SN Computer Science (2025) 6:115
https://doi.org/10.1007/s42979-024-03629-z

ORIGINAL RESEARCH

(Semi-)automatic Extraction of Urban Planning Rules in French for Better Management of Land Artificialization

Maksim Koptelov[1,2,3] · Margaux Holveck[4] · Bruno Cremilleux[1] · Justine Reynaud[1] · Mathieu Roche[3,5] · Maguelonne Teisseire[2,3]

Received: 16 February 2024 / Accepted: 15 December 2024

Abstract
Land artificialization is a significant modern concern, as it is irreversible, diminishes agriculturally suitable land and causes environmental problems. Our project, Hérelles, aims to address this challenge by developing a framework for land artificialization management. In this framework, we associate urban planning rules in text form with clusters extracted from time series of satellite images. To achieve this, it is crucial to understand the planning rules with two key objectives: (1) to verify if the constraints derived from the rules are verifiable on satellite images and (2) to use these constraints to guide the labelling (or semantization) of clusters. The first step in this process involves the automatic extraction of rules from urban planning documents written in the French language. To solve this problem, we propose a method based on the multilabel classification of textual segments and their subsequent summarization. This method includes a special format for representing segments, in which each segment has a title and a subtitle. We then propose a cascade approach to address the hierarchy of class labels. Additionally, we develop several text augmentation techniques for French texts that can improve prediction results. Finally, we reformulate classified segments into concise text portions containing necessary elements for expert rule construction. We adapt an approach based on Abstract Meaning Representation (AMR) graphs to generate these portions in the French language and conduct a comparative analysis with ChatGPT. We experimentally demonstrate that the resulting framework correctly classifies each type of segment with more than 90% accuracy. Furthermore, our results indicate that ChatGPT outperforms the AMR-based approach, leading to a discussion of the advantages and limitations of both methods.

Keywords Natural language processing · Supervised learning · Data augmentation · Knowledge extraction

Abbreviations

3M	Montpellier Méditerranée Metropolis
AIR-FUD+	Automatic Identification of Rules in French Urban Documents extended
AMR	Abstract meaning representation
CV	Cross validation
FN	False negative
FP	False positive
LLM	Large language model
ML	Machine learning
MRR	Mean reciprocal rank
NLP	Natural language processing
NN	Neural network
PDF	Portable document format
PLU	Plan Local d’Urbanisme (local land plan)
POS	Part-of-speech
PPRI	Plan de Prévention des Risques naturels d’Inondation (Natural flood risk prevention plan)
SGD	Stochastic gradient descent
SVM	Support vector machines
TF	Term frequency
TF-IDF	TF-inverse document frequency
TP	True positive

✉ Mathieu Roche
mathieu.roche@cirad.fr

✉ Maguelonne Teisseire
maguelonne.teisseire@inrae.fr

1 UNICAEN, ENSICAEN, CNRS-UMR GREYC, 14000 Caen, France

2 INRAE, 34398 Montpellier, France

3 UMR TETIS, Univ. Montpellier, AgroParisTech, CIRAD, CNRS, INRAE, 34090 Montpellier, France

4 ICube, Université de Strasbourg, 67412 Illkirch, France

5 French Agricultural Research for Development (CIRAD), Montpellier, France

Published online: 28 January 2025

Introduction

Land artificialization is a serious problem in modern society. It is considered one of the principal factors contributing to biodiversity loss and climate change on our planet [9]. Additionally, land artificialization results in a net loss of forested and natural areas [11]. For example, any transformation of a natural area can lead to the disappearance of plants or animals from that area [33]. Moreover, artificial soil no longer absorbs CO_2, which contributes to the increase in global warming [9]. Moreover, land artificialization increases the risk of natural disasters such as floods and wildfires, which are very costly to society [25]. By definition, sealed ground does not absorb rainwater, so in the case of heavy rains, the risk of flooding is amplified [10]. Finally, land artificialization is irreversible. Once land is sealed, it is lost permanently, and the process cannot be reversed. As a consequence, the amount of land suitable for agriculture is decreasing over time.

Nevertheless, land artificialization meets the needs of human society by fulfilling demands for housing, industrial, and service infrastructures. These human activities tend to be concentrated in towns, which continue to expand [15, 86]. Further artificialization could be avoided through densification of already developed areas. Therefore, to reduce the impacts of artificialization, control over this process is necessary. Studies on land artificialization and natural risk management aim to address this problem, and our project, Hérelles[1], is a step forward in improving it.

In our project, we develop a methodology and software to enhance the management of land artificialization. Our approach associates constraints derived from urban planning documents with clusters obtained from time series of satellite images. These documents, written in French, include regulations such as authorizations, obligations, and prohibitions related to land use and development. The process begins by verifying whether these constraints, derived from the regulations, are verifiable via satellite images. We then label the clusters to assess the constraints. The initial phase involves extracting rules from urban planning documents related to our research sites. A *rule* in this context is defined as a formal regulation that can be translated into a constraint. For instance, the sentence *"If a piece of land can be built on then there must be a road that connects to it"* represents an ideal rule, as it can be translated into a straightforward *"if... then..."* constraint. Applying these rules involves certain considerations, and our clustering framework allows the user to reformulate the rules into constraints.

[1] https://herelles-anr-project.cnrs.fr.

However, before constraints can be formulated, they must be identified in the documents. A common approach for automatic rule extraction from texts consists of employing a machine learning classifier [5, 31, 45]. The main challenge in this modelling is data representation. Segmentation can be performed on the level of words [45], sentences [89], or segments (text parts containing multiple sentences) [73]. Segment representation better suits our needs because the rules of interest can be longer than one sentence. However, in this form, the segments cannot be directly added to the clustering process and must undergo further processing. This is why we explore the possibility of a two-step approach: first, to classify the segments and then to process them so that they can be used in our clustering approach.

Therefore, we develop a pipeline based on machine learning to automate the extraction of the rules. The first part of our pipeline provides a solution for detecting parts of the document containing constraints and identifying their types (e.g., *verifiable* and *non-verifiable* by satellite images) in an automatic or semiautomatic manner. We refer to this part of the pipeline as *rule identification*. We then continue the process by the reformulation of those parts of documents, referred to as *segments*, into concise portions of text containing all necessary elements for formulating the constraints. This process, which represents the second part of our pipeline, is termed *rule formulation*. The resulting text portions are intended for input into our clustering framework by an expert user who will validate and incorporate them as constraints into the clustering process.

Most of the state-of-the-art rule identification methods exploit supervised learning setting. However, data in French are less available, especially annotated data, and in the domain of our study in particular. The latter is not available at all, at least in an open access form. To address this, we constructed our own corpus by manually annotating the rules and defining a format for their representation [47]. Our data are labelled using four different classes (Verifiable and Non-verifiable by satellite images, Informative and Not pertinent) with a hierarchical structure. To perform their classification, we develop a specific framework that is based on multiple classifiers. To address the small size of our dataset, we perform data augmentation [48]. Finally, to validate our framework, we perform a set of experiments using both traditional natural language processing (NLP) methods and a state-of-the-art deep learning model. The results demonstrate that our framework can identify rules of different categories in French urban planning documents with high accuracy.

With respect to rule formulation, this task typically requires an ontology or a set of examples, both of which are unavailable. Moreover, our objective is to explore the possibility of semiautomatically formulating constraints for an expert. Therefore, we explore *abstractive text*

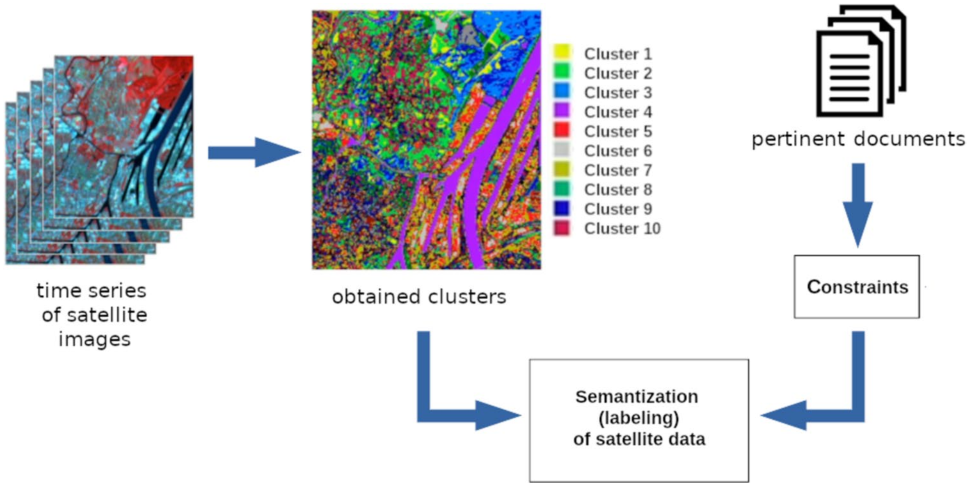


Fig. 1 Schematic representation of the framework for collaborative clustering. Possible labels of the clusters following semantization: *water* (Cluster 4), *forestry and wooded areas* (Cluster 3), *port and industrial area* (Cluster 5), *urban housing* (Cluster 6)

summarization, utilizing it to generate text portions for expert rule construction from segments classified as pertinent in the previous step. Classical unsupervised abstractive summarization involves the use of Abstract Meaning Representation (AMR) graphs, which are labelled graphs abstracting from grammatical structures and are suitable for various tasks. Recent advances include large language models (LLMs), particularly ChatGPT, which are trained on vast multilingual corpora. These models offer broader task-solving capabilities but provide less control and interpretability than AMR graphs. In this work, we explore both approaches.

A particular challenge related to AMR graphs is the lack of a model accessible for French texts. To overcome this, we employ an automatic translation system, enabling us to work with AMR graphs designed for English texts. Evaluating generated text portions without reference examples presents a separate challenge. We first conduct an automatic evaluation via a state-of-the-art approach that is based on the contextual similarity between the generated portions and the original segments. This provides an initial approximation of the resulting quality. Considering that our interest lies not only in summaries but also in their potential application to our project, we subsequently perform a manual evaluation with a group of nine annotators.

To sum up, in this work, we study the possibility of extracting constraints from French urban planning documents via both classical and modern state-of-the-art methods. There are five main contributions of this article:

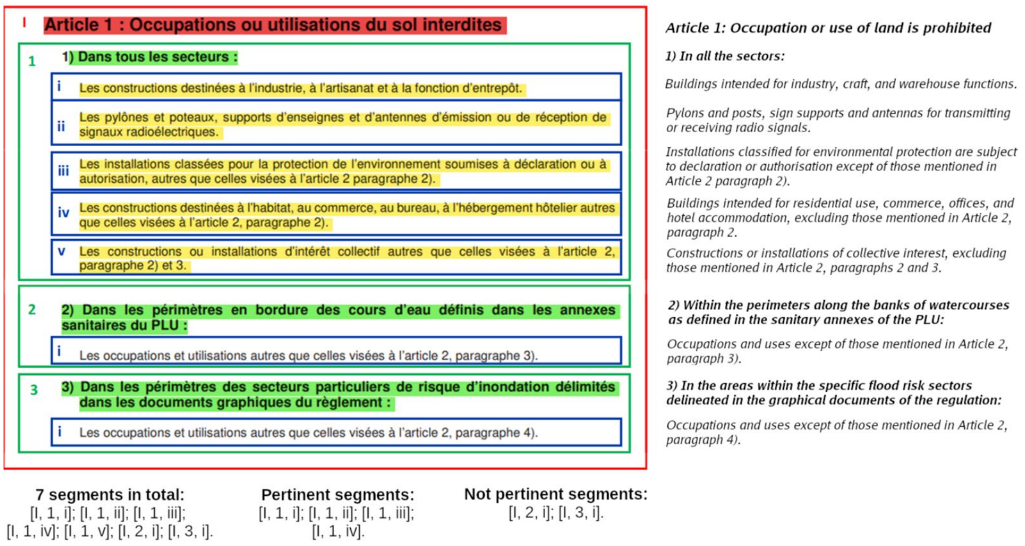
1. We develop an original pipeline to address a specific problem of rule identification in the context of urban planning and natural risk management. To our knowledge, this is the first work of its kind in these domains.
2. We propose a cascade approach for the multilabel classification of hierarchies of classes.
3. We design several text augmentation methods for the French language, which can improve the results of text classification. This is particularly uncommon for the text in French, especially on a topic outside of biological or financial domains.
4. We present an innovative pipeline based on AMR graphs for generating abstracts of textual segments containing rules in the French language.
5. We conduct a comparative analysis between widely used AMR graphs and ChatGPT, which no one has performed before to the best of our knowledge.

The rest of the article is organized as follows. "Background" provides basic definitions and presents the context of the problem. "Related Work" discusses related work on rule identification, text augmentation, and rule formulation. "The AIR-FUD+ Framework" presents our framework for rule identification and rule formulation. "Experimental Evaluation" describes the data used for the experiments and the experimental setup and presents the results. Finally, "Conclusion" concludes and highlights future perspectives.

Background

The main research topics of the Hérelles project are the effects of urbanization and natural risk management. The principal research site of the project concerns Montpellier Méditerranée Metropolis (3M) in France, a rapidly evolving area exposed to natural risks.

