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

Tartalomjegyzék

[Összefoglaló 6](#_Toc59896130)

[Abstract 7](#_Toc59896131)

[1 Bevezetés 8](#_Toc59896132)

[1.1 Formázási tudnivalók 8](#_Toc59896133)

[1.1.1 Címsorok 8](#_Toc59896134)

[1.1.2 Képek 8](#_Toc59896135)

[1.1.3 Kódrészletek 8](#_Toc59896136)

[1.1.4 Irodalomjegyzék 8](#_Toc59896137)

[2 Utolsó simítások 10](#_Toc59896138)

[Irodalomjegyzék 11](#_Toc59896139)

[Függelék 12](#_Toc59896140)

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. 02.

...…………………………………………….

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

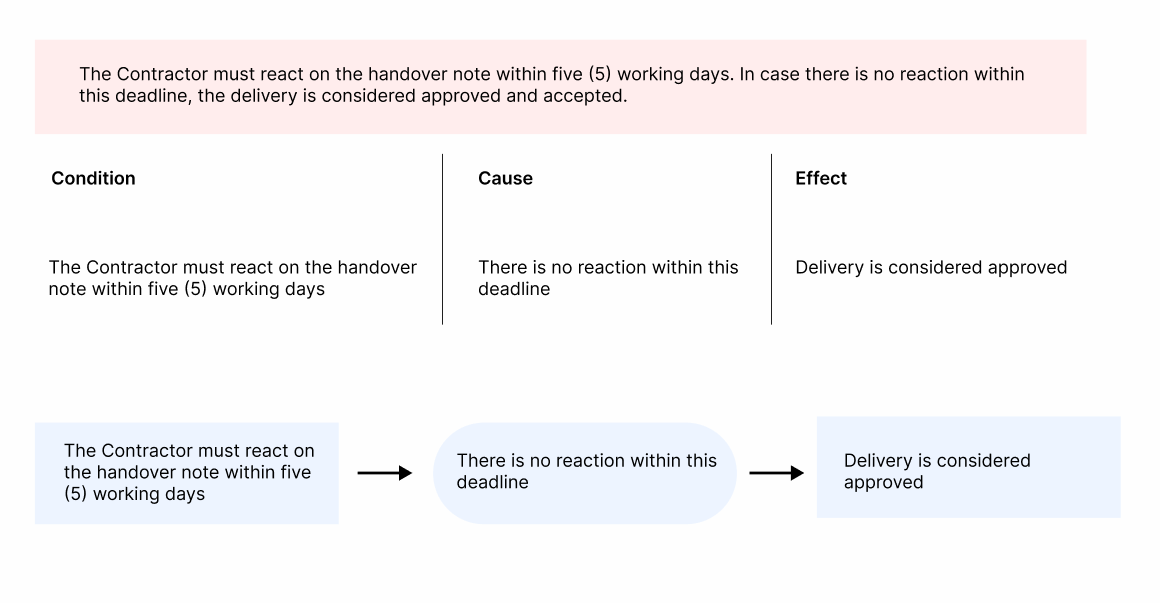
## Motivation

## Tools

# Text mining in legal documents

## 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.

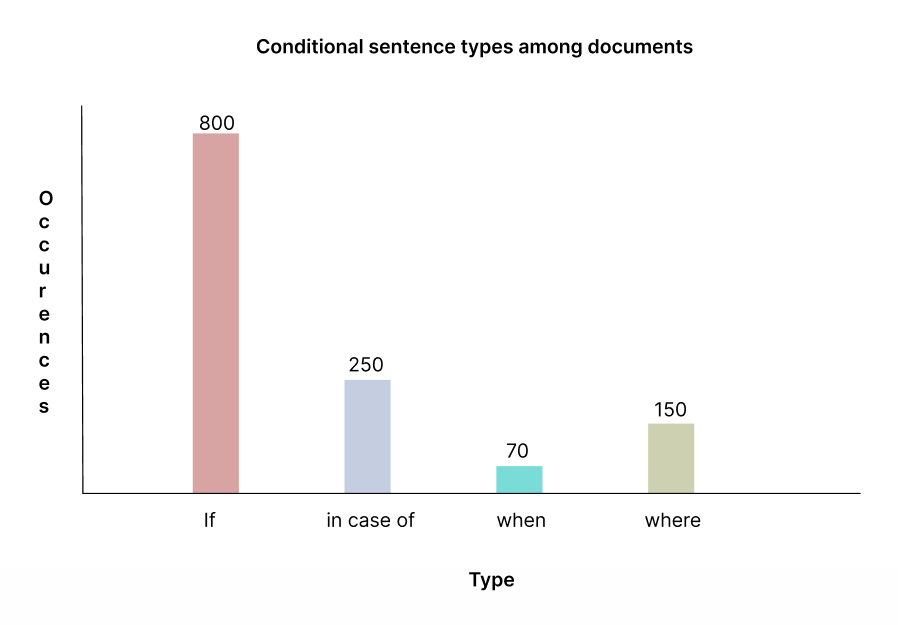


### Nature of the sentences

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.

