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](#_Toc102951054)

[Abstract 6](#_Toc102951055)

[1 Introduction 7](#_Toc102951056)

[1.1 Motivation 7](#_Toc102951057)

[1.2 Applied tools 7](#_Toc102951058)

[1.3 Structure 9](#_Toc102951059)

[2 Text mining in legal documents 10](#_Toc102951060)

[2.1 Dataset 10](#_Toc102951061)

[2.2 Logical expression extraction 11](#_Toc102951062)

[2.2.2 Nature of conditional 12](#_Toc102951063)

[2.2.3 Detection of logical patterns 12](#_Toc102951064)

[2.2.4 Extract cause-effect 18](#_Toc102951065)

[2.2.5 Building flowchart 20](#_Toc102951066)

[2.3 Cross-references 22](#_Toc102951067)

[2.3.1 External cross-reference identification 23](#_Toc102951068)

[2.3.2 Internal cross-reference identification 26](#_Toc102951069)

[2.3.3 Cross-reference detection framework 26](#_Toc102951070)

[2.4 Title detection 28](#_Toc102951071)

[2.4.1 Preprocessing data 28](#_Toc102951072)

[2.4.2 Manual algorithms 30](#_Toc102951073)

[2.4.3 Classification 31](#_Toc102951074)

[3 Summary 33](#_Toc102951075)

[Irodalomjegyzék 34](#_Toc102951076)

Hallgatói nyilatkozat

Alulírott **Rezeda Kázmér**, 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. 09.

...…………………………………………….

Rezeda Kázmér

Összefoglaló

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

Abstract

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

# Introduction

## Motivation

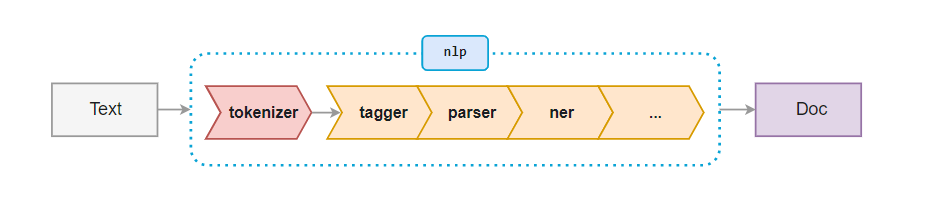
In legal reasoning and decision-making, the content of the contracts, ruling, treaties or legislative acts plays a critical role. Whether it is a legal dispute to undermine the other or is a decision-making based on different conditions, the extraction, analyzation, relating and commenting of the proper information crucial for the everyday work of a legal professional. Day by day, lawyers, judges deal with a vast and complex network of interrelated texts, and 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 machine help either. More recently, with electronic documentation, legal professional can process the provisions, search in the case bases quickly. There are companies, who provides 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 a careful human reading.

Tools like text mining 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 case base, facilitate in decision making and legal reasoning or for automatic identification of legal arguments. Research in the fields of information extraction, natural language processing, artificial intelligence and expert system has augmented text mining process for enhancing the knowledge discovery process in this domain.

## Applied tools

By applying advanced analytical techniques, and other deep learning algorithms, we are able to explore and discover hidden relationships within their unstructured data. We need a tool in order 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 trained pipeline components like “textcat”, “ner”, “sentencizer”, “word2vec”, etc. for a variety of languages, with them we can perform machine learning algorithms. In order to use a pipeline, we need to install the language package first. These packages include language-dependent components like lexical entries, binary weights, word vectors which later used by the pipeline. In our research we are using text classification (“textcat” pipe) and named entity recognition (“ner” pipe). Each component consumes different format of training data and needs a different configuration. Spacy provides us an user firendly and end-to-end approaches to train the different pipelines, so the user do not have to deal directly with the underlying neural network architecture.



To perform a classification, we have **TextCategorizer** as an optional and trainable pipeline component. In order to train it, we need to provide examples and their class labels. Later, the categorized text could be find in the “*doc.cats*” property. Since its 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 called “**EntityRecognizer**”, which is 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 labelled entities could be find in the “*doc.ents*” property.

In the following research, we often referring 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 try to visualize the results to have a comprehensive picture on the real-life usage. In section 2.3, we investigate the issue of the 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 which 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 about 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, and its use and use in applied research is still in its infancy. Few studies of this kind have been written so far, so we have a great freedom in terms of choice.

## Dataset

The given data set used for the following experiments consists of 10 legal contracts in English. These contracts were originally embedded in HTML pages, from which the contracts 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

Important to note, that our data set is unlabeled, and we do not have any labeled data during our study. Therefore, each task is preceded by a data processing and labeling process, where we label our data set according to the given task.

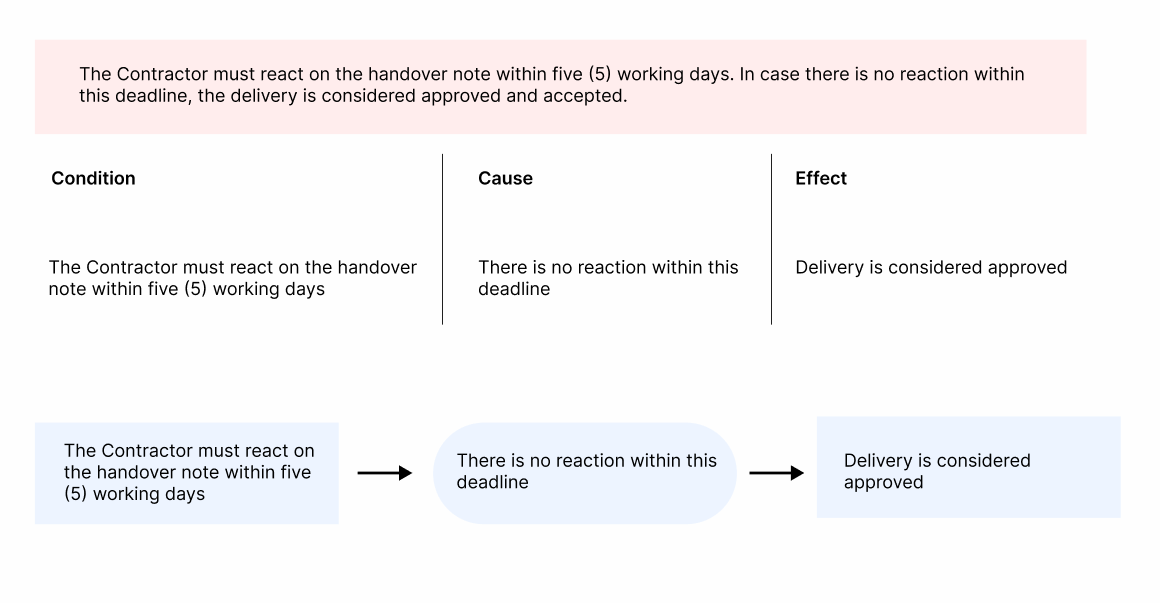
To get a more comprehensive picture and see how our results perform in other environments, we have another 10 different contracts at our disposal. The source of these is the publicly available EUR-lex webpage (https://eur-lex.europa.eu/homepage.html?locale=hu), which is a collection for EU legal texts. We note that these contracts are plain unformatted texts, and as our initial dataset, these contracts are also required pre-processing and labelling.