The final goal of Hérelles is to develop software for collaborative clustering [21] with a focus on analysing time series data from satellite images. We define a *cluster* as a group of elements sharing common attributes, in our case, derived from images and consisting of pixels within an image. In practice, these clusters represent geographical



I Article 1 : Occupations ou utilisations du sol interdites

1 1) Dans tous les secteurs :

i Les constructions destinées à l'industrie, à l'artisanat et à la fonction d'entrepôt.

ii Les pylônes et poteaux, supports d'enseignes et d'antennes d'émission ou de réception de signaux radioélectriques.

iii Les installations classées pour la protection de l'environnement soumises à déclaration ou à autorisation, autres que celles visées à l'article 2 paragraphe 2).

iv Les constructions destinées à l'habitat, au commerce, au bureau, à l'hébergement hôtelier autres que celles visées à l'article 2, paragraphe 2).

v Les constructions ou installations d'intérêt collectif autres que celles visées à l'article 2, paragraphe 2) et 3.

2 2) Dans les périmètres en bordure des cours d'eau définis dans les annexes sanitaires du PLU :

i Les occupations et utilisations autres que celles visées à l'article 2, paragraphe 3).

3 3) Dans les périmètres des secteurs particuliers de risque d'inondation délimités dans les documents graphiques du règlement :

i Les occupations et utilisations autres que celles visées à l'article 2, paragraphe 4).

Article 1: Occupation or use of land is prohibited

1) In all the sectors:

Buildings intended for industry, craft, and warehouse functions.

Pylons and posts, sign supports and antennas for transmitting or receiving radio signals.

Installations classified for environmental protection are subject to declaration or authorisation except of those mentioned in Article 2 paragraph 2).

Buildings intended for residential use, commerce, offices, and hotel accommodation, excluding those mentioned in Article 2, paragraph 2.

Constructions or installations of collective interest, excluding those mentioned in Article 2, paragraphs 2 and 3.

2) Within the perimeters along the banks of watercourses as defined in the sanitary annexes of the PLU:

Occupations and uses except of those mentioned in Article 2, paragraph 3).

3) In the areas within the specific flood risk sectors delineated in the graphical documents of the regulation:

Occupations and uses except of those mentioned in Article 2, paragraph 4).

7 segments in total:
[I, 1, i]; [I, 1, ii]; [I, 1, iii];
[I, 1, iv]; [I, 1, v]; [I, 2, i]; [I, 3, i].

Pertinent segments:
[I, 1, i]; [I, 1, ii]; [I, 1, iii];
[I, 1, iv].

Not pertinent segments:
[I, 2, i]; [I, 3, i].

Fig. 2 An extract from the PLU ZONE-5AU document (on the left), its translation to English (on the right) and the segments constructed from it (on the bottom). The title is highlighted in red, subtitles are in green and pertinent rules are in yellow

objects, with exact labels assigned by the user of the system (Fig. 1). One of the objectives of the project is to find new methods to facilitate the labelling, or semantization, of clusters obtained from these images. To address this, one proposed solution involves associating textual elements of interest (corresponding to the study themes and the spatiotemporal perimeter of the time series) with the extracted clusters. For example, on the basis of textual elements containing urban planning rules, we can infer that the label cannot be "a road", or alternatively, there must be a road-labelled cluster, as every building in this area requires access roads.

In the project workflow, we permit a user to add constraints into the clustering process to improve the results of the latter and to speed up the process [26]. We help the user formulate the constraints, for which we use text resources to extract the constraints, and we formulate them in the form of rules. For example, the sentence *"Pour être constructible, un terrain, doit avoir un accès à une voie publique ou privée ouverte au public"*[2] contains an obligation with regards to land use, and it can be converted to the following constraint: *"S'il n'y a pas de route adjacente à la zone constructible → erreur"*[3]. In fact, we intend to use formulated constraints twice: to improve the clustering process itself and to facilitate the labelling of the extracted clusters.

[2] In English: *"To be authorized for development, land must have access to a public road or a private road open to the public"*.

[3] In English: *"If there is no road adjacent to the buildable area → error"*.

To enable the implementation of the automatic extraction of constraints, the first identification of potential rules was manually performed by the expert within the documents of interest. These documents originate from the thematic expert corpus [41] and have been chosen for their richness in potential rules: they are the written regulations of the local land plans (PLU—*Plan Local d'Urbanisme*) and the natural flood risk prevention plans (PPRI—*Plan de Prévention des Risques naturels d'Inondation*) of the areas studied (see Fig. 2 for an example).

Not all the textual segments of a selected document can be taken into account. Moreover, some rules have an informative value, whereas others represent a strict constraint. Finally, the application of these rules is not always observable in satellite data. Therefore, the following classification of textual segments has been defined (Fig. 3).

A text segment is called *pertinent* if it can provide information within the scope of the project, corresponds to our research topics, and contains information in the context of selected research territories. The adequacy of research topics is verified by the presence of words from a nomenclature in the segment. A *nomenclature* is a collection of thematic words describing the research topics, for example, "chemin de fer" (railroad in English) and "stationnement" (parking). In total, *the Hérelles nomenclature*[4] contains 67 thematic concepts. In remote sensing, a key application area of our land artificialization management system, the utilization of nomenclature and/or ontologies is essential for the labelling process [4, 57]. We thus use them as a source of labels for

[4] https://doi.org/10.57745/OXACT8.

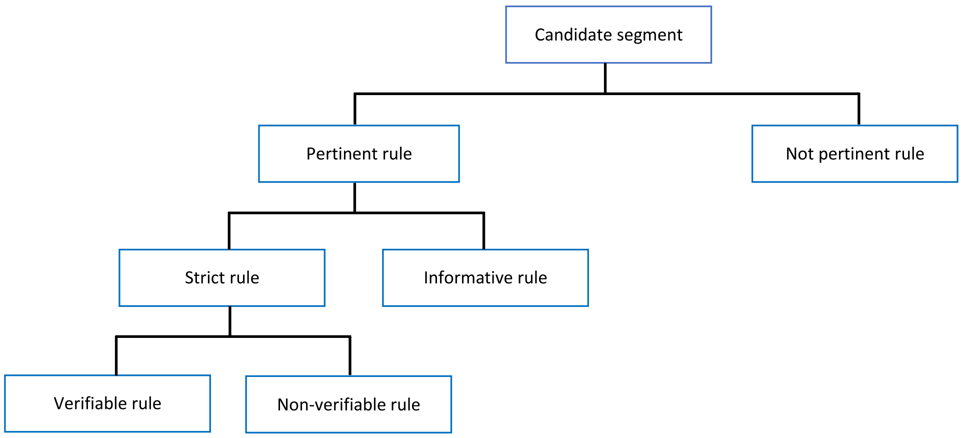


Fig. 3 Hierarchical representation of segments containing rules (Pertinent class) and not (Not pertinent)

our clusters. Specifically, we employ nomenclature from two perspectives: first, for the annotation of textual data, and second, as a basis for labelling satellite images.

All other texts that have not been classified as pertinent are considered to be *not pertinent*. These segments may be reminders of the law or definitions, elements that do not belong to the scope of our study. This includes layout elements, bibliography, headers, footers, and so on.

A *strict rule* concerns instructions that have legal force and are therefore enforceable by law. There is no ambiguity in its application (a strict rule clearly states what must be done, what is forbidden, what is allowed).

An *informative rule* refers to segments that provide detailed information about the study topic and the territory. This information is intended to help understand the results of the proposed solutions. The informative rules also include the segments presented as recommendations.

Some of the strict rules might be difficult to verify, for example, with satellite images. We therefore distinguish them by *verifiable* and *non-verifiable*. This distinction is important for the next steps in the project for selecting the constraints.

For more details on the context and concrete examples of the rules, please refer to the description of our corpus in [37].

Related Work

Rule Identification

Constraint (or rule) extraction or identification can be performed in multiple ways. The majority of approaches in the literature use traditional methods based on bag-of-words and classical NLP preprocessing. Kiziltan et al. [45] developed a system that automatically detects parts of text that describe constraints. They construct a dataset in which words in the sentences are labelled as belonging to constraints or not. To represent words, the authors exploit the stemmed representation of the words, part-of-speech (POS) tagging, and bag-of-words. A machine learning classifier based on support vector machines (SVMs) is then employed to solve the problem. Winter and Rinderle-Ma [89] focused not only on the extraction of constraints but also on grouping them and detecting and displaying relations between constraints. The authors use term frequencies and k-means clustering to achieve their tasks. In the preprocessing step, each document is chunked into sentences and POS tagged. Some constraints are not directly included in sentences. To overcome this problem, lemmatized representations of words are used. Anwar et al. [8] automatically extracted verification constraints from technical documents. The proposed framework is based on 3 core NLP concepts: sentence splitting, tokenization, and POS tagging.

In addition to traditional NLP, word embedding [6] can be used to extract information. Ba et al. [12] used word embedding vectors to derive spatiotemporal characteristics and special indicators from text documents describing food security problems. The method is then used for analysing the food crisis in West Africa. The problem can also be represented by combining word and image embeddings. Radford et al. [72] demonstrated that both images and text can be mapped into a common artificial space and that similar vectors can be used to match a caption with an image. Neptune and Mothe [64] solves a similar problem to ours. The authors do not extract constraints explicitly. In contrast, they map both images and text to a common space. The proposed framework can automatically annotate the change images with labels extracted from scientific documents related to the study area. The main disadvantage of this type of modelling is a certain lack of control over the process. In this type of approach, it is not possible to intervene in the vector representation or add other constraints.

The most efficient approach for information extraction is to use an encoder-decoder neural network (NN) [63, 93]. This network is pretrained on many texts to obtain their semantics in the form of high-dimensional vectors [44]. Then, using additional training, the NN can learn a specific task based on that representation. This additional training requires a smaller corpus than pretraining because the semantic information has already been acquired during pretraining [75]. The advantage of this approach is the abstraction from the grammar through the use of lexical embeddings. Its main limitation is that most available NNs are trained or pretrained only on English-language corpora, making them impossible to directly apply to other languages, and existing multilingual models are not sufficiently effective [68].

One of the most common encoder-decoder NNs is the BERT model [44]. In [75], the authors use an encoder-decoder model of type BERT pretrained on a large number of texts, which allows lexical embeddings to be obtained to represent semantics as high-dimensional vectors. Another model is then used for learning particular tasks on a smaller corpus. The problem is represented as a multilabel classification of sentences including constraints or not. To improve the applicability of the extracted information, ontologies can be used on top of the model. Wu et al. [91] propose a hybrid approach that uses an NN model to extract constraints and predefined rules to properly extract their relationships. This approach is limited by the availability of predefined rules and known relationships.

In this work, we experiment with both types of approaches: traditional NLP and a state-of-the-art encoder-decoder model. We represent the problem as a text classification task. The classifier that we develop can detect constraints in text segments constructed from documents of interest. Since our documents are in the French language, we are obliged to use one of the BERT extensions for that language. The most common among them are CamemBERT [58] and FlauBERT [51]. The first is a more general model, while the latter is better suited for downstream tasks [35]. In addition, CamemBERT outperforms FlauBERT [35, 43]. We thus use the former as the state-of-the-art approach implementation.

Text Augmentation

Data augmentation is usually used for mitigating a lack of data [48, 75] or for improving data imbalance [2]. Text augmentation can be performed in numerous ways, from straightforward implementations to the use of an LLM.

The straightforward approach can include shuffling words and deleting or replacing random words in the original sentences [24]. These types of methods introduce slight variation in the data but produce grammatical and syntactical errors. We test this type of approach in our experiments because of the ease of implementation.

A more advanced approach includes replacing selected words by their synonyms derived from specialized dictionaries or by a model of type BERT [44]. New sentences generated via this method introduce small variation to the original data and contain most of the linguistic features included in the original sentences. We implement these types of methods in our work.

The LLMs can generate semantically similar text without overlap at the level of words with original phrases. The resulting phrases might be too different from the original data; e.g., they may not contain important linguistic features. Additionally, there is not much control over the process; therefore, we do not use this type of model in our work.

Finally, existing solutions for automatic text augmentation [24] are available for texts in English only. In the case of multilanguage models, there are numerous limitations: in particular, the length of the input [74]. Few existing works that use the augmentation of French texts [48] have no publicly available code. We thus have no alternative to implementing our augmentation methods by ourselves.[5]

Rule Formulation

The task of rule formulation, often referred to as argument mining [50], typically requires an ontology or a set of examples. As constructing an ontology and gathering a set of examples is a resource-intensive process, at this stage of our project, we aim to explore the possibility of semiautomatically formulating constraints with an expert, who serves as the user of our clustering framework. To achieve this objective, we intend to explore text summarization as a method to generate concise text portions that contain essential elements, facilitating the expert in defining the constraints.

Given the large volume of work on text summarization [30, 88], a detailed review is beyond the scope of this article. In general, there are two principal types of text summarization: extractive and abstractive [32]. Extractive summaries consist of the original text's vocabulary and include only the most salient text units, whereas abstractive summaries are not constrained to the input vocabulary and have the ability to paraphrase and generalize. Abstractive summarization, known for providing shorter and more comprehensive results [30], aligns well with the requirements of our task.

Abstractive summarization can be carried out in a supervised or unsupervised manner. In a supervised setting, a model, typically an NN, is fine-tuned on a corpus of textual fragments and reference summaries [70]. In contrast, an unsupervised setting eliminates the need to train or fine-tune a model on text and reference summaries.

[5] The code of our implementations can be found in our framework repository: https://github.com/koptelovmax/AIR-FUD.

Instead, in this setting, summaries can be obtained directly by querying the model. Given that we lack reference summaries for the rules in our project and that obtaining them would be costly, an unsupervised approach is better suited for our needs.

