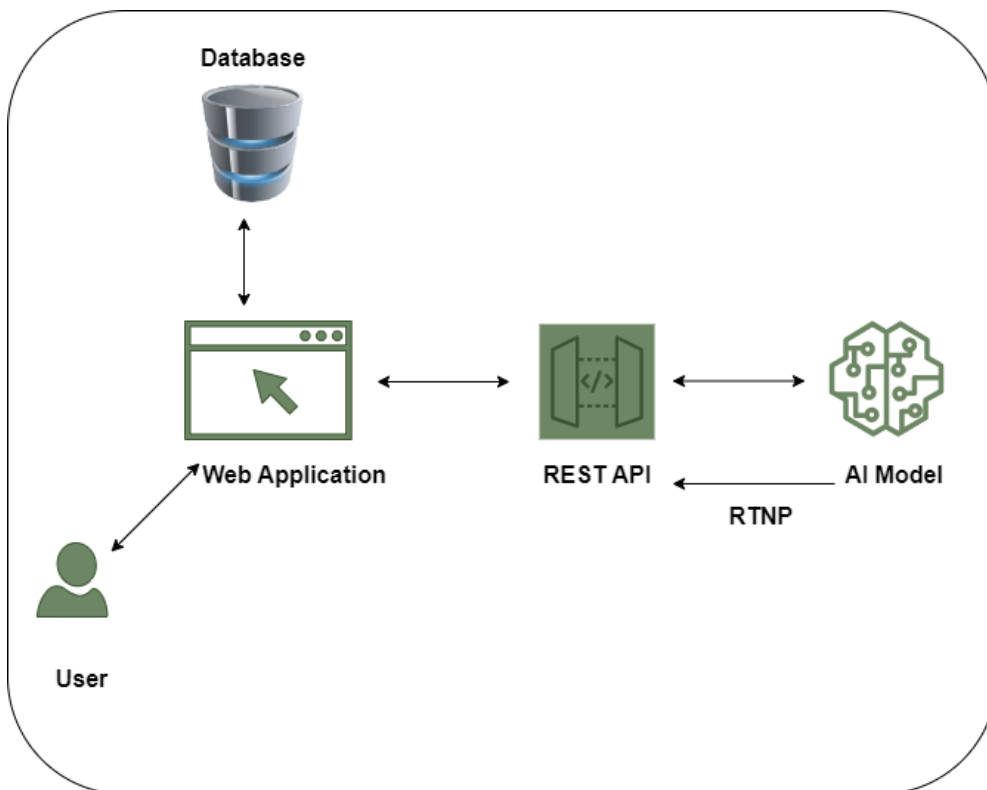


Figure 32. RASE automation tool architecture.

3.3.3.1.3 Tool Components

1. User Interface (Web Application)

- **Purpose:** To allow users to interact with the tool, upload text files, and receive YAML outputs.
- **Functionality:** Users upload their text files through the web interface. The web application sends these files to the REST API for processing and displays the resulting YAML files to the user.

2. REST API

- **Purpose:** To serve as an intermediary between the web application and the AI model.
- **Functionality:** The REST API receives text files from the web application, forwards them to the AI model for processing, and returns the generated YAML files back to the web application. It ensures smooth communication and data transfer within the system.

3. AI Model

- **Purpose:** To convert the regulatory text files into YAML format using the RASE method.
- **Functionality:** The AI model, fine-tuned on a dataset of regulatory texts and YAML files, processes the input text files and generates structured YAML outputs. The model handles the complexities of the regulatory language and maintains the hierarchical structure required for effective compliance checking.

4. Database

- **Purpose:** To store user data, input text files, and generated YAML files.
- **Functionality:** The database ensures that all data is securely stored and easily retrievable. It supports the web application by providing persistent storage for ongoing and past conversions.

5. Real-Time Notification Provider (RTNP)

- **Purpose:** To notify users about the status of their file processing in real time.
- **Functionality:** The RTNP component sends real-time notifications to users about the progress and completion of their text file conversion. This ensures that users are kept informed throughout the process without having to manually check the status.

3.3.3.1.4 Process Flow

1. File Upload:

- The user uploads a text file through the web application.

2. Data Transfer:

- The web application sends the uploaded file to the REST API.

3. AI Processing:

- The REST API forwards the file to the AI model, which processes the file and generates the corresponding YAML output.

4. Storage and Retrieval:

- The generated YAML file is stored in the database, and the REST API retrieves it for the web application.

5. User Notification:

- The RTNP notifies the user about the completion of the conversion process, and the web application displays the YAML file to the user.

3.3.3.1.5 Conclusion

This tool leverages the power of AI to automate the conversion of regulatory text files into YAML format, significantly enhancing efficiency and accuracy in compliance-related tasks. The well-structured architecture ensures seamless operation and real-time user updates, making it a robust solution for automated compliance checking. The screenshots of the UI for the RASE-LLM Tool are presented in ANNEX D.

3.3.4 Resource Index

This section serves as a comprehensive index of all open-source resources, including datasets, models and workflows, developed by the task for Artificial Intelligence for Natural Language Processing of Building Codes within the ACCORD project.