In the following chapters, the initial dataset is referenced as “Initial dataset” and the contracts from the EUR-lex are referenced as “EUR-dataset”.

\*TODO: valamit írt a dataelőfeldolgozó szakmáról

## Logical expression extraction

Conditionals describe the result of a certain condition. These sentences are statements of an “if-then”, “unless-then” situation (although “then is not used”), but other keyword such as “when”, “where” or “in cases” are can be found in these kinds of sentences either. In legal documents, especially contracts, often contain parts where certain paragraphs come into force 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 document to help the readers better and quicker understanding the particular paragraph. In this study, we conduct an experiment in which we try to collect the conditional sentences found in the legal documents, then break them down into cause and effect and display them in a flowchart-like manner.



A képen szöveg látható

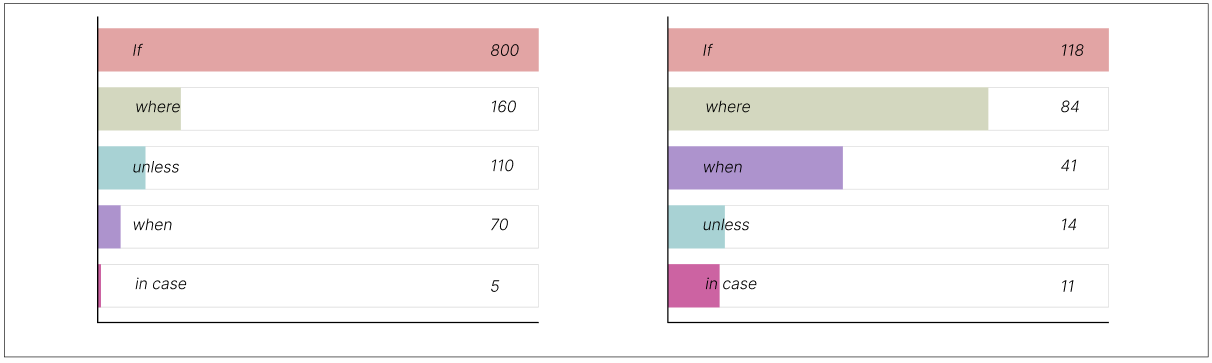
Automatikusan generált leírás

### Nature of conditional

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 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 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”. There are also many other connectives which could introduce conditionals, e.g., “in case”, “unless”, “when”, “assuming”, “where”. The table (w) shows the distribution of these keywords among our datasets.



The left chart shows the results from the initial dataset, and on the right, we see the results from the EUR-lex dataset. In both sets, the vast majority of cases the keyword “if” occurred most frequently and the distribution of the remaining keyword are similar. We note that the documents from the EUR-lex dataset are shorter than the documents in the initial dataset.

### Detection of logical patterns

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

*A képen asztal látható

Automatikusan generált leírás*

The Figure above shows us two examples for these cases. Both sentences contain a conditional keyword, although the meaning from 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

#### 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 sentences tag such sentences which contain this logical structure. We have two classes, conditional and conditional, we divide the sentences into two groups accordingly and we hope that the model we have built will be able to differentiate between them. Once we are able to identify the logical pattern within a document, further processing can be done with other text mining tools to distinguish the cause-effect relation.

Spacy’s classification model works by giving it a properly structured set of teachings 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”, “non-conditional”), we can start teaching the model. I used a model which architecture 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

Since it is a classification task, we need conditional and non-conditional sentences as a 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 40 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:

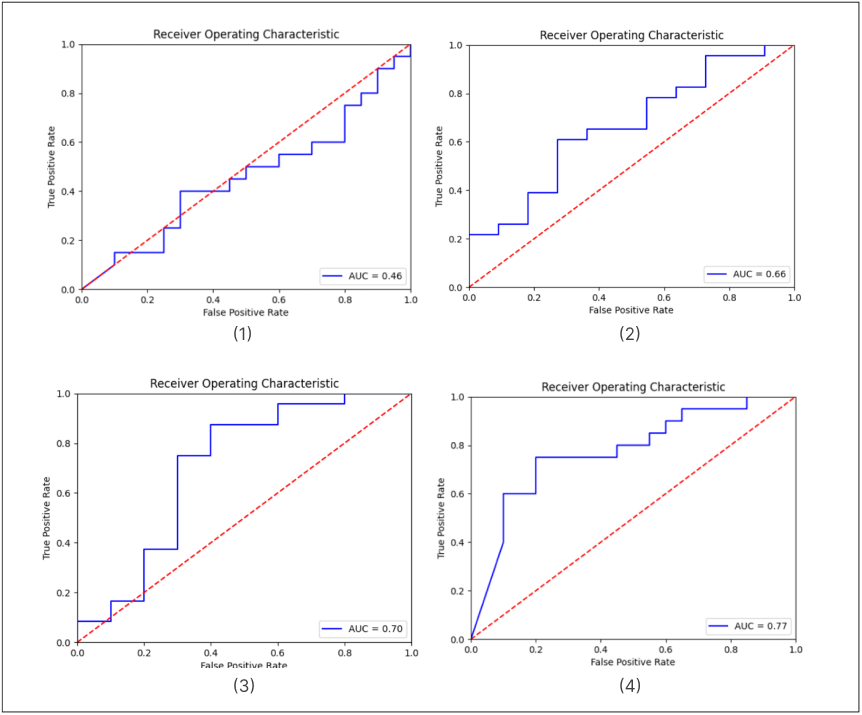
* Train the model with x examples
* Validate the model on a validation set
* Look for anomalies, manually add these examples to the train set with a correct tag
* Retrain a model with the anomalies