A classical unsupervised approach for abstractive summarization involves the use of AMR graphs, which are directed, labelled graphs containing a semantic representation of a given text [13]. AMR graphs abstract away from grammatical structure and ignore stop words, making them suitable for applications such as question answering, machine translation, and text summarization [54]. The AMR formalism aims to provide a consistent representation for sentences with the same underlying meaning. For example, all the following phrases are represented by the same AMR graph: “He loves dogs,” “Dogs are loved by him,” and “He will love a dog” [96].

The authors of [56] initially proposed the use of AMR graphs to generate a summary of a text. They merge several graphs representing sentences in the text into a single AMR structure. They subsequently extract a subgraph that represents a summary via statistical learning. For this purpose, they employ a supervised learning setting and benefit from the use of an existing dataset with gold-standard AMR annotations. Dohare and Karnick [27] proposed an unsupervised learning setting. They present a hybrid approach: first, a few of the most important sentences are extracted; then, an AMR graph is constructed via the extracted sentences. Later, Dohare et al. [28] proposed an unsupervised approach for extracting a story graph—an AMR graph containing the summary of the input text—using co-reference resolution, the task of finding all expressions that refer to the same entity in a text. They identify the most important nodes via the inverse version of the term frequency [79] and then extract the story graph on that basis. Liao et al. [54] extends this idea to multiple document summarization. They select similar sentences and then construct an AMR graph representing them. In our approach, we utilize a combination of these ideas: extracting several subgraphs from the main graph representing the input segment and then merging them to obtain the summary graph.

A particular challenge related to AMR graphs is the lack of a model accessible for French texts. AMR graphs were originally designed for English texts [92]. Existing language-specific models are not adapted to French [7, 59, 92] or are not available in open access [84]. Uhrig et al. [83] recently demonstrated that it is indeed possible to use translated texts to exploit English-based AMR models. We, therefore, employ an automatic translation system, enabling us to work with AMR graphs in English with input texts in French.

Finally, unsupervised text summarization can be performed via LLMs [67, 70]. The development of ChatGPT [65], one of the most widely used LLMs, significantly changed all the downstream NLP tasks, including summarization [46]. According to Pu et al. [70], human evaluators prefer LLM summaries over other summaries generated by NNs. ChatGPT-based approaches for summarizing textual documents have numerous applications, particularly in the biomedical domain [39, 80, 94]. These approaches can even be used for summary evaluation without reference summaries produced by humans [81]. In this work, we test both a classical approach based on AMR graphs and an LLM model, more precisely ChatGPT, to solve our task of generating text portions with rule summaries for expert rule construction.

The AIR-FUD+ Framework

To automate extraction of the rules from thematic documents, we developed a framework that we refer to as *AIR-FUD+* (*Automatic Identification of Rules in French Urban Documents extended*). The AIR-FUD+ workflow has three main parts: segment preparation, segment classification, and rule formulation (Fig. 4). To train a classifier, we use a dataset that was already constructed in [37]. To construct that dataset, 1934 textual segments were manually annotated by the expert as belonging to one of the 4 classes: *Verifiable*, *Non-verifiable*, *Informative*, and *Not pertinent*. The details of the dataset are presented in “Experimental Evaluation”. In the following, we detail how segments are constructed

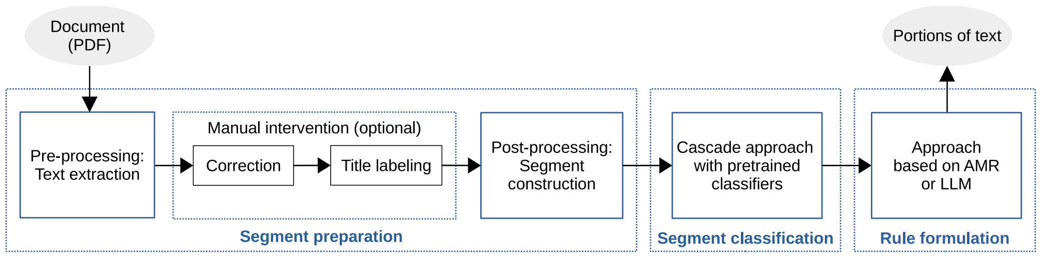

Document (PDF)
Portions of text
Pre-processing: Text extraction
Manual intervention (optional)
Correction
Title labeling
Post-processing: Segment construction
Cascade approach with pretrained classifiers
Approach based on AMR or LLM
Segment preparation
Segment classification
Rule formulation

Fig. 4 The AIR-FUD+ workflow presenting different steps in generation of portions of text containing elements for expert rule construction in new documents

from new documents and which methods are used to perform their classification. In addition, we present text augmentation techniques that we develop for improving the quality of the results. Finally, we detail the rule formulation module, which we use to generate portions of text containing necessary elements for expert rule construction.

Segment Preparation

Segment preparation consists of three steps: text extraction, manual intervention, and segment construction (Fig. 4).

Text Extraction

Both document types in our thematic corpus, the PLU and PPRI, are originally in the portable document format (PDF). Therefore, we first extract text from the PDF files of these documents in the preprocessing step. The output of this step is the set of text fragments in the form of a plain text file. We define a *fragment* as one or several sentences separated by empty lines.

Manual Intervention

In this step, we manually correct the extracted text. This includes cleaning of the text, in which we remove unnecessary fragments such as the tables of the contents and figure descriptions. In addition, we perform *title labelling*. For that, we label all the fragments that are titles and subtitles via sets of special characters, as described in [37]. Note that this step is optional since our implementation has a fully automatic mode, in which titles and subtitles are extracted automatically from new unseen documents. However, according to our experiments, manual intervention significantly improves the quality of the results; thus, manual intervention is strongly recommended.

Segment Construction

In postprocessing, we perform automatic construction of text segments from labelled fragments. A *segment* in our representation must have a title, subtitle, and a rule, whereas the presence of a subsubtitle is not mandatory (Fig. 2). Subsubtitles are detected automatically by our segment construction module via a set of predefined patterns. In these patterns, the decision is made on the basis of the presence of certain characters in the fragment. A *rule* in our representation is a fragment that is not a title, subtitle, or subsubtitle. The dataset that we constructed contains detailed examples of segments constructed from different numbers of fragments [37].

Segment Classification

Once the segments are constructed, the next step is to perform their classification. To address this, we propose a *cascade* approach, developed with a logical sequence and structured steps. Initially, we focus on identifying relevant rules and filtering out irrelevant. Next, we examine whether these rules are visible in satellite data. Finally, we determine whether they can be verified via satellite images. This cascade classification aligns closely with our practical needs, where sometimes only Verifiable rules are needed, and sometimes broader rules of the Pertinent class are sufficient.

Specifically, we develop our cascade approach as follows. We split the task into three binary classifications applied one by one (Fig. 5). In the *1st classification*, we classify segments by Pertinent (containing the rules) and Not pertinent (all other text). For that, we treat the Verifiable, Non-verifiable, and Informative classes all together as Pertinent. In the *2nd classification*, we classify segments by Strict (containing strict rules) and Informative (containing not strict rules), for which we treat the Verifiable and Non-verifiable classes as a whole. Finally, we classify the Verifiable and Non-verifiable classes in the *3rd classification*.

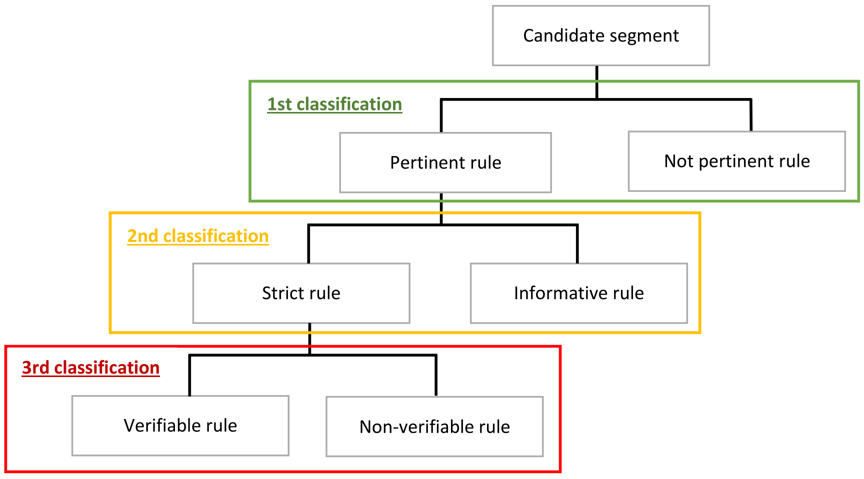


Fig. 5 The cascade classification of hierarchy of classes

For each classification, we select a binary classification model that performs best. To make this selection, we develop and test several baseline methods (trigger words, vector similarity, and machine learning using frequency vectors) and a state-of-the-art approach based on deep learning (CamemBERT). We define these methods in "Classification Methods". In addition, we develop text augmentation techniques, which we use to enrich the annotated corpus and improve the prediction results. We define these techniques in "Text Augmentation".

Classification Methods

Trigger Words

In this method, we exploit a list of trigger words extracted from the expert corpus to facilitate the automatic extraction of rules. A *trigger word* indicates the presence of a rule in the fragment or its neighbourhood. We have 43 such words in total, which were manually chosen by a geographical expert, for example, "être interdit" ("is prohibited" in French) or "admettre" ("admit"). The full list can be found in our code (see footnote 5). In this method, we first find their stemmed representation. For given examples, it respectively corresponds to "être interd" and "admettr". Next, we analyse their appearance in segments. By default, all the segments are assigned to the negative class. We check whether a trigger is present in a neighbourhood of segments of size n, the exact value of which is determined experimentally ("Model Parameters"). If yes, all of these segments are considered to be in the positive class.

Vector Similarity Model

For preprocessing, we perform tokenization of segments, remove stop words, and obtain a stemmed representation of the remaining words. We use the result to compute the term frequency (TF) [79]:

$$TF(t,d) = \frac{freq(t,d)}{\sum_m freq(t,d)},$$

where $freq(t, d)$ is the frequency of term t in segment d, m is the total number of terms, and the TF-inverse document frequency (TF-IDF) [16, 53] is:

$$TF\text{-}IDF(t,d) = TF(t,d) \cdot \log \frac{N}{df(t)},$$

where N is the total number of segments and where $df(t)$ is the number of segments containing t. We use both frequencies to construct the frequency vectors. To achieve this, we represent each segment d by a vector of term frequencies, $F(d)$, with a size of m. Each element t of the segment in this vector corresponds to $TF(t, d)$ or $TF\text{-}IDF(t, d)$. We then use the resulting vectors for solving the binary classification task, which is modelled as follows. When a new segment d_{new} arrives, we compute the mean of its similarity with all segments of the positive class and then with all segments of the negative class:

$$sim_{class}(d_{new}) = mean\Bigg(\sum_{i \in \{d_{class}\}} F(d_{new}) \times F(i) \Bigg),$$

where d_{class}—segments are labelled *class*. If $sim_{pos}(d_{new}) > sim_{pos}(d_{new})$, d_{new} receives the positive class and is negative otherwise.

Machine Learning (ML) Using Frequency Vectors

In this method, we use the same vectors $F(d)$ to represent segments as in the previous method. The difference is that this time, we employ a machine learning model to learn a binary classifier. When a new segment d_{new} arrives, we classify it with the trained model.

CamemBERT

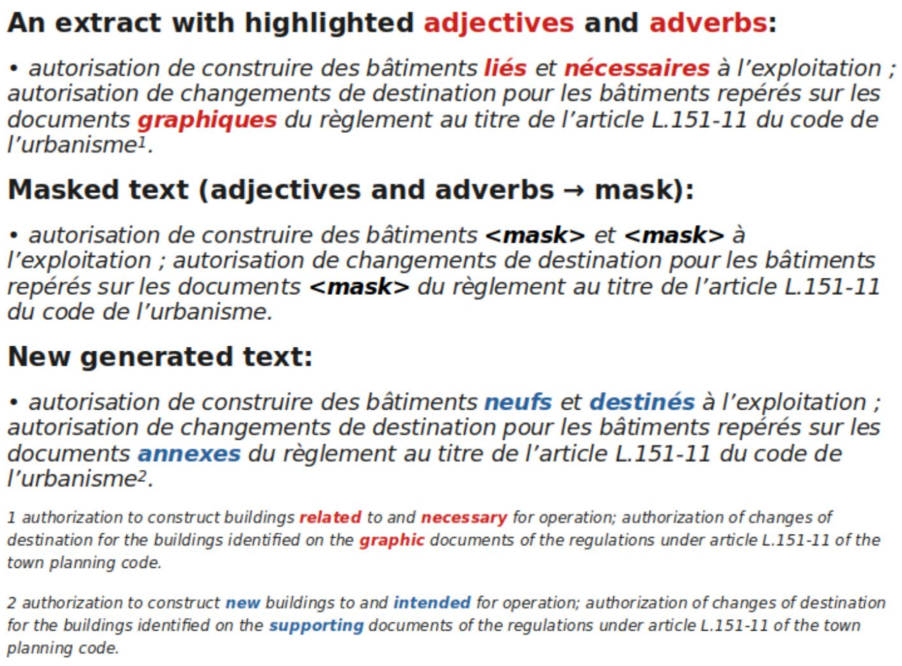
Following our discussion in "Related Work", we employ CamemBERT as the state-of-the-art approach for text classification. In this approach, we fine-tune the original CamemBERT model for binary classification tasks via labelled segments from our dataset. We then use the fine-tuned model to classify each unseen segment.

Text Augmentation

To improve the results of the CamemBERT model, we augment the training data, which are then used for fine-tuning the model. We use each segment k times to generate k new segments. Using this method, we increment the number of examples of an underrepresented class, which improves the imbalance of our data. In the following, we detail different strategies that we use to generate new text. They are based on grammatical information, semantic information, etc.