3.3.4.1 Datasets

Dataset	Description	Section Reference
CODE-ACCORD Entities	CODE-ACCORD entity-annotated data, considering four entity categories: (1) object, (2) property, (3) quality and (4) value	Section 3.3.1.3
CODE-ACCORD Relations	CODE-ACCORD relation-annotated data, considering ten relation categories: (1) selection, (2) necessity, (3) part-of, (4) not-part-of, (5) greater, (6) greater-equal, (7) equal, (8) less-equal, (9) less and (10) none	Section 3.3.1.3
Augmented CODE-ACCORD Relations	Oversampled CODE-ACCORD relations dataset with a more balanced category distribution	Section 3.3.2.2
Regulatory Text Corpus	Raw text corpus developed utilising England's approved document collection	Section 3.3.2.2

3.3.4.2 Pre-trained Models

Model	Description	Section Reference

ACCORD-NLP/ner-roberta-large	RoBERTa large model fine-tuned for sequence labelling/entity classification using the <i>CODE-ACCORD Entities</i> dataset	Section 3.3.2.1 and 3.3.2.2
ACCORD-NLP/ner-bert-large	BERT large (cased) model fine-tuned for sequence labelling/entity classification using the <i>CODE-ACCORD Entities</i> dataset	Section 3.3.2.1 and 3.3.2.2
ACCORD-NLP/ner-albert-large	ALBERT large model fine-tuned for sequence labelling/entity classification using the <i>CODE-ACCORD Entities</i> dataset	Section 3.3.2.1 and 3.3.2.2
ACCORD-NLP/re-roberta-large	RoBERTa large model fine-tuned for relation classification using the <i>CODE-ACCORD Relations</i> dataset	Section 3.3.2.1 and 3.3.2.2
ACCORD-NLP/re-bert-large	BERT large model fine-tuned for relation classification using the <i>CODE-ACCORD Relations</i> dataset	Section 3.3.2.1 and 3.3.2.2
ACCORD-NLP/re-albert-large	ALBERT large model fine-tuned for relation classification using the <i>CODE-ACCORD Relations</i> dataset	Section 3.3.2.1 and 3.3.2.2
ACCORD-NLP/roberta-large-lm	RoBERTa large model pre-trained on the <i>Regulatory Text Corpus</i> using the Masked Language Modelling (MLM) objective	Section 3.2.1.2 and 3.3.2.2
ACCORD-NLP/ner-roberta-large-lm	RoBERTa large model fine-tuned for sequence labelling/entity classification using the <i>CODE-ACCORD Entities</i> dataset, following language modelling using the <i>Regulatory Text Corpus</i>	Section 3.3.2.1 and 3.3.2.2
https://github.com/accord-project/RaseLLM/blob/main/few_shots_gpt_text_to_html.py https://github.com/accord-project/RaseLLM/blob/main/few_shot_text_yaml_rase.py	Few shots for both: 1. Text to HTML 2. Text to YAML	Section 3.3.3.1
https://github.com/accord-project/RaseLLM/blob/main/fine_tuning_use_model.py	Fine-tuning for Text to YAML	Section 3.3.3.1

https://github.com/accord-project/RaseLLM/tree/main/RASE_Automation_Tool	RASE Automation Tool (Web Application)	Section 3.3.3.3
ACCORD-NLP/re-roberta-large-lm	RoBERTa large model fine-tuned for relation classification using the <i>CODE-ACCORD Relations</i> dataset, following language modelling using the <i>Regulatory Text Corpus</i>	Section 3.3.2.1 and 3.3.2.2

3.3.4.3 Workflows

Workflow	Description	Section Reference
SNOWTEC	Information extraction pipeline to convert a self-contained regulatory sentence into a knowledge graph(s) of entities and relations	Section 0

3.3.5 Future Improvements

In the future we aim to automate the interpretation of images and tables found in the regulations and other documents. Also, we aim to enhance the NLP models with the capability to interpret and convert into structure graph format not only single self-contained sentences but also paragraphs of text.

4. Conclusions

This deliverable has documented the outcomes of Tasks 2.4 “Artificial Intelligence for Natural Language Processing of Building Codes” and 2.5 “Design and Implementation of Rule Formalisation Tool” of the ACCORD project. The specified objectives have been successfully realized through the accomplishment of the following goals:

- Creation of a web tool provided with the necessary user interfaces to allow experts in regulations and construction codes to formalize regulatory documents in PDF format in graphs as instances of AEC3PO ontology.
- Creation of a tool that leverages the power of AI to automate the conversion of regulatory text files into YAML format, significantly improving efficiency and accuracy in tasks related to regulatory compliance.

The AEC3PO is a component of the Compliance and Permitting Semantic Framework developed in the ACCORD project reported in Deliverable 2.2, together with the rule formalisation methodology developed in Task 2.3 and BCRL, a domain specific rule language. All these components have been successfully implemented in the rule formalisation tool, providing a way for regulatory experts to generate versions of regulations in a machine-processable format.

ACCORD is a pioneer in deploying LLM models for interpreting and structuring textual regulatory content and the progress of AI based rule interpretation has gone beyond the initial aim of the project which was limited to only identify important entities and relationships in regulatory text.

