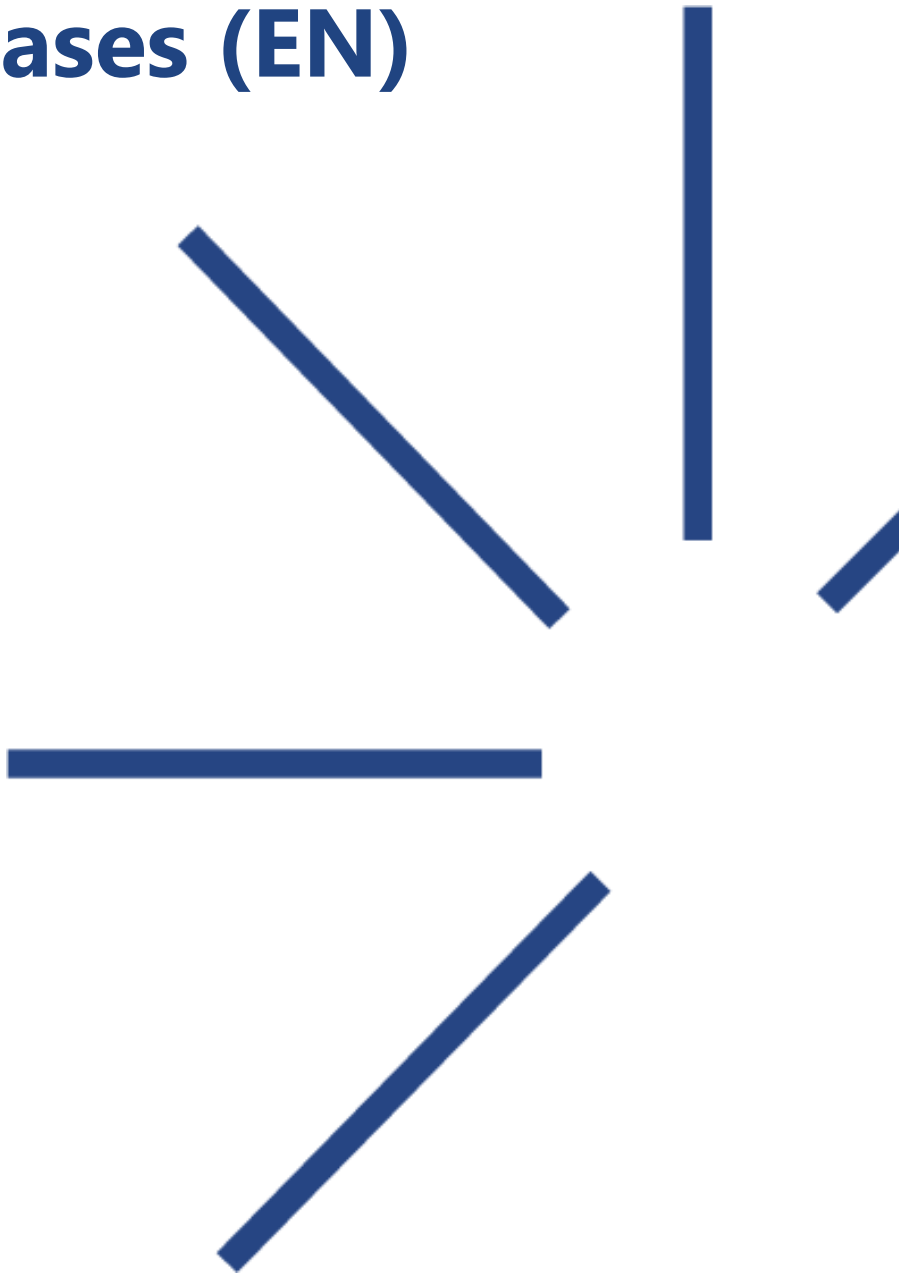
D07 – Method for extraction of process-related phrases (EN)



Project funded by the European Union's Next Generation EU through the Slovak Republic's Recovery and Resilience Plan under project no. 09I05-03-V02-00049.


Financované
Európskou úniou
NextGenerationEU

PLÁN OBNOVY

ÚRAD PODPREDSEDU VLÁDY
SLOVENSKEJ REPUBLIKY
PRE PLÁN OBNOVY
A ZNALOSTNÚ EKONOMIKU

VAIA
VÝSKUMNÁ
A INOVAČNÁ
AUTORITA

VÝSKUMNÁ
AGENTÚRA

INTRODUCTION

The project **"Automation of Legal Text Analysis based on Machine** Learning" (hereinafter "ALTAML") represents a key initiative aimed at integrating innovative approaches in the field of data processing and subsequent analysis, with a focus on the field of law, which also includes information and communication technology law. The aim of this project is to develop and validate effective methods for the automated analysis of legal texts using machine learning techniques. This involves, in particular, the development of tools that facilitate the processing and analysis of large volumes of data in the form of various legal documents, as well as the extraction of relevant information (attributes) from such documents, including the identification of key terms, references to legislation and other attributes.

The aim of the ALTAML project is therefore to contribute to a more efficient approach to legal information and to the acceleration of legal processes, thereby ensuring a higher degree of legal certainty and improving the accessibility of legal texts, the results of their analysis and relevant legal information for both professionals and the general public.