POS-Driven Method

In this method, we replace certain words [48] in each segment with semantically meaningful phrases derived by CamemBERT. To achieve that, we mask certain parts of speech in the segment and then ask CamemBERT to solve the masked word prediction task, a main principle of language models of type BERT [44]. In this task, the model tries to predict the original vocabulary of the masked content on the basis of only its context. For the selection of parts of speech, we mask all adjectives and adverbs as in [48] because they are usually not thematic key phrases. The aim of this method is to introduce a variety of newly generated segments without changing the main content. In practice, newly generated sentences are grammatically correct but do not always have the same meaning (Fig. 6). We also experiment with masking all verbs and all nouns in the segments to generate more diverse examples.



An extract with highlighted adjectives and adverbs:

• *autorisation de construire des bâtiments **liés** et **nécessaires** à l'exploitation ; autorisation de changements de destination pour les bâtiments repérés sur les documents **graphiques** du règlement au titre de l'article L.151-11 du code de l'urbanisme*[1].

Masked text (adjectives and adverbs → mask):

• *autorisation de construire des bâtiments **<mask>** et **<mask>** à l'exploitation ; autorisation de changements de destination pour les bâtiments repérés sur les documents **<mask>** du règlement au titre de l'article L.151-11 du code de l'urbanisme.*

New generated text:

• *autorisation de construire des bâtiments **neufs** et **destinés** à l'exploitation ; autorisation de changements de destination pour les bâtiments repérés sur les documents **annexes** du règlement au titre de l'article L.151-11 du code de l'urbanisme*[2].

*1 authorization to construct buildings **related** to and **necessary** for operation; authorization of changes of destination for the buildings identified on the **graphic** documents of the regulations under article L.151-11 of the town planning code.*

*2 authorization to construct **new** buildings to and **intended** for operation; authorization of changes of destination for the buildings identified on the **supporting** documents of the regulations under article L.151-11 of the town planning code.*

Fig. 6 An example of a text augmentation using CamemBERT and masked word prediction

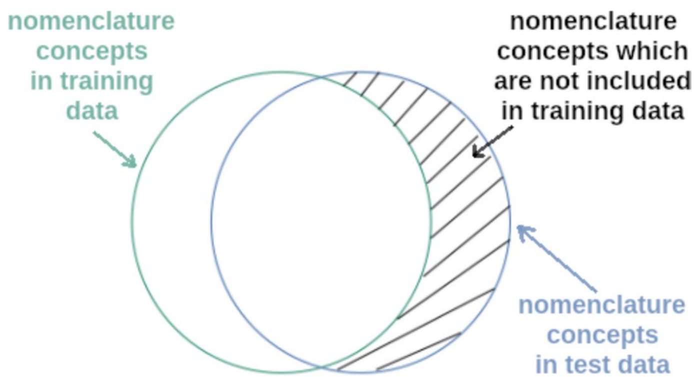

nomenclature concepts in training data
nomenclature concepts which are not included in training data
nomenclature concepts in test data

Fig. 7 Euler diagram representing distribution of nomenclature concepts in training and test data. The set of nomenclature concepts included in train data is shown in green color and the set of nomenclature concepts presented in test data is shown in blue. The set of nomenclature concepts which are included in test data but not included in train data are shown using black stripes

Semantic-Driven Method

This method is based on the hypothesis that an ideal classifier would classify segments by the presence of words from the nomenclature. In this hypothesis, we assume that not all nomenclature words included in the test data are present in the training data. Therefore, we can attempt to artificially include them in the training data (Fig. 7). To achieve this, we replace a random word in the segment with a random concept from an *enriched nomenclature*. Since we do not have the full list of nomenclature concepts, but only 67 thematic words, we use a special dictionary [19, 60, 61] to enrich our thematic words with synonyms. We select s synonyms at most for the dictionary (the exact value is detailed in "Model Parameters"). A new segment generated by this method is not always grammatically correct, but it is guaranteed to include at least one nomenclature concept from an enriched vocabulary.

Table 1 Number of concepts in the nomenclature, expert nomenclature and their extended versions

Type of nomenclature	Number of concepts
Nomenclature	67
Enriched nomenclature (WordNet, s = 5)	134
Enriched nomenclature (Agrovoc, s = 5)	120
Enriched nomenclature (DES, s = 5)	153
Expert nomenclature	207
Enriched expert nomenclature (WordNet, s = 5)	406
Enriched expert nomenclature (Agrovoc, s = 5)	429
Enriched expert nomenclature (DES, s = 5)	487

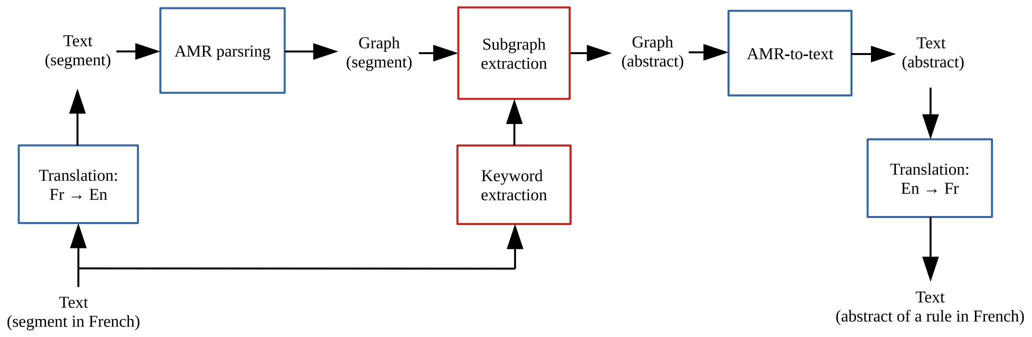


Fig. 8 The workflow presenting different steps of the AMR-based approach. Modules in red are developed by ourselves and modules in blue are existing implementations

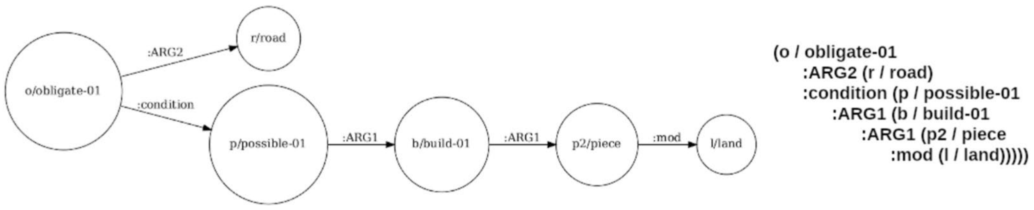


Fig. 9 An example of AMR parsing for the sentence *"If a piece of land can be built on then there must be a road"* in the graph format (on the left) and corresponding to its description in the Penman format (on the right)

Combined Approach

This method is based on two previous ideas. First, we check the presence of the words from an extended *expert nomenclature*[6] in the segment (Table 1). If at least one of these words is present in the segment, we use the latter to generate a new segment with POS-Driven Method.

After identifying relevant segments, we move to the last part of our framework: rule formulation. In that part, we generate the necessary elements for expert rule construction by creating summaries of the relevant segments. In the following, we define the methods used for this process.

Rule Formulation

In this section, we detail two different approaches used for summarizing the relevant rules determined in the previous parts of our framework: an approach based on classical AMR graphs and the state-of-the-art LLM-based approach. The first approach is developed by us, while the latter is used out of the box. In the following, we provide a detailed explanation of both of these approaches.

[6] A nomenclature, manually extended by an expert (included in our code).

Approach Based on AMR Graphs

The overall workflow of the approach based on AMR graphs consists of several principal modules: automatic translation, AMR parsing, keyword extraction, summary graph extraction, and AMR-to-text (Fig. 8). In the following paragraphs, we elaborate on each of these modules. Except for keyword extraction and summary graph extraction, we use existing implementations available in open access. For the exceptions, we develop these modules ourselves.

Automatic translation

To utilize AMR graphs, which were originally designed for English texts [92], we employ an automatic translation system, which we detail later in "Implementation Details". The purpose of this system is to first automatically translate our input segments from French to English (Fr → En) and then automatically translate the resulting summaries from English to French (En → Fr).

AMR Parsing and AMR-to-Text

There are two main procedures associated with AMR graphs: AMR parsing and AMR-to-text. AMR parsing is a process of constructing an AMR graph from a given text (Fig. 9), and AMR-to-text is the generation of text from a given AMR graph.

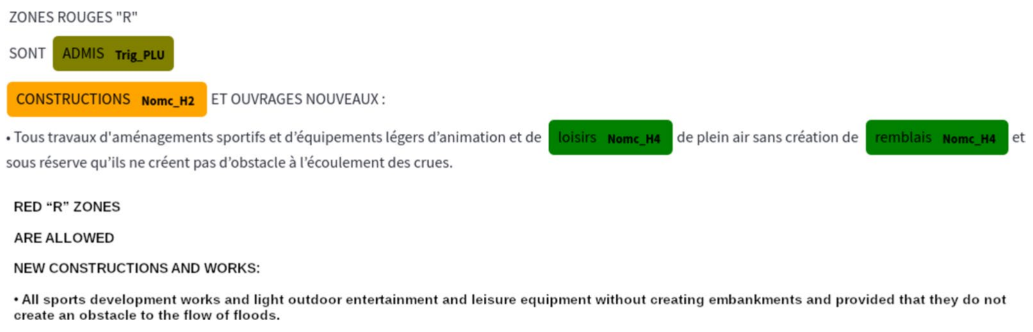
AMR parsing and AMR-to-text can be performed in several ways. The implementation we use for both tasks is based on a neural network, which was trained on a corpus of annotated examples with reference summaries. After parsing, the graph is stored in the Penman format [42], a serialization format originally designed for encoding semantic dependencies represented by directed, rooted graphs (Fig. 9). Penman is widely used in AMR applications, and we employ this format to extract the summary graph in the dedicated module. The same format is used as an input for the AMR-to-text module.

Keyword Extraction

Our AMR-based approach requires the identification of the most important words, which we refer to as *keywords*, and we exploit an existing implementation of a keyword extractor to find them. By a keyword, we understand a nomenclature concept, a trigger word, or a named entity. We already used the notions of trigger words and nomenclature in the previous section dedicated to segment construction.

A *named entity* is a real-world object such as a person, location, organization, product, etc. that can be identified by a proper name [62]. The implementation that we use automatically searches for geopolitical entities (such as countries and cities) and locations (geographical objects) on the basis of an open-source library. All the keywords are detected automatically in this implementation via their stemmed representation (Fig. 10). For example, for trigger words *"être interdit"* and *"admettre"*, this representation corresponds to *"être interd"* and *"admettr"*, respectively. More details on the implementation are provided in "Implementation Details". Once the keywords are found, we pass them to the summary graph extraction module (Fig. 8).

Summary Graph Extraction AMR allows for the rephrasing of input text into a more compact fragment, avoiding some details. However, it does not permit significant compression of an input fragment without additional manipulations with the graph. One way to achieve that is to extract a subgraph called the *summary graph* [56], which contains the most essential information related to the task. To extract such a graph, we use *keywords*, which we find automatically with the keyword extraction module described above. For each keyword, we search for a corresponding node called the *target node* in the graph. Once this node is found, we extract the subtree associated with it. In addition to the subtree, we extract the full path


ZONES ROUGES "R"

SONT ADMIS Trig_PLU

CONSTRUCTIONS Nomc_H2 ET OUVRAGES NOUVEAUX :

• Tous travaux d'aménagements sportifs et d'équipements légers d'animation et de loisirs Nomc_H4 de plein air sans création de remblais Nomc_H4 et sous réserve qu'ils ne créent pas d'obstacle à l'écoulement des crues.

RED "R" ZONES

ARE ALLOWED

NEW CONSTRUCTIONS AND WORKS:

• All sports development works and light outdoor entertainment and leisure equipment without creating embankments and provided that they do not create an obstacle to the flow of floods.

