UNIVERSITEIT GENT

Project AI: Een aanbevelingssysteem voor juridische documenten

Arnoud De Jonge - 01808870
Arno Vermote - 01806792

Academiejaar 2020-2021

1 Inleiding

1.1 Probleemstelling

Er moet een aanbevelingssysteem opgesteld worden. Dit aanbevelingssysteem neemt als input een lijst van rechtszaken en een geselecteerde rechtszaak. Er worden dan 5 vergelijkbare rechtszaken teruggegeven.

1.2 Gebruikte datasets

Voor dit probleem zullen wij gebruik maken van de dataset 'Open Data van de Rechtspraak'[8]. Deze dataset is publiek beschikbaar via rechtspraak.nl. Uit deze data worden het vonnis en de gebruikte wetteksten gefilterd. De bedoeling is dan om op basis van deze 2 factoren vergelijkbare rechtszaken te vinden. Om gelijkenissen te detecteren, hebben we natuurlijk algoritmes nodig.

1.3 Algoritmes

1.3.1 Soorten algoritmes

In de literatuur wordt er gesproken over verschillende algoritmes om aan Natural Language Processing (NLP) te doen. We kunnen ze opsplitsen in 2 groepen: tekstgebaseerde en netwerkgebaseerde methodes.

Deze eerste groep zal vaak proberen de tekst om te zetten naar vectoren. Deze vectoren stellen woorden of zinnen voor en de context waarin ze verschijnen. Om gelijkaardige woorden te vinden, kan dus gekeken worden naar de afstand tussen 2 vectoren. Indien deze klein is, zijn ze gelijkaardig.

De tweede groep zal proberen een netwerk te maken van alle teksten, gebaseerd op de metadata van de tekst. Verbanden kunnen onder andere gemaakt worden door een verwijzing uit tekst A naar tekst B. Hoe sterker de verwijzing is tussen 2 teksten, hoe beter ze op elkaar zullen lijken. [2]

	(hoger is beter) Silhouette score	(hoger is beter) Dunn Index	(dichter bij 0 is beter) S_Dbw	(dichter bij 0 is beter) Davies-Bouldin score
TF-IDF	-0.16536	2.59E-05	0.82416	4.6662
Doc2Vec	-0.14014	1.5727E-05	0.367126	3.2904
Word2Vec	-0.077212	2.96568E-05	0.34176	1.7568
LDA	-0.14586	0.000003168	0.426	2.11464
Dispersion	-0.21879	4.62073E-05	1.6544	57.985704

Tabel 1: Clusteranalyse van verschillende NLP-technieken
Groen is beste, geel is 2de beste

1.3.2 Selectie

Er bestaan dus verschillende algoritmes met uiteenlopende resultaten. Om te bepalen welke algoritmes zullen geïmplementeerd worden in het aanbevelingssysteem, moeten deze met elkaar vergeleken worden. Bij deze vergelijking maken we gebruik van een clusteranalyse.

Een clusteranalyse is een techniek die men gebruikt om te kijken hoe goed een algoritme de gegeven data kan sorteren in verschillende groepen (clusters). De gebruikte analyse bestaat uit het berekenen van enkele scores: Silhouette index, Dunn index, s_Dbw index en de Davies-Bouldin score. Iedere score controleert op een bepaalde eigenschap van een goed geclusterd model en kent deze een waarde toe. Bij de Silhouette index wordt er bijvoorbeeld gemeten of een object gelijkend is aan anderen binnen de eigen cluster, en verschillend is tegenover andere clusters. De Davies-Bouldin index geeft een score gebaseerd op de gemiddelde afstand van punten ten opzichte van het centrum van hun cluster, en de afstand tussen cluster centra [2].

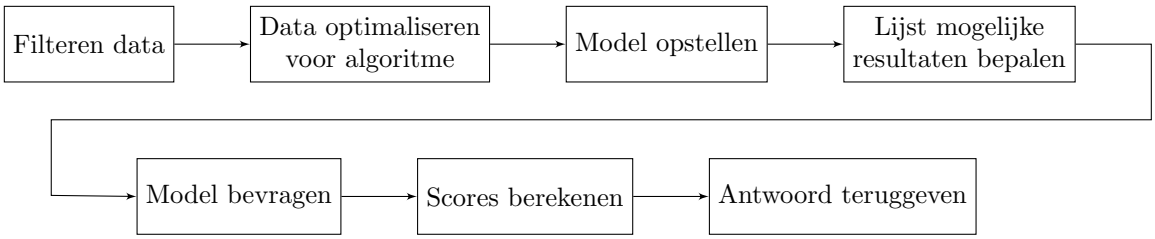
Van een vooraf bepaalde lijst van algoritmes werden diegene gekozen die de beste resultaten behaalden volgens de bovenstaande beschreven clusteranalyse: Word2Vec en Doc2Vec (Tabel 1). Onze resultaten komen in grote lijnen overeen met die in de masterproef van Bert Commeine [2].

