FELADATKIÍRÁS

A feladatkiírást a **tanszék saját előírása szerint** vagy a tanszéki adminisztrációban lehet átvenni, és a tanszéki pecséttel ellátott, a tanszékvezető által aláírt lapot kell belefűzni a leadott munkába, vagy a tanszékvezető által elektronikusan jóváhagyott feladatkiírást kell a Diplomaterv Portálról letölteni és a leadott munkába belefűzni (ezen oldal HELYETT, ez az oldal csak útmutatás). Az elektronikusan feltöltött dolgozatban már nem kell megismételni a feladatkiírást.



Budapesti Műszaki és Gazdaságtudományi Egyetem

Villamosmérnöki és Informatikai Kar

XXX Tanszék

Péter Bircher

Analysis of legal documents using text mining techniques

Supervisor

Csaba Gáspár

BUDAPEST, 2022

Table of contents

[Összefoglaló 5](#_Toc103287631)

[Abstract 6](#_Toc103287632)

[1 Introduction 7](#_Toc103287633)

[1.1 Motivation 7](#_Toc103287634)

[1.2 Applied tools 7](#_Toc103287635)

[1.3 Structure 9](#_Toc103287636)

[2 Text mining in legal documents 10](#_Toc103287637)

[2.1 Dataset 10](#_Toc103287638)

[2.2 Logical expression extraction 11](#_Toc103287639)

[2.2.2 Nature of conditionals 12](#_Toc103287640)

[2.2.3 Detection of logical structures 13](#_Toc103287641)

[2.2.4 Extract condition-effect 20](#_Toc103287642)

[2.2.5 Building flowchart 23](#_Toc103287643)

[2.3 Cross-references 25](#_Toc103287644)

[2.3.1 External cross-reference identification 26](#_Toc103287645)

[2.3.2 Internal cross-reference identification 29](#_Toc103287646)

[2.3.3 Cross-reference detection framework 30](#_Toc103287647)

[2.4 Title detection 34](#_Toc103287648)

[2.4.1 Preprocessing data 34](#_Toc103287649)

[2.4.2 Rule-based algorithms 35](#_Toc103287650)

[2.4.3 Classification 37](#_Toc103287651)

[3 Summary 39](#_Toc103287652)

[Irodalomjegyzék 41](#_Toc103287653)

Hallgatói nyilatkozat

Alulírott **Bircher Péter** szigorló hallgató kijelentem, hogy ezt a szakdolgozatot/ diplomatervet (nem kívánt törlendő) meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy hitelesített felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Kelt: Budapest, 2022. 05. 11.

...…………………………………………….

Rezeda Kázmér

Összefoglaló

A gyorsan fejlődő információs technológiának köszönhetően a jogrendszerben eletronikusan is feldolgozható dokumentumok mennyisége rohamosan növekszik. Ezeknek az adatoknak a feldolgozás mind emberi, mind gépi szinteken egyre nagyobb kapacitást igényel. Már a 80-as évek óta foglalkozik az adatbányászat ezen dokumentumok elemzésével és feldolgozásával. Célja a látens hasznos információk kinyerése a nem struktúrált adatbázisokból illetve különböző algortimusok segítségével ezen adatok feldolgozása.

Dolgozatban jogi dokumentumokat elemzünk feldolgozhatóság és adatkinyerés szempontjából. Különböző stratégiákat dolgozunk ki, melyeknek segítségével hasznos adatok nyerhetőek ki szerződésekből illetve egyéb dokumentumokból, Mindezek pedig hasznosnak bizonyulhatnak egy hozzáértő számára.

Az első részben ismertetjök a motivációt, bemutatásra kerül az elemzéshez használt eszköz, aminek segítségével szövegbányászati algoritmusokat hajtunk végre.

A második fejezetben három elemzés kerül bemutatásra, logikai struktúrák kinyerése szövegből, külső és belső referenciák felismerése és feldolgozása, illetve cím detektálás. Ezeknek az eredményeit kiértékeljük, egyes megoldásokhoz felhasználái javaslatot is bemutatunk.

Utolsó fejezetben röviden összefoglalásra kerülnek a kísérletek, illetve a dolgozat eredményei.

Abstract

Ide jön a ½-1 oldalas angol nyelvű összefoglaló, amelynek szövege a Diplomaterv Portálra külön is feltöltésre kerül.

# Introduction

## Motivation

In legal reasoning and decision-making, the content of the contracts, rulings, treaties, or legislative acts plays a critical role. Whether it is a legal dispute to undermine the other or is decision-making based on different conditions, the extraction, analysis, relating, and commenting of the proper information is crucial for the everyday work of a legal professional. Lawyers and judges deal with a vast and complex network of interrelated texts day by day. These texts have been around for centuries. Hundreds of legal texts are generated day by day and are becoming increasingly difficult to process, therefore in today’s automated world, the world of law cannot go without the help of a machine either. Recently, with electronic documentation, a legal professional can process the provisions, and search on the case bases quickly. There are companies, which are able to provide access to legal case bases. However, these tools are fairly accurate; it needs a legal professional to determine the relevance of the results, so these techniques cannot replace careful human reading.

With the help of text mining, we can perform information extraction. Using such tools, legal professionals can identify detailed properties and relationships within and among cases. Information can be carried out and can be made available to legal researchers on new cases automatically. The main goal of it is to extract structured, relevant information from unstructured machine-readable texts. The extracted knowledge is used to simplify the preparation of the case base. It also facilitates decision-making and legal reasoning or for automatic identification of legal arguments. The research in the fields of information extraction, natural language processing, artificial intelligence, and expert system has augmented the text mining process for enhancing the knowledge discovery process in this domain.

## Applied tools

By applying advanced analytical techniques, we are able to explore and discover hidden relationships within their unstructured data. We need a tool to perform various text mining techniques. In this paper, we are using a tool called Spacy. Spacy is a free, open-source library for advanced Natural Language Processing (NLP) in Python. Spacy is designed specifically for production use and helps you build applications that process and “understand” large volumes of text. It can be used to build information extraction or natural language understanding systems or to pre-process text for deep learning.

Spacy provides a wide range of features, some of them refer to linguistic concepts, while others are related to more general machine learning functionality. In Spacy, we have a pipeline where we have pre-trained components like “textcat”, “ner”, “sentencizer”, “word2vec”, etc. for a variety of languages. With these components, we can perform machine learning algorithms. To use a pipeline, we need to have a language pack first. These packages include language-dependent components like lexical entries, binary weights, and word vectors which are later used by the pipeline. In our research, we are going to use text classification and named entity recognition.

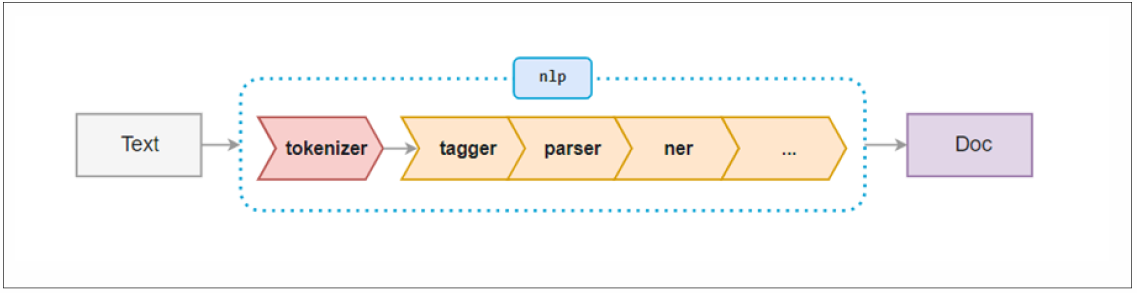


Figure 1 – Block diagram of Spacy's pipeline

**Text classification** is one of the most important techniques in text mining, commonly used for topic analysis, opinion mining, intent detection, or sentiment analysis. The model learns to make classifications based on past observations and can learn the different associations between pieces of text, and that a particular output (i.e., tags) is expected for a particular input (i.e., text). A “tag” is the pre-determined classification or category that any given text could fall into. To evaluate a classification model, we are using ROC curve and AUC score.

**Named entity recognition** (NER) aims to locate and categorize key information, i.e., entities, in text data. These ‘entities’ can be any word or any sequence of words that consistently refer to the same thing. Most modern named entity recognition systems make use of a machine learning/deep learning model. The usual measures to evaluate the quality of a NER system are called precision, recall, and F1 score. In this paper we are using these metrics to measure our models, however, NER can fail in many other ways, many of which are arguable "partially correct", and should not be counted as complete success or failure, but described later in this paper.

Each component consumes a different format of training data and needs a different configuration. Spacy provides a user-friendly and end-to-end system to train the different pipelines, so the user does not have to deal directly with the underlying neural network architecture.

To perform classification, we have **TextCategorizer** as an optional and trainable pipeline component. To train it, we need to provide examples and their class labels. Later, the categorized text could be found in the “doc.cats” property. Since it is an optional component, we need to add it to the pipeline first, then we are able to train it and evaluate the results. On the other hand, the “ner” component is called “**EntityRecognizer**”, an essential and by default a pretrained component. However, we have the ability to custom train this component, we only need to have a labeled training set with entities, their labels, and their positions within an example sentence. The labeled entities could be found in the “doc.ents” property.

In the following research, we often refer to these pipe components while performing various experiments on legal texts.

## Structure

In the next chapter, we are presenting various analyzes of legal texts using the above-mentioned Spacy text mining tool.

In *Section 2.2*, we propose a solution to extract logical patterns that are common in legal texts. We talk about the nature of these patterns, their distributions among the datasets, and possible solutions to recognize and extract these patterns. Then, we also visualize the results to have a comprehensive picture of real-life usage. In *Section 2.3*, we investigate the issue of cross-references. We examine cross-references across legal texts, try to identify them, then make plans on an automated cross-reference detector framework. In the last part of this section, we present a solution that can identify and distinguish titles and paragraphs, which could be core for an automatic table of contents generator.

Then *Section 3* concludes the article where we give a brief summary of the results of our experiments.