Fig. 10 An example of a textual segment with highlighted keywords (at the top) and its translation to English (at the bottom). "Trig_PLU" corresponds to trigger words and "Nomc_H*n*" to nomenclature concepts with *n* indicating the level of hierarchy [36]

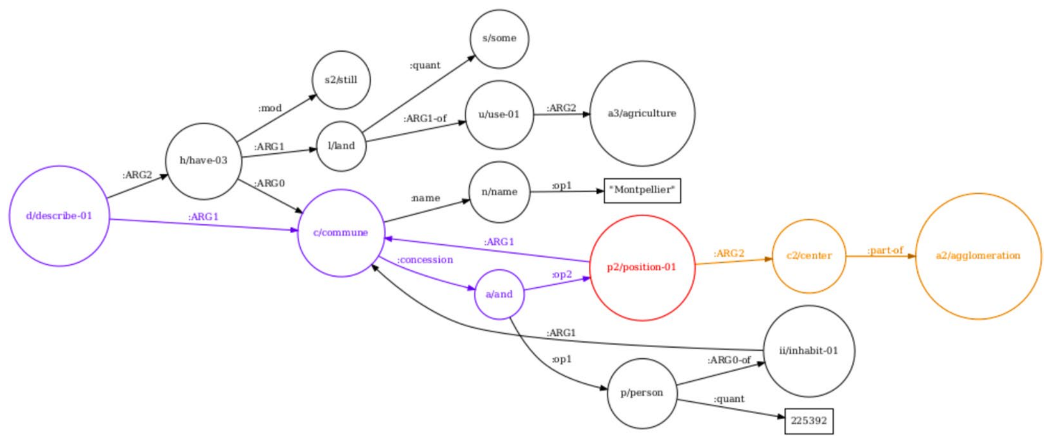
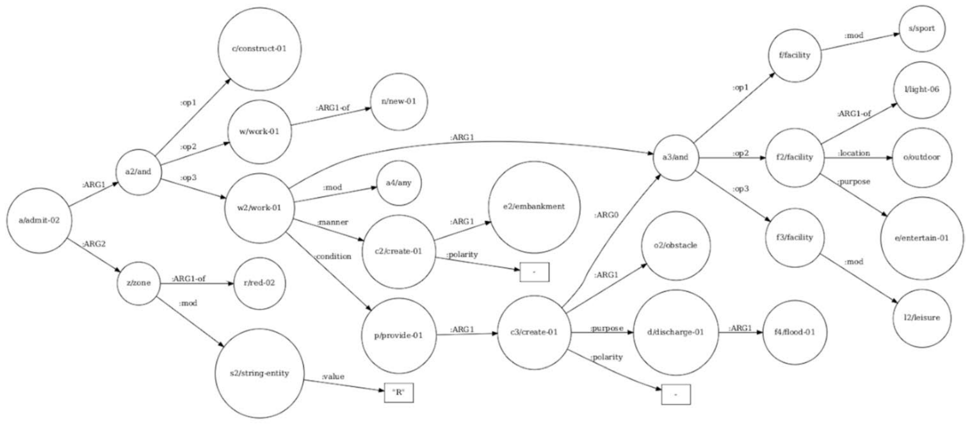


Fig. 11 En example of a subgraph (shown in different colors) extracted with the keyword "position". The target node is highlighted in red, the subtree is in beige and the path from the root to the target node is in violet

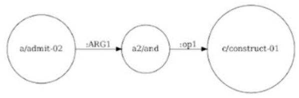
from the root node to the target node, which includes all the nodes and edges directly connected to the target node (Fig. 11).

Once the subgraphs corresponding to different keywords are extracted, we sum them by merging the same nodes and edges. This is realized by the standard functionality of the AMR libraries that we use (see "Implementation Details" for details). The resulting "fusionned" graph is then used to generate the final summary (Fig. 12).

We perform subgraph extraction for all segments that are "large enough". We use a heuristic to make this decision: if a segment contains fewer than k keywords, we use the full AMR graph to generate a summary; otherwise, we extract subgraphs using keywords. In our implementation, we set $k = 3$, which is guided by intuition: a segment with too few keywords (i.e., fewer than 3) is semantically poor; thus, the full segment must be used for summarization without the need to extract a summary graph.



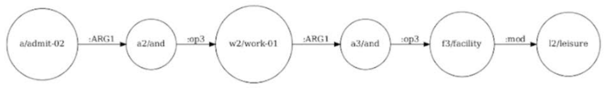
Subgraph 1 corresponding to the keyword «CONSTRUCTIONS» (associated word in English is «construct»):



Generated text in English: Admitting construction and.

Automatic translation to French: Admettre la construction et.

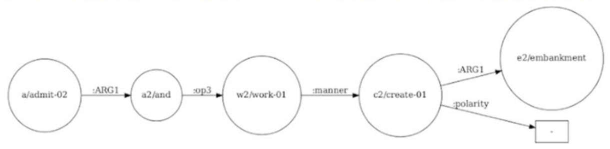
Subgraph 2 corresponding to the keyword «loisirs» (associated word in English is «leisure»):



Generated text in English: admitted to work in leisure facilities and

Automatic translation to French: admis au travail dans des installations de loisirs et

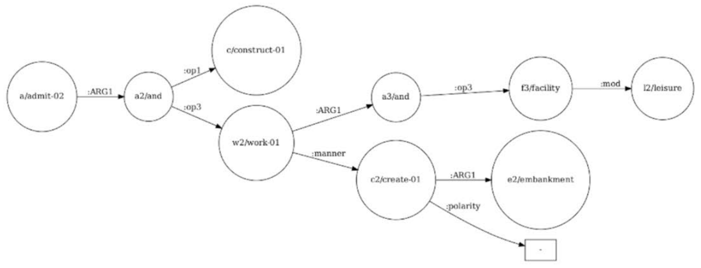
Subgraph 3 corresponding to the keyword «remblais» (associated word in English is «embankment»):



Generated text in English: Admitting that, and that the work was done without creating embankments.

Automatic translation to French: Admettant cela, et que le travail a été fait sans créer de remblais.

Final summary graph obtained by fusioning subgraph 1, 2 and 3:



Generated text in English: Admitted is construction and work on leisure facilities without creating embankments.

Automatic translation to French: Admis est la construction et le travail sur les installations de loisirs sans créer de remblais.

Fig. 12 An example of constructing a summary graph using keywords. The AMR graph corresponding to the segment from Fig. 10 is at the top, the final summary graph is at the bottom, and the subgraphs extracted using different keywords are in the middle

Table 2 Number of segments corresponding to each class and each document

Document	Number of segments								
	1st classification			2nd classification			3rd classification		
	Pertinent	Not pertinent	Total	Strict	Informative	Total	Verifiable	Non-verifiable	Total
PLU Montpellier ZONE-A	29	42	71	27	2	29	8	19	27
PLU Montpellier ZONE-N	48	78	126	39	9	48	12	27	39
PLU Montpellier ZONE-AU0	31	58	89	28	3	31	6	22	28
PLU Montpellier ZONE-14AU	23	59	82	21	2	23	8	13	21
PLU Montpellier ZONE-5AU	30	64	94	27	3	30	4	23	27
PLU Montpellier ZONE-4AU1	65	101	166	55	10	65	7	48	55
PPRI Montpellier	88	37	125	83	5	88	22	61	83
PPRI Grabels	54	45	99	47	7	54	33	14	47
PLU Grabels	306	776	1082	261	45	306	47	214	261
Total	674	1260	1934	588	86	674	147	441	588
	35%	65%	100%	87%	13%	100%	25%	75%	100%

The last problem we faced in summary graph extraction is how to find French keywords in the English versions of segments. This process is called *text alignment* [18], and we benefit from using an existing implementation [78].

Approach Based on LLM

As discussed in "Rule Formulation", the application of an LLM in computer science systems is not a novel concept [67, 80]. The main limitation of this approach is the probabilistic nature of LLMs, resulting in nondeterministic outputs. However, their ease of use and statistically favourable results, particularly in text summarization tasks [70], are notable advantages.

To generate portions of text containing elements for expert rule construction, we automatically query ChatGPT, an instance of an LLM. In this process, we use the following query in the French language: *"résumé abstrait à 20 mots maximum:"*,[7] which is followed by a textual segment for which a text portion needs to be obtained. This query was selected experimentally, and a limit of 20 words was added to avoid long outputs or multiple sentences. Without this restriction, the system might produce a sentence as long as the original segment or output multiple sentences. Given that our AMR-based summaries contain 20 words on average, we use the same number of words in the query for ChatGPT. Importantly, ChatGPT does not consider this a strong constraint. As a result, it occasionally generates phrases longer than the specified limit.

[7] In English: "abstractive summary in 20 words maximum".

Experimental Evaluation

Data

Our dataset [37], which we use for the experiments, contains 1934 labelled segments extracted from 9 PLU and PPRI documents. In the data, the segments are labelled as belonging to one of 4 classes: Verifiable, Non-verifiable, Informative, and Not pertinent. We combine the classes Verifiable and Non-verifiable to derive the class *Strict*, and we combine the classes Strict and Informative to derive the class *Pertinent* (Fig. 3). The detailed statistics for each type of segment and each document are presented in Table 2.

To evaluate the rule formulation, we selected 64 segments of the class Verifiable, 32 segments from Informative, and 32 from Non-verifiable [37]. By fixing the number of segments in this way, we ensured diverse selection encompassing all types within the Pertinent class (Fig. 3). We prioritize the class Verifiable because of its greater importance for the continuation of our project. Verifiable segments were selected by removing all duplicate or similar segments that did not pass the threshold of 30 by the Levenshtein distance [52]. Informative and Non-verifiable segments were selected uniformly at random. The same procedure was applied (the Levenshtein distance with a threshold of 30) to verify that there are no duplicate or similar segments of the Informative and Non-verifiable classes.

Experimental Settings

Model Parameters

We evaluate our methods via the following parameters. In turn, we fix $n \in [1..10]$. In the vector similarity model, we use two types of term frequencies: TF and TF-IDF. In the ML method, we use two types of vectors: those based on TF and those based on TF-IDF. In addition, we experiment with 4 classifiers: decision trees [71], random forests [17], SVM [22], and stochastic gradient descent (SGD) [77]. We report only the best parameter settings with respect to each baseline method.

In the CamemBERT implementation, we use the parameters recommended in [44]: the learning rate 2×10^{-5} and $\epsilon = 10 \times 10^{-8}$. We also fix the number of epochs to 10 and the batch size to 16. We repeat each experiment 10 times to address the model instability problem [95] and report the best and average results, which we compute via the mean function.

For data augmentation, we test each of the methods presented in "Text Augmentation" with $k \in [1..5]$. As before, we report only the results corresponding to the best performing values of k. To implement Semantic-Driven Method and Combined Approach, we test 3 different dictionaries as sources of synonyms: WordNet [60], Agrovoc [19], and DES [61]. For each of the dictionaries, we fix $s = 5$ on the basis of our preliminary experiments (Table 1).

Finally, in the AMR-based approach, we use the value $k = 3$ to determine if a segment is large enough, which was selected experimentally. To evaluate each segment in our selection described in "Data", we generate 1 summary via the approach that is based on AMR graphs and 3 summaries via LLM. We use ChatGPT of version *3.5-turbo-0301* as an instance of an LLM. Compared with ChatGPT, the AMR approach is deterministic; for this reason, we generate only 1 summary using AMR graphs and several with ChatGPT.

Evaluation Protocol

Segment Classification

We do not require any specific validation framework for trigger words since there is no training phase in the method. To evaluate the vector-similarity model, we use *leave-one-out* cross validation (CV) implemented as follows: each of the segments is used for testing, while all the others are used for training. For the ML method, which is based on frequency vectors, we use a more common validation framework. To evaluate this method, we implement *10-fold* CV, with each fold containing 10% of all segments. The model is trained on 9 folds, while the last fold is used for validation. The process is repeated 10 times until each fold is used as a test set. Finally, to evaluate CamemBERT, we use *stratified* sampling implemented as follows. The data are split into 2 parts: 80% of the segments are used for learning, whereas the other 20% are used for validation. The split is performed such that the proportion of positive and negative examples for each type of classification remains the same (Table 2).

Rule formulation

We evaluate text portions for expert rule construction in two ways: automatically and manually. For automatic evaluation, we compute the semantic textual similarity between each segment and each text portion via BERT [44]. We employ the cosine similarity as the similarity measure and use BERT to compute embedding vectors of the text portions and the original segments. Since ground truth text portions are very costly to obtain, we do not use standard metrics such as ROUGE [55]. Instead, we developed our own metric. We then rank all the portions of each segment via the following algorithm. If a similarity score is less than 0.5, the portion receives a rank of 0; otherwise, it receives a rank starting from 1, where 1 corresponds to the most similar. For example, if 4 portions receive similarity scores *0.49*, *0.65*, *0.48*, *0.72*, they will be ranked as *0*, *2*, *0*, *1* for 1st, 2nd, 3rd, and 4th, respectively.

In the manual evaluation, we ask a group of volunteers to rank the generated portions in 2 steps, following a similar procedure to that of the automatic evaluation. In the 1st step, we ask them to assign a score of 0 to all portions that do not contain elements for expert rule construction or are not pertinent. In the 2nd step, we ask them to rank all the portions that did not receive a score of 0 in the previous step by assigning a rank starting from 1, where 1 corresponds to the best-quality portion. Similar to the automatic evaluation, there cannot be 2 portions with the same rank. For example, if two portions passed the 1st step, one of them must receive a rank of 1, and the other 2. In this way, we evaluate each segment by 2 different people to minimize the risk of error. In total, we asked 9 people to participate in this experiment, each of whom annotated approximately 30 segments (2 people annotated 33 each, 5 annotated 30 each, and 2 annotated 20 each). To assess the degree of agreement between different annotators, we computed the kappa coefficient [20]. We obtained 0.41 as the average value, which corresponds to fair agreement [49].

Quality Measures

Segment Classification In the 1st classification, we use the precision, recall, and F$_1$ score to assess the quality of our prediction:

$$Prec = \frac{TP}{TP + FP},$$

$$Rec = \frac{TP}{TP + FN},$$

$$F_1 = 2 \cdot \frac{Prec \cdot Rec}{Prec + Rec},$$

where *TP* represents true positive examples, *FP* represents false positive, and *FN* represents false negative. In the 2nd and 3rd classifications, we compute each of those values for both classes. To determine which of the results are the best, we use *weighted* accuracy. For that, we assign a classification cost of 1 to examples of an overrepresented class (Strict and Non-verifiable) and cost *new_cost* to examples of an underrepresented class (Informative and Verifiable), derived by:

$$new\_cost = \frac{|D|}{2 \cdot |N|},$$

where |*D*| is the number of examples of both classes for the classification task and |*N*| is the number of examples of an underrepresented class. We then perform an evaluation on the basis of the costs defined: the FNs and TNs receive a score of *new_cost* for every example of an underrepresented class w.r.t. its real class, whereas the FPs and TPs receive a score of 1 for positives. We benefit from using weighted accuracy twice: to determine the best performing epoch in each experiment and to select the best result among 10 runs.