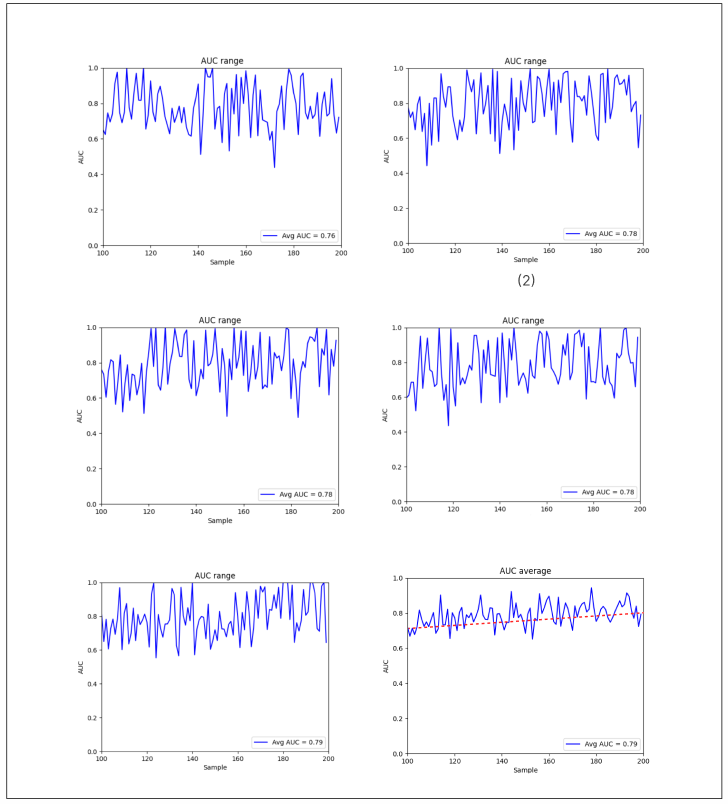
\*TODO reinforcement learning hasonlóságok

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

#### Results

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



From the results, we can assume that as the teacher patterns grow, so does the outcome. With this small amount of train and test set, we cannot state this with absolute certainty, so for this we should also see the result between the samples. We again training our model, but this time we are incrementing the training set one by one, and after each round we are measuring the AUC score. We do this five times to 

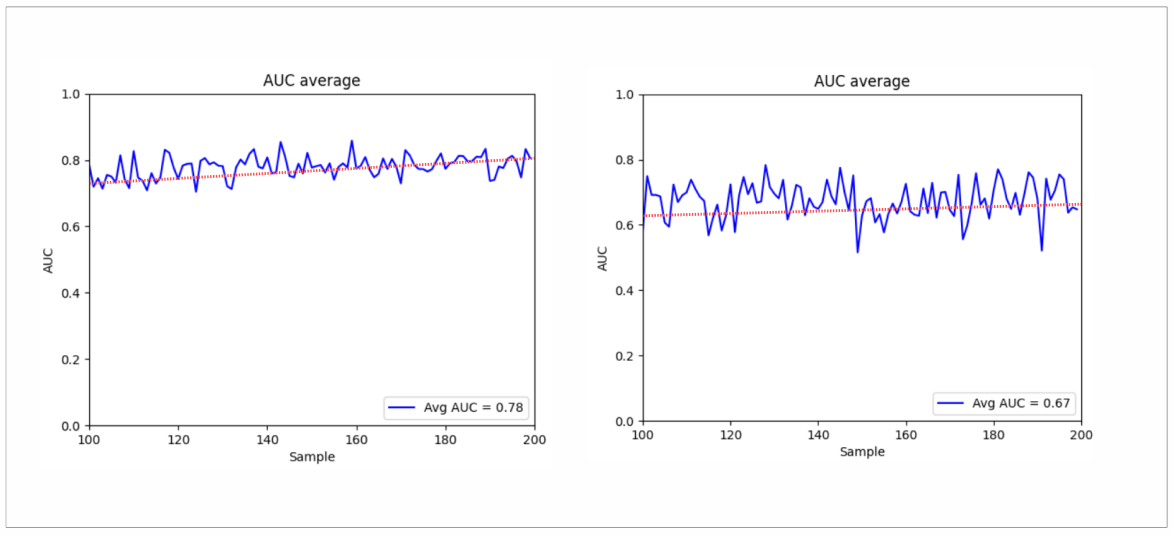
The last figure shows how the average AUC score is increasing over the number of samples. We could come to a conclusion, that increasing the number of samples could result to a better classification model, therefore it could recognize the semantic nature of the conditional sentences.

Lets see, how the EUR dataset performs in the classification.

A képen szöveg látható

Automatikusan generált leírás

In this case the average score shows a bit increasing but much smaller than the previous model with the initial dataset. 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 an initial training set on the EUR dataset and vice versa.



The first figure shows how the initial dataset performs on the EUR test set. We can see the increasing in the AUC score as clear as we see on (figuewx), so increasing the number of samples results in better scores. However, in the next figure we see how the EUR dataset performs on the initial dataset, and the result is not as clear as we have on the first figure. The increasing in the AUC score is present as well, but not as much as on the first figure, rather it is similar on figure (x) when it was tested on the EUR dataset.

Overall, the initial dataset had a better grasp of the sentences, performing better on both its own and the other dataset. The increase was much smaller in both cases than in the other dataset.

### Extract cause-effect

We assume that our model above can successfully decide on a sentence whether it is conditional or not with an appropriate number of samples. In order to get the exact logical relation from the sentences, we need another approach to successfully build a flowchart.

#### Named entity recognition

The approach for the above-mentioned problem is a named entity recognition, where we tag the cause and the effects in a conditional sentence. 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 which contains the condition, and “Effect”, the part which applies if the condition is met.

A képen szöveg látható

Automatikusan generált leírás

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

A képen szöveg látható

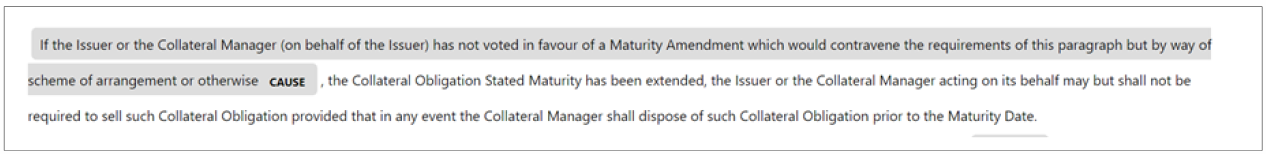
Automatikusan generált leírás

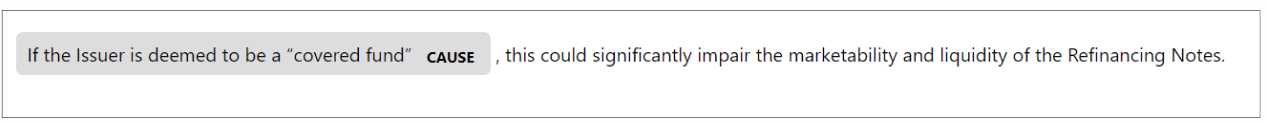
The training and validation method is the same as we can see in Section 2.2.2, so training, validation, and retraining the model with new examples. In the first round, thirty examples were collected as the training set, and validated over a validation sample. Evaluating the results, we see that there were cases, where our model could successfully recognize the condition-cause relation:

A képen szöveg látható

Automatikusan generált leírás

Of course, there were plenty of misrecognized pattern as we can see in the figure. However, after looking forward, we find an interesting pattern, namely, that our model barely can detect the “effect” part of the sentences. Usually, it finds the condition, but could not be able to match the “effect” part.