# Text mining in legal documents

While data mining and text mining a in international business and legal systems it is given domestic emphasis, its use and use in applied research are still in their infancy. We have several experiments that we can perform using legal documents, in general, the basic tasks of text mining can apply as such. We can classify documents into categories (these are subcontracts, these are employment contracts, these are licenses, etc.), highlighting the difference between previous contracts and the new contract (new versions often come, but where is the substantial change). We can develop smart search (semantic, ontological) or highlight concepts, and definitions, producing your glossary. In this section, we selected three main experiments, namely logical structure extraction, cross-reference recognition, and title detection using text mining techniques.

## Dataset

The given dataset used for the following experiments consists of ten legal contracts in English. These contracts were originally embedded in HTML (**H**yper**t**ext **M**arkup **L**anguage) pages, from which they were parsed individually HTML tag by HTML tag into a raw text format and saved into a text file. In addition, each contract is accompanied by another “tags” file that contains these HTML tags for each line.

A képen szöveg látható

Automatikusan generált leírás

Figure 2 - Raw text and tags

It is important to note, that our data set is unlabeled, and we do not have any labeled data during our first two experiments *(Section 2.2 and Section 2.3).* Therefore, each task is preceded by a data formatting and labeling process, where we give labels to our data set according to the given task.

To get a more comprehensive picture and see how our results perform in other environments, we have another ten different contracts at our disposal. The source of these documents is the publicly available EUR-Lex webpage which is a collection of EU legal contracts, treaties, rules, and other EU-related legislative documents. We note that these contracts are plain unformatted texts, and these contracts are also required pre-processing and labeling as our initial dataset.

In the following chapters, the initial dataset is referenced as “**Legal-Contracts**” and the contracts from the EUR-Lex are referenced as “**EUR-Lex**”.

\*TODO: valamit írt a dataelőfeldolgozó szakmáról

## Logical expression extraction

Conditionals describe the result of a certain condition in a particular scenario. These sentences are statements of an “if-then”, “unless-then” situation *(although “then is not used”),* but other keywords such as “when”, “where” or “in cases” can be found in these kinds of sentences either. In legal documents, especially in contracts, there are often parts where certain paragraphs are in favor due to the fulfillment or non-fulfillment of a condition. What if there is a tool that is able to identify these structures in a document and could create a diagram next to the document to help the readers better and quicker understand that particular part of the contract? In this study, we conduct an experiment in which we collect the conditional sentences found in the legal documents, then break them down into a cause-effect relation and display them in a flowchart-like manner.

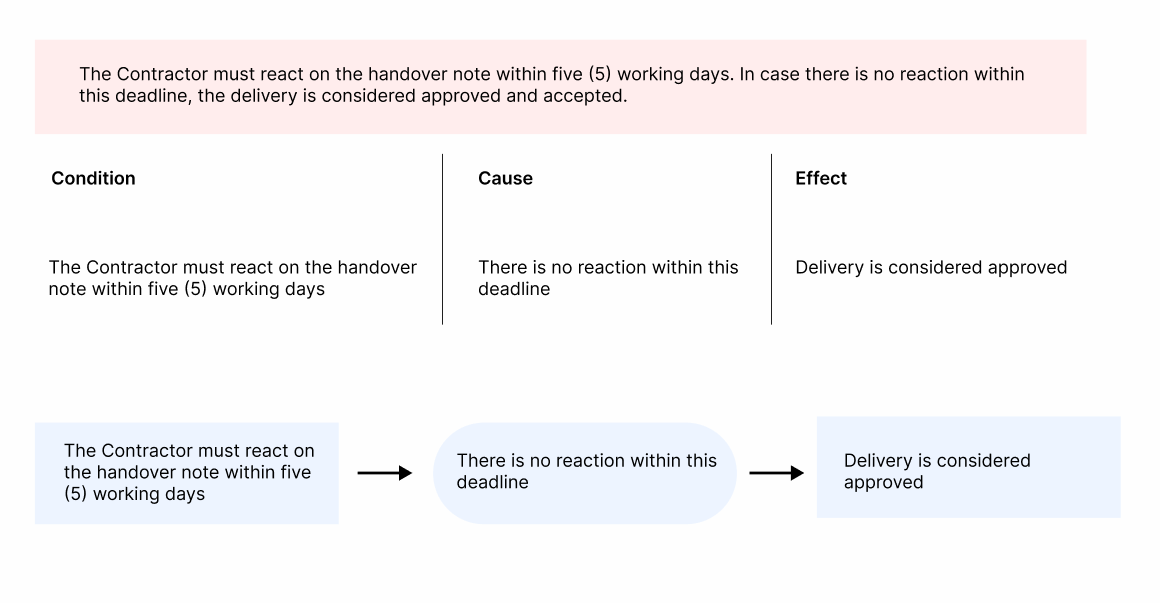


Figure 3 - Cause-effect relation in a sentence

A képen szöveg látható

Automatikusan generált leírás

Figure 4 - Logical pattern detection in a document

### Nature of conditionals

The word “if” can almost always be used to introduce a case or condition in legal documents. However, there are cases, when other words are a more appropriate choice than a simple “if-else” structure, it depends heavily on the context and what we want to express. In case of time or timing is important to the rule or describing a rare or once-only situation, or there is some certainty that an event will occur, “when” is a better choice. “Where” usually is to introduce adverbial clauses that refer to a specific physical place. Of course, this does not preclude the use of “in cases where”, “where”, or “in circumstances where” when they seem more logical and natural in the particular context.

In our dataset, we can find examples of all the above-mentioned keywords. Examining the distribution of the conditional sentences among the contracts, we find a very similar amount of each type.

A large majority of conditional sentences are introduced by the subordinating conjunction” if”. Many other connectives could introduce conditionals, e.g., “in case”, “unless”, “when”, “assuming”, and “where”. Table (w) shows the distribution of these keywords among our datasets.

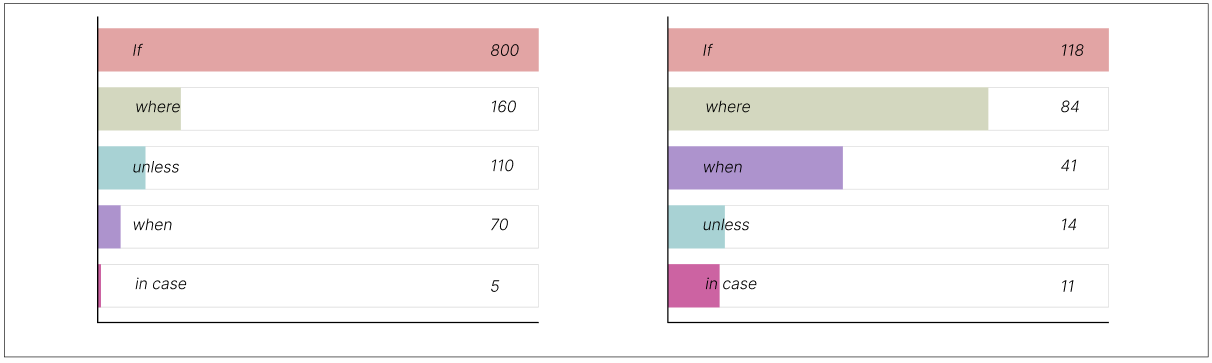


Figure 5 - Distribution of keywords across the two datasets

The left chart shows the results of the search in the **Legal-Contracts** dataset, and on the right, we see the results in the **EUR-Lex** dataset. In both sets, in the vast majority of cases, the keyword “if” occurred most frequently while the distribution of the remaining keywords is quite similar to each other. We note that the documents from the **EUR-Lex** dataset are significantly less than the documents in the **Legal-Contracts** dataset.

### Detection of logical structures

A question naturally arises about whether a basic search pattern could be sufficient to recognize conditional sentences. The answer is yes because a large proportion of conditional sentences could be filtered by relevant keywords (detailed above in Figure 3)*.* However, in our study, our objective is to extract logical patterns in legal documents, where particular provisions or actions come into force in case the related conditional is met.

*A képen asztal látható

Automatikusan generált leírás*

Figure 6 - Difference between conditionals

The figure above shows us two examples of these cases. Both sentences contain a conditional keyword, although the meaning from a legal perspective is different. While the first one is a recommendation, the second sentence is a cause-effect structure, the pattern we are looking for in our study.

It is important to note, that in many cases the conditional case is not focused on a single sentence. In many cases, the condition itself and the reference to and the cause of the condition are separated in the following/previous sentences. We need to pay attention to these patterns as we consider taking the causal relationship separately.

A képen szöveg látható

Automatikusan generált leírás

Figure 7 - Complex conditional

#### Classification

As highlighted before, we do not want to perform a simple keyword-based search but want to run a semantic analysis on the document. Now that we have a comprehensive picture of the data available and the nature of the structure that needs to be extracted, we could prepare an automatic approach with the help of machine learning algorithms. In this paper, we use Spacy for building and using text-mining machine learning models. In our first approach to the problem, we perform semantic analysis using text classification. The model should tag sentences that contain this logical structure. We have two classes to categorize them, “conditional” and “non-conditional” and we have to divide the sentences into these two groups accordingly to this. In the end, we expect that the model we have built will be able to differentiate between them. Once we can identify the logical patterns within a document, further processing can be done with other text mining tools to distinguish the cause-effect relation to display it as a flowchart.

Spacy’s classification model works by giving it a properly structured set of training sets from which Spacy builds the classification model. Compiling the right set of training data is therefore vital to our success. We must first manually review the contracts for examples with the appropriate logical structure. In addition, of course, we need such sentences in the training set that does not contain these patterns.

Once we have created the relevant components for classification in Spacy and created the corresponding labels (“conditional” and “non-conditional”), we can start teaching the model. we used a model in which architecture was built upon a convolutional neural network. Convolution neural networks (CNN) are often used for NLP tasks, especially for text classification and semantic analysis, because CNNs are very good at pattern detection.