Rule Formulation

To assess the quality of the text portions generated by different approaches, we use the mean reciprocal rank (MRR) [85], precision at *k* (P@k), and recall at *k* (R@k) with $k \in [1, 4]$. The MRR calculates the average of the reciprocal ranks at which the first relevant text portion is retrieved via a given approach. We define MRR as:

$$MRR = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{rank_i},$$

where *Q* refers to the set of segments used for evaluation and where $rank_i$ represents the rank position of the first relevant portion of the text generated by a given approach for the *i*-th segment. If the approach does not produce a relevant portion for the *i*-th segment, $\frac{1}{rank_i}$ is set to 0 [85]. For example, if an approach is evaluated on 4 segments and for segment 1, it is ranked 1st, for segment 2, it is ranked 2nd, for segment 3, it is ranked 4th, and for segment 4, there is no relevant result, then the MRR of this approach is $\frac{1}{4}(1 + \frac{1}{2} + \frac{1}{4} + 0) = 0.4375$. We compute the MMR for each approach and each user and report an average between users. We define P@k and R@k as:

$$P@k = \frac{|\{relevant@k\} \cap \{generated\}|}{|\{generated\}|},$$

$$R@k = \frac{|\{relevant@k\} \cap \{generated\}|}{|\{relevant\}|},$$

Table 3 Evaluation results with different methods on 1st classification task

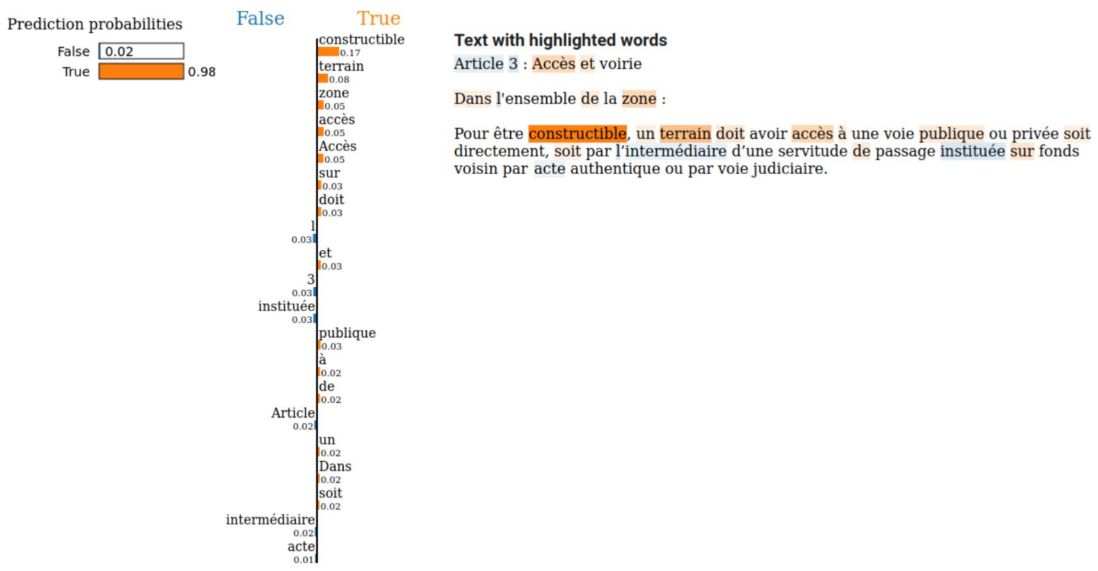
Method	Results		
	Precision	Recall	F_1
Trigger words (n = 10)	0.35	0.98	0.51
Vector similarity (TF-IDF)	0.66	0.96	0.78
ML approach (TF, SVM)	0.87	0.81	0.83
CamemBERT	0.86	0.96	0.91

where *relevant* corresponds to the portions of text that are relevant to a given approach, *generated* corresponds to the portions generated by a given approach, and *k* corresponds to the lower bound of rank *k*. P@k shows the fraction of generated text portions that are relevant with respect to a given approach and rank *k*. In this way, P@4 is identical to the percentage of text portions that passed the 1st step of the evaluation. R@k gives a measure of how many of the relevant text portions passed the lower bound of rank *k* out of all the relevant items. If there is at least one portion generated by the approach, its R@4 is always 100%, and we thus do not output this measure. Similar to MRR, we compute P@k and R@k for each approach and user and then find an average between users. We use MRR, P@k, and R@k for both the automatic and manual evaluation.

Implementation Details

To perform segment classification, we implemented baseline methods and state-of-the-art and text augmentation approaches in Python (see footnote 5). We used the NLTK library [14] to implement tokenization, remove stop words, and find stemmed representations of segments for the baseline methods. In addition, we used the Stanford POS-Tagger [82] for the French language to determine part-of-speech in POS-Driven Method and Combined Approach for data augmentation. We used the scikit-learn library [66] to implement the decision trees, random forests, SVM, and SGD classifiers. We also used this library to implement precision, recall, F_1 score, and weighted versions of accuracy. Finally, we used the *CamembertForSequenceClassification* model from the HuggingFace library [90] as the CamemBERT implementation.

For rule formulation, we used [40] as an implementation of the keyword extraction module, which is based on [38]. In addition, we used *amrlib* [1] for the AMR parsing and AMR-to-text modules and [78] for text alignment. We also used [34] to manipulate AMR graphs in the Penman format. For the automatic translation system, we employed [29] to convert input segments from French to English and then generated portions from English back to French. Our approach is not constrained to a specific translation system,



Text with highlighted words

Article 3 : Accès et voirie

Dans l'ensemble de la zone :

Pour être constructible, un terrain doit avoir accès à une voie publique ou privée soit directement, soit par l'intermédiaire d'une servitude de passage instituée sur fonds voisin par acte authentique ou par voie judiciaire.

Fig. 13 An example of a segment analysis using Lime, from left to right: the classification results with probability scores, the features which led to these results, original segment with negative and positive features highlighted in blue and orange consequently (with some of the features coming from the expert nomenclature concepts: *terrain*, *zone* and *accès*)

Table 4 Evaluation results with different methods on 2nd classification task

Method	Results						
	Class Strict			Class informative			Accuracy*
	Precision	Recall	F_1	Precision	Recall	F_1	
Trigger words (n = 1)	0.92	0.56	0.70	0.18	0.67	0.29	0.60
Vector similarity (TF-IDF)	0.99	0.92	0.95	0.62	0.92	0.74	0.92
ML approach (TF-IDF, SGD)	0.97	0.99	0.98	0.93	0.81	0.85	0.93
CamemBERT	0.99	1.00	1.00	1.00	0.94	0.97	0.98

*corresponds to the weighted version

and we selected this translator, as it is widely used in the literature and has demonstrated good results [3, 23, 69, 87]. Finally, we use [66] to compute the kappa coefficient, and we implement (see footnote 5) MRR, P@k, and R@k ourselves.

Results

Segment Classification

1st Classification The summary of results[8] is presented in Table 3. Trigger words can discover almost all pertinent segments (recall 98%). However, the low precision of this method results in an average performance equal to that of random guessing (F_1 score 51%). Compared with trigger words, the vector similarity method results in identical recall but improves precision by twofold. The ML approach further improves precision, but its recall decreases compared with that of the previous method. Nevertheless, it slightly outperforms the latter. Finally, CamemBERT smooths out this difference by providing precision similar to that of the ML approach and recall identical to that of the vector similarity method. The overall result of CamemBERT (F_1 score 91%) demonstrates the very good performance of the method. We thus use the classifier trained by this method in our framework.

To verify the quality of the resulting classifier, we perform a detailed study of the segments classified by CamemBERT. For each example in the test data classified as TP (128 segments out of 135 positive examples in the test data), we collect all the features that led to this result via *Lime* [76] (Fig. 13). As a result, 35.06% of all positive features are expert nomenclature concepts. This is a very good result, showing that CamemBERT can capture thematic concepts

[8] For more detailed results w.r.t. each method and each parameter, please refer to our annex provided with our code: https://github.com/koptelovmax/AIR-FUD/blob/main/classification_results.ods.

and use them as indicators of positive examples. Another result of this analysis is that 10.0% of all the top 1 distinct features are trigger words. Despite the relatively low percentage, this is also a good result given that the features should not only consist of nomenclature concepts.

2nd Classification A summary of the results (see footnote 8) is shown in Table 4. Trigger words provide satisfactory performance for the class Strict (F_1 score 70%) and quite a low result for the Informative class. This can be explained by the fact that trigger words are included in both classes, which makes this method ineffective. The vector similarity method improves the results for both classes; however, the class Strict outperforms the Informative class due to high imbalance of classes (Table 2). The ML approach compensates for this shortcoming by improving the results of the underrepresented class. The overall performance of the resulting classifier (weighted accuracy 93%) is sufficient for its choice for the framework. The results of CamemBERT, with a 100% F_1 score on class Strict and 100% precision on class Informative, may indicate that the model does not generalize well. To avoid potential overfitting, we have decided not to employ CamemBERT in the final version of the framework for this task.

3rd Classification

A summary of the results (see footnote 8) is presented in Table 5. The trigger words perform similarly to the 2nd classification task, with the only difference being that the class Verifiable has worse results than does the class Non-verifiable. The two next methods, vector similarity and the ML approach, improve the results of the trigger words, maintaining the trend of better performance of the Non-verifiable class. The latter can be explained by the fact that the Non-verifiable class is better represented in the data than the Verifiable class is (Table 2). CamemBERT slightly improves the situation by minimizing this difference to 13% (F_1 class Verifiable 82% vs. F_1 class Non-verifiable 94%). We attempt to minimize this difference further by performing data augmentation of the underrepresented class ("Text Augmentation"). To achieve that, we continue applying our methods defined in "Text Augmentation".

Text Augmentation

Most of the best results for data augmentation (see footnote 8) correspond to the settings with the positive class (i.e., the class Verifiable in our case) augmented to 50% or more (Table 10). This correlation can be observed mostly in POS-Driven Method and Semantic-Driven Method (class Verifiable max F_1 score 86–87%). Nevertheless, Combined Approach allows the data to be kept unbalanced and offers the same performance (or even better, taking into account weighted accuracy). Surprisingly, Semantic-Driven Method and Combined Approach can improve the results. According to our initial hypothesis, all pertinent segments should include nomenclature concepts. Following the results on augmented data, we can conclude that the class Verifiable contains more nomenclature concepts than does the class Non-verifiable. Ultimately, the best result in our experiments (see footnote 8) corresponds to Combined Approach with the DES dictionary as a source of synonyms (Table 5). Compared with the setting with nouns, the setting in which adjectives and adverbs are replaced requires fewer cycles of data augmentation ($k = 1$), which requires repeating the augmentation process twice ($k = 2$). Using this method, we

Table 5 Evaluation results with different methods on 3rd classification task

Method	Results						
	Class verifiable			Class non-verifiable			Accuracy*
	Precision	Recall	F_1	Precision	Recall	F_1	
Trigger words (n = 1)	0.31	0.70	0.43	0.83	0.49	0.62	0.57
Vector similarity (TF-IDF)	0.53	0.97	0.68	0.98	0.71	0.83	0.81
ML approach (TF, Decision Trees)	0.66	0.78	0.71	0.94	0.89	0.91	0.85
CamemBERT	0.78	0.86	0.82	0.95	0.92	0.94	0.90
CamemBERT+data augmentation	0.82	0.93	0.87	0.98	0.93	0.95	0.93

*corresponds to the weighted version

Table 6 Automatic evaluation of the AMR-based approach and several runs of ChatGPT

Approach	Quality measures							
	MRR	P@1	P@2	P@3	P@4	R@1	R@2	R@3
AMR-based	0.43	0.23	0.38	0.51	0.82	0.29	0.47	0.62
ChatGPT (run 1)	0.54	0.30	0.54	0.80	0.94	0.32	0.58	0.86
ChatGPT (run 2)	0.50	0.21	0.52	0.84	0.97	0.22	0.54	0.86
ChatGPT (run 3)	0.52	0.26	0.55	0.81	0.95	0.27	0.58	0.86

can improve the performance of CamemBERT for the Verifiable class by 6% (max F_1 score from 82% to 87%). We thus use the resulting method in the AIR-FUD+ framework for processing new documents.

Rule formulation The results of the automatic evaluation are presented in Table 6.

As seen from the results, ChatGPT generally performs better than the approach based on AMR graphs, except for a few cases where the AMR-based approach outperforms its competitor. These exceptions include P@1 of ChatGPT (run 2) and R@1 of ChatGPT (runs 2, 3). This indicates that the automatic evaluator favours text portions generated by the AMR-based approach over its competitors and ranks them in the first place. Overall, this is a positive result, demonstrating that an approach based on AMR graphs remains competitive with the state-of-the-art methods. Next, the results of ChatGPT vary across runs. This is not surprising given the probabilistic nature of LLMs. The text portions generated by ChatGPT exhibit variability across different runs, even if the query remains the same. Unfortunately, this behaviour is not stable: MRR, P@1, and R@1 show the best performance in the first run; P@3 and P@4 in the second; and P@2, R@2, and R@3 in the last run. The results do not show a consistent trend of improvement or deterioration across different runs. Our experiments did not detect such a trend.