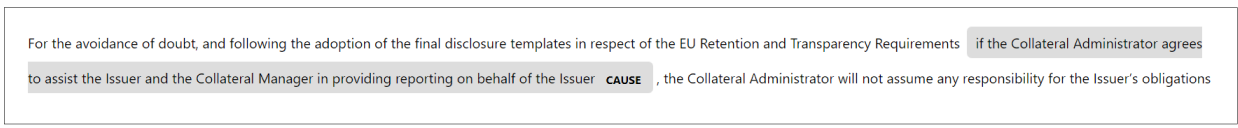




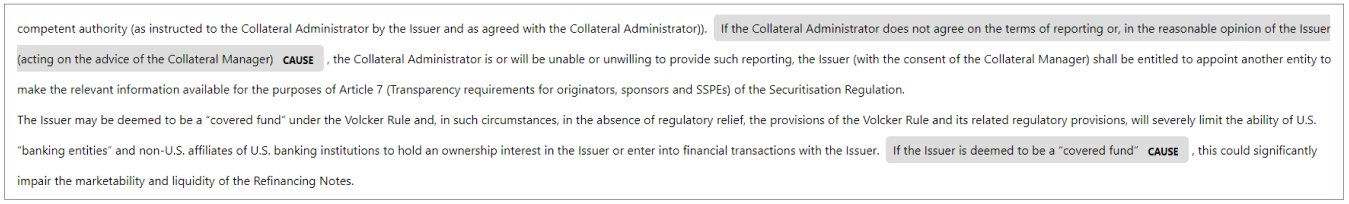
A képen asztal látható

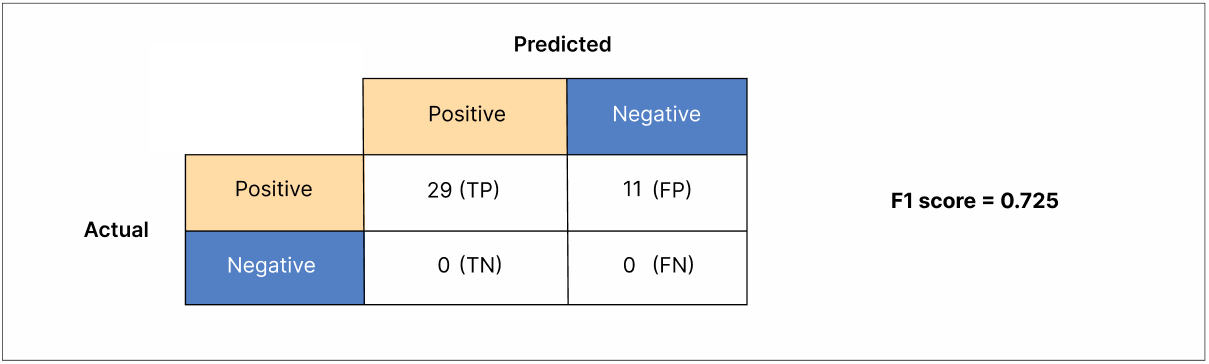
Automatikusan generált leírás

We retrain the model with similar examples as the missed or wrongly recognized entities, collecting around an additional 20 examples, and checking the results again.



Our model getting better and better 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 a bit. Since the condition part has been identified with an approximately success, and we only have conditional sentences, the other part of the sentence which has not been recognized as the effect, in fact, that leaves the effect part, so we do not need to recognize them in the first place. With a slightly changes on the training dataset (removing the “effect” tags from the entities objects), and training the model, we got the result as we expected. The model was able to identify the condition part of the sentences with a high success rate. We have a test set containing 40 example conditional sentences. From these 40 samples, our model detected 29/40 condition and no false matching, only true positive matchings. We got F1 score of 0.725 which considered a good score.