#### Train and test sets

The problem is a classification task; therefore, we need to categorize our dataset into conditional and non-conditional sentences, which will be the training set. Our dataset is untagged, raw data, so in our case, before we could even validate our concept, we need to gather sufficient examples, which means, we need to manually go through the documents to look for conditional sentences (and non-conditionals!). It is a time-consuming task since the documents are unformatted but crucial. For the following classification task, from both datasets, a total of 100-100 example sentences for conditional and for non-conditional were collected across 9-9 contracts, and 1-1 contracts were used for the testing set. The training set in both cases consists of forty example sentences where conditional and non-conditional sentences are evenly distributed.

Furthermore, when collecting the right sentences, another method was used, which resembles the term “reinforcement learning”. The method is the following:

1. Train the model with x examples
2. Validate the model on a validation set
3. Look for anomalies, manually add these examples to the train set with a correct tag
4. Retrain a model with the anomalies

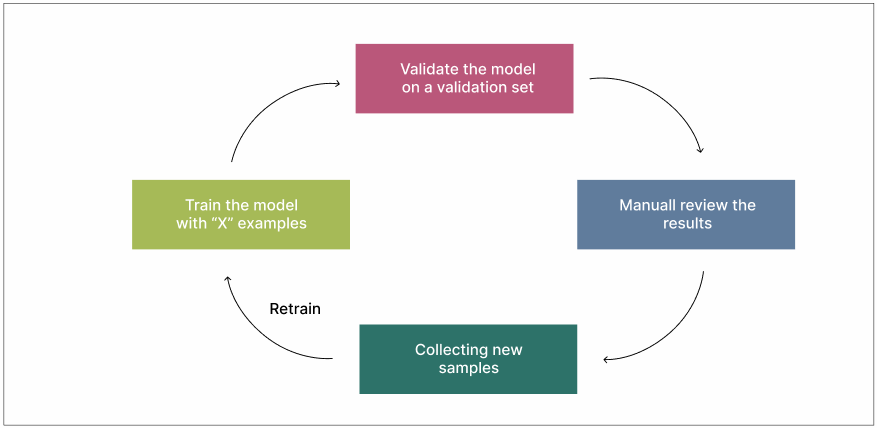


Figure 8 - Manual reinforcement learning

In this classification task, there were three manual-retraining phases, each round there has been an additional 50-50 examples added to our training set.

#### Results

First, the **Legal-contracts** dataset was used. Our classification model was trained with 50 sentences for each category. Then we validated the model with the validation set, looked for anomalies, erroneously classified results, and retrained the model with the expanded training set. We run these experiments three times (50-50, 100-100, 150-150) and evaluated the model on the test set.

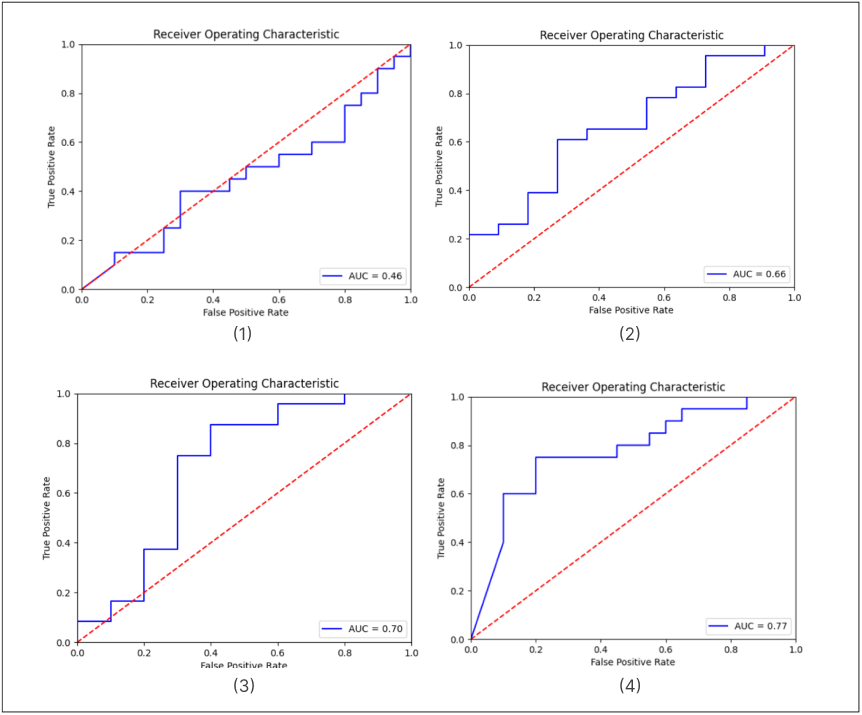


Figure 9 - AUC score with different number of samples

From the results, we can assume that as the training dataset grows, so does the outcome. The first figure (1) with an AUC score of 0.46 is a simple guessing model, but over the increase of the number of training samples, we are getting better and better scores (2), (3). The model which was trained by 200 examples, achieved an AUC score of 0.77 which is considered a good classification model since it can classify approximately into correct categories.

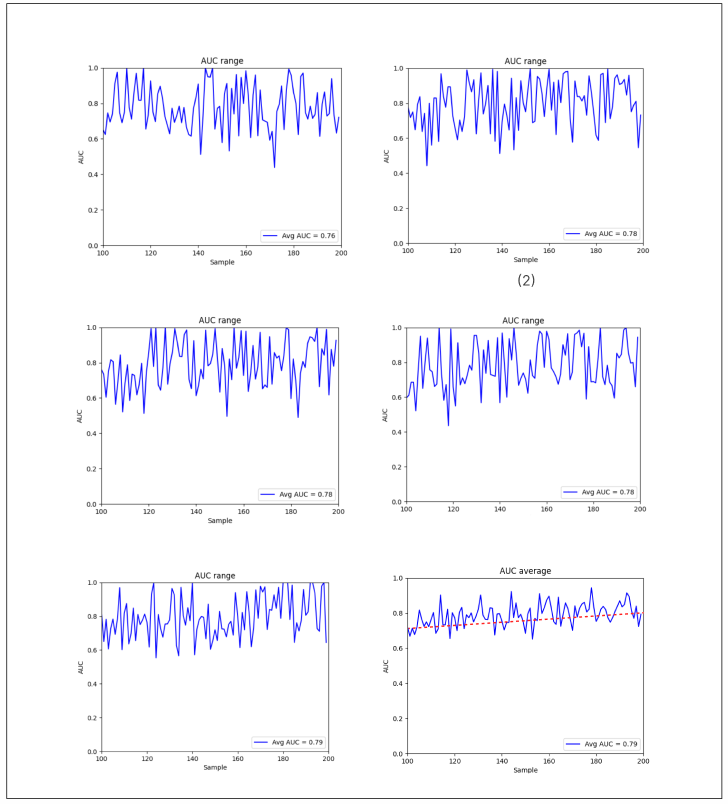
Nevertheless, having this small amount of train and test set, we cannot state with absolute certainty that our model is an acceptable classification model, so for further validation, we may also see the result between the samples. We are training our model, but this time we are incrementing the training set one by one, and after each round,d we are measuring the AUC score. We do these five times and check the results. 

Figure 10 - AUC score with increasing samples (Legal-contracts dataset)

The first five-figure shows the AUC scores in five loops. The most important figure is the last one, where we can see how the average AUC score is increasing over the number of samples. The red line shows a monotonic increase in the AUC scores. The first value is 0.71 and the last one is 0.79, so we can see an increase of 0.08 over the number of samples. Hence, we could conclude, that increasing the number of samples could result in a better classification model, we need further clarification on this topic.

Now we have our first results, let’s see, how the **EUR-Lex** dataset performs in the classification.

A képen szöveg látható

Automatikusan generált leírás

Figure 11 - AUC score with an increasing number of samples (EUR dataset)

In case, the average score shows an increase in the AUC score, from 0.58 we reach 0.65, and the efficiency increased by 0.07, which is a similar value as we got in the previous model, however, the overall AUC scores are smaller.

The last experiment is a cross-testing when we test the model on a different dataset, which means we will test a model that was trained with the **Legal-contracts** training set on the **EUR-Lex** dataset and vice versa.

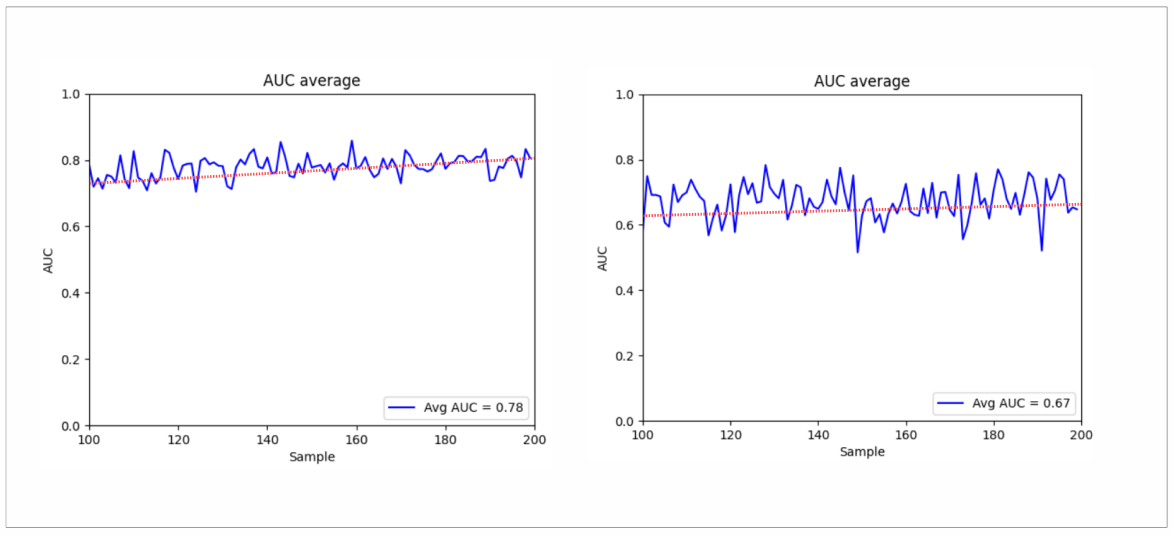


Figure 12 – Cross-validation of both models

The first figure shows how the **Legal-Contracts** dataset performs on the **EUR-Lex** test set. We can see the increase in the AUC score as clearly as we see in (Figure 10). In the first figure, it increases from 0.8 to 0.86 (0.06) and in the second it is from 0.58 to 0.64 (0.06). We note that we got better AUC scores with the Legal-Contracts dataset testing on EUR-Lex than otherwise.

