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. 04. 25.

...…………………………………………….

Rezeda Kázmér

Összefoglaló

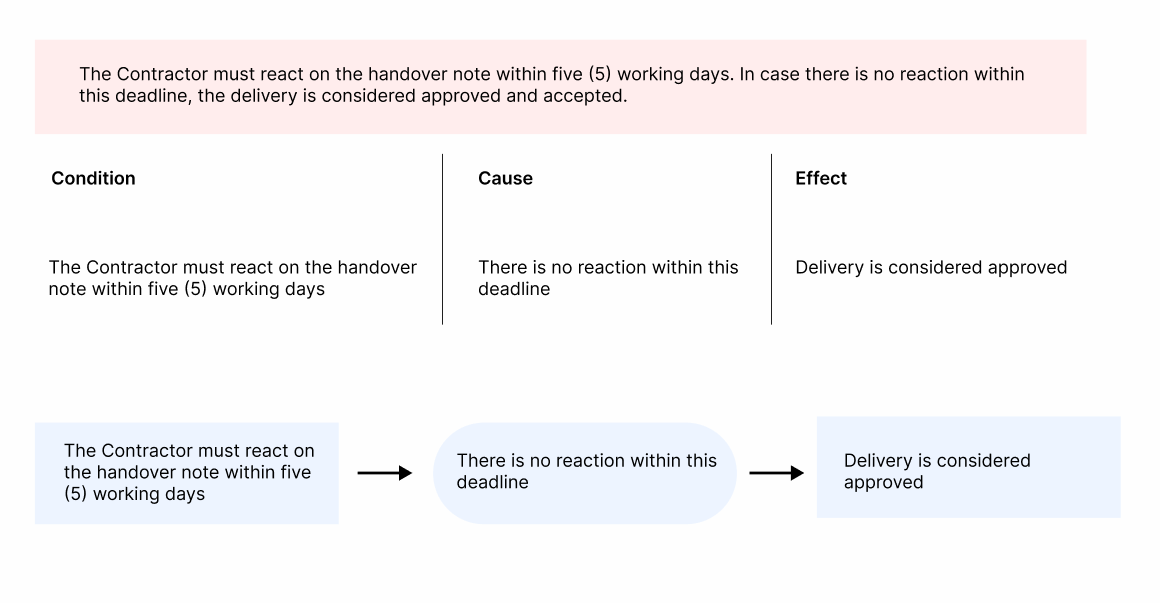
Ide jön a ½-1 oldalas magyar nyelvű összefoglaló, melynek szövege a Diplomaterv Portálra külön is feltöltésre kerül.