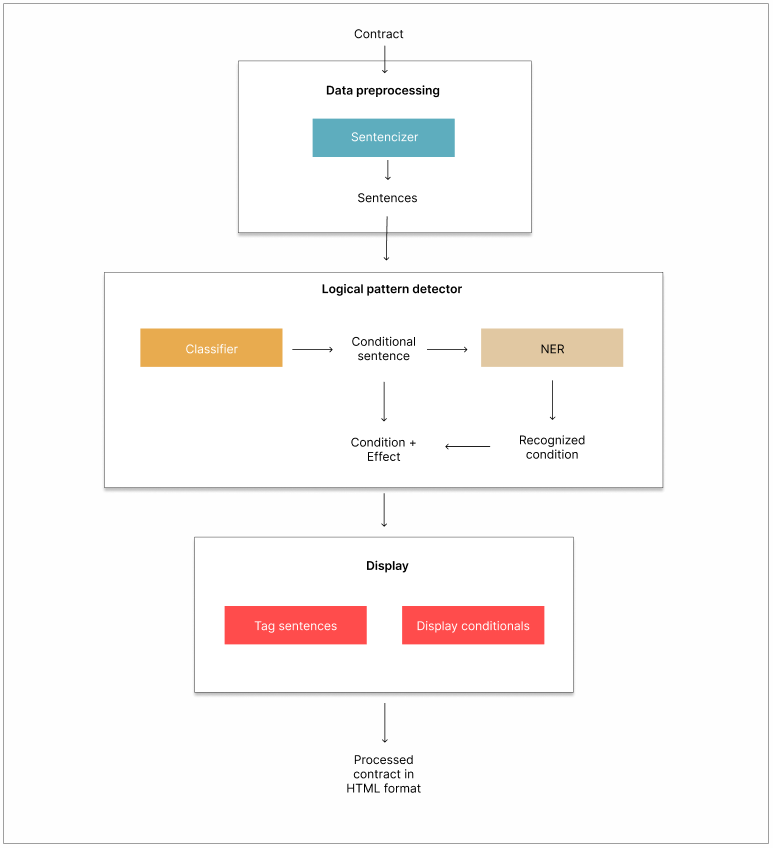




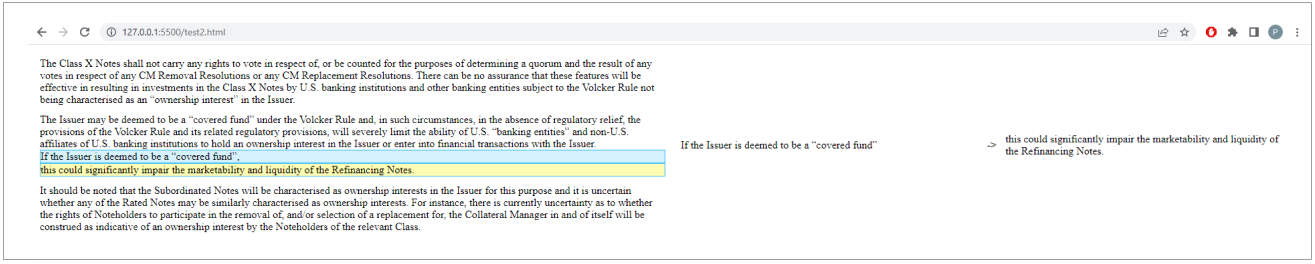
In the next round we add some In the 50-example case, TODO eredmények + 100??

### Building flowchart

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



The picture above shows the blockdiagram 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 it to a JSON file. An HTML webpage then loads the JSON file and the contract, it connects the paragraphs and the conditionals and displays it the user next to each other.



The whole is a 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), we highlight the condition-effect in the text and in the other column the relation gets displayed. This column also contains a table where we place the condition and the effect.

## Cross-references

In a contract, a provision often refers to clause in another contract or in 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

There are some recommendations to keep the number of the cross-references to a minimum, since it improves the readability and it helps better understanding the provision on its own without having to turn or look up other contracts, treaties. Not to mention, a high number of cross-references increases the chances of dead provisions in contracts. Writers often relies 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.

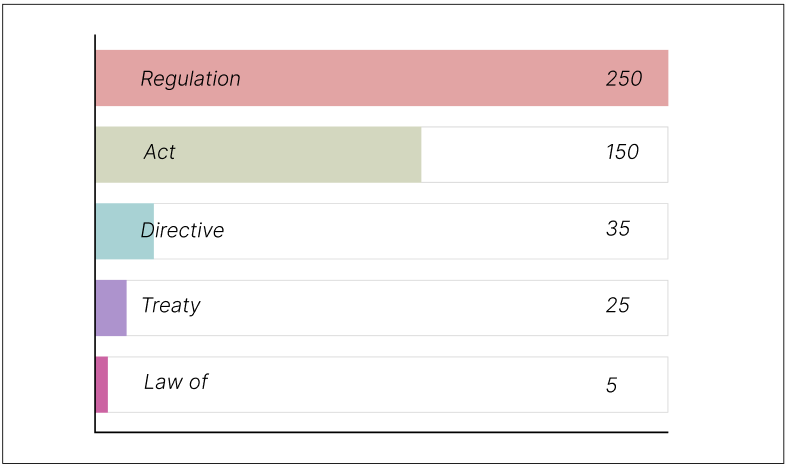
Checking cross-references and schedules is an extremely important task which contract drafters must, but often do not, undertake. There are some real-life examples where wrongly addressed cross-references led to lost cases or severe fines. In the following study, we are going to study cross-reference in contracts, try to identify cross-references and connect the provisions with these references. Then I propose to build a framework that could automatically perform cross-reference recognition in legal texts. There are 10 different contracts available for us as before, furthermore, I gathered 10 different contracts from EUR-lex, the official website of European Union law and other public documents of the EU.

### 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 a various shape, 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:

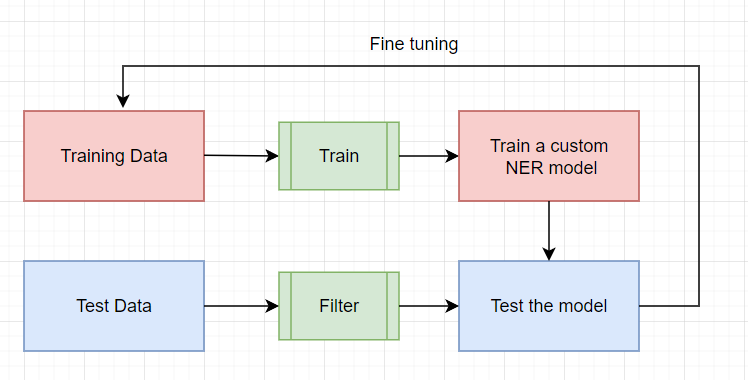
**

The references often consist of a prefix, which defines the specific paragraph within 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 10 documents.



The identification of these entities (as the wording suggests) 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, to validate our solution for a real-life situation, we try to link the references to their external source, build a reusable dictionary and create a HTML document with the links.

#### Model building



The model building consists of two phases. In the first phase, where the training is happening, we need to gather sufficient examples as training dataset to train a custom NER model with the help of Spacy. Then, in the second phase, the evaluation and retraining where first we construct a test dataset only for to retrain our model. For this, I used a whole contract itself, then retrained the model with such examples what we missed identify or incorrectly recognized.

In the first round of training our model, I used 30 training examples. The structure of the training dataset follows the same pattern as we see in Section 2.1. I, labeled these entities as “External-reference”.A képen szöveg látható

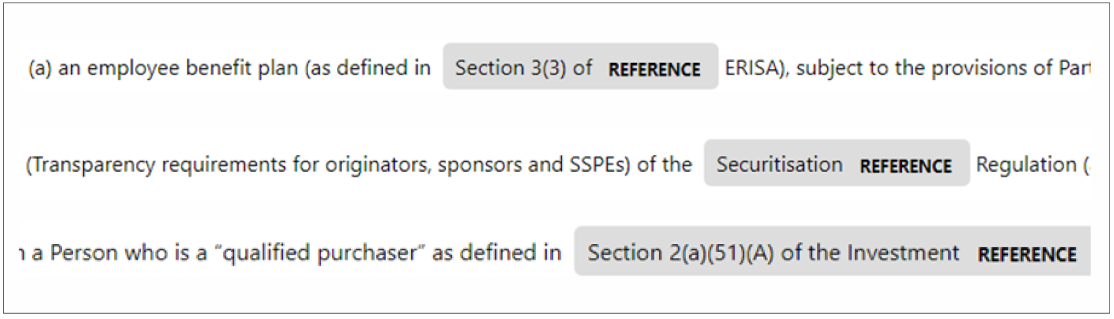
Automatikusan generált leírás

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

A képen szöveg látható

Automatikusan generált leírás

The result after the first round is promising. Our model was capable to recognize several external-references even though the size of the training dataset. There were some mistakes which the input to the next training round would be, for instance:



So, in the second we are going to add these erroneous results to the training dataset to fine tune our model. We do this in each round, and as a result we got a model that recognizes an external reference with high accuracy.