5. References

- [1] Z. Zhang, L. Ma and N. Nisbet, "Unpacking Ambiguity in Building Requirements to Support Automated Compliance Checking," *Journal of Management in Engineering*, vol. 39, 2023.
- [2] E. Hjelseth and N. Nisbet, "Capturing normative constraints by use of the semantic mark-up RASE methodology," in *Proceedings of CIB W78-W102 Conference*, 2011.
- [3] A. Dridi, E. Vakaj, P. Patlakas, H. Hettiarachchi, T. Beach, J. Yeung, G. Costa, P. Hradil and H. Tan, "D2.2 BCO Ontology and Rules Format," 2023.
- [4] M. Sporny, D. Longley, G. Kellogg, M. Lanthaler, P. Antoine and N. Lindström, "JSON-LD 1.1," 2020. [Online]. Available: <https://www.w3.org/TR/json-ld11/>. [Accessed 2024].
- [5] C. W. E. & A. R. Pautasso, *REST: Advanced Research Topics and Practical Applications*, New York: Springer, 2014.
- [6] S. Fuchs and R. Amor, "Natural language processing for building code interpretation: A systematic literature review," in *Proceedings of the Conference CIB W78*, 2021.
- [7] C. D. Manning, P. Raghavan and H. Schütze, *Introduction to Information Retrieval*, Cambridge University Press, 2008.
- [8] Y.-C. Zhou, Z. Zheng, J.-R. Lin and X.-Z. Lu, "Integrating NLP and context-free grammar for complex rule interpretation towards automated compliance checking," *Computers in Industry*, vol. 142, 2022.
- [9] T. H. Beach, Y. Rezgui, H. Li and T. Kasim, "A Rule-Based Semantic Approach for Automated Regulatory Compliance in the Construction Sector," *Expert Systems with Applications*, vol. 42, p. 5219–5231, 2015.
- [10] J. Zhang and N. M. El-Gohary, "Semantic NLP-Based Information Extraction from Construction Regulatory Documents for Automated Compliance Checking," *Journal of Computing in Civil Engineering*, vol. 30, 2016.
- [11] J. Zhang and N. M. El-Gohary, "Integrating semantic NLP and logic reasoning into a unified system for fully-automated code checking," *Automation in Construction*, vol. 73, pp. 45-57, 2017.
- [12] P. Zhou and N. El-Gohary, "Ontology-based automated information extraction from building energy conservation codes," *Automation in Construction*, vol. 74, pp. 103-117, 2017.
- [13] R. Zhang and N. El-Gohary, "A deep neural network-based method for deep information extraction using transfer learning strategies to support automated compliance checking," *Automation in Construction*, vol. 132, 2021.

- [14] X. Wang and N. El-Gohary, "Deep Learning-Based Named Entity Recognition from Construction Safety Regulations for Automated Field Compliance Checking," in *Computing in Civil Engineering 2021*, 2021, pp. 164-171.
- [15] X. Wang and N. El-Gohary, "Deep learning-based relation extraction and knowledge graph-based representation of construction safety requirements," *Automation in Construction*, vol. 147, p. 104696, 2023.
- [16] Y. Shen, X. Wang, Z. Tan, G. Xu, P. Xie, F. Huang, W. Lu and Y. Zhuang, "Parallel Instance Query Network for Named Entity Recognition," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2022.
- [17] A. Plum, T. Ranasinghe, S. Jones, C. Orasan and R. Mitkov, "Biographical Semi-Supervised Relation Extraction Dataset," in *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2022.
- [18] Y. Shen, Z. Tan, S. Wu, W. Zhang, R. Zhang, Y. Xi, W. Lu and Y. Zhuang, "PromptNER: Prompt Locating and Typing for Named Entity Recognition," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2023.
- [19] S. Yang, M. Choi, Y. Cho and J. Choo, "HistRED: A Historical Document-Level Relation Extraction Dataset," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2023.
- [20] J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019.
- [21] OpenAI, "GPT-4 Technical Report," ArXiv, 2023.
- [22] S. Minaee, T. Mikolov, N. Nikzad, M. Chenaghlu, R. Socher, X. Amatriain and J. Gao, "Large Language Models: A Survey," *ArXiv*, vol. abs/2402.06196, 2024.
- [23] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser and I. Polosukhin, "Attention is All you Need," *Advances in Neural Information Processing Systems*, vol. 30, 2017.
- [24] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer and V. Stoyanov, "RoBERTa: A Robustly Optimized BERT Pretraining Approach," *CoRR*, vol. abs/1907.11692, 2019.
- [25] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma and R. Soricut, "ALBERT: A Lite BERT for Self-supervised Learning of Language Representations," *International Conference on Learning Representations*, 2020.