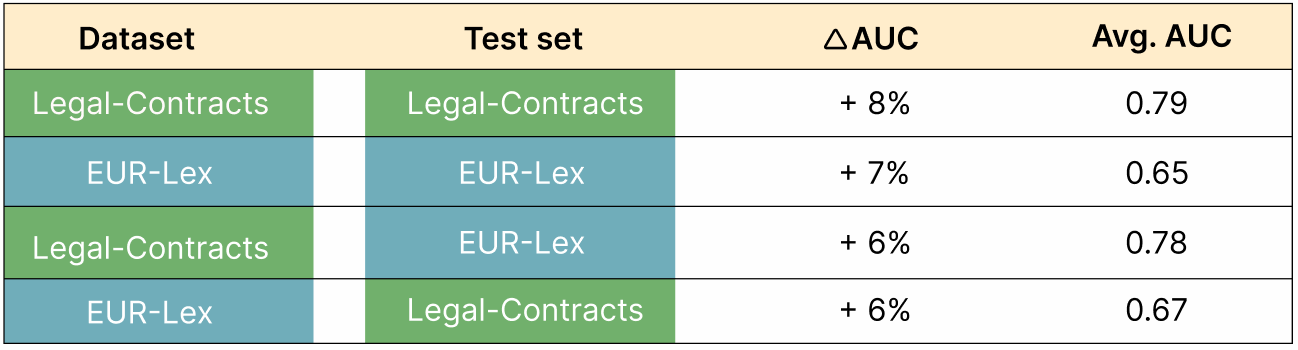


Figure 13 - Summary of the model performance

Overall, the **Legal-Contracts** dataset had a little better grasp of the sentences, performing better on both its own and on the other dataset.

### Extract condition-effect

We assume that our model mentioned above can successfully decide on a sentence whether it is conditional or not with an appropriate number of samples. In the next step, we want somehow to get the exact logical relation from the sentences, so separate the condition and the effect in a sentence. Hence it is not as straightforward as it looks as we need to find the border of the condition and the effects. It is not efficient to split a sentence on a comma or with a rule-based algorithm, we need another approach to extract them and successfully build a flowchart.

#### Named entity recognition

The approach for the above-mentioned problem is named entity recognition (NER), where we tag the condition and the effects in a conditional sentence. In the following experiment, we explore the possibility to recognize coconditionsnd effects with the help of NER.

The process starts with constructing a proper training dataset. For NER, Spacy needs the sentence itself completed with an additional object called “entities”, which holds the positions of our custom entity labels. In this approach, we have two labels, “Cause” is the part of the conditional sentence that contains the condition, and “Effect”, is the part that applies if the condition is met.

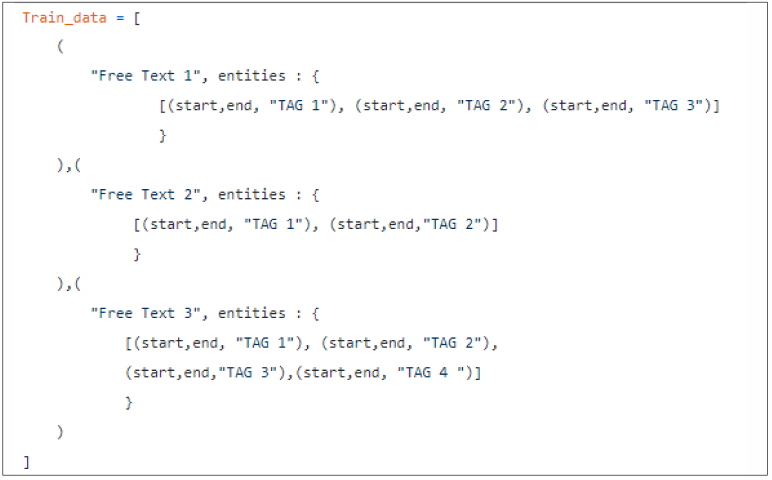


Figure 14 - Training data format

Since we need the exact positions of the entities, it is not enough just to collect conditional sentences, we need to calculate the positions. For constructing the training set to the Spacy format, we created a program. In this program, we need to select with the cursor the demanded part of the sentences, and the program calculates the position of the selection. Then clicking on a button, it gets added to a JSON file, which is later converted to the Spacy format. This method can significantly reduce the sample collection time.

A képen szöveg látható

Automatikusan generált leírás

Figure 15 - JSON config file

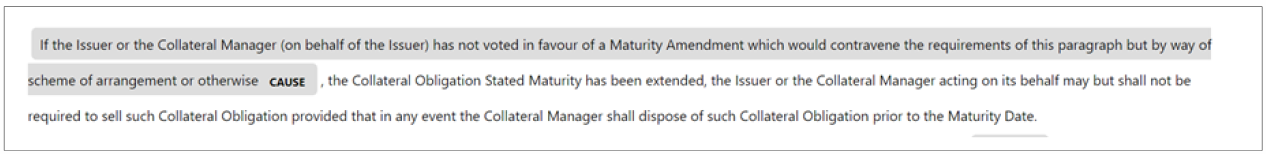
The training and validation method is the same as we can see in *Section 2.2.2*, so training, validation, and retraining of the model with new examples. In the first round, thirty examples were collected as the training set and validated over a test dataset. After looking at the results created by Spacy, we see that there were cases where our model can successfully recognize the condition-cause relation.

A képen szöveg látható

Automatikusan generált leírás

Figure 16 - Result of NER (30)

There were plenty of misrecognized patterns as we can see in the figure. However, after looking more into the data, we find a pattern, namely, that our model barely can detect the “effect” part of the sentences. It was able to find the condition but could not be able to match the “effect” part. This can be explained that while the condition part of the sentences could be similar, the effect part of the sentences could be drafted in anyways.



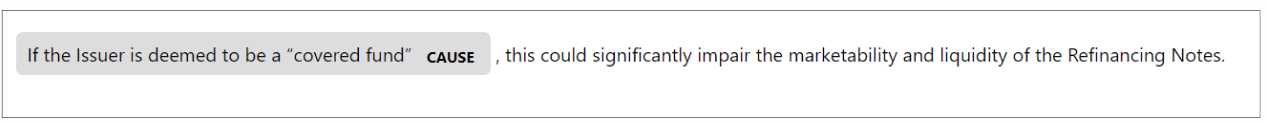


Figure 17 - Correct results of NER

We retrain the model with similar examples as the missed or wrongly recognized entities. This time we are collecting around an additional twenty examples, then checking the results again.

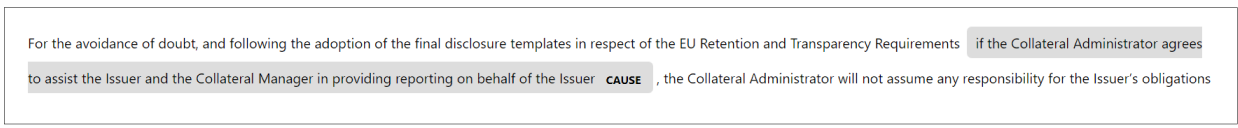
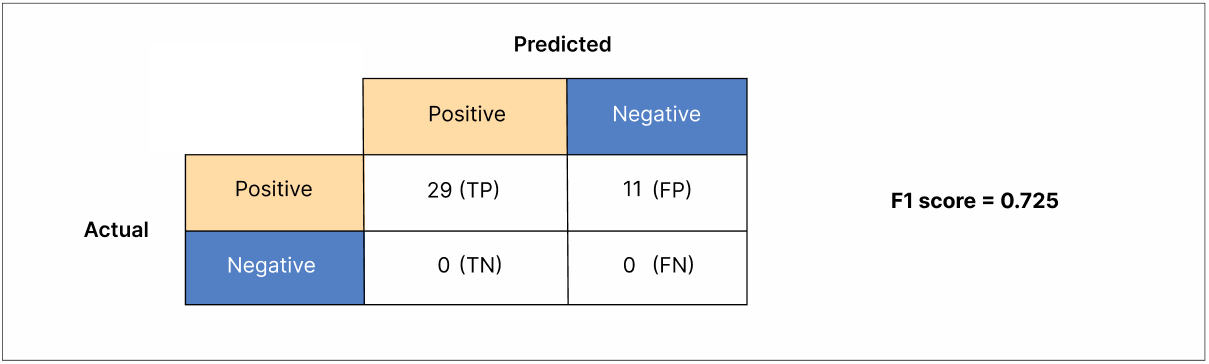


Figure 18 - Result of NER (50)

The model already performs acceptably when it comes to tagging the condition part of the sentences, however, it could not be able to identify the “effect” part. Only a few effects have been identified. Starting from this, we will change our approach. Since the condition, part has been identified with an approximate success, and we only have conditional sentences, the other part of the sentence which has not been recognized as the effect leaves the effect part, so we do not need to recognize them in the first place. With changes to the training dataset (removing the “effect” tags from the entities objects), and training the model, we got the result as we expected.

To evaluate the model, as mentioned in Section 1.2, we use a confusion matrix, precision, and recall. We have a test set that consists of forty examples both from conditional and non-conditional sentences. The model recognized the conditions with a high success rate; 

### Building flowchart

We have a model which recognizes the conditional sentences across a document. We have another model that identifies the condition within a conditional sentence. Now, we just need to create a solution that can read a document and then displays the logical patterns within a paragraph. For this, one contract will be used as the input. Then, it gets processed into a sentence with the help of the built-in sentence splitter in Spacy called “sentencizer”. The sentences will be the input of the classifier and the output is the conditional sentences, and it provides the feed for the NER model, which then separates the condition and the effect within a sentence. The result then gets saved to a JSON file, and later, this JSON file is used to provide the “conditional” configuration for an application, which reads this JSON and visualizes the condition. For this, we validate the concept behind the application.