Abstract

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

# 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.

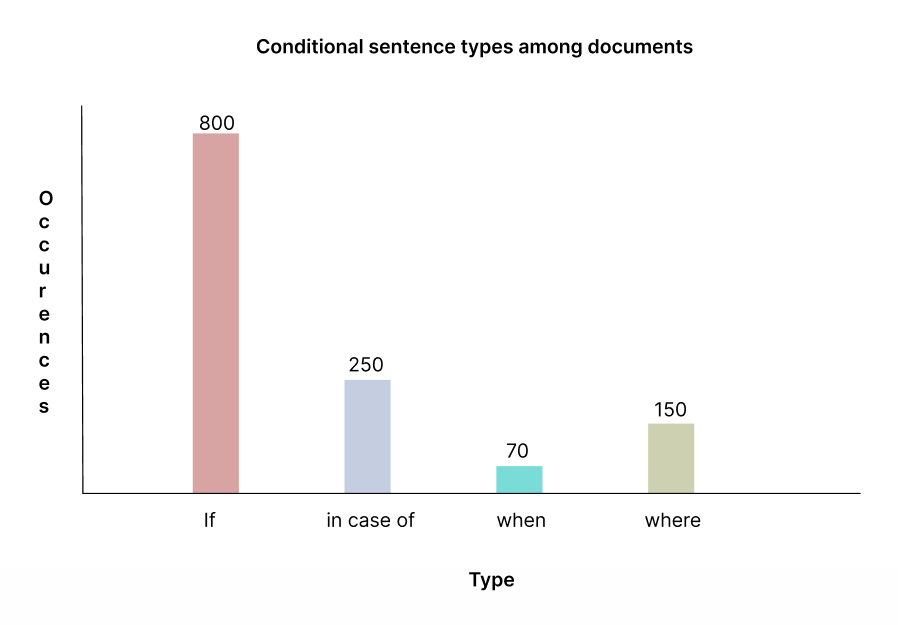


## Nature of the sentences

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

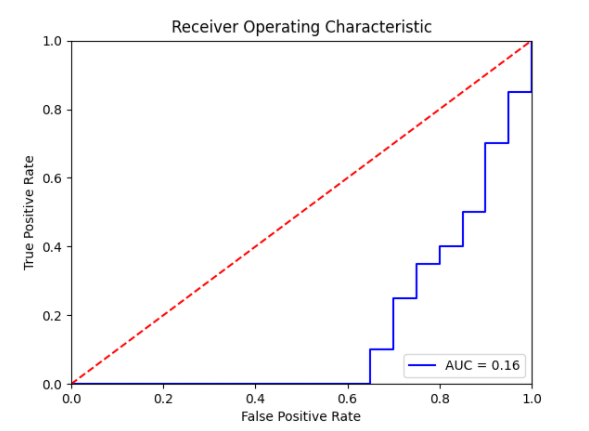
Most, hogy van egy kisebb képünk a rendelkezésre álló adatokról és a kinyerni való struktúra természetéről, már cask le kell fordítanunk gép nyelvre. Első körben szemantikai analízist próbálunk végezni, melyben azokat a paragrafusokat próbáljuk megjelölni, amelyek tartalmaznak feltételes/logikai szerkezeteket. 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

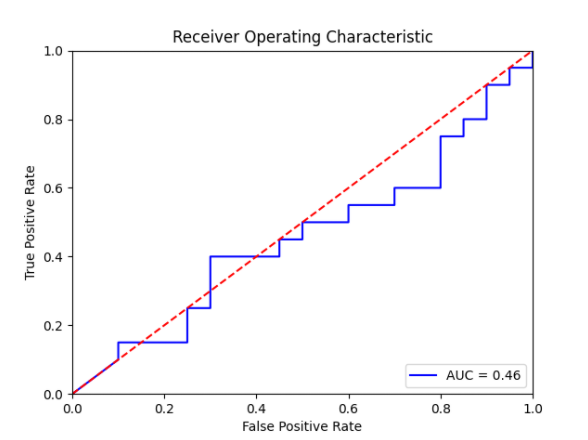
Miután spacyben ltérehoztam a klasszifikációs modellt, szerettem volna minél előbb validálni magát a koncepciót, így egy 10-10 darabos train-teszt adatom le is teszteltem egyből a modellt. A példákat az első dookumetumből kézzel válogattam ki, egy példa egy paragrafust tartalmaz, amiben van vagy feltételes 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

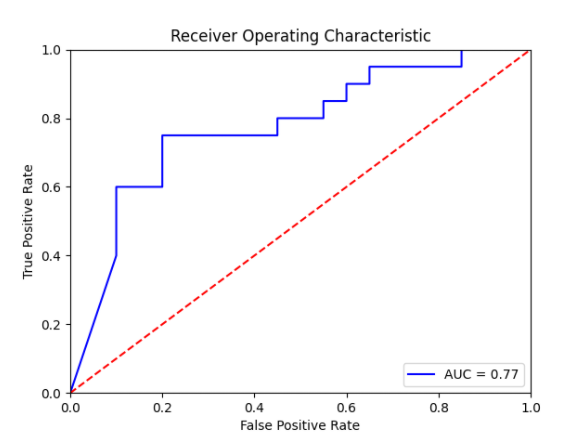
Most, hogy van már egy rendesen összterakott konfiguráció, már cask jó train adatok kellenek, hogy téynelgesen validájlni tudjuk, vajon a kitalált eljárás működhet-e. Ebben az iterációban már 50-50 példát gyűjöttem conditional szerkezetre a dokumentumokban, az eredmény pedig meglehetősen javult.