The dataset, which I used in my following experiments, consists of 10 legal contracts. As previously mentioned, they come in a plain unformatted .txt format. In these documents, we can find examples of all above mentioned keywords. Examining the distribution of the conditional sentences among the documents, we find a very similar amount of each type.

Sentences with “if” ~ 800 occurrences, “in case of” or similar is around 200-250 occurrences. The “when” keyword usually listed around 70 occurrences, while conditional sentences with the keyword “where” is around 150 occurrences.



The most commonly used word in conditional sentences if “if”, in the studies I mostly focus on that kind of sentences.

### Detection

The question immediately arises, why not using some basic search pattern to identify the conditional sentences. It would be enough just to search for a few keywords with the help of regex (we identified above) and we could filter these sentences. However, in practice, it’s not just that simple. There are a plenty of cases, where filter sentences with the keywords results in such sentences, that are not conditionals, but they contain a keyword, as the following example shows:

*“If you are in any doubt about the contents of this Prospectus you should consult your stockbroker, bank manager, solicitor, account or other financial adviser.”*

Even if the sentence itself a conditional sentence, and contains the keyword “if”, we would not think of it as a legal condition. Below there is an example of a “good” example for conditional sentence.

*“If the Issuer is deemed to be a "covered fund", this could significantly impair the marketability and liquidity of the Notes.”*

So, in order to succeed we need to be cautious not to include this kind of sentences into our dataset. Other issue could be that often the conditions are in the previous/following sentence and are separated from the effect.

*“The Dodd-Frank Act requires that federal banking agencies amend their regulations to remove reference to or reliance on credit agency ratings, including but not limited to those found in the federal banking agencies’ risk-based capital regulations. New regulations have been proposed but have not yet been fully implemented in all respects. When such regulations are fully implemented, investments in asset-backed.”*

The first sentence declares the main condition, however in the next sentence which starts with an “if” refers to the first sentence as the condition and the cause in the second sentence. In this case, we need to tag the two sentences as “conditional” part of the text.

#### Classification

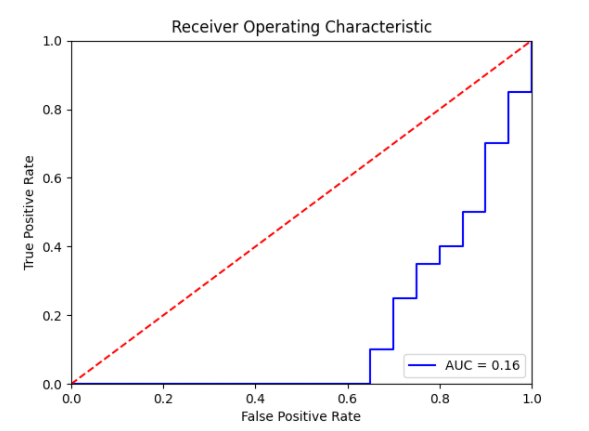
Most, hogy van egy kisebb képünk a rendelkezésre álló adatokról és a kinyerni való struktúra természetéről, már cask le kell fordítanunk gép nyelvre. Első körben szemantikai analízist próbálunk végezni, melyben azokat a paragrafusokat próbáljuk megjelölni, amelyek tartalmaznak feltételes/logikai szerkezeteket. Azért paragrafusokat vizsgálunjk, mert mint fentebb is láthattuk, hogy többször több mondatot ölel fel egy-egy logikai rész. Ezt a feladatot klasszifikáció segítségével fogjuk végezni, megpróbáljuka a modellünket úgy betanítani, hogy a feltételes/logiaki struktúrát természetére próbáljanak rátanulni. Miután azonsoítani tudtuk, hogy igen, ebben a szövegrészben logikai strultúra van, a további feldolgozást már egyéb text mining eszközökkel folytatjuk, melynek célja az less, hogy a folyamatábránkat fel tudjuk építeni

A modellünk sikerességét nagyon nagybani fogja befolyásolni egy jó training data set. 10 szerződés áll rendelkezésünkre, formázatlan txt formátumban. Ezekből 9 szerződésből fogok válogatni training adathoz, validáláshoz pedig 1 szerződést fogok használni. Hogyan kell kinéznie a training adatnak, ahhoz, hogy haszánlható eredményt kapjunk? Fentebb láthattuk, hogy nagyon nagyrészt if mondatokat tartalmaznak a szerződések, sok in case és when is található, szóval a training dataset-et is próbáljuk ehhez araányosan felépíteni.Ezeken kívül természetesen olyan mondatok is kellenek a training halmazba, amelyek nem tartalmaznak feltételes/logikai struktúrákat.