To verify whether there is a variance in performance across different segment types, we recomputed the quality measures with respect to different classes (Table 7). Compared with all classes, the performance gap between the AMR-based approach and ChatGPT is lower for the Verifiable and Non-verifiable classes and greater for the Informative class. Moreover, the AMR approach provides a better R@1 than ChatGPT does for the Informative and Non-verifiable classes. This is not the only difference we noticed. Interestingly, compared with the all-class setting, the AMR-based approach performs better for the Verifiable (MRR, P@2, P@3, P@4, R@2, R@3) and Non-verifiable classes (P@1, P@2, P@3, R@1, R@2, R@3), whereas the performance of ChatGPT for these classes decreases (MRR, P@2, P@3, P@4, R@2, R@3). Additionally, the Informative class demonstrates the opposite behaviour: the performance of the AMR approach decreases, but ChatGPT improves in performance compared with the all-class setting (MRR, P@2, P@3, P@4, R@2, R@3). This can be explained by the fact that segments of the Informative class are usually longer and contain fewer important keywords, making it more difficult for the AMR approach to capture semantics, whereas ChatGPT usually performs better on general texts than on specific tasks [46]. Segments of the Verifiable and Non-verifiable classes are more concise, making our approach easier and more challenging for ChatGPT.

An automatic evaluation demonstrated that ChatGPT performs globally better than the AMR-based approach does. This study also demonstrated the potential of the

Table 7 Automatic evaluation of the AMR-based approach and ChatGPT (averaged among 3 runs) based on different classes of segments

Approach	Quality measures							
	MRR	P@1	P@2	P@3	P@4	R@1	R@2	R@3
	All classes							
AMR-based	0.43	0.23	0.38	0.51	0.82	0.29	0.47	0.62
ChatGPT (mean)	0.52	0.26	0.54	0.82	0.95	0.27	0.56	0.86
	Verifiable							
AMR-based	0.45	0.23	0.44	0.58	0.86	0.27	0.51	0.67
ChatGPT (mean)	0.51	0.26	0.52	0.79	0.93	0.27	0.55	0.85
	Informative							
AMR-based	0.38	0.22	0.25	0.34	0.78	0.28	0.32	0.44
ChatGPT (mean)	0.54	0.26	0.58	0.86	0.97	0.27	0.60	0.89
	Non-verifiable							
AMR-based	0.43	0.25	0.41	0.53	0.78	0.32	0.52	0.68
ChatGPT (mean)	0.52	0.25	0.53	0.82	0.97	0.26	0.55	0.85

Table 8 Manual evaluation of the AMR-based approach and several runs of ChatGPT

Approach	Quality measures							
	MRR	P@1	P@2	P@3	P@4	R@1	R@2	R@3
AMR-based	0.10	0.06	0.11	0.13	0.17	0.34	0.66	0.77
ChatGPT (run 1)	0.50	0.30	0.60	0.75	0.76	0.40	0.79	0.99
ChatGPT (run 2)	0.52	0.34	0.57	0.77	0.77	0.44	0.74	0.99
ChatGPT (run 3)	0.47	0.25	0.54	0.75	0.76	0.33	0.71	0.98

Table 9 Manual evaluation of the AMR-based approach and ChatGPT (averaged among 3 runs) based on different classes of segments

Approach	Quality measures							
	MRR	P@1	P@2	P@3	P@4	R@1	R@2	R@3
	All classes							
AMR-based	0.10	0.06	0.11	0.13	0.17	0.34	0.66	0.77
ChatGPT (mean)	0.50	0.30	0.57	0.76	0.76	0.39	0.75	0.99
	Verifiable							
AMR-based	0.06	0.02	0.05	0.09	0.14	0.18	0.39	0.71
ChatGPT (mean)	0.54	0.32	0.61	0.82	0.83	0.38	0.73	0.99
	Informative							
AMR-based	0.13	0.08	0.17	0.17	0.20	0.42	0.89	0.89
ChatGPT (mean)	0.50	0.30	0.58	0.74	0.76	0.40	0.76	0.99
	Non-verifiable							
AMR-based	0.15	0.11	0.17	0.17	0.20	0.51	0.80	0.80
ChatGPT (mean)	0.42	0.27	0.47	0.61	0.62	0.43	0.76	0.98

Table 10 Results of text augmentation for 3rd classification and their comparison with performance on original data

Method	Size of training data			Results on test data				
				Class Verifiable		Class Non-verifiable		Accuracy*
	Number of segments		% positive	F_1 score		F_1 score		
	Total	Positive		avg	max	avg	max	
Original data	470	118	25	0.80	0.82	0.93	0.94	0.90
POS-Driven Method (adj+adv, k = 1)	573	221	39	0.82	0.86	0.94	0.96	0.92
POS-Driven Method (nouns, k = 2)	706	354	50	0.81	0.84	0.94	0.95	0.91
POS-Driven Method (verbs, k = 3)	818	466	57	0.83	0.86	0.94	0.95	0.92
Semantic-Driven Method (WordNet, k = 1)	588	236	40	0.82	0.85	0.93	0.95	0.92
Semantic-Driven Method (DES, k = 2)	706	354	50	0.82	0.87	0.94	0.95	0.93
Semantic-Driven Method (Agrovoc, k = 3)	824	472	57	0.82	0.86	0.94	0.95	0.92
Combined Approach (DES, adj+adv, k = 1)	540	188	35	0.82	0.87	0.94	0.95	0.93
Combined Approach (DES, nouns, k = 2)	628	276	44	0.82	0.87	0.94	0.96	0.93
Combined Approach (DES, verbs, k = 4)	786	434	55	0.83	0.85	0.94	0.95	0.92

*corresponds to the weighted version

AMR-based approach and highlighted some of the limitations of ChatGPT. However, our generated portions of text are intended for use in expert rule construction; thus, manual verification should be performed. We continue to evaluate the generated portions manually to confirm or disprove our findings. The results of this manual evaluation are presented in Table 8.

According to the new results, only R@1, R@2, and R@3 demonstrate a similar performance of the AMR-based approach to ChatGPT. Other measures show a greater performance gap between the two approaches than before. The number of times the AMR-based approach did not pass the first step is much lower than that in the automatic evaluation, as indicated by the P@4 measure (only 17% of text portions passed the first human evaluation, compared with 82% in the automatic evaluation). This could occur if a majority of the generated portions are not pertinent (e.g., due to errors in automatic translation) or incomplete (e.g., lacking important information from the original segment). The latter is less likely considering that the automatic evaluation did not detect it. In contrast, the former is quite possible since a small change in vocabulary may not be effectively detected by computing semantic similarity with the original segment, but it is much easier for humans to recognize. Such a low result in the first step directly impacts the overall performance of the approach. The new values of R@1, R@2, and R@3 indicate that humans still prefer portions of text generated by ChatGPT rather than by the AMR-based approach. Finally, the results of ChatGPT still vary from run to run, but a common trend is now visible. The best result corresponds to the second run according to most quality measures (MRR, P@1, P@3, P@4, R@1), and the last run is the worst according to all measures. This phenomenon can be explained by the fact that querying ChatGPT is performed during one session, and the system can improve the quality of the output based on previous iterations up to a certain level. It was surprising that we were not able to detect this phenomenon with automatic evaluation. Evaluation by humans is more accurate for such tasks, and it demonstrated almost perfect agreement.

With respect to the results of different classes, the common trend slightly changed: the Informative class was replaced by Verifiable, which has the highest performance gap in the manual evaluation (Table 9).

Detailed analysis shows that the results of the AMR approach improve for the Informative and Non-verifiable classes even though they become worse for the Verifiable class.

ChatGPT behaves the opposite way: it improves on Verifiable (except for R@1, R@2), performs identically on Informative and drops on Non-verifiable (except for R@1, R@2). This is an interesting outcome demonstrating that manual evaluation can present results from another angle. In summary, there is a significant gap in the results between AMR and ChatGPT, with the latter leading. For some measures, this gap can reach 10 fold. This is because AMR cannot successfully pass the first step of human evaluation. On the other hand, R@1 and R@2 of the Non-verifiable class indicate that among the portions that passed the first step, humans prefer those generated by the AMR. This makes the AMR-based approach not useless and relegates it to some specific cases.

Conclusion

In this work, we presented the AIR-FUD+ framework for (semi)automatic identification and formulation of rules in urban planning documents in the French language. The framework is used within the scope of the Hérelles project, which aims to improve the management of land artificialization. The first step in the framework involves extracting constraints from urban planning documents by identifying segments that contain rules and determining their types. The next step is to reformulate these segments into concise text portions that include all the necessary elements for formulating the constraints.

We showed experimentally via a manually annotated corpus that AIR-FUD+ can correctly identify the rules within the hierarchy of classes. We proposed a cascade approach for this purpose, and we demonstrated a good performance. In addition, we developed several text augmentation methods based on text mining and a language model that can solve the data imbalance problem and improve the overall results for the latest classification task.

We demonstrated that it is indeed possible to use identified segments to generate portions of text containing all necessary elements for expert rule construction. We propose two solutions: a classical approach based on AMR graphs and a state-of-the-art solution using an LLM (ChatGPT). The results demonstrated a performance gap between the two approaches, with the LLM approach leading. In our project, we will continue using the LLM, as it better meets our performance requirements. However, in some sensitive applications where reproducibility and interpretability are more important, we suggest exploring AMR-based approaches. Our findings indicate that, in specific cases, the AMR-based approach has the potential to be competitive with the LLM.

To date, we have only conducted evaluations on both parts—segment classification and rule formulation—using the corpus that we constructed. It would be interesting to apply our framework to new unseen documents. Depending on the results obtained, adjustments to the choice of classifiers and the data augmentation method could be made for each classification task. Finally, it would be interesting to continue the work on enhancing the reproducibility of an LLM and improving the results of the AMR-based approach. A promising direction would be to explore the combination of these two approaches to increase the reproducibility of the ChatGPT results. This could also improve its effectiveness, especially in cases where it may not perform optimally.

Acknowledgements We thank Rodrique Kafando and Rémy Decoupes for providing valuable consultations on the methodological part. We also thank Baptiste Lafabrègue, Nadia Guiffant, Mariya Borovikova, Bahdja Boudoua, Sarah Valentin, Léa Tardieu and Emmanuel Capliez for their assistance.

Funding Open access funding provided by CIRAD. This work is partially funded by the French National Research Agency (ANR)—HERELLES project (Contract No. ANR-20-CE23-0022).

Data availability The data used in this article for the experiments are publicly available: https://doi.org/10.57745/DWYGMB.