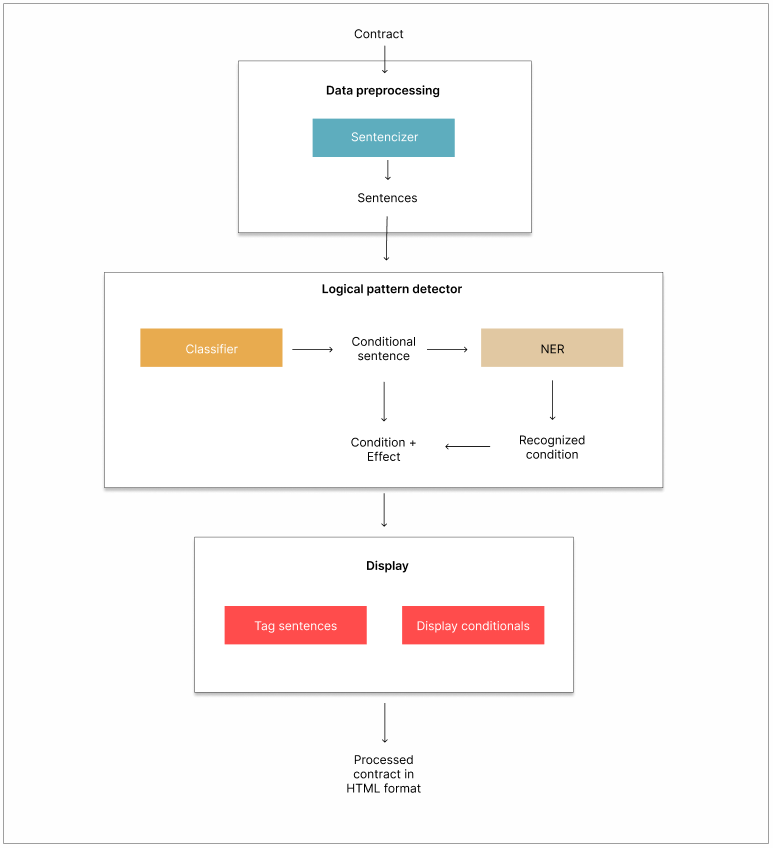


Figure 19 – Block diagram of the application

The picture above shows the block diagram of the proposed application. In our test, a python script processes a contract, then breaks it down into sentences and feeds the logical pattern extractor. The model identifies the conditional sentences, the conditions, and the effects and then saves them to a JSON file. An HTML webpage then loads the JSON file and the contract, it connects the paragraphs and the conditionals and displays them to the user next to each other.

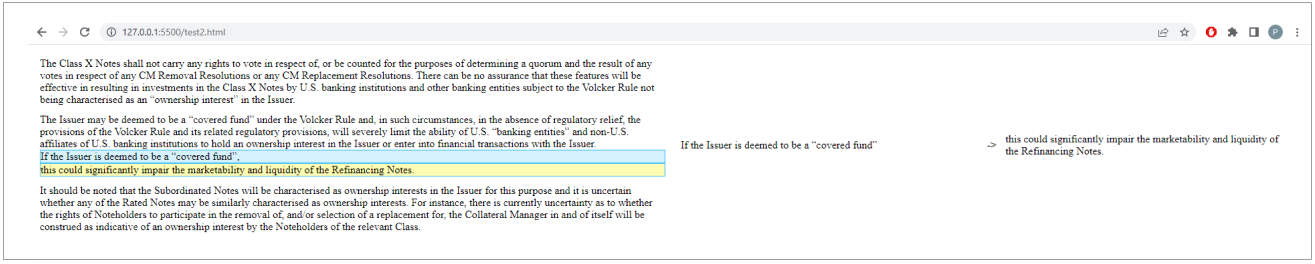


Figure 20 - Result of logical extraction

The whole contract is an HTML table. The first column contains the contract, each paragraph is a separate table cell (<tr>). The other column is for the conditional relation showing. In case the current paragraph in a row has got a conditional relation in it (comparing the JSON texts with the current paragraph), first, we highlight the condition-effect in the first column, and in the other column, the relation gets displayed (as we can see on the figure above). This column also contains a table where we place the condition and the effect.

## Cross-references

In a contract, a provision often refers to a clause in another contract or the same contract, or it can also refer to itself (*this is Section 2*). A cross-reference shows how two provisions interact with each other, whether one is an elaboration on the other, prevailing, or subordinated.

A képen szöveg látható

Automatikusan generált leírás

Figure 21 - Cross-reference examples in legal documents

There are recommendations to keep the number of the cross-references to a minimum since it improves the readability and helps better understand the provision on its own without having to turn or look up other contracts, and treaties. Not to mention, a high number of cross-references increases the chances of dead provisions in contracts. Writers often rely on suffixes like “below”, “above” or “hereof” to the cross-references, which indicates that the reference is an internal cross-reference. It is also a recommendation to omit these drafting to avoid confusion and later misunderstandings.

Checking cross-references and schedules is an extremely important task that contract drafters must, but often do not undertake. There are some real-life examples where wrongly addressed cross-references led to lost cases or severe fines. In the following study, we are going to study these cross-references in contracts. We identify them within documents and later we connect the provision’s source or chapters with these references. Then, in the end, we propose a framework that could automatically perform cross-reference recognition in legal texts. During the experiments, we are going to use the **“Legal-Contracts”** dataset only.

### External cross-reference identification

Detecting and resolving cross-references in a legal text requires precise knowledge of the structure of those references. They come in various shapes, therefore step one is to identify the nature of the external cross-reference format in our dataset. Analyzing the contracts, the following examples can be listed as possible external references:

**

Figure 22 - External cross-reference examples

The references often consist of a prefix, which defines the specific paragraph within the law. In our approach, we are not going to consider these prefixes. The distribution of the types across the documents follows the same ratio shown below. The values ​​represent the average of the ten documents.

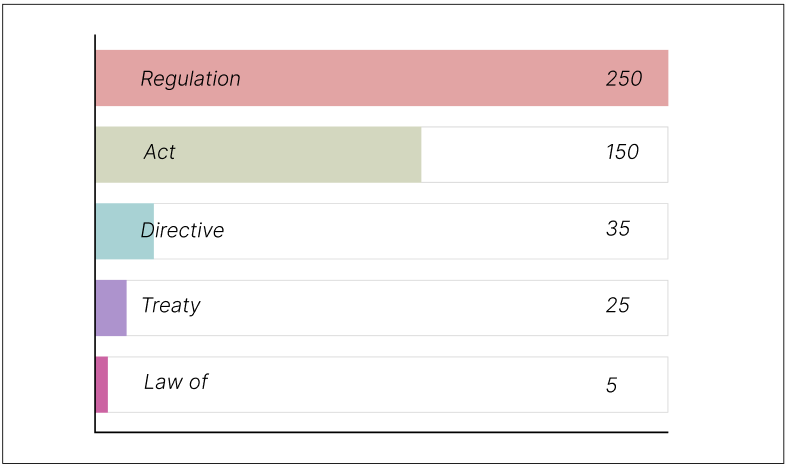
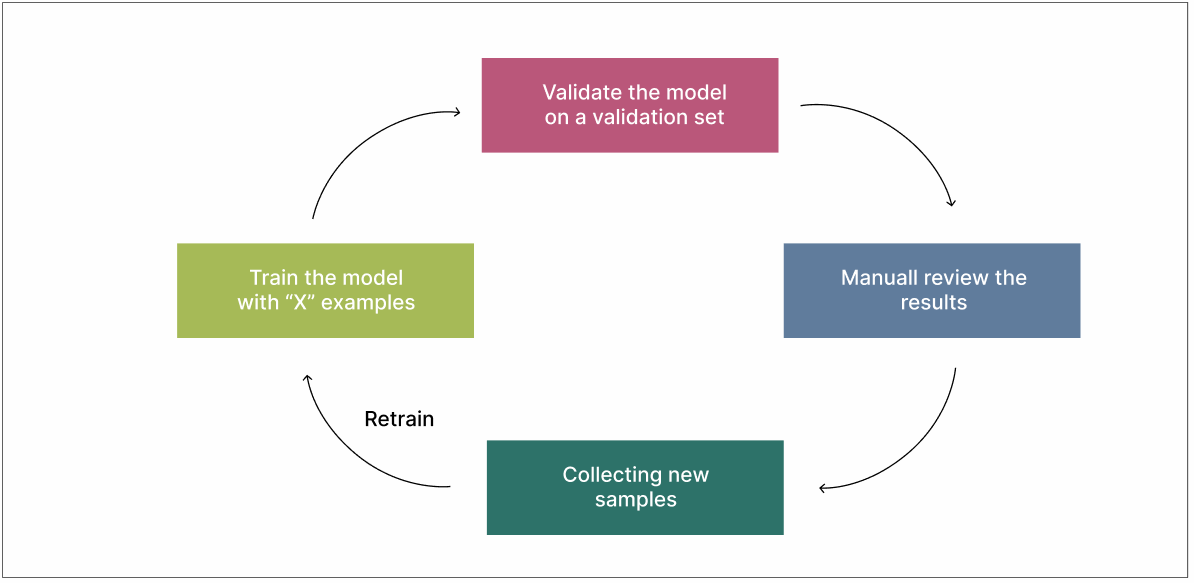


Figure 23 - Keyword distribution across the contracts

The identification of these entities leads us to a classic named entity recognition, where we need to train our custom NER model to identify these acts, regulations, directives, etc. After we successfully created a model and detected the external references, we validate our solution for a real-life situation. We link the references to their external source, build a reusable dictionary and create an HTML document with the links. https://www.unite.ai/how-does-text-classification-work/

#### Model building



The model building part is very similar to what we performed in Section 2.2.3.2. In the first phase, where the training is happening, we need to gather sufficient examples as a training dataset to train a custom NER model with the help of Spacy and to have valuable results. In the second phase, where the evaluation and retraining happen we construct a test dataset only to retrain our model. To meet this requirement, I used a whole contract itself, then retrained the model with such examples as what we missed identified or incorrectly recognized.