Klasszifikációt a Spacy nevű text mining tool TextCategorizer komponensével végezzük. Ehhez először is a spacy által használt “textcat” (TextCategorizer) nevű pipe-ot létre kell hoznunk. Miután létrehoztuk, címkéket is létre kell hoznunk. Jelen esetünkben két címke less, “conditional” és “nonconditional”. Utána már cask elég a training adatok segítségével “trainelnünk” a text cat pipe-ot, majd kiértékelni az eredményt. Első körben 10-10, 50-50, majd 100-100 pédát gyűjtöttem ki “conditionalre” és nonconditionlre.

#### Train and test set

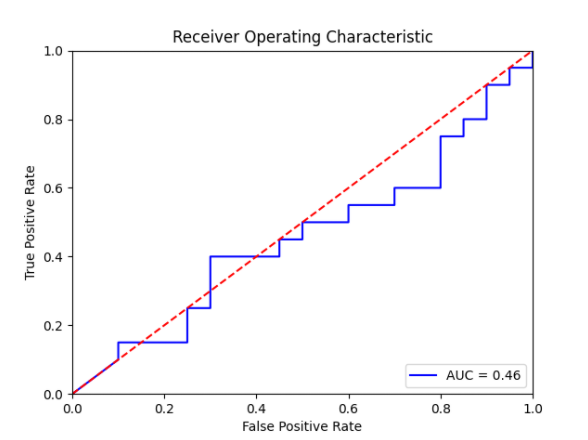
Miután spacyben ltérehoztam a klasszifikációs modellt, szerettem volna minél előbb validálni magát a koncepciót, így egy 10-10 darabos train-teszt adatom le is teszteltem egyből a modellt. A példákat az első dookumetumből kézzel válogattam ki, egy példa egy paragrafust tartalmaz, amiben van vagy feltételes mód, vagy nincs. Válogatás után egyből lefutattam és a következő roc\_auc score-t kaptam.



Sok következtetést nem vonunk még le az eredményből, de láthatjuk, hogy 20 példa még kevés less a kiértékeléshez.

#### 50-50 adat

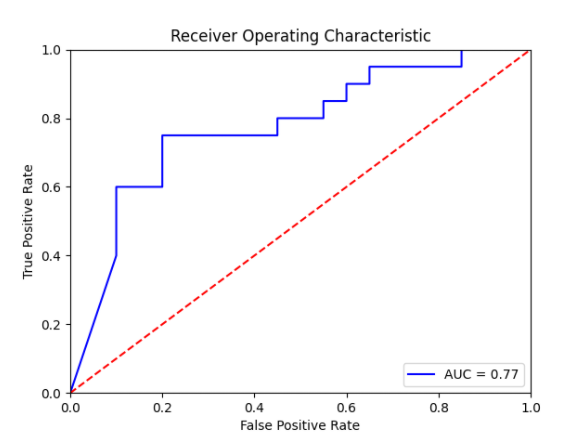
Most, hogy van már egy rendesen összterakott konfiguráció, már cask jó train adatok kellenek, hogy téynelgesen validájlni tudjuk, vajon a kitalált eljárás működhet-e. Ebben az iterációban már 50-50 példát gyűjöttem conditional szerkezetre a dokumentumokban, az eredmény pedig meglehetősen javult.



Sajnos a kb. 0.5 AUC score nem jelent mást, mint, hogy a modellünk cask tippelni tud egyelőre. Ezekután a teljes teszt szerződésre is ráengedtem a modellt, azért, hogy elemezni tudjam, milyen esetekben dönt rosszul a modellünk. Ezeket az eseteket kigyűjtööhgetem, majd hasonló példáűkkal újratanítom a modellemet, kvázi megersőítéses tanulás manuális.