Ten slotte selecteren we op aanraden van professor Saeys ook BERT, een recente ontwikkeling in NLP.

2 Methodiek

2.1 Pijplijn

Bij het zoeken naar vergelijkbare rechtszaken, zullen alle documenten door een pijplijn gestuurd worden. Deze pijplijn bestaat in grote lijnen uit de volgende onderdelen:



2.2 Data filteren

De data van rechtspraak.nl wordt aangeboden in .xml-formaat. De eerste stap die we ondernemen is de (voor ons) waardeloze metadata wegfilteren. Dit gaat dan over de plaats van de rechtbank, de datum, interne ID's... We willen enkel het vonnis en de gebruikte wetten en wetsartikels bijhouden. De uitkomst wordt opgeslagen in een .txt-bestand, zodat deze stap niet steeds hoeft herhaald te worden.

`De relevante code staat in src/XMLParser/XMLParser.py.`

2.3 Data optimaliseren voor algoritme

In deze stap is het de bedoeling om enkel betekenisvolle woorden en zinnen uit het vonnis te halen. Wat 'betekenisvol' is, hangt af per algoritme.

2.3.1 Word2Vec & Doc2Vec

Word2Vec en Doc2Vec presteren beter als er duidelijke verbanden tussen woorden gevonden worden. Het is daarom belangrijk om die verbanden zo duidelijk en expliciet mogelijk te maken. Er zijn veel manieren en combinaties mogelijk om dit te doen. Wij gebruikten de volgende:

2.3.1.1 Stemming

Stemming is een techniek die woorden van een tekst omzet naar hun stam. Bijvoorbeeld 'beslissing' en 'beslist' worden allebei omgezet naar 'beslis'. Zo worden woorden met dezelfde stam makkelijk aan elkaar gelinkt.

2.3.1.2 Stopwoorden

Door stopwoorden (de, maar, een, en...) weg te filteren wordt de tekst minder leesbaar voor de mens, maar blijven alle betekenisvolle woorden aanwezig in de tekst. Dit zorgt dus voor minder ruis in de data.

`De relevante code staat in`
`thesis_master/src/models_implementation/preproces_pipeline.py.`

2.3.2 BERT

BERT werkt goed met natuurlijke zinnen. Daarom moet de data ook gesplitst worden per zin. Maar dit is niet zo simpel als de tekst opsplitsen als we een '.', '?' of '!' tegenkomen. We zijn vertrokken van een implementatie voor Engelstalige zinnen[6]. Om dit naar een bruikbare Nederlandstalige versie om te zetten, zullen we zorgen dat de Nederlands afkortingen die een punt bevatten zoals 'Dr.', 'Artk.' niet worden gezien als het einde van een zin. Idem voor getallen die één of meerdere punten bevatten zoals '1.000.000' of '1.'. Ten slotte merkten we op dat in de vonnissen gebruik werd gemaakt van '(...)' om delen over te slaan, hier moeten we ook rekening mee houden.

2.4 Model opstellen

2.4.1 Word2Vec

Word2Vec wordt getraind aan de hand van de vonnissen. Ieder woord wordt voorgesteld door een vector. Vervolgens wordt ieder woord in een vonnis overlopen en wordt er gekeken naar het woord zelf, en de context rond dat woord. De gelijkheid (op basis van afstand) van de vectoren wordt gebruikt om de kans te berekenen dat dat woord voorkomt gegeven die context, of omgekeerd. De vectoren worden dan zodanig aangepast om de kans te maximaliseren. Dit proces herhaalt zich voor ieder vonnis. Door meer data wordt de voorspelling beter. Zo kan dan een voorspelling gemaakt worden voor ieder woord, gegeven een context.