- [26] T. Ranasinghe, C. Orasan and R. Mitkov, "TransQuest: Translation quality estimation with cross-lingual transformers," in *Proceedings of the 28th International Conference on Computational Linguistics*, 2020.
- [27] K. Nassiri and M. Akhloufi, "Transformer models used for text-based question answering systems," *Applied Intelligence*, vol. 53, pp. 10602--10635, 2023.
- [28] A. Merchant, E. Rahimtoroghi, E. Pavlick and I. Tenney, "What Happens To BERT Embeddings During Fine-tuning?," in *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, 2020.
- [29] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo and Y. Iwasawa, "Large language models are zero-shot reasoners," *Advances in neural information processing systems*, vol. 35, pp. 22199--22213, 2022.
- [30] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell and others, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877--1901, 2020.
- [31] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou and others, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in neural information processing systems*, vol. 35, pp. 24824--24837, 2022.
- [32] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun and H. Wang, "Retrieval-augmented generation for large language models: A survey," *ArXiv*, vol. abs/2312.10997, 2023.
- [33] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray and others, "Training language models to follow instructions with human feedback," *Advances in neural information processing systems*, vol. 35, pp. 27730--27744, 2022.
- [34] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar and others, "LLaMA: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.
- [35] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann and others, "PaLM: Scaling language modeling with pathways," *Journal of Machine Learning Research*, vol. 24, pp. 1--113, 2023.
- [36] D. Jurafsky and J. H. Martin, *Speech and Language Processing (2Nd Edition)*, Prentice-Hall, Inc., 2009.
- [37] T. Perry, "LightTag: Text Annotation Platform," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 2021.
- [38] L. Ramshaw and M. Marcus, "Text Chunking using Transformation-Based Learning," in *Third Workshop on Very Large Corpora*, 1995.

- [39] A. Bastos, A. Nadgeri, K. Singh, I. O. Mulang, S. Shekarpour, J. Hoffart and M. Kaul, "RECON: Relation Extraction Using Knowledge Graph Context in a Graph Neural Network," in *Proceedings of the Web Conference 2021*, Ljubljana, Slovenia, 2021.
- [40] Z. Zheng, Y.-C. Zhou, X.-Z. Lu and J.-R. Lin, "Knowledge-informed semantic alignment and rule interpretation for automated compliance checking," *Automation in Construction*, vol. 142, p. 104524, 2022.
- [41] S. Fuchs, M. Witbrock, J. Dimyadi and R. Amor, "Neural Semantic Parsing of Building Regulations for Compliance Checking," *IOP Conference Series: Earth and Environmental Science*, vol. 1101, p. 092022, 2022.
- [42] H. Hettiarachchi, M. Adedoyin-Olowe, J. Bhogal and M. M. Gaber, "TTL: transformer-based two-phase transfer learning for cross-lingual news event detection," *International Journal of Machine Learning and Cybernetics*, 2023.
- [43] L. Baldini Soares, N. FitzGerald, J. Ling and T. Kwiatkowski, "Matching the Blanks: Distributional Similarity for Relation Learning," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.
- [44] S. Y. Feng, V. Gangal, J. Wei, S. Chandar, S. Vosoughi, T. Mitamura and E. Hovy, "A Survey of Data Augmentation Approaches for NLP," in *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, 2021.
- [45] C. Shorten, T. M. Khoshgoftaar and B. Furht, "Text data augmentation for deep learning," *Journal of big Data*, vol. 8, pp. 1--34, 2021.
- [46] S. Longpre, Y. Wang and C. DuBois, "How Effective is Task-Agnostic Data Augmentation for Pretrained Transformers?," in *Findings of the Association for Computational Linguistics: EMNLP 2020*, 2020.
- [47] D. Jin, Z. Jin, J. T. Zhou and P. Szolovits, "Is BERT Really Robust? A Strong Baseline for Natural Language Attack on Text Classification and Entailment," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 8018-8025, 2020.
- [48] X. Han, Z. Zhang, N. Ding, Y. Gu, X. Liu, Y. Huo, J. Qiu, Y. Yao, A. Zhang, L. Zhang and others, "Pre-trained models: Past, present and future," *AI Open*, vol. 2, pp. 225-250, 2021.
- [49] H. Hettiarachchi and T. Ranasinghe, "Explainable Event Detection with Event Trigger Identification as Rationale Extraction," in *Proceedings of the 14th International Conference on Recent Advances in Natural Language Processing*, 2023.
- [50] K. Clark, M.-T. Luong, Q. V. Le and C. D. Manning, "ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators," in *International Conference on Learning Representations*, 2020.
- [51] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz and others, "Transformers: State-of-the-Art Natural Language Processing," in

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, 2020.

- [52] H. Nakayama, “seqeval: A Python framework for sequence labeling evaluation,” 2018. [Online]. Available: <https://github.com/chakki-works/seqeval>.
- [53] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimeshein, L. Antiga and others, “PyTorch: An Imperative Style, High-Performance Deep Learning Library,” in *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, 2019.
- [54] N. Nisbet, L. Ma and G. Aksenova, “Presentations of rase knowledge mark-up,” in *Proceedings of the European Conference on Computing in Construction, 2022*.

Annex A. SNOWTEC: Model Hyper-parameters

We used a common hyper-parameter setup summarised in Table 17 to generate comparable results while maintaining consistency among different entity and relation classification architectures. We also believe these parameters will provide a basis for future experiments. However, depending on the training data availability, we set the evaluation steps to eight for the entity classifier and 16 for the relation classifier, allowing five and 15 evaluations per epoch, respectively. These evaluations were conducted on a validation split of 10% of training data while using the remaining 90% for training.