#### 100-100 adat

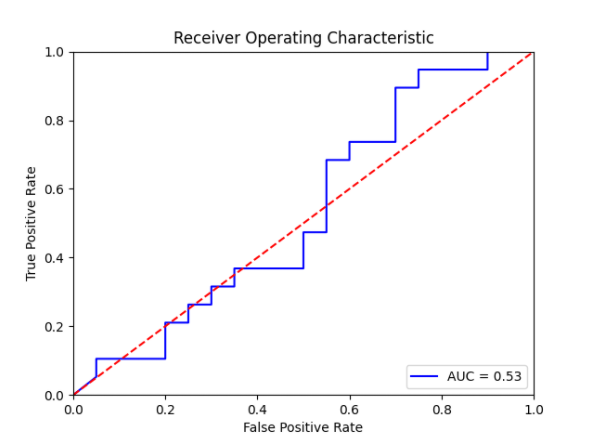
Átnézve a dokumentumokat, szemezgetve rossz döntéseket, ezekhez hasonló példákat gyűjtve a training setbe, most már 200 train adattal indulok neki a következő modell tanításnak. Áteresztve az új training adatokon, majd kiértékelve a 40 darabos teszt példán, a következő eredményeket kapjuk:



0.7-08 közötti AUC score már nem annyira rossz, egy úgymond elfogadtható tartomány. Ezen felül megnéztem még a f1, precision, recall és accuracy-t:

* Precision: 0.615
* Recall: 0.800
* Accuracy: 0.650
* F1 Score: 0.696

Szintén nem túlságosan meggyőző eredmények, de sokkal jobbak, mint 50 példánál. Ezen a ponton kijelenthetjük, hogy megfelelő számú tanítópélda után létre tudunk hopzni egy olyan modellt, amely képes szemantikailag valamennyire ráérezni a logiaki struktőrákra. Érdekesség képpen keresztvalidáltam egy msik dokumentumon, amit a interneten is elérhető EUR-LEX oldalról szedtem. Ez az összes Európao szerződés nyilvánosa elérhető, innen letöltötem pár szerződést, hogy kipróbáljam rajta a modellemet.

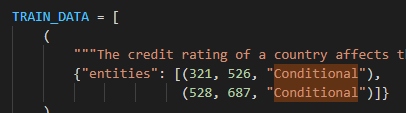


Sajnos itt is egy 0.53-as AUC score-t kaptunk, ami megint cask azt jelenti, hogy a modellünk tippelget. Mivel ez 100 példa után is cask egy tippelgetésre jó, kijelenthetkük, hogy nem sikerült a klasszifikációs modellünknek rátanulnia alogikai struktúrák szemantikájára.

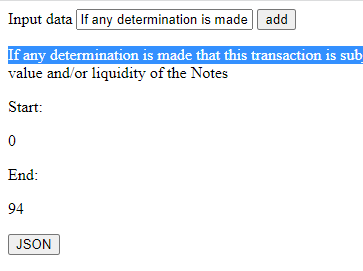
### Named entity recognition

Fentebbi metódusnál láthattuk, hogy sima klasszifikáció nem vezetett kielégítő eredményre, így egy másik megoldáshoz kell folyamodni ahhoz, hogy a feltételes mondatok felismerése sikerhez vezessen, így a spacy által szintén támogatott named entity recognitionhoz fordultam. Named Entity Recognition is a process which deals with identifying and classifying named entities. This named entites usually place, person, organization, time, object, or geographic entity, but in my following approach, I’m going to tag the conditional sentences as named entities.

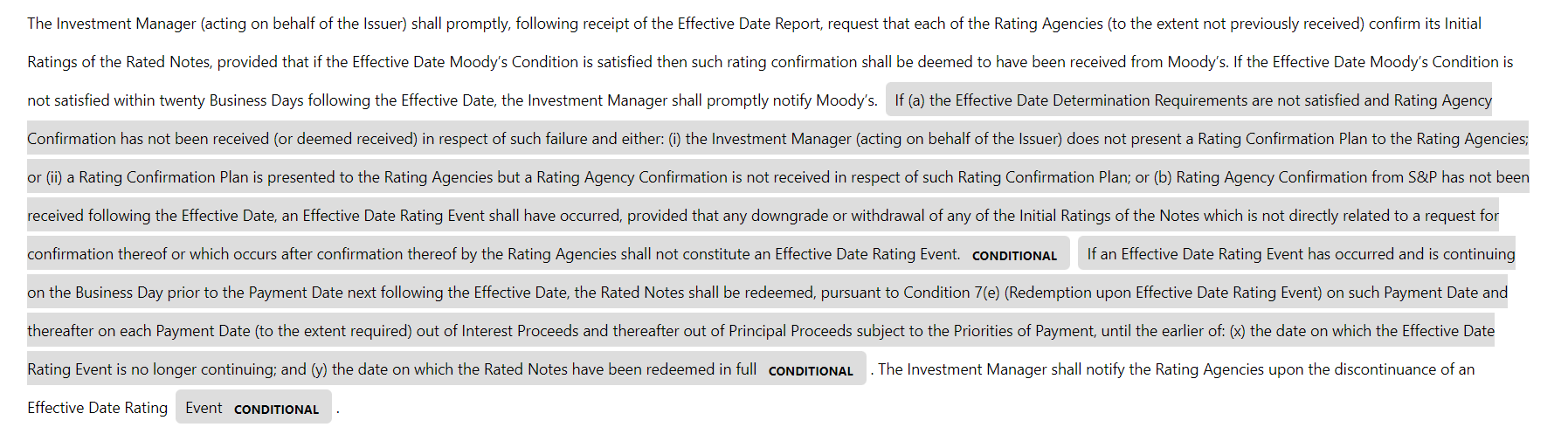
Spacy has a fast statistical entity recognition system, but we can train it with our own for business specific reasons, the spacy model general performs well for all types of text data. All we need is a correct train data format, which looks like this:



It’s a tuple in python, which first’s property is the current paragraph, the second one is an object containing our custom NER tags. I have tried two different approaches. The first one is to try identify the logical expressions within a document by tagging it with a custom NER tagger. To gather examples for the training set, again, I have examined the contracts, I have used 9 contracts to search for examples, and 1 is for testing. In this case, its not enough just to tag the paragraph whether it contains conditionals or not, we need to be more exact and tag the exact position of the logical structure. Doing the search and tagging it could be time-consuming since we need the position numbers, I have created a small HTML program to determine the positions of the conditionals. We just need to add the paragraph and select the condiontal manually by the mouse, click on the save and we got a json file with an object which contains the format we saw above earlier.



For the first round, I have gathered a hundred examples for the training set, and have tested it against a training contract (!not test contract). The algorithm here is similar as I have applied in the classification algorithm, so first train then apply the model on a training contract to search for false prediction, and retrain the model with similar examples.



Spacy has a visualizer called “diSplacy”, which supports the entity visualization alongside others. After ~70 examples and going through the whole document checking for elements for the retraining. The picture above shows as a great example for it. As clear we got the logical structure, we also got a wrong tag for “Event”. In this case, we add the paragraph to the train set explicit say to the model not to tag this kind of examples. We had plenty of similar examples in the document for the next training round. This time, with ~200 examples.

### Dependency graph

## 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.

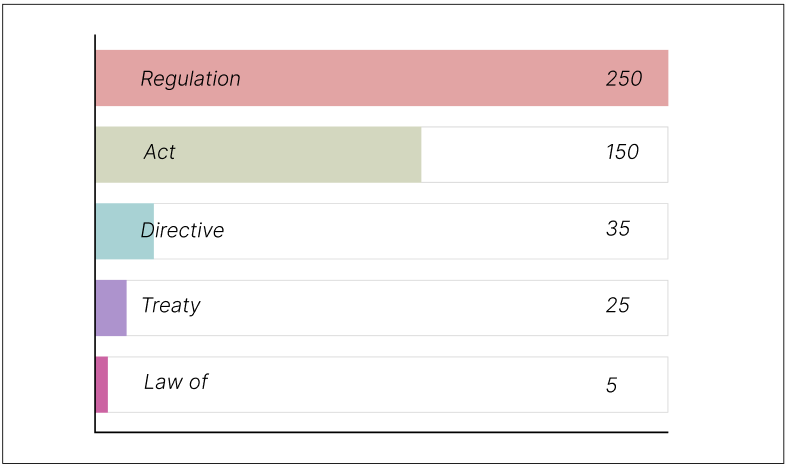
Checking cross-references and schedules is an extremely important task which contract drafters must, but often do not, undertake. There are some real-life examples where wrongly addressed cross-references led to lost cases or severe fines. In the following study, we are going to study cross-reference 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:

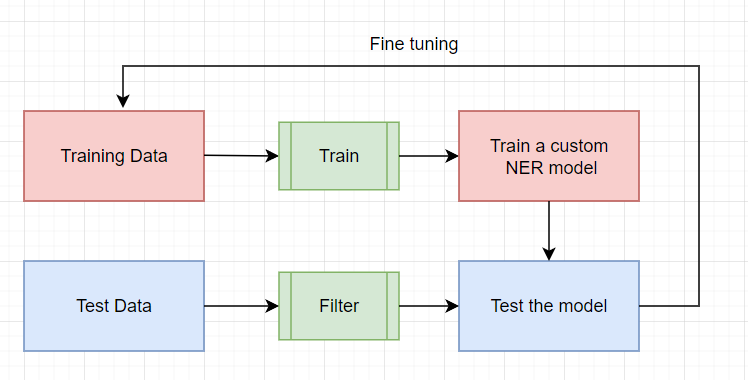
**

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 then train a custom NER model with the help of spaCy. Then comes 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.

#### Analyzation of the entities

As I mentioned when introducing external references, references can look different, they contain accurate section information of a provision. In the model building part, we intentionally left the section markings. In the next part, we are going to build another NER model to identify the section markings and the reference itself within a recognized entity.