In the first round of training our model, we used thirty training examples. The structure of the training dataset follows the same pattern as we see in *Section 2.1.1*. We labeled our custom entities as “External-reference”.A képen szöveg látható

Automatikusan generált leírás

When constructing the training dataset, it is important to follow the same ratio of the individual types as we see in Figure 21 above.

A képen szöveg látható

Automatikusan generált leírás

Figure 24 - Recognized external references

The result after the first round is similar to intended. Our model was capable to recognize several external references even though the size of the training dataset. There were some mistakes, we collect them and next round we are going to add these erroneous results to the training dataset to fine-tune our model.

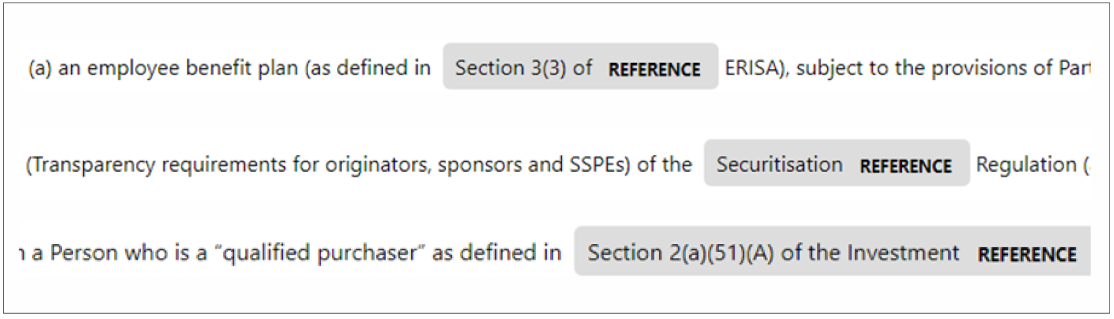


Figure 25 – Mis-detected external references

We do this several rounds, to reach a point where our model looks capable to recognize external references. To evaluate the NER model as in *Section 1.2* mentioned, we are using confusion matrix, recall, and precision to have a comprehensive picture of the performance. In external references, we consider partial recognition as the correct one if it can be determined the act from it. For instance, we have a few characters surpluses or less. The reason for this is that we can match partial strings later to link with their source, we do not need exact matches at all.

We have a test document, which contains forty occurrences of external references. We validate the model which was trained with hundred examples. From these forty occurrences, it found 29 entities, including partially correct ones as well. There were cases when it wrongly detected words or characters. However, overall, we calculated a precision of 0.85, which means how many entities have been identified by the model divided by all the relevant entities, in percentage. The recall is 72%.

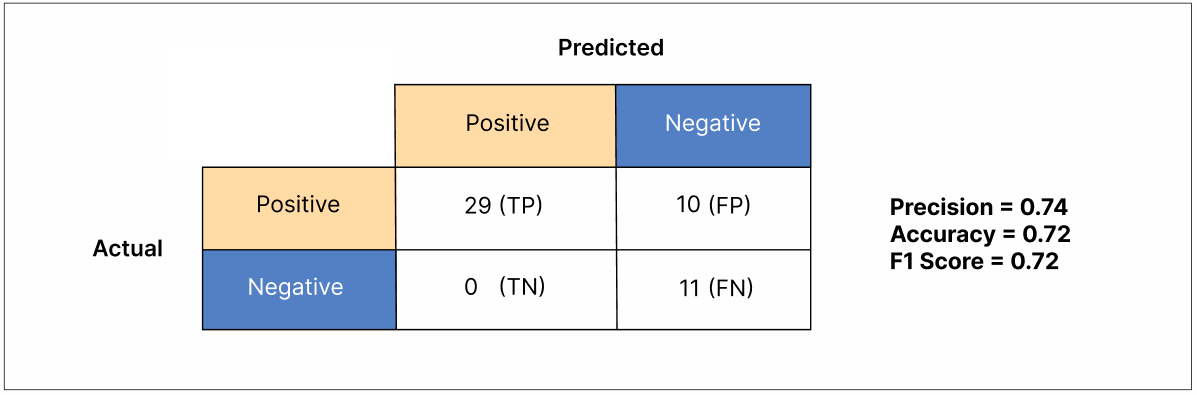


Figure 26 - Confusion matrix of the NER model

Despite we only using the **“Legal-Contracts”** during this experiment, we intended to cross-validate the solution on the **EUR-Lex** dataset. Performing a NER on a EUR contract, we expect that the model will be able to recognize such external references that are similar to those we can find in the **“Legal-Contracts”** dataset.



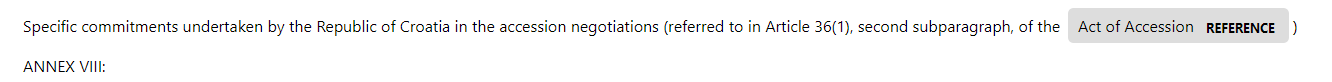




Figure 27 - Result of the model on a EUR contract

We are seeing that the model could recognize several entities. Going through the results, we are seeing several misrecognized entities, however, the model was approximately able to find the external references. We expected this since external references have a more fixed format and the NER model could apply these settings to this dataset as well.

At this point, we are not proceeding with this experiment since the task is not to create an excellent solution with perfect models, but to validate the concept of detecting external references. Next, we are examining the topic of internal references.

### Internal cross-reference identification

Internal cross-references are used to cite text and notes within the same work. Often preceded or followed by words like “below”, “see”, “herein”, and “above”, and could be a reference for a different section or in the same section but a different paragraph.

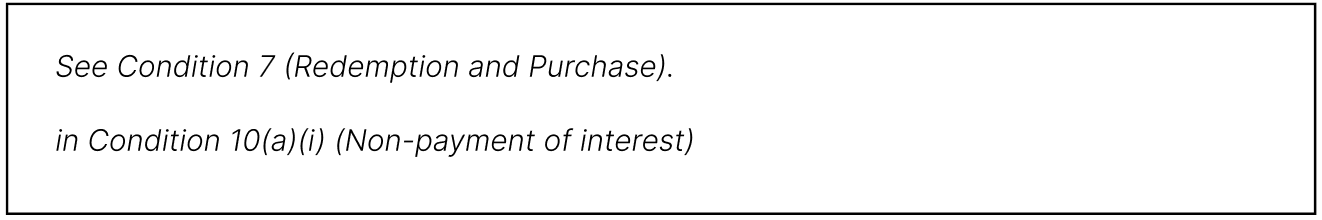


Figure 28 - Internal reference examples

Since it is hard to distinguish the internal references, because they strongly depend on the drafting and they also vary from text to text, therefore in this section, we only focus on the chapter references and not the references within a chapter.

#### Model building

As the external reference identification, the internal reference identification is a NER task either and the whole process is similar except for the training data and evaluation. For constructing the training data dataset, we are creating an example set of fifty samples. Collecting the right example set method is the same as we see in the external reference section. We expect to see a similar result as well.

Going through a result, it is noticeably that this model performs better than the external reference one. The reason behind this could be that there are fewer types of internal references than external references ones. Also, we emphasize, that the text context of internal references is visibly simpler, therefore the NER model could better distinguish or detect the references based on the surrounding words.

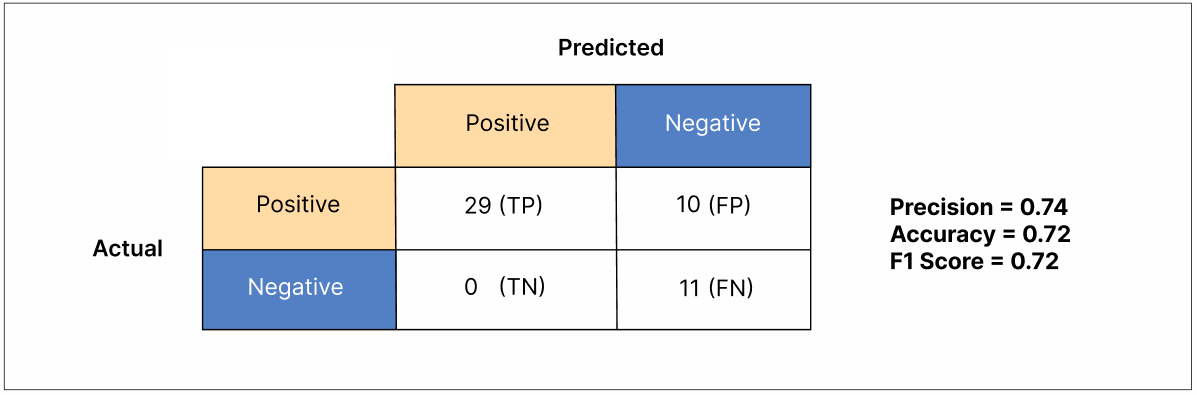


Figure 29 - Evaluation of the NER model

However, testing the model against one contract from the EUR-Lex dataset gives us a different result. We are expecting that the internal references are strongly dependent on the drafting, therefore our model will perform worse. The result shows us how the model can hardly identify entities. We concluded that the NER model relies on the drafting more than classification and because internal references are strongly dependent on the wordings, it is harder to build a model which can recognize references with certainty.

### Cross-reference detection framework

Now that we have created an approximately working model, I will try to present the planned workflow through a real example. An application, which could process a contract presented as user input, could recognize the external reference within the contract, and as an output, displays the contract in a web browser as an HTML page in which the external references are linked to an external source.