Table 17. Hyper-parameter specifications

Parameter	Value
Learning rate	$1e^{-5}$
Batch size	16
Number of epochs	5
Early stopping patience	10
Maximum sequence length	128
Optimiser	Adam optimiser

Our experiments involved two model optimisation techniques: (1) hyper-parameter tuning and (2) language modelling. During hyper-parameter tuning, we particularly focused on optimising the model's learning rate. Our initial experiments revealed that adjusting this parameter yielded a more noticeable impact on the final results compared to others. More details about hyper-parameter tuning are discussed together with each classifier's results in Section 3.3.2.2. For language modelling, we used the maximum sequence length of 128 following the sequence length distribution in Figure 29. We set the batch size to 16 and the learning rate to $3e^{-5}$ with Adam optimiser, following our initial experiments. We allowed the model to train on 25 epochs with an early stopping patience of 10, considering the sensitivity of this task.

Annex B. SNOWTEC: Error Analysis of Entity Classifier

The best-performed entity classifier's (i.e. RoBERTa-Large with language modelling) results were further analysed to identify the causes of its performance limitations. Our analysis revealed two major factors: (1) unique attributes of entity categories and (2) contextual complexities that directly affect the model performance, as elaborated below.

Impact by unique attributes of entity categories: In entity classification, we aim to identify four categories (i.e. object, property, quality and value) defined in the CODE-ACCORD dataset (Section 3.3.1.2), which hold their unique characteristics. For instance, objects and values were mostly formed by one or two tokens/words. Additionally, a high proportion of values have numbers mostly coupled with units. In contrast, properties and qualities have a composition of short and long text sequences. Overall, qualities have more lengthy text spans than all other categories. We showcase

a few randomly selected entities from each category in Table 18 to further emphasise their distinctive attributes. Table 19 presents the outcomes of our analysis regarding the model's predictive performance for each entity category. Objects and values, the entity categories characterised by more definitive structures with shorter sequences, demonstrated comparatively higher F1 scores, as evident from the results. Since values have the fewest annotated samples, these results suggest that the integrity of the entity structure has more influence on the model's learning than the sample count. This fact is further confirmed by the results obtained under quality. Even though quality has more annotated samples ($\times 6$ than value samples), it resulted in the lowest F1 score due to its complex structure. Overall, our findings suggest that the inherent nature or structure of an entity category significantly influences the learning process of a transformer-based entity classifier, often surpassing the impact of the sample count within the training data.

Table 18. Entity categories with a few samples

object	property	quality	value
building	height	mechanical	one
doors	width	insulated	900mm
parking spaces	U-values	wheelchair-accessible	500 millimetres
windows	target primary energy rate	for entering a dwelling	0.7 metres
ventilation systems	floor area	insulated at ceiling level	0.15m/s

Table 19. Results of the best-performed entity classifier (RoBERTa-Large) on validation and test datasets across different entity categories

Category	Validation				Test			
	Support	P	R	F1	Support	P	R	F1
object	146	0.7955	0.7192	0.7554	317	0.5212	0.5047	0.5128
property	48	0.6154	0.6667	0.6400	98	0.4750	0.3878	0.4270
quality	139	0.7281	0.5971	0.6561	317	0.3805	0.3817	0.3811
value	28	0.9231	0.8571	0.8889	48	0.4773	0.4375	0.4565
Macro Average	361	0.7655	0.7100	0.7351	780	0.4635	0.4279	0.4444

Impact by the contextual complexities: In natural language, the organisational structure of textual elements within a context significantly influences entity identification. To measure contextual complexity in our experiments, we adopted the entity count within a sentence, where a higher count denotes a more detailed and intricate context. Table 20 provides a summary of the entity classifier's performance variations based on contextual complexities. As can be seen, the model demonstrates heightened accuracy in identifying entities within simpler contexts or those with fewer entities. Conversely, sentences with a high number of entities, often indicating longer and more complex

structures, yield less accurate results. Overall, the model may require exposure to more data to effectively grasp intricate contextual structures and improve the prediction performance of complex sentences.

Table 20. Results of the best-performed entity classifier (RoBERTa-Large) on test datasets depending on the entity count per sentence.

Entity Count	Category	Support	P	R	F1
≤ 3	object	70	0.6494	0.7143	0.6803
	property	17	0.6429	0.5294	0.5806
	quality	87	0.4674	0.4943	0.4804
	value	7	0.5000	0.4286	0.4615
	Macro Average	181	0.5649	0.5416	0.5507
> 3 and ≤ 5	object	100	0.5500	0.5500	0.5500
	property	44	0.5000	0.4091	0.4500
	quality	103	0.3846	0.4369	0.4091
	value	19	0.5556	0.5263	0.5405
	Macro Average	266	0.4975	0.4806	0.4874
> 5	object	147	0.4231	0.3741	0.3971
	property	37	0.3667	0.2973	0.2973
	quality	127	0.3028	0.2598	0.2797
	value	22	0.4000	0.3636	0.3810
	Macro Average	333	0.3731	0.3237	0.3465

Annex C: SNOWTEC: Error Analysis of Relation Classifier

The best-performed relation classifier's (i.e. RoBERTa-Large with data augmentation) results were further analysed to identify the factors that hinder its performance. The overall assessment displayed consistent performance across all relation categories, with a validation F1 score exceeding 80%, as depicted in Table 21. Although there were slight declines in the greater-equal and less-equal categories in the test set, given their association with a relatively small sample count, these drops can be deemed negligible. Altogether, these findings suggest that the relation category has no notable impact on the model's final performance.