### 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)

10. Events of Default

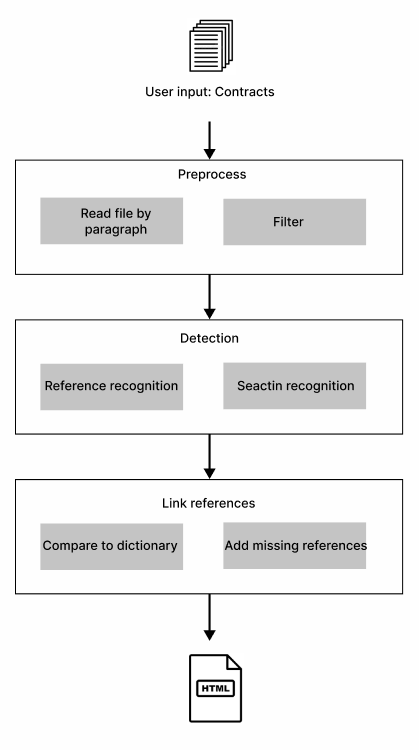
(a) Events of Default

Any of the following events in this paragraph (a)(i) to (a)(viii) shall constitute an “Event of Default”:

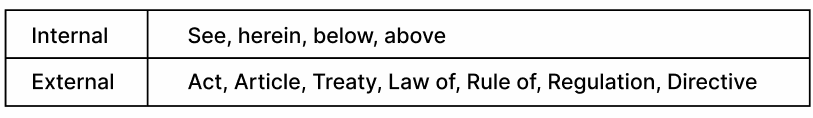
1. Non-payment of interest

### 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 linked references. Az előfeldolgozási fázisban első lépésként beolvassuk a szöveget paragrafusonként. Mielőtt azonban ezeket bemenetként odaadnánk a felismerőnek feldolgozása, előtte megtisztítjuk, kiszűrjük azokat az eredményeket, amelyek nem tartalmazzák az úgynevezett trigger szavakat. Mint az korábban láthattuk, referenciáknak van egy tipikus felépítése, és minden esetben hasonló szavak környékén találkozhatnuk vele. Így a felsimerő modellünk már egy előre megszűrt adathalmazt kap, ami reményeink szerint segít kiszűrni a fals eredményeket.



A felismerő fázisban az immáron feldolgozott adatainkat áteresztjük először azon a NER modellen, mely tisztán a referenciát ismeri fel, elmenti ezeket, majd inputként továbbadja a második NER modellnek, amely már a felismert referenciákon megy keresztül, és különszedi őket “section”, “order” és rövidítésre. Ennek az eredményét, tehát a felismert és különszedett referenciákat egyenként egy dictionarybe mentjük ki.

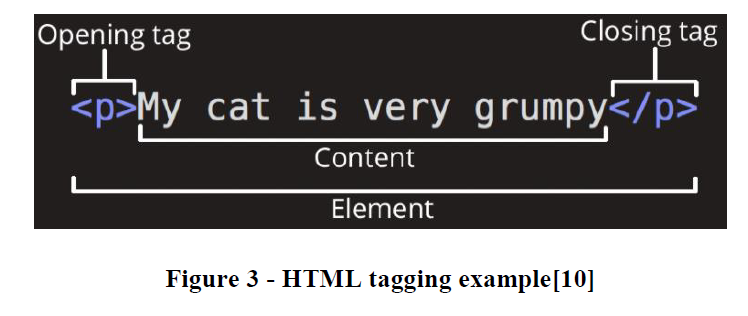
Az alkalmazásunk úgy működne, hogy létezne egy szótárunk törvényekkel és a hozzájuk kapcsolódó forrással. Minden egyes alkalommal, amikor feldolgozunk egy fájlt és felismertük a referenciákat, átfutunk ezen a szótáron még nem létező rekordok után kutatva. Ha találunk ilyet, akkor azt pótoltatjuk a userrel. Majd miután a kimaradt referenciák felöltése is megtörtént, következhet a törvények linkelése. Ehhez újból végigeresztjük a modellünkön, de ebben az esetbe