As part of work package KPB2, the output Method for extraction of process-related phrases was developed. The proposed system is a hybrid extractor of procedural legal events, combining regular expressions, semantic vector representations and a large language model. The aim of the system is the automatic identification of procedural acts and outcomes of proceedings from Slovak court decisions, such as the dismissal of an appeal, the quashing of a decision, the discontinuation of proceedings, or the referral of a case back for further proceedings.

Extraction is limited to a predefined ontology of process categories, thereby ensuring the consistency of outputs and minimising the risk of misinterpreting the legal text. The combination of a rule-based and semantic approach enables the effective processing of linguistically variable formulations typical of Slovak court decisions.

The system employs a multi-layered process for identifying procedural events. In the first stage, regular expressions are applied to the text of the decision, enabling the precise identification of explicit procedural formulations. If the rule-based approach fails to yield a result, the system utilises a locally deployed large language model to extract candidate process phrases. These phrases are then semantically classified using embeddings generated by the SlovakBERT language model, which has been fine-tuned for the task of semantic similarity between sentences.

The result is a structured set of process events containing the identified event category, the extracted process phrase, the semantic similarity score, and the original

sentence from the decision from which the event was extracted. A briefer description of the system is provided below.

Hybrid extractor of procedural events from court decisions

The procedural event extractor is a software module implemented in Python, designed to automatically identify procedural legal acts from Slovak court decisions. The system combines rule-based natural language processing methods, semantic vector representations and a large language model (LLM) to achieve robust and terminologically accurate results.

The system is designed to identify the procedural outcomes of proceedings, in particular:

- dismissal of the appeal,
- successful appeal,
- discontinuation of proceedings,
- annulment of a decision,
- referral of the case for further proceedings.

System functionality

The main functionality of the extractor consists of the following steps:

1. Preparation of input data

The system works with a structured corpus of Slovak court decisions in JSON format. For each decision, the full text of the decision is processed.

The texts are normalised prior to processing using simple transformation rules:

- removal of redundant spaces,
- unification of fragmented headings,
- adjusting text formatting.

After normalisation, the text is divided into individual sentences, which form the basic unit for further analysis.

1.1. Filtering of meaningful sentences

Not all sentences in a decision contain process-relevant information. The system therefore performs heuristic filtering of sentences based on:

• the presence of process-related keywords,
• minimum and maximum sentence length,
• the occurrence of legal terminology typical of the operative parts of decisions.

This step significantly reduces the number of sentences analysed and increases processing efficiency.

1.2. Definition of procedural ontology

Process events are represented by means of a predefined ontology of process categories. Each category contains a set of representative process formulations used as reference prototypes in semantic comparison.

Examples of categories:

• appeal_rejected,
• appeal_successful,
• proceedings_discontinued,
• case_returned,
• decision_revoked.

2. Extraction of procedural events

In the first phase, the system uses regular expressions to identify explicit procedural formulations in the text of the decision. This approach enables highly accurate detection of standardised legal sentences.

Examples of detected formulations:

• "the appeal is dismissed",
• "the contested judgment is set aside",
• "the proceedings are discontinued".

Each identified phrase is immediately assigned to a specific procedural category.
Extraction using a language model

If the rule-based approach fails to identify any procedural event, the system utilises a locally deployed large language model via the Ollama tool.

The language model is controlled by a strictly defined prompt in Slovak, which ensures:

- extraction of legal phrases relating solely to the proceedings,
- generation of concise outputs without explanations,
- separation of individual phrases into separate lines,
- deterministic behaviour of the model.

This step also allows for the identification of less standard or linguistically variable formulations of procedural actions.

3. Semantic classification of events

The extracted process phrases are subsequently transformed into a vector representation using the SlovakBERT model, fine-tuned for the task of semantic similarity between sentences.
For each process phrase, the cosine similarity between the phrase's embedding and the embeddings of the reference formulations for each process category is calculated.

The process event is assigned to the category with the highest semantic similarity, whereby:

- a similarity score is calculated,
- the closest reference prototype is identified,
- a minimum similarity threshold is applied.

This approach enables robust classification even in the face of linguistic variations in legal formulations.

4. Post-processing and deduplication

After extracting and classifying events, the system removes duplicate records.

Duplicates are identified based on a combination of:

- event category,
- normalised text representation of the phrase.

The resulting events are then sorted according to their semantic similarity scores.

System output

The system output is a JSON structure containing:

- decision identifier,
- the number of identified events,
- a list of process events.

Each process event contains:

- the event category,
- extracted phrase,
- similarity score,
- reference prototype,
- the original decision sentence,
- extraction method.

The resulting output enables further analytical processing of procedural decisions and the creation of structured databases of procedural events.

Requirements and technologies

The system is implemented in Python and utilises the following technologies:

- Sentence Transformers for semantic embeddings,
- SlovakBERT STS model for representing legal sentences,
- scikit-learn for calculating cosine similarity,
- regular expressions (regex) for rule-based extraction,
- a local LLM (LLaMA 3.3) deployed via Ollama,
- CUDA acceleration for efficient embedding processing.

Recommended:

- At least 32 GB of RAM,
- a GPU with CUDA support,
- Local deployment of the language model for high-performance batch processing of large legal corpora.

Source files:

The project source code and data set are maintained in a repository and are available at: https://gitlab.science.upjs.sk/altaml/process-related-phrases