Table 21. Results of the best-performed relation classifier (RoBERTa-Large) on validation and test datasets across different relation categories

Category	Validation				Test			
	Support	P	R	F1	Support	P	R	F1
equal	77	0.9625	1.0000	0.9809	15	0.7143	1.0000	0.8333
greater	57	1.0000	1.0000	1.0000	11	0.8182	0.8182	0.8182
greater-equal	89	0.9889	1.0000	0.9944	21	0.7273	0.7619	0.7442
less	9	1.0000	1.0000	1.0000	2	1.0000	1.0000	1.0000
less-equal	73	0.9865	1.0000	0.9932	14	0.6842	0.9286	0.7879
necessity	82	0.8902	0.8902	0.8902	206	0.9167	0.8544	0.8844
none	80	0.8592	0.7625	0.8079	200	0.8289	0.7750	0.8010
not-part-of	13	1.0000	1.0000	1.0000	2	0.6667	1.0000	0.8000
part-of	65	0.8657	0.8923	0.8788	162	0.7910	0.8642	0.8260
selection	93	0.8947	0.9140	0.9043	233	0.8498	0.8498	0.8498
Macro Average	638	0.9448	0.9459	0.9450	866	0.7997	0.8352	0.8011

Secondly, we explored the misclassifications to identify potential confusion between relation categories. This in-depth analysis revealed that complex textual structures influence the model's performance, leading to misclassification. Details of these findings are outlined below.

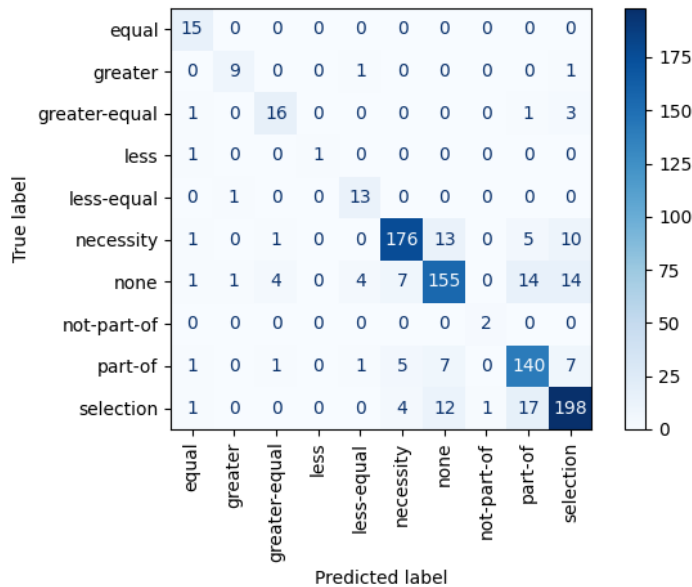


Figure 33. Confusion matrix of the best-performed relation classifier (RoBERTa-Large) on the test dataset

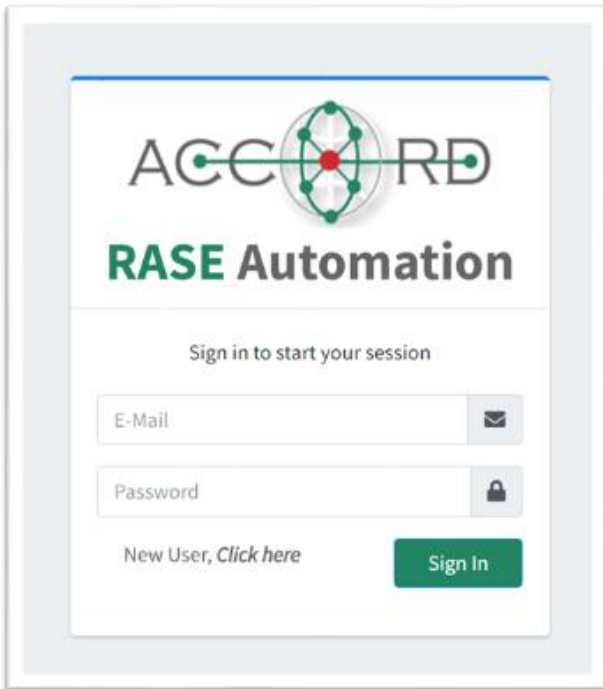
Impact by complex textual structures: The confusion matrix of the best-performed relation classifier on the test dataset is illustrated in Figure 33. This matrix exhibits clear patterns, confirming

the model's overall strong performance. Diagonal dominance signifies a majority of accurate predictions, accompanied by low off-diagonal values, which indicate minimal misclassification. However, we further investigated the misclassifications (particularly, the off-diagonal values above 10) to understand the factors contributing to the model's confusion. Our analysis pinpointed that most of these instances of confusion arose from complexities within the textual structure. To explain this finding, we summarise the frequently identified textual complexities in Table 22 together with relevant data samples.