## Title detection

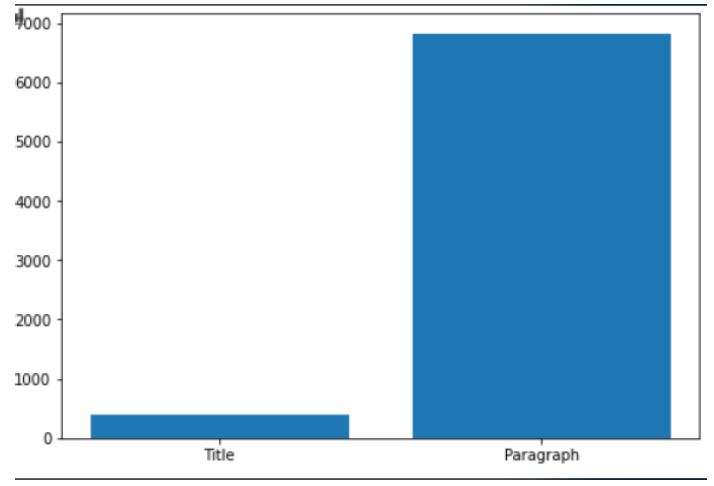
The detection of the title and paragraph in texts has started from finding an algorithm that can be useful for quickly searching in a large contract or text, or even generating a table of contents. Recognizing titles and paragraphs are a (binary) classification problem that can be done with ease. A text is either a title or if it’s not, then we assume, it is a paragraph (although it can be other type as well, but we are not dealing with it yet). However, the main problem here is - as at every data mining task – to find the right test data. In the real world, from the legal field, documents can come in many forms, raw txt, image or searchable pdf, word document, raw HTML text, etc.…

### Data set and preprocessing

For the first experiment, I have got legal contracts parsed from raw HTML text based on their HTML tags, exactly 10 contracts. HTML is a markup language that defines the structure of your document or text. HTML consists of a series of elements, which you use to wrap, different parts of the content to make it appear in a certain way. The enclosing tags can make a word or image hyperlink to somewhere else, can italicize words, can make the font bigger or smaller, and so on.[10]



The contracts were parsed by their HTML tag. For example, a title is enclosed by a so called “h1, h2, etc ...” tag, and when the parser finds a closing tag (i.e. </h1>) it adds the text to a row. Unfortunately, parsing alone is not that simple. The difficulty of processing was that while we can indicate certain things in an HTML text using CSS (Cascading Style Sheets, use for formatting text in HTML) a member 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 its formatted properly. Furthermore, the text and the tags were in a different file, so first I had to zip into the same file (csv). After zipping, I had to correct some tags manually since it was crucial for the evaluation and the training as well. Fortunately, the original formatted HTML of the contracts was available, so it was possible to do that. After cleansing and preprocessing, I could start analyzing and developing algorithms.

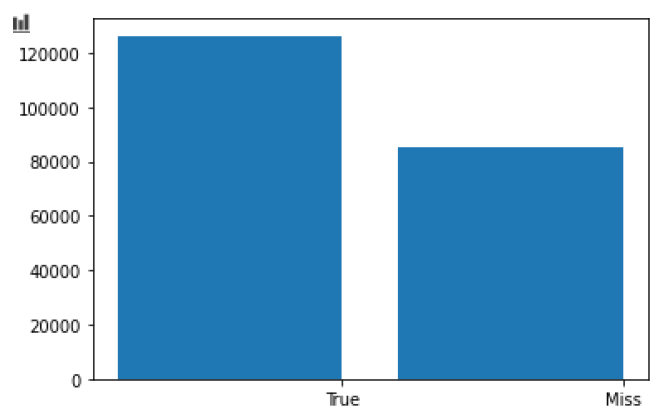


### Experiment with “manual” algorithms

Before I even immersed myself in the depths of text mining tools, I started developing more “primitive” algorithms to find out the task is even solvable manually. I have organized the data into CSV, (comma separated text files), because it is easier to handle the data in CSV format with Python. The text is one column and the tag is another. After reading texts programmatically, we process the data with an 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:

* Is the text’s length short?
* Does the text contain few verbs?

If the above conditions are met, we can say, that the text is possibly a title.

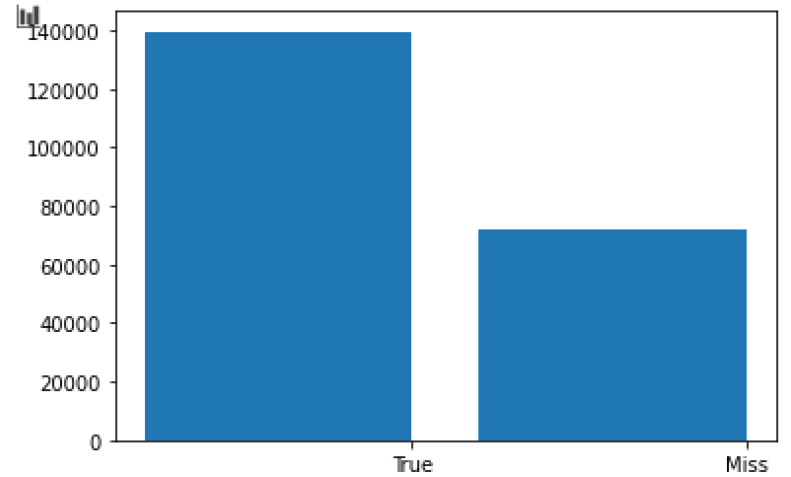


We got 39826 hit and 126289 miss out of 211346 segment means 59% probability. It seems that we have to make some improvements in our algorithm.