Verwijder nu de output-layer van het model: Gegeven een woord, wordt er dan geen voorspelling meer teruggegeven, maar een vector. Zo kunnen we de afstand tussen 2 woorden bevragen.[5]

2.4.2 Doc2Vec

Het principe van Word2Vec kan ook toegepast worden op een zin, paragraaf of tekst. Het idee blijft dus hetzelfde, maar in plaats van een woord wordt nu een document omgezet naar een vector. Er wordt opnieuw geprobeerd het volgende woord te voorspellen, maar deze keer 1). aan de hand van de vorige woorden (= paragraaf) en 2). aan de hand van een unieke paragraafvector. Die paragraafvector wordt gekozen op basis van het volledige document: er wordt een gemiddelde genomen van (een gesampled deel van) alle woordvectoren van het document. Zo wordt er dus rekening gehouden met de betekenis van de woorden (zoals in Word2Vec) en ook met de woordvolgorde van een zin. Als 2 rechtszaken gelijkaardig zijn, zullen de Doc2Vec-representaties ook dicht bij elkaar liggen. [2]

2.4.3 BERT

Bidirectional Encoder Representations from Transformers [3] [4], of kortweg BERT, maakt gebruik van het encodermechanisme van Transformer[10]. Ook dit algoritme werkt aan de hand van vectoren. Dit encodermechanisme leest een volledige zin in één keer in. Hierdoor wordt zowel de context links als rechts van een woord mee in rekening genomen.

De implementatie van BERT die wij gebruikten, focuste zich op het zoeken naar semantisch gelijke zinnen, waar de 2 andere algoritmen meer afhankelijk waren van identieke woordenschat.

Het gebruikte BERT-model hebben wij niet zelf opgesteld. Hiervoor hadden wij niet genoeg data, kennis van het juridisch vakgebied en computationele kracht. Daarom kozen we om gebruik te maken van een vooraf getrainde algemene dataset. [9] [7]

2.5 Lijst mogelijke resultaten bepalen

Door te kijken naar de gebruikte wetten en wetsartikelen van zaak 1 en die van zaak 2 kunnen we al enkele conclusies trekken: 2 rechtszaken die geen enkele gebruikte wet gemeenschappelijk hebben, worden op voorhand al uitgesloten bij het zoeken naar vergelijkbare rechtszaken. Dit doen we hoofdzakelijk omdat we slechts een beperkte computationele kracht hebben. Door zaken op voorhand uit te sluiten, besparen we veel tijd. Dit zal vooral een verschil maken bij BERT.

2.6 Model bevragen

Nu de data gefilterd is en de modellen zijn opgesteld, wordt het tijd om voor ieder model een rangschikking te kunnen maken van vergelijkbare rechtszaken. Aan de hand van het model kunnen we een gegeven rechtszaak omzetten in een vector. Het is nu de bedoeling dat we dit doen voor iedere

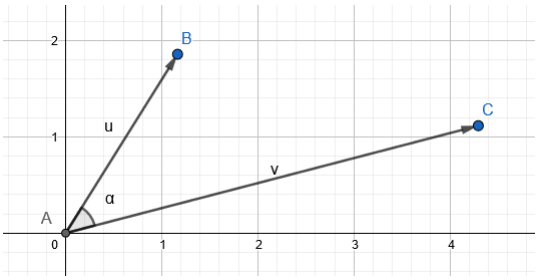
rechtszaak. Vervolgens moet men de afstand tussen de gegeven rechtszaak en iedere andere berekenen. Ten slotte sorteren we op afstand. Hoe kleiner de afstand, hoe beter deze zaken overeenkomen.

2.6.1 Afstand berekenen

De afstand tussen 2 vectoren kan berekend worden op verschillende manieren. Zo bestaan er de Euclidische afstand en de cosinusgelijkenis. Omdat bij de gebruikte modellen de grootte van de vectoren niet relevant is, is het beter de cosinusgelijkenis te gebruiken. Die houdt enkel rekening met de zin en de richting van de vectoren.

$$cosinusgelijkenis(A, B) = \frac{A \cdot B}{||A|| * ||B||}$$

We zeggen dat de afstand tussen 2 vectoren klein is als de cosinusgelijkenis dicht bij 1 ligt. De afstand tussen 2 vectoren is groot als deze dicht bij 0 ligt. Figuur 1 toont de meetkundige voorstelling van de cosinusgelijkenis.



Figuur 1: $cosinusgelijkenis(u, v) = cos(\alpha)$

2.7 Scores berekenen

We hebben nu dus lijsten van scores ten opzichte van een gegeven document. Iedere lijst stelt de uitvoering van een algoritme voor. Er is dus een lijst voor Word2Vec, Doc2Vec en BERT. Het valt onmiddellijk op dat als een bepaald document een hoge score haalt voor het ene algoritme, dit daarom niet een hoge score betekent in een ander algoritme. We moeten dus afwegen welk algoritme onze voorkeur geniet. We hebben dit probleem opgelost door een totaalscore op te stellen. Ieder algoritme heeft invloed op een bepaald procent van die totaalscore. Ook houden we nog extra rekening met de gebruikte wetteksten.