Table 22. Frequently misclassified relations with identified causes and samples

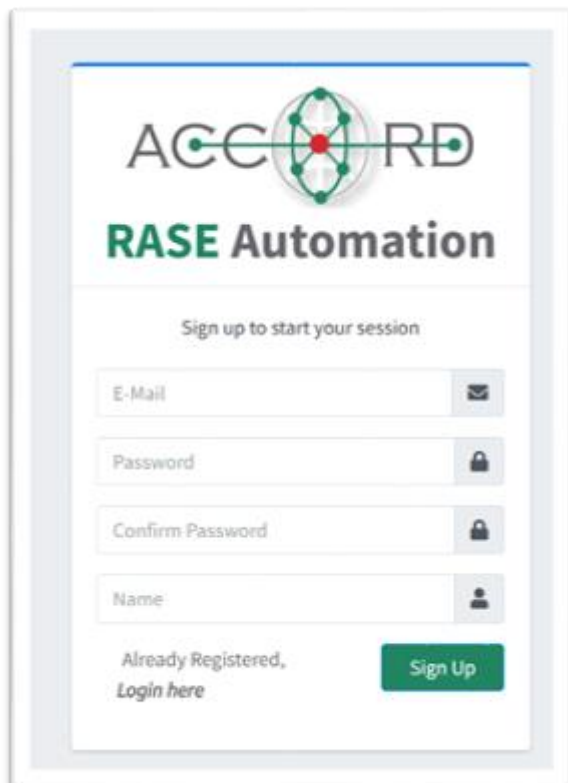
True Label	Predicted Label	Sample	Reason
necessity	none	Any <e1>walls</e1>, doors and windows should be insulated and <e2>draught-proofed</e2> to at least the same extent as in the existing building.	multiple necessities
		Ground arrays including <e1>header pipes</e1> and manifolds should be <e2>flushed as one system to remove all debris and purged to remove all air</e2>.	
none	part-of	The <e1>roof structures</e1> and <e2>joints</e2> must be properly inclined and sealed for water drainage.	conjoined elements
		The natural or mechanical <e1>ventilation system</e1> of a building shall be strong and its <e2>air-tightness</e2> shall be at least class B.	coreferences
none	selection	There shall be <e1>adequate means</e1> of ventilation <e2>provided for people in the building</e2>.	hierarchical relationships
		The air extract rate should be 20 litres per second per <e1>machine</e1> <e2>during use</e2>.	
selection	none	In an existing building, for the principal or <e1>main staff</e1> entrance or <e2>entrances</e2> to be accessible, an alternative accessible entrance should be provided.	multiple selections
		A building must be provided with a fire wall if <e1>it</e1> is situated adjacent to a neighbouring building or <e2>so close to a neighbouring building that the spread of fire is evident</e2>.	
selection	part-of	<e1>Requirements</e1> for <e2>accessibility</e2> should be balanced against preserving historic buildings or environments.	confusing text patterns
		The <e1>impact</e1> of the <e2>ground</e2> and crawl space on heat loss shall be taken into account in heat loss calculations.	

Annex D: RASE-LLM Tool Interface



The screenshot shows the login interface for RASE Automation. At the top, the ACC RD logo is displayed above the text "RASE Automation". Below this, the instruction "Sign in to start your session" is centered. The form contains two input fields: "E-Mail" with an envelope icon and "Password" with a lock icon. A link "New User, Click here" is positioned to the left of a green "Sign In" button.

Figure 34. Login Screen



The screenshot shows the sign up interface for RASE Automation. At the top, the ACC RD logo is displayed above the text "RASE Automation". Below this, the instruction "Sign up to start your session" is centered. The form contains four input fields: "E-Mail" with an envelope icon, "Password" with a lock icon, "Confirm Password" with a lock icon, and "Name" with a person icon. A link "Already Registered, Login here" is positioned to the left of a green "Sign Up" button.