### Internal cross-reference identification

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

* See Condition 7 (Redemption and Purchase).
* in Condition 10(a)(i) (Non-payment of interest)
* Any of the following events in this paragraph (a)(i) to (a)(viii) shall constitute an “Event of Default”:

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.

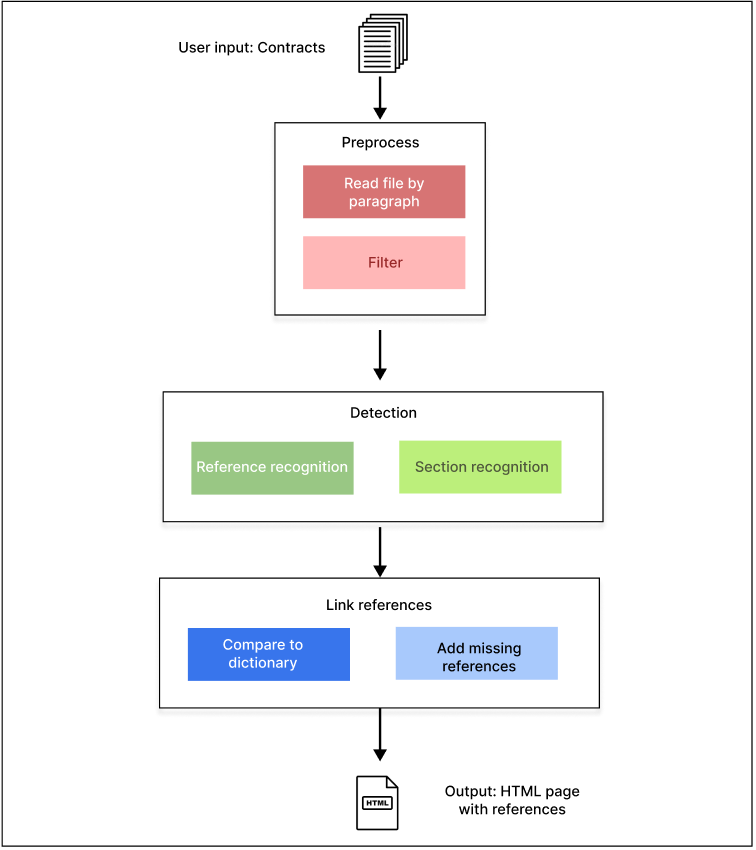
#### Model building

As the external reference identification, the internal reference identification is a NER task either. For constructing the training data dataset, we are creating an example set of 50 samples. Collecting the right example method is the same as we have seen in the external reference. We expect to see a similar result as in the previous experiment.

Here, we have test set of 40 example sentence. Checking the results after the last iteration, we got a quite promising result, the success rate is TODO.

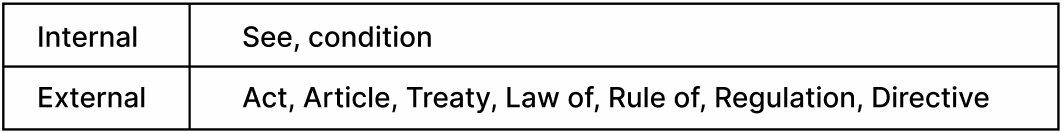
### Cross-reference detection framework

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 a user input, could recognizes 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.



In Figure (x+1) 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 try to 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.

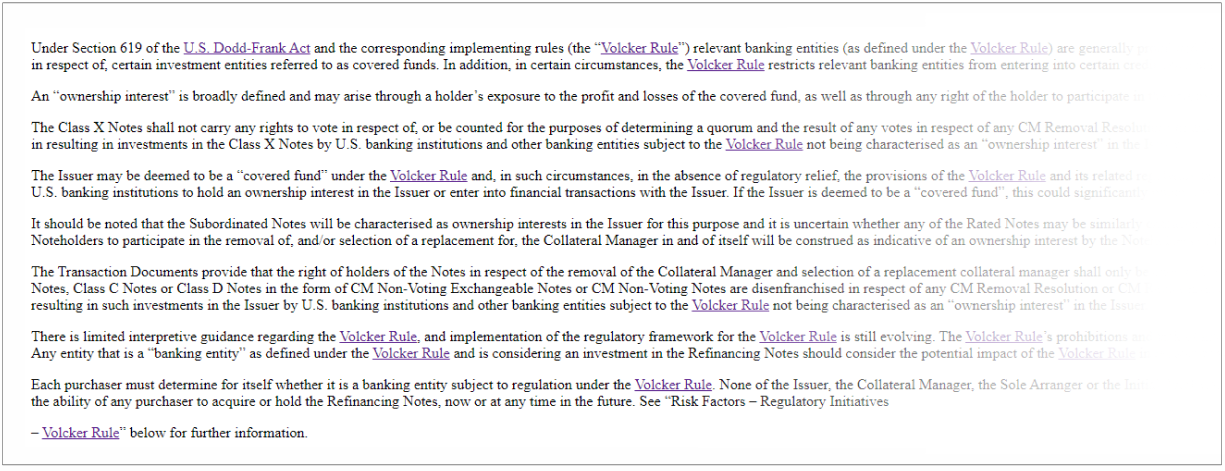


During the whole process, we build the output file parallel. When we have paragraph, which contains one of the trigger words, we feed our pretrained ner model to identify the entities within that paragraphs. First the whole reference recognition, then the second model detect the section within a reference. 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 contains 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 an <p> tag whie the identified internal reference got an <a> tag with a href attribute which points to the row id. The row id also gets saved when we reach that rownumber in the document, that paragraph will got an id with the riwnumber in it.

Other case is, when we find an external reference. For external references, we do not have the source which we need to link for. So we have two options here: one is to manually add the corresponding links to the references, and the other is to have a dictionary with predefined link-reference connection. When we find an external reference entity, first we go through in 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 an <a> link tag. But if we do not find, 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.

In the end, we have an HTML file filled with internal and external references. In 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.



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 on the figure above.