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

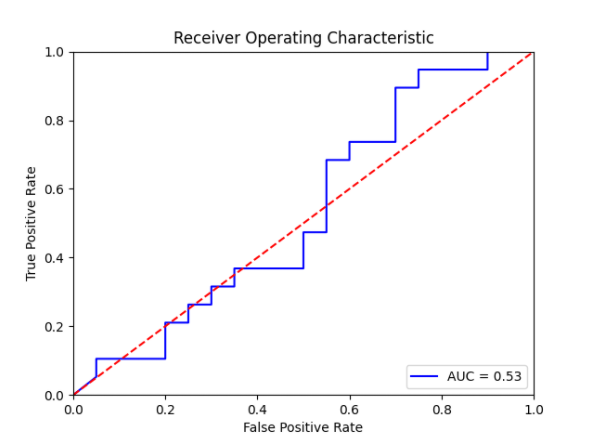
Átnézve a dokumentumokat, szemezgetve rossz döntéseket, ezekhez hasonló példákat gyűjtve a training setbe, most már 200 train adattal indulok neki a következő modell tanításnak. Áteresztve az új training adatokon, majd kiértékelve a 40 darabos teszt példán, a következő eredményeket kapjuk:



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.

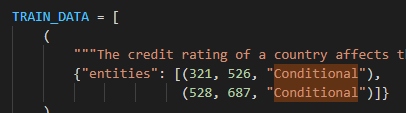


Sajnos itt is egy 0.53-as AUC score-t kaptunk, ami megint cask azt jelenti, hogy a modellünk tippelget. Mivel ez 100 példa után is cask egy tippelgetésre jó, kijelenthetkük, hogy nem sikerült a klasszifikációs modellünknek rátanulnia alogikai struktúrák szemantikájára.

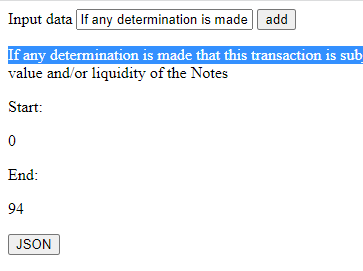
### Named entity recognition

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.

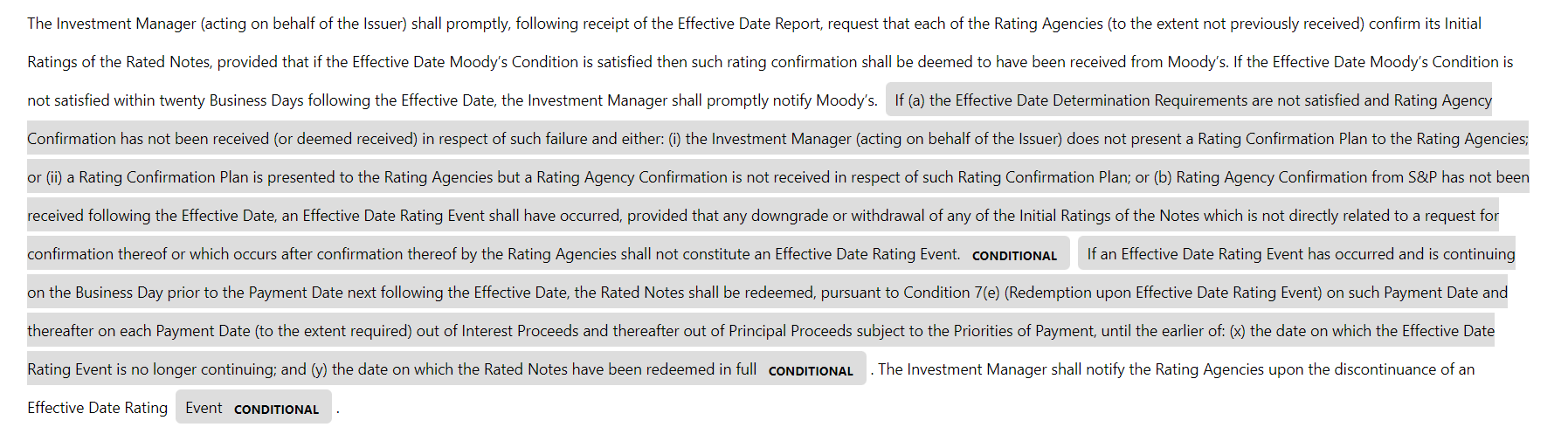
Spacy has a fast statistical entity recognition system, but we can train it with our own for business specific reasons, the spacy model general performs well for all types of text data. All we need is a correct train data format, which looks like this:



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

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