Figure 35. Sign Up Screen

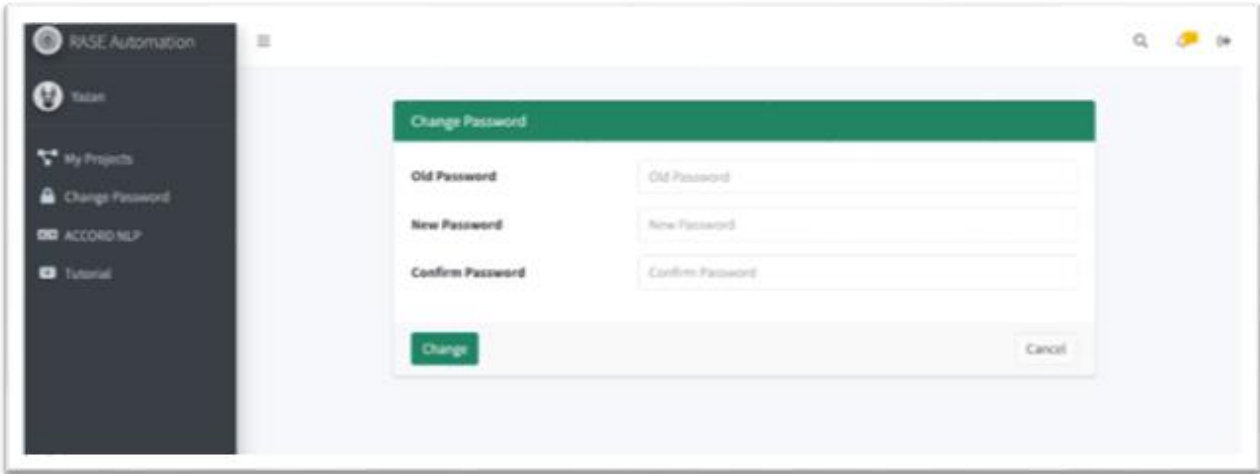


Figure 36. Change Password

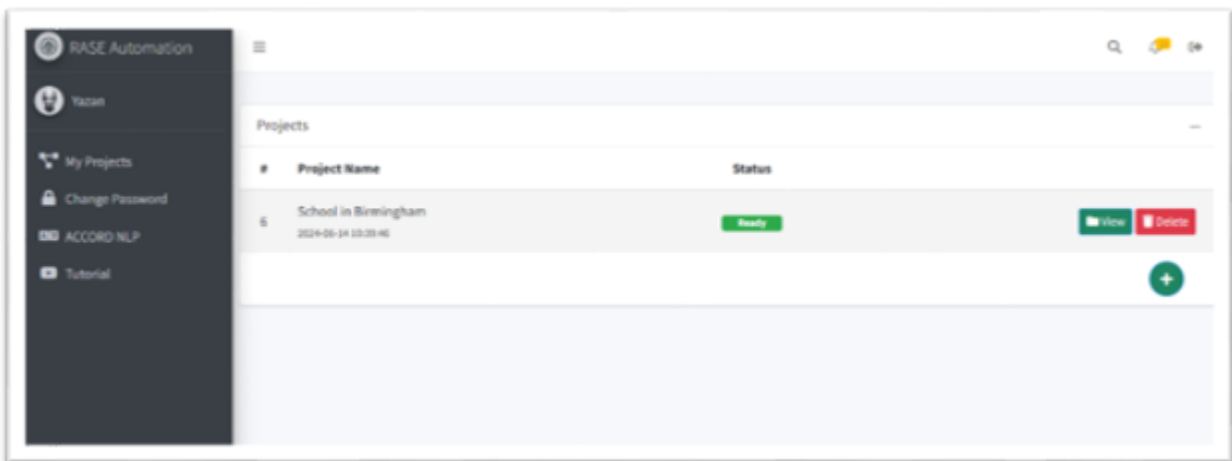


Figure 37. Projects List

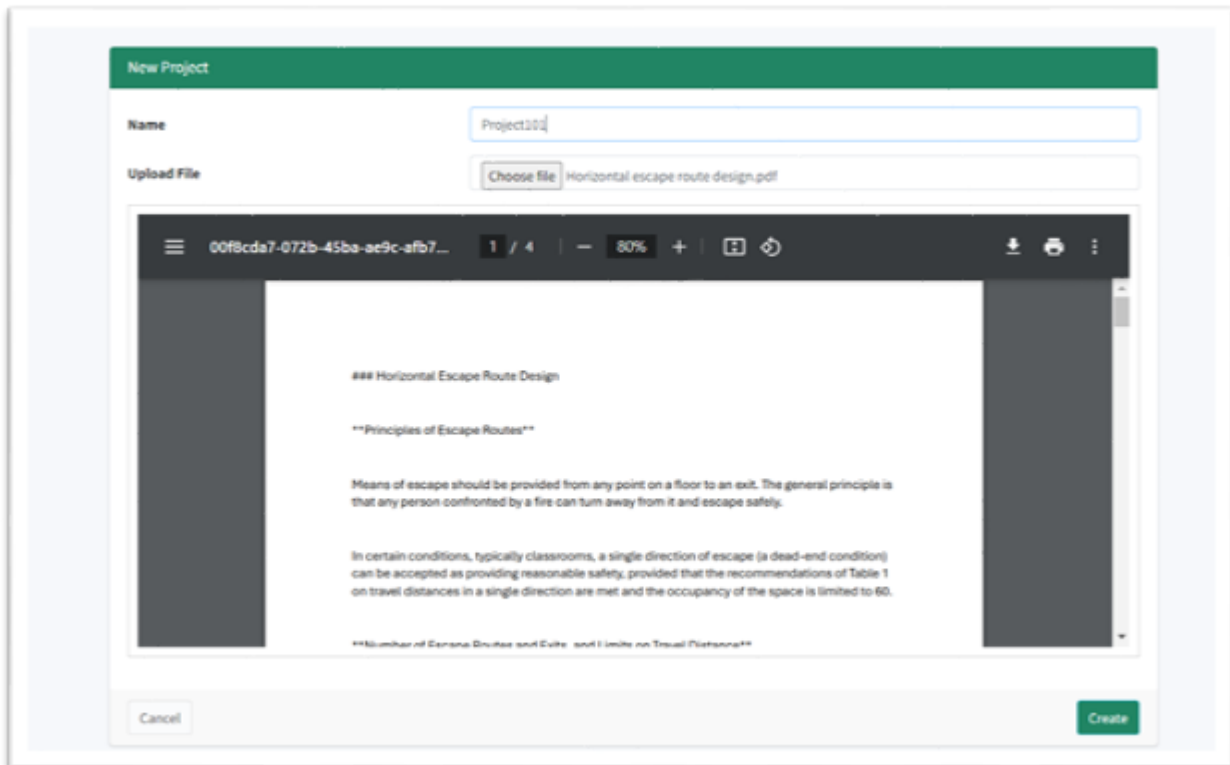


Figure 38. Add Project