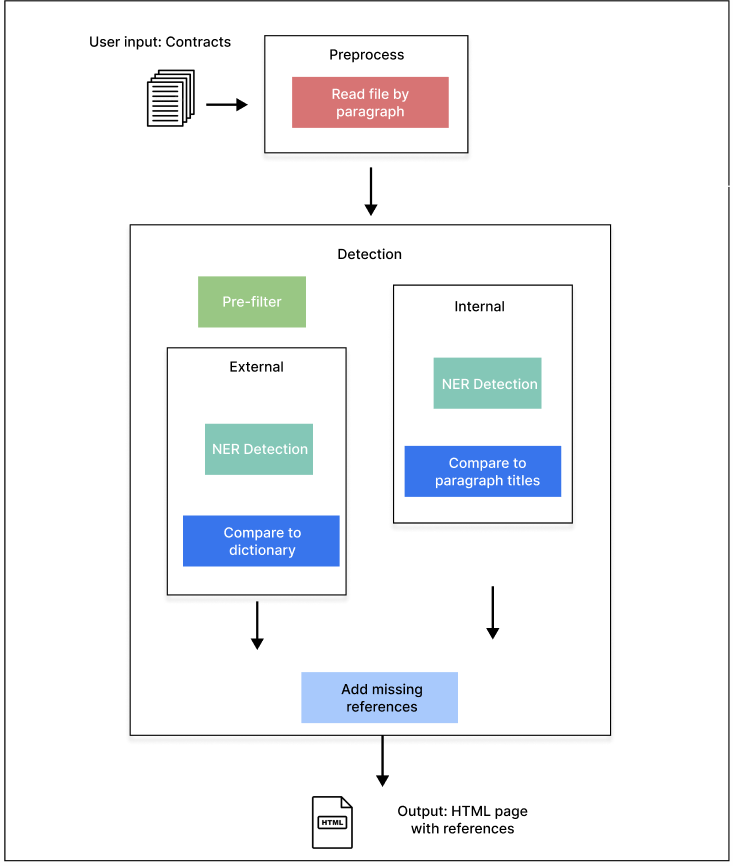


Figure 30 - Block diagram of the reference recognition process

In *Figure 26* we see the block diagram of the application. The input of this service is a document, and the produced output is an HTML source code filled with the linked references.

In the preprocessing phase as a first step, we read the input provided by the user. We read the contract line by line and then, we prefilter our dataset. On prefilter, we mean that as we explored the nature of the references, we see that the internal and the external references have the same words almost in every case. Therefore, before we feed our recognition model, we filter the paragraphs by these trigger words. Internal and external references have different trigger words. Thus, our uplifting model already gets a pre-filtered data set, which we hope will help filter out false results more.

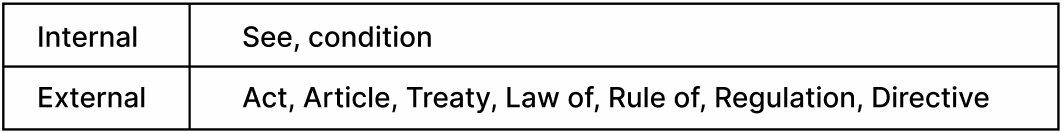


Figure 31 - Most common keywords

During the whole process, we build the output HTML file in parallel. When we have a paragraph, which contains one of the trigger words, we feed our pretrained NER model to identify the entities within that paragraph. After the detection, if we found a reference, we need to tag these lines and find their internal or external source.

When it comes to the internal references, we need to get the chapters. If we have the recognized internal reference, first we do a simple search, where the search word is the identified chapter name. We have the whole document’s text as well so if we find such paragraphs that only contain the search word, then we assume, it is the corresponding chapter (we can do the chapter-matching with classification and title detection, as we see *in Section 2.4*). The current paragraph got a <p> tag while the identified internal reference got a <a> tag with a “*href*” attribute which points to the row ID. The row ID also gets saved when we reach that particular row in the document. Then the paragraph containing the row will get an id with the row ID in it (this way when we click on the internal reference, we get directed to the paragraph that the internal reference refers to).

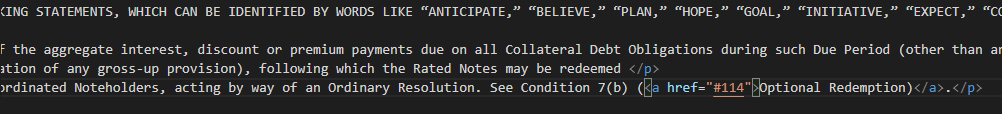


Figure 32 - Internal reference HTML code

Another case s, when we find an external reference. For external references, we do not have the source which we need to link for. In this case, we have two options here: one is to manually add the corresponding links to the references, and the other is to have a dictionary wia th predefined link-reference connection. When we find an external reference, as the first step, we go through our dictionary to look for matches. If we find a match, we just simply add the link to the “href” attribute and wrap it in a <a> link tag. But if we do not find it, then we have to manually add these links, but this would only happen after we went through the whole document. Then the application notifies the user about the missing references and asks the user to fill these with the correct links, then the remaining external reference HTML tag gets filled.

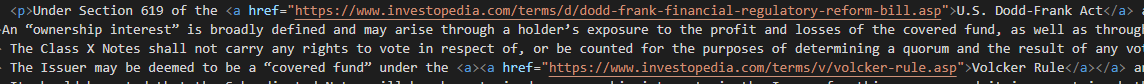


Figure 33 - External reference HTML code

In the end, we have an HTML file filled with internal and external references. For the sake of validating the concept, we created a small python script to process, recognize and connect the references. In this case, we did that separately, so first the external references, then the internal references.

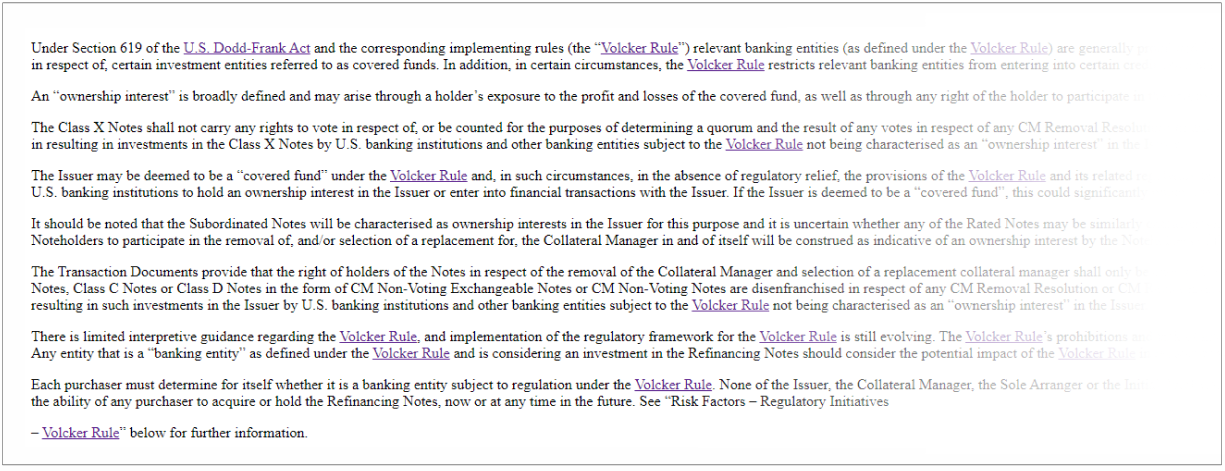


Figure 34 - Contract with linked references on an HTML page

We sliced a contract into a small chunk, went through the possible external references, and filled the dictionary with it. Then running the python script we see the results in the figure above.

In the case of internal references, we used a document that contained a few chapters and a few internal references in it. Then a python script read the document, and if it finds an internal reference, the whole document gets searched for that reference. If one of the occurrences exactly matches the internal reference (so no other word in that paragraph’s meaning it’s a title) then the row id of that paragraph gets saved into a dictionary, and later when the program reaches that line, it gets an id with the row number. The result is an HTML document containing the internal references.

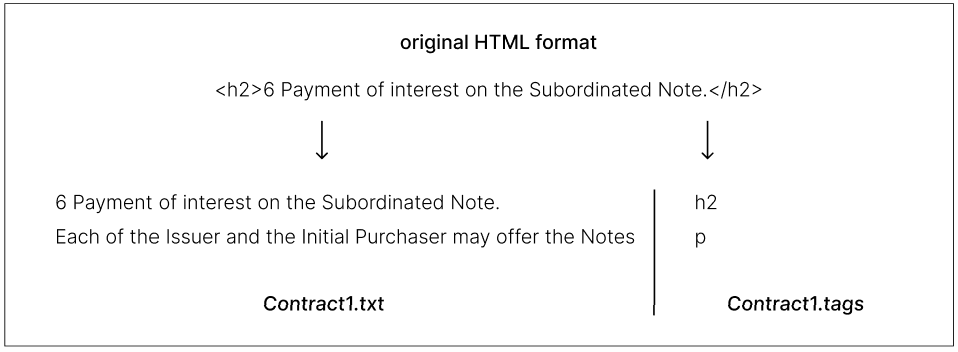
## Title detection

The legal domain generates a huge amount of information in the form of text and documents. The generated data are being stored, categorized, and are being searched. The detection of titles within documents improves the information process. It is also a preliminary task for a table of contents generation. This, and the title detection enrich the access to later searchable documents. Furthermore, it can also help other text mining tasks which rely on titles (as we can see in section 2.3.2, “Internal cross-reference identification”).

Recognizing titles is a binary classification problem that can be done with ease since a paragraph or text within a document is either a text or not. In the following section, we propose a method of detecting titles. We start from a simple rule-based search algorithm, refining the algorithm, which then ends in a binary classification problem.

### Preprocessing data

To detect titles within documents, we need to examine our dataset first. We have our **Legal-Contracts** dataset which contains the contracts. While it contains unformatted text files, every contract has another file attached. In these attached files, there are HTML tags per line, each tag represents the same line in the original document meaning what HTML tag the related line was parsed from.



Normally, the HTML tag determines the nature of the text enclosed. In the case of a title or a subtitle, usually, a heating element is used, and the corresponding tag is an “<h1>, <h2>,…” element, in the case of a paragraph, an <p> element. Unfortunately, parsing alone is not that simple. The difficulty of processing is that while we can indicate certain things in an HTML text using CSS (Cascading Style Sheets, used for formatting text in HTML) a tag alone does not necessarily carry enough information about the original use of the text, in this case, that title or paragraph because a <p> paragraph can also be a title if it is formatted properly. Furthermore, the text and the tags were in a different file, so first, we have to zip it into the same file (CSV). After zipping, we had to correct some tags manually since it was crucial for the evaluation and the training as well to have proper training labels. Fortunately, the original formatted HTML of the contracts was available, so it was possible to do that. After cleansing and preprocessing, we can start analyzing and developing algorithms.