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

1. A python library that makes AMR parsing, generation and visualization simple. 2022. https://amrlib.readthedocs.io.
2. Afzal S, Maqsood M, Nazir F, et al. A data augmentation-based framework to handle class imbalance problem for Alzheimer's stage detection. IEEE Access. 2019;7:115528–39.
3. Akbari M, Karimi AH, Saeedi T, et al. A Persian benchmark for joint intent detection and slot filling. 2023. arXiv preprint arXiv:2303.00408
4. Alirezaie M, Kiselev A, Längkvist M, et al. An ontology-based reasoning framework for querying satellite images for disaster monitoring. Sensors. 2017;17(11):2545.
5. Allahyari M, Pouriyeh S, Assefi M, et al. A brief survey of text mining: classification, clustering and extraction techniques. 2017. arXiv preprint arXiv:1707.02919
6. Almeida F, Xexéo G. Word embeddings: a survey. 2019. arXiv preprint arXiv:1901.09069
7. Anchiêta RT, Pardo TA.S A rule-based AMR parser for Portuguese. In: Advances in artificial intelligence-IBERAMIA 2018: 16th Ibero-American Conference on AI, Trujillo, Peru, 2018; November 13–16, 2018, proceedings 16. Springer. pp 341–353
8. Anwar MW, Ahsan I, Azam F, et al. A natural language processing (NLP) framework for embedded systems to automatically extract verification aspects from textual design requirements. In: Proceedings of the 2020 12th international conference on computer and automation engineering. 2020. pp. 7–12.
9. Argüeso D, Evans JP, Fita L, et al. Temperature response to future urbanization and climate change. Clim Dyn. 2014;42:2183–99.
10. Artificialisation des sols. 2022. https://www.ecologie.gouv.fr/artificialisation-des-sols. Accessed 13 Mar 2023
11. Artificialised land and artificialisation processes: determinants, impacts and levers for action. 2017. https://www.inrae.fr/en/news/artificialised-land-and-artificialisation-processes. Accessed 13 March 2023
12. Ba CT, Choquet C, Interdonato R, et al. Explaining food security warning signals with YouTube transcriptions and local news articles. In: Proceedings of the 2022 ACM conference on information technology for social good. 2022. pp. 315–322.
13. Banarescu L, Bonial C, Cai S, et al. Abstract meaning representation for sembanking. In: Proceedings of the 7th linguistic annotation workshop and interoperability with discourse. 2013. pp. 178–186.
14. Bird S, Klein E, Loper E. Natural language processing with Python: analyzing text with the natural language toolkit. Sebastopol: O'Reilly Media Inc; 2009.
15. Boori MS, Netzband M, Voženílek V, et al. Urban growth in last three decades in Kuala Lumpur, Malaysia. In: 2015 Joint urban remote sensing event (JURSE). IEEE;2015. pp. 1–4.
16. Bougouin A, Barreaux S, Romary L, et al: TermiTH-Eval: a French standard-based resource for keyphrase extraction evaluation. In: LREC-language resources and evaluation conference. 2016.
17. Breiman L. Random forests. Mach Learn. 2001;45:5–32.
18. Brown PF, Cocke J, Della Pietra SA, et al. A statistical approach to machine translation. Comput Linguist. 1990;16(2):79–85.
19. Caracciolo C, Stellato A, Morshed A, et al. The AGROVOC linked dataset. Semant Web. 2013;4(3):341–8.
20. Cohen J. A coefficient of agreement for nominal scales. Educ Psychol Meas. 1960;20(1):37–46.
21. Cornuéjols A, Wemmert C, Gançarski P, et al. Collaborative clustering: why, when, what and how. Inf Fusion. 2018;39:81–95.
22. Cortes C, Vapnik V. Support-vector networks. Mach Learn. 1995;20:273–97.
23. Costa-jussà M, Basta C, Domingo O, et al. OccGen: selection of real-world multilingual parallel data balanced in gender within occupations. Adv Neural Inf Process Syst. 2022;35:1445–57.
24. Damodaran P. Parrot: paraphrase generation for NLU. 2021. https://github.com/PrithivirajDamodaran/Parrot_Paraphraser.
25. Daniell J, Wenzel F, Schaefer A. The economic costs of natural disasters globally from 1900-2015: historical and normalised floods, storms, earthquakes, volcanoes, bushfires, drought and other disasters. In: EGU general assembly conference abstracts. 2016. pp. EPSC2016–1899
26. Davidson I, Ravi S: Agglomerative hierarchical clustering with constraints: theoretical and empirical results. In: Knowledge discovery in databases: PKDD 2005: 9th European conference on principles and practice of knowledge discovery in databases, Porto, Portugal, October 3–7, 2005. Proceedings 9. Springer;2005. pp. 59–70.
27. Dohare S, Karnick H: Text summarization using abstract meaning representation. https://api.semanticscholar.org/CorpusID:260498172.
28. Dohare S, Gupta V, Karnick H. Unsupervised semantic abstractive summarization. In: Proceedings of ACL 2018, student research workshop. 2018. pp. 74–83.
29. EasyNMT—Easy to use, state-of-the-art neural machine translation. 2021. https://github.com/UKPLab/EasyNMT.
30. El-Kassas WS, Salama CR, Rafea AA, et al. Automatic text summarization: a comprehensive survey. Expert Syst Appl. 2021;165:113679.
31. Eskenazi S, Gomez-Krämer P, Ogier JM. A comprehensive survey of mostly textual document segmentation algorithms since 2008. Pattern Recognit. 2017;64:1–14.
32. Espejel JL. Automatic abstractive summarization of long medical texts with multi-encoders transformer and general-domain summary evaluation with wikiSERA. PhD thesis, Université Paris-Nord-Paris XIII 2021.
33. Espigares T, Moreno-de las Heras M, Nicolau JM. Performance of vegetation in reclaimed slopes affected by soil erosion. Restor Ecol. 2011;19(1):35–44.
34. Goodman MW: Penman: an open-source library and tool for AMR graphs. In: Proceedings of the 58th annual meeting of the association for computational linguistics: system demonstrations. 2020. pp. 312–319.
35. Guo Y, Rennard V, Xypolopoulos C, et al. BERTweetFR: domain adaptation of pre-trained language models for French tweets. In: Proceedings of the seventh workshop on noisy user-generated text (W-NUT 2021). 2021. pp. 445–450.
36. Holveck M. Nomenclature HÃrelles. 2023. https://doi.org/10.57745/OXACT8.
37. Holveck M, Koptelov M, Roche M, et al. 2023. Segments textuels Hérelles. https://doi.org/10.57745/DWYGMB.
38. Honnibal M, Montani I. spaCy 2: natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. 2017. https://github.com/explosion/spaCy.
39. Jeblick K, Schachtner B, Dexl J, et al. ChatGPT makes medicine easy to swallow: an exploratory case study on simplified radiology reports. Eur Radiol. 2023;34:2817–25.
40. Kafando R. Hérelles rules visualization. 2023. https://rdius-herelles-ner-app-smoynm.streamlit.app.
41. Kafando R, Decoupes R, Teisseire M, et al. Constitution de corpus thématique: Pour un meilleur suivi du territoire de la métropole de montpellier méditerranée. In: SAGEO'21 16ème Conférence

Internationale de la Géomatique, de l'Analyse Spatiale et des Sciences de l'Information Géographique. 2021.
42. Kasper RT A flexible interface for linking applications to penman's sentence generator. In: Speech and natural language: proceedings of a workshop held at Philadelphia, Pennsylvania, February 21–23, 1989. 1989.
43. Kelodjoue E, Goulian J, Schwab D. Performance of two French BERT models for French language on verbatim transcripts and online posts. In: Proceedings of the 5th international conference on natural language and speech processing (ICNLSP 2022). 2022. pp. 88–94.
44. Kenton JDMWC, Toutanova LK. BERT: pre-training of deep bidirectional transformers for language understanding. In: Proceedings of 2019 annual conference of the North American chapter of the Association for Computational Linguistics: human language technologies. 2019. pp. 4171–4186.
45. Kiziltan Z, Lippi M, Torroni P, et al. Constraint detection in natural language problem descriptions. In: IJCAI, international joint conferences on artificial intelligence. 2016. pp. 744–750.
46. Kocoń J, Cichecki I, Kaszyca O, et al. ChatGPT: jack of all trades, master of none. Inf Fusion. 2023;99: 101861.
47. Koptelov M, Holveck M, Cremilleux B, et al. A manually annotated corpus in French for the study of urbanization and the natural risk prevention. Sci Data. 2023;10(1):818.
48. Laifa A, Gautier L, Cruz C. Impact of textual data augmentation on linguistic pattern extraction to improve the idiomaticity of extractive summaries. In: Big data analytics and knowledge discovery: 23rd international conference, DaWaK 2021, Virtual Event, September 27–30, 2021, proceedings 23. Springer;2021. pp. 143–151.
49. Landis JR, Koch GG. The measurement of observer agreement for categorical data. Biometrics. 1977;33:159–74.
50. Lawrence J, Reed C. Argument mining: a survey. Comput Linguist. 2020;45(4):765–818.
51. Le H, Vial L, Frej J. FlauBERT: unsupervised language model pre-training for French. In: Proceedings of the 12th language resources and evaluation conference. 2020. pp. 2479–2490.
52. Levenshtein VI, et al. Binary codes capable of correcting deletions, insertions, and reversals. In: Soviet Physics Doklady, Soviet Union. 1966. pp. 707–710.
53. Liang M, Niu T. Research on text classification techniques based on improved TF-IDF algorithm and LSTM inputs. Procedia Comput Sci. 2022;208:460–70.
54. Liao K, Lebanoff L, Liu F. Abstract meaning representation for multi-document summarization. In: Proceedings of the 27th international conference on computational linguistics. 2018. pp. 1178–1190.
55. Lin CY. Rouge: a package for automatic evaluation of summaries. In: Text summarization branches out. 2004. pp. 74–81.
56. Liu F, Flanigan J, Thomson S, et al. Toward abstractive summarization using semantic representations. In: Proceedings of the 2015 conference of the North American chapter of the Association for Computational Linguistics. Association for Computational Linguistics;2015. pp. 1077–1086.
57. Luo H, Li L, Zhu H, et al. Land cover extraction from high resolution ZY-3 satellite imagery using ontology-based method. ISPRS Int J Geo-Inf. 2016;5(3):31.
58. Martin L, Muller B, Suárez PJO, et al. CamemBERT: a tasty French language model. 2019. arXiv preprint arXiv:1911.03894.
59. Migueles-Abraira N, Agerri R, de Ilarraza AD. Annotating abstract meaning representations for Spanish. In: Proceedings of the eleventh international conference on language resources and evaluation (LREC 2018). 2018.
60. Miller GA. WordNet: an electronic lexical database. Cambridge: MIT Press; 1998.
61. Morel M, François J. Le Dictionnaire Electronique des Synonymes du CRISCO: un outil de plus en plus interactif. Revue française de linguistique appliquée. 2015;20(1):9–28.
62. Nadeau D, Sekine S. A survey of named entity recognition and classification. Lingvisticae Investigationes. 2007;30(1):3–26.
63. Nayak T, Ng HT. Effective modeling of encoder–decoder architecture for joint entity and relation extraction. In: Proceedings of the AAAI conference on artificial intelligence. 2020. pp. 8528–8535.
64. Neptune N, Mothe J. Automatic annotation of change detection images. Sensors. 2021;21(4):1110.
65. OpenAI GPT-4 Technical Report. 2023. https://doi.org/10.48550/arXiv.2303.08774.
66. Pedregosa F, Varoquaux G, Gramfort A, et al. Scikit-learn: machine learning in Python. J Mach Learn Res. 2011;12:2825–30.
67. Pilault J, Li R, Subramanian S, et al. On extractive and abstractive neural document summarization with transformer language models. In: Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP). 2020. pp. 9308–9319.
68. Pires T, Schlinger E, Garrette D. How multilingual is multilingual BERT? In: Proceedings of the 57th annual meeting of the Association for Computational Linguistics. 2019. pp. 4996–5001.
69. Pretkalnins IJ, Sprogis A, Barzdins G. CLIP augmentation for image search. In: COMPLEXIS. 2022. pp. 71–78.
70. Pu X, Gao M, Wan X. Summarization is (almost) dead. 2023 arXiv preprint arXiv:2309.09558.
71. Quinlan JR. Induction of decision trees. Mach Learn. 1986;1:81–106.
72. Radford A, Kim JW, Hallacy C, et al. Learning transferable visual models from natural language supervision. In: International conference on machine learning. PMLR;2021. pp. 8748–8763.
73. Ramakrishnan C, Patnia A, Hovy E, et al. Layout-aware text extraction from full-text PDF of scientific articles. Source Code Biol Med. 2012;7:1–10.
74. RASA Paraphraser. 2021. https://github.com/RasaHQ/paraphraser.
75. Remaud A. Extraction de contraintes dans des spécifications de validation de données. Extraction et Gestion des Connaissances. 2022. EGC'2022 38.
76. Ribeiro MT, Singh S, Guestrin C. "Why should I trust you?" Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. 2016. pp. 1135–1144.
77. Robbins H, Monro S. A stochastic approximation method. Ann Math Stat. 1951;22:400–7.
78. Sabet MJ, Dufter P, Yvon F, et al. SimAlign: high quality word alignments without parallel training data using static and contextualized embeddings. EMNLP. 2020;2020:1627–43.
79. Salton G, Buckley C. Term-weighting approaches in automatic text retrieval. Inf Process Manag. 1988;24(5):513–23.
80. Shaib C, Li ML, Joseph S. Summarizing, simplifying, and synthesizing medical evidence using GPT-3 (with varying success). In: Proceedings of the 61st annual meeting of the Association for Computational Linguistics (volume 2: short papers). Association for Computational Linguistics;2023. pp. 1387–1407.
81. Shen C, Cheng L, Nguyen XP, et al. Large language models are not yet human-level evaluators for abstractive summarization. Find Assoc Comput Linguist EMNLP. 2023;2023:4215–33.
82. Toutanova K, Klein D, Manning CD. Feature-rich part-of-speech tagging with a cyclic dependency network. In: Proceedings of the 2003 human language technology conference of the North American chapter of the Association for Computational Linguistics. 2003. pp. 252–259.
83. Uhrig S, Garcia Y, Opitz J, et al. Translate, then parse! A strong baseline for cross-lingual AMR parsing. In: Proceedings of the 17th international conference on parsing technologies and the

IWPT 2021 shared task on parsing into enhanced universal dependencies (IWPT 2021). 2021. pp. 58–64.
84. Vanderwende L, Menezes A, Quirk C. An AMR parser for English, French, German, Spanish and Japanese and a new AMR-annotated corpus. In: Proceedings of the 2015 conference of the North American chapter of the Association for Computational Linguistics: demonstrations. 2015. pp. 26–30.
85. Voorhees EM, et al. The Trec-8 question answering track report. In: Trec. 1999. pp. 77–82.
86. Weber C, Hirsch J. Processus de croissance et limites urbaines. Cybergeo Eur. J. Geogr. 2000.
87. Wein S, Schneider N. Accounting for language effect in the evaluation of cross-lingual AMR parsers. In: Proceedings of the 29th international conference on computational linguistics. 2022. pp. 3824–3834.
88. Widyassari AP, Rustad S, Shidik GF, et al. Review of automatic text summarization techniques & methods. J King Saud Univ Comput Inf Sci. 2022;34(4):1029–46.
89. Winter K, Rinderle-Ma S. Detecting constraints and their relations from regulatory documents using NLP techniques. In: On the move to meaningful internet systems. OTM 2018 conferences: confederated international conferences: CoopIS, C &TC, and ODBASE 2018, Valletta, Malta, October 22–26, 2018, proceedings, part I. Springer;2018. pp. 261–278.
90. Wolf T, Debut L, Sanh V, et al. Transformers: state-of-the-art natural language processing. In: Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations. Association for Computational Linguistics;2020. pp. 38–45.
91. Wu C, Wu P, Wang J, et al. Developing a hybrid approach to extract constraints related information for constraint management. Autom Constr. 2021;124: 103563.
92. Xue N, Bojar O, Hajic J, et al. Not an interlingua, but close: comparison of English AMRs to Chinese and Czech. In: LREC, Reykjavik, Iceland. 2014. pp. 1765–72.
93. Yang J, Han SC, Poon J. A survey on extraction of causal relations from natural language text. Knowl Inf Syst. 2022;64(5):1161–86.
94. Yang X, Li Y, Zhang X, et al. Exploring the limits of ChatGPT for query or aspect-based text summarization. In: Proceedings of the 61st annual meeting of the association for computational linguistics—student research workshop. Association for Computational Linguistics. 2023. pp. 1–18.
95. Zhang T, Wu F, Katiyar A, et al. Revisiting few-sample BERT fine-tuning. In: Proceedings of the ninth international conference on learning representations (ICLR). 2020.
96. Zhijiang G. Learning with graphs in natural language processing. PhD thesis, Singapore University of Technology and Design. 2021.