In case of internal references, we used a document which contained a few chapter and few internal references in it. Then a python script read the document, and if it finds an internal reference, the whole document get searched for that reference. If one of the occurrences exactly matches the internal reference (so no other word in that paragrahs 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 get an id with the rownumber. The result is an HTML document containing the internal references.

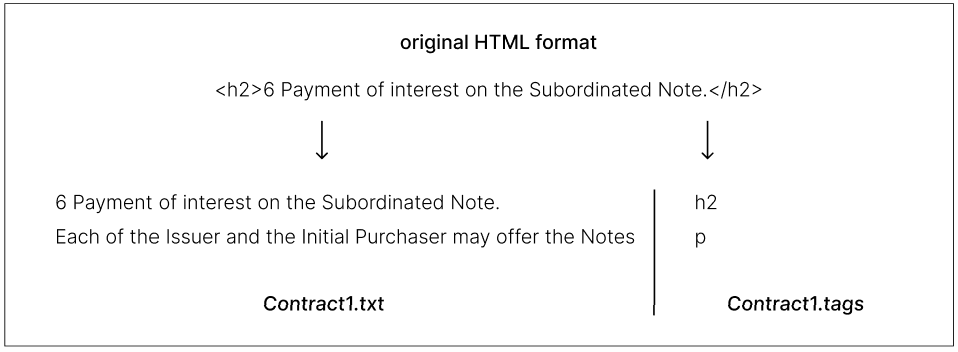
## Title detection

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 title within documents improves the information process. It is also a preliminary task for 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”).

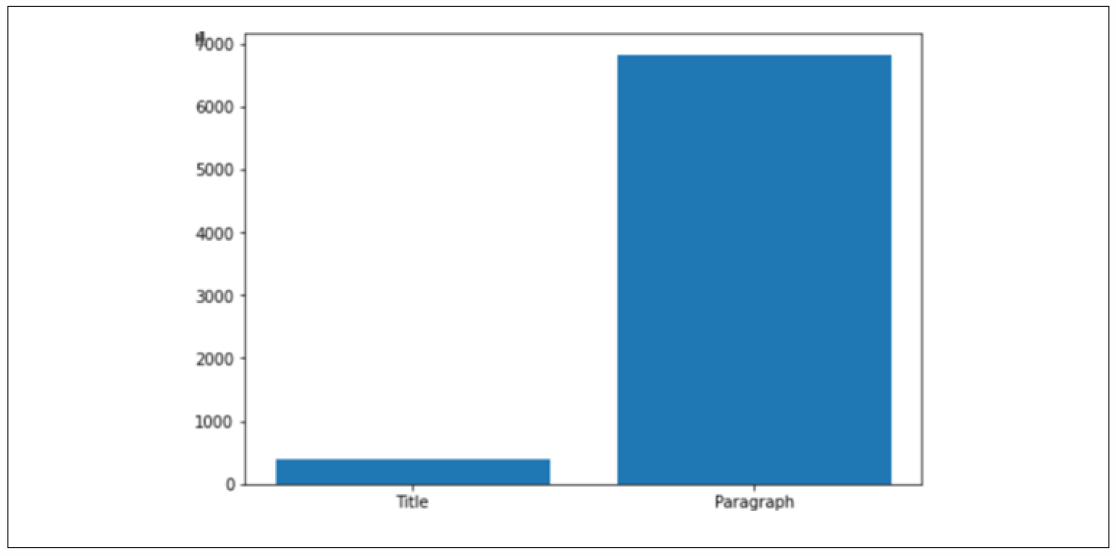
Recognizing titles are a binary classification problem that can be done with ease since a paragraph or text within document is either a text or if not. In the following section, we propose a method of detecting titles. We start from a simple rule-based search algorithm, refining the algorithm, which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

### Preprocessing data

In order to detect titles within documents, we need to examine our dataset first. We have our initial 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 lines, 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 by. In case of title or a subtitle, usually a heading element is used, and the corresponding tag is an “<h1>, <h2>,…” element, in case of a paragraph, a <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, use 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 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.



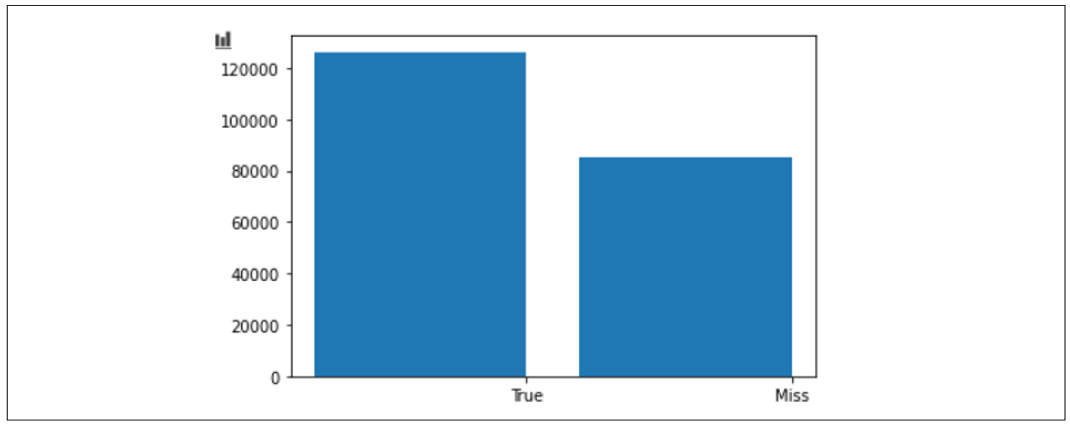
On the figure above, we can examine the distribution of the titles and the paragraphs within a document. We note that the number of titles are fairly less than the paragraphs.

### Manual algorithms

Before we start developing machine learning algorithms, first we are experimenting more simpler ways to determine a text is a title or not. After we have organized our data into a CSV, where one column is the text, the other column is the related HTML tag. After reading texts programmatically, we process the data with a manual algorithm which 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.