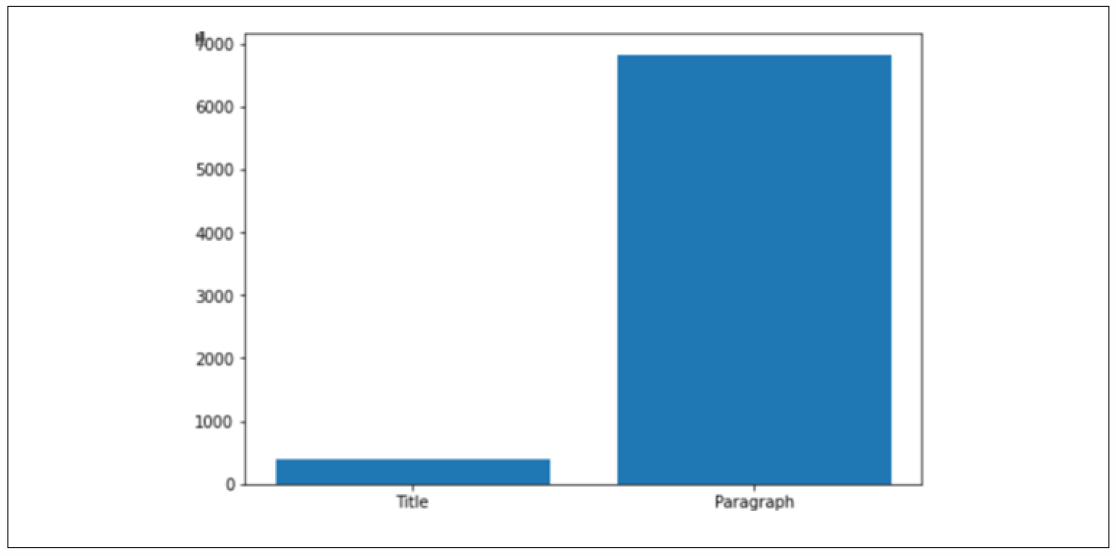


Figure 35 – Title-paragraph ratio in a document

In the figure above, we can examine the distribution of the titles and the paragraphs within a document. We note that the number of titles is less than the paragraphs.

### Rule-based algorithms

Before we start developing machine learning algorithms, we are experimenting with simpler ways to determine whether a text is a title or not. A rule-based text classification algorithms work using explicitly predefined linguistic rules. The algorithm uses these rules to determine which class a given text should belong to, for instance, it could look for topic-related words, a list of keywords, phrases, or other relevant patterns. Rule-based systems could be interpreted by humans and improved by manual intervention. However, these types of classification methods are hardly flexible and as they can only adhere to the predefined rules, they are hard to generalize.

After the data procession, we organize our data into a CSV, where one column is the text, and the other column is the related HTML tag. After reading texts programmatically, we process the data with a rule-based algorithm that classifies the texts into titles or paragraphs. This algorithm goes through every line, and it checks the following properties of the text during processing:

* The length of the text
* The number of the verbs in the text

If the above conditions are met, we can say, that the text is possibly a title. We got 39826 hit and 126289 miss out of the 211346 segment means a 59% probability. The next step is to make some improvements to our algorithm.

During the refining of the algorithm, we add the following:

* How the text starts (uppercase, lowercase, title case)
* Contains anything besides number (e.g., not date)
* Whether it starts with a number (chapter number) or not

This has already given us a slightly better result; we got 139505 hit and 71841 miss with a probability of 66%. The goal is to reach a probability of around 70%.

Last refining iteration:

* We check whether the text before and after the examined row is a paragraph according to our algorithm and if our extended conditions are met, then we are sure that the row is a title.

The final algorithm shows the following results: we got 39826 hits out of 141652 means a 0.7188 probability. It seems like we have achieved better results with these refinements. We can perform classification by a rule-based algorithm on a simple task like title detection, it is mostly because titles are much more recognizable and easier to distinguish from other parts of a text. In the next part, we are going to use classification for the same task and we are expecting to have a significantly better result.

### Classification

As we see in the previous section above, rule-based systems have limitations. However, in contrast, the classification models are based on machine learning algorithms that use algorithms to analyze data sets based on samples with a particular class.

After testing how the rule-based classification of the text can be solved, we also use machine learning algorithms to perform the original classification problem mentioned above. We are going to use Spacy for that. As training data, we use eight datasets from the ten contracts, and for testing purposes, we use the remaining two contracts. The training data needs a preprocessing to have the shape that Spacy expects. We assume, that every heading tag is a title, and the other tags are classified as a paragraph, “non-title”, therefore we divide our training dataset into two sets, “Title” and “Non-Title”. At this time, the data set is labeled, and we have a sufficient amount of training samples. Each document contains around 150 000 - 200 000 rows as training samples, so we have roughly 1 520 000 samples for testing, and 380 000 samples for testing.

After successful dataset preparation, we feed Spacy with the training data, add labels like “Title” and “Non-Title” and evaluate the result on the test dataset.

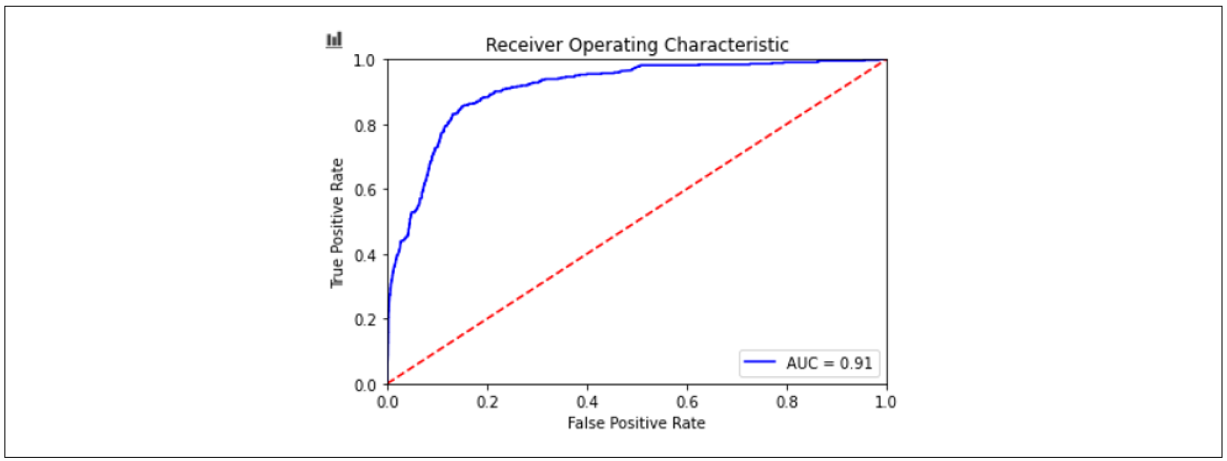


Figure 36 - ROC-AUC curve of the title detection model

Checking the AUC score we got 91 which is a much better score than our rule-based model. It means, that our model can classify text into titles and paragraphs with any major issues. The model uses a machine-learning algorithm (CNN) and with that, we are more flexible and accurate than a rule-based model. While we can add more and more rules to fine the rule-based model, it is not flexible and time-consuming. In contrast, a machine learning algorithm relies on the quality of the training set, it can identify complex relations within a text.

# Summary

In the 21st century, with the general spread of digitalization, more and more emphasis is being placed on business, administrative, and scientific data analysis. Their application is new perspectives in case law analysis can open several automating a current document management process and retrieving direct non-retrievable information. A properly executed text mining analysis is capable of such latent structures explored in the resolution texts, which are manually only a significant amount of manual labor.

In this paper, we explored the possibilities of processing, mining, and testing different experiments on an unstructured unlabeled dataset. We showed how easily one can make text mining experiments using an open-source tool Spacy. We made algorithms for labeling and preparing our dataset to use a machine learning algorithm. Furthermore, we showed a few real-life usages of analyzing legal documents and proposed frameworks for possible applications.

In *Section 2.2,* we extracted logical patterns in legal documents. First, we explored our dataset and constantly fine-tuned our dataset to have a better model. In this case, we used a classification model, and we tested our results on a different dataset as well, to check how our model performs in a different environment. Although our training dataset was quite small, we presented a promising result, we could see how the AUC score increased over the number of training samples. In the end, we proposed a plan for a real-life application which able to process and extract logical patterns in legal documents, also, to validate the concept, we created a working solution to demonstrate how the whole process looks from start to end.

In *Section 2.3*, we explored the cross-references. We start by identifying internal and external references across legislative documents. We compared the two approaches and validated them on different datasets. Then, we built a model which can recognize these references and link them with their correct source. This approach used named entity recognition to detect the reference entities.

In the last part of *Section 2*, we made proposals for detecting titles in a document. We showed the efficiency of a rule-based model, and what possibilities we have to detect titles without any machine learning algorithm. Then, we build a classification model to compare it with the manual algorithms. As we had around a 71% probability rate of detecting titles, we got an AUC score of 96% with the classification problem, which is a significant difference between the two approaches.

Biblioteuqe

1. Levendovszky, J., Jereb, L., Elek, Zs., Vesztergombi, Gy.: Adaptive statistical algorithms in network reliability analysis, Performance Evaluation - Elsevier, Vol. 48, 2002, pp. 225-236
2. National Istruments: LabVIEW grafikus fejlesztői környezet leírása, <http://www.ni.com/> (2010. nov.)
3. Fowler, M.: UML Distilled, 3rd edition, ISBN 0-321-19368-7, Addison-Wesley, 2004
4. Wikipedia: Evaluation strategy, <http://en.wikipedia.org/wiki/Evaluation_strategy> (revision 18:11, 31 July 2012)