So, refining the algorithm, we add the following:

* Contains only uppercase letters, or begins with an uppercase letter, or title case
* Contains anything besides number (e.g., not date)
* It starts with a number, then continues with text

This has already given us a slightly better results:

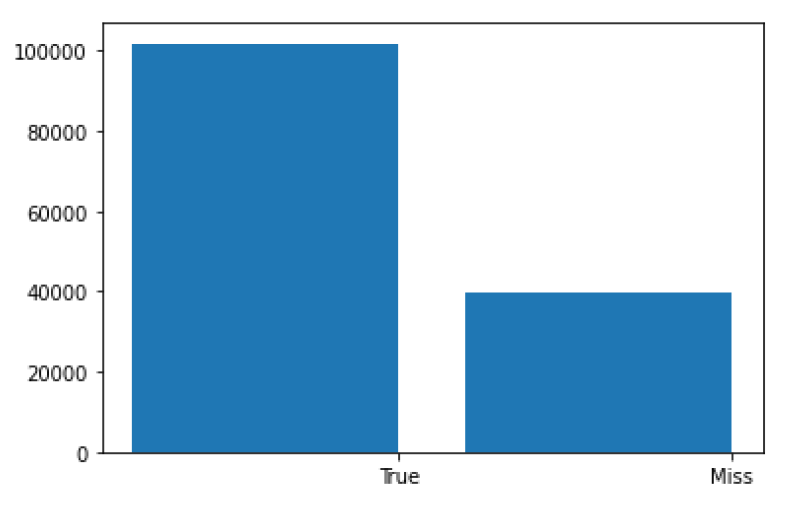


We got 139505 hit and 71841 miss with a probability of 66%. The goal is to reach a probability around 70%.

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:

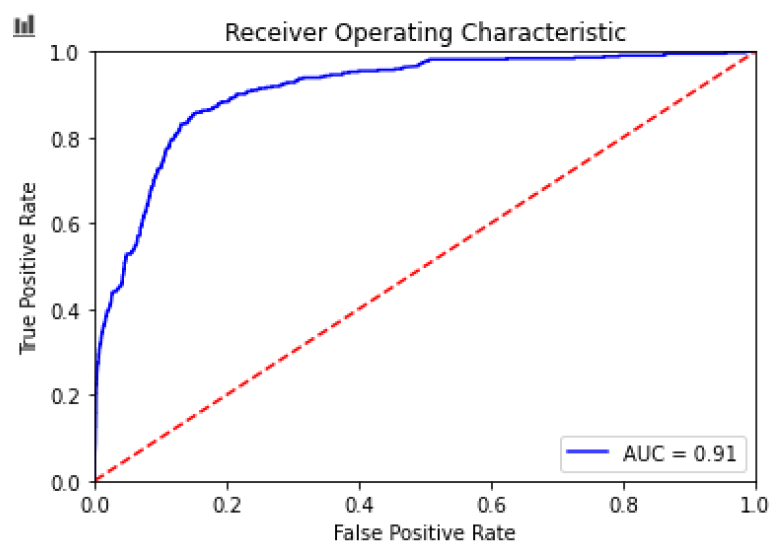


We got 39826 hit out of 141652 means 0.7188 probability. It seems like we have achieved better results with these small refinements.

After testing how manually the classification of the text can be solved, we also try to use machine learning algorithms to perform the classification’s problem mentioned above.

In our example, the text attribute contains the text wrapped by the HTML tag, and the cats attribute indicates whether the segment is a title or a paragraph. First, we must shape our dataset to the above-mentioned way. For training data, I have used 8 datasets from the 10 contracts, and for testing purpose I have used the remaining two contracts. The documents were parsed by the paragraphs’ HTML tags, so when a paragraph had the tag “<hx>” (where x is 1,2,3 or 4), then it got 1 for ‘Title’ and zero for ‘Paragraph’ label and vice versa.

After successful preprocessing, we created our training and test dataset consumable for Spacy pipeline. We have already prepared the data for the classification, now we only need to configure our model. In Spacy, we find various text manipulation models called as pipes (such as the tokenizer mentioned earlier). The text categorizer is represented in Spacy as the “textcat” component. Once its added, we have the option to configure the neural network. After configuration and labeling, our model can be taught. By default, we will use a simple convolutional neural network.



We got an AUC score of 91 which seems like a much better score than our manually developed model, although this text classification model was the simplest one which can be created with Spacy.

# Further work

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)

Függelék