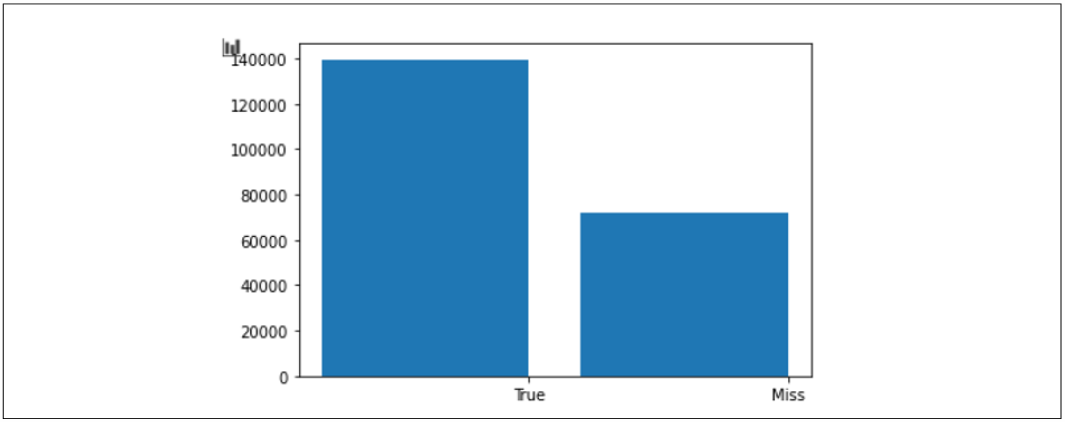


We got 39826 hit and 126289 miss out of 211346 segment means a 59% probability. Next step is to make some improvements in our algorithm.

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:

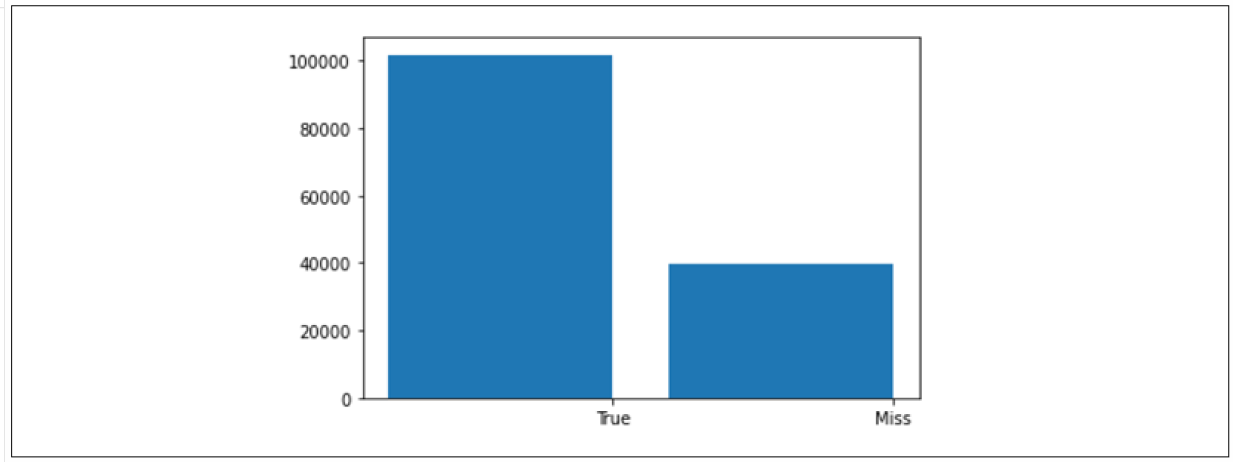


We got 139505 hit and 71841 miss with a probability of 66%. The goal is to reach a probability around 70%.

Last refining iteration:

* We check whether the text before and after the examined row is 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:

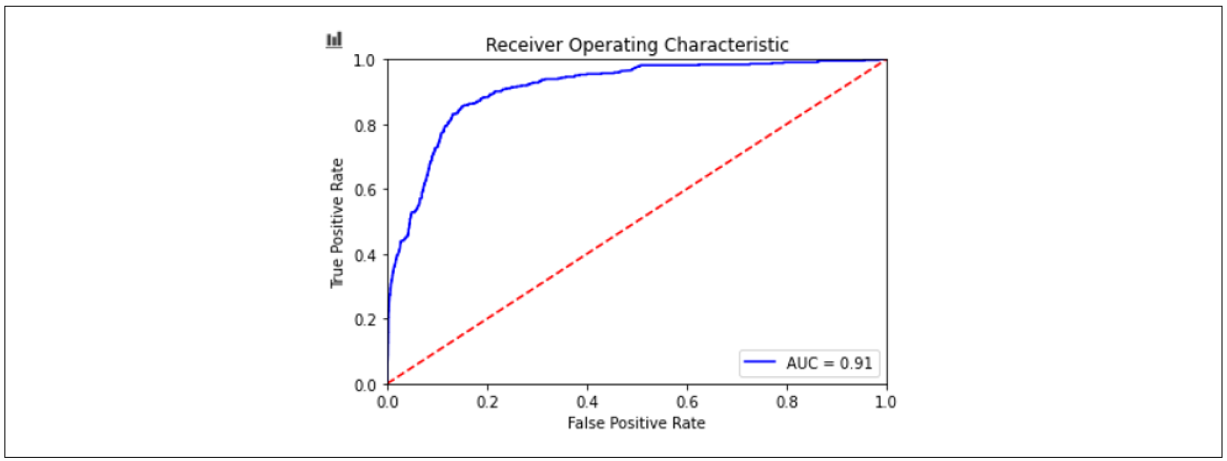


We got 39826 hit out of 141652 means a 0.7188 probability. It seems like we have achieved better results with these small refinements.

### Classification

After testing how manually the classification of the text can be solved, we also try to use machine learning algorithms to perform the original classification problem mentioned above, and we use Spacy again for that. As training data, we use 8 datasets from the 10 contracts, and for testing purpose we use the remaining two contracts. The training data needs a bit preprocessing to have the shape which Spacy expects. We assume, that every heading tag is a title, and the other tags are classified as paragraph, “non-title”.

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



Checking the AUC score we got 91 which seems like a much better score than our manually developed model. It means, that our model can classify text into title and paragraph with a strong confident.

# Summary

In this paper, we explored the possibilities of processing, text mining and testing our approaches 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 labelling 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 tried extracting logical patterns in legal documents. First, we explored our dataset, constantly prepared our dataset in order to fine our model. In this case, we used a classification model, and we tried to We compared our results with a different dataset as well, to check our solutions validity. Although our training dataset was quite small, we presented a promising result, we could see how the AUC score increasing through the number of training samples. 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 also created a small working solution.

In Section 2.3, we explored the cross-references, identified internal and external references across legislative documents. We tried to build a model which can recognize these references and link them with their correct source. This approach used named entity recognition, and

In the last part of Section 2, we made proposals for detecting titles in a document. We showed first, without any machine learinig alfgorithm what possibilities do we have to detect titles. Then, we build a classification model to compare it with the manual algorithms. As we had around a 80% rate detecting titles, we got an AUC score of 96% with the classification problem, which is a significant different in the two approaches.

Irodalomjegyzék

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)