2.7.1 Vergelijken van wetteksten

De wetteksten van rechtszaak 1 en 2 worden ook in rekening gebracht in de totaalscore. Om die score te bepalen, wordt het volgende algoritme gebruikt:

```
Tekst 1 = De input rechtszaak
Tekst 2 = De rechtszaak waarmee we vergelijken

a1 ... an wetteksten uit rechtszaak 1 (input)
b1 ... bm wetteksten uit rechtszaak 2 (waarmee we vergelijken)
```

```
# score tussen 0 en 1 voor teksten uit b die in a zitten.
score_b_in_a = 0
voor elke bi:
    # indien dezelfde wettekst gebruikt maar een verschillend artikel
    indien in a1 ... an dezelfde wettekst zit: score_b_in_a += 0.7

    # indien dezelfde wettekst gebruikt werd met hetzelfde artikel
    # dit maakt een totaalscore van 1
    indien het gebruikte artikel ook hetzelfde is: score_b_in_a += 0.3

# nu delen we score_b_in_a door de het aantal gebruikte wetteksten in tekst 2
score_b_in_a = score_b_in_a / m

# nu doen we het analoog voor de teksten uit a die in b zitten
# score tussen 0 en 1 voor teksten uit a die in b zitten.
score_a_in_b = 0
voor elke ai:
    # indien dezelfde wettekst gebruikt maar een verschillend artikel
    indien in b1 ... bm dezelfde wettekst zit: score_a_in_b += 0.7

    # indien dezelfde wettekst gebruikt werd met hetzelfde artikel
    # dit maakt een totaalscore van 1
    indien het gebruikte artikel ook hetzelfde is: score_a_in_b += 0.3

# nu delen we score_a_in_b door de het aantal gebruikte wetteksten in tekst 1
score_a_in_b = score_a_in_b / n

# nu zullen we de totaalscore berekenen:
# dit is een gewogen gemiddelde van score_a_in_b en score_b_in_a
# de gewichten zijn niet gelijk omdat we het iets belangrijker
# vinden dat alle gebruikte wetteksten in a ook in b zitten
# dan omgekeerd. Dit komt omdat het minder erg is dat
# tekst 2 uitgebreider dan dat het te kort zou zijn.

return score_a_in_b * 0.6 + score_b_in_a * 0.4
```

Kort samengevat zal dit algoritme 0 teruggeven indien er geen gemeenschappelijke wetteksten zijn, en zal de score stijgen hoe preciezer de wetteksten overeenkomen. Hierdoor trachten we fouten van de algoritmes te reduceren door teksten die qua woordenschat gelijkend zijn, maar inhoudelijk niet, een lagere score toe te kennen. Het algoritme zou denken dat het een goede match is, maar wegens het gebrek aan overeenkomstige wetteksten geven wij het toch een lagere score. Dit doen we omdat wij denken dat het aantal overeenkomstige wetteksten wel een belangrijke factor is.

Een belangrijke extra opmerking is dat rechtszaken waarbij de Wettekstscore 0 is, niet in aanmerking zullen komen om geselecteerd te worden in Word2Vec, Doc2Vec en BERT.

2.7.2 Gewichten van algoritmes

We beschikken zelf niet over de juridische kennis om correct te bepalen wat een gelijkaardige rechtszaak is. Met onze beperkte kennis leek het voor ons echter wel dat de topscore van ieder algoritme

effectief een gerelateerde rechtszaak voorstelde. Daarom besloten we ieder algoritme evenveel te laten doorwegen.

$$totaalscore = 0.3 * w2v + 0.3 * d2v + 0.3 * BERT + 0.1 * WETTEKST$$

```
De implementaties van deze algoritmes staan in
thesis_master/src/models_implementation/doc2vec.py,
thesis_master/src/models_implementation/word2vec.py en
src/BERT/bert.py.
```

3 Resultaten

3.1 Evaluatiemethode

Als niet-juristen was het voor ons niet altijd even makkelijk om de resultaten te beoordelen. We bouwden een model, vroegen er de beste matches voor een gegeven rechtszaak aan en controleerden manueel of we vonden dat de volgorde die het algoritme had toegekend klopte voor de eerste 5 resultaten.

3.1.1 Verificatie resultaten

Na een opmerking tijdens de presentatie hebben we gekeken naar een andere manier om onze resultaten te verifiëren. Hiervoor zullen we de volgende procedure volgen:

1. Kies een inputfile, en houd bij over welk soort recht het gaat (zoals bestuursrecht).
2. Voer een algoritme (zoals Doc2Vec) uit over alle rechtszaken van 2005, dus zonder rekening te houden met de gebruikte wetteksten.
3. Bekijk de resultaten van het algoritme en vergelijk de soort recht van de ieder resultaat met dat van de input.

We verwachten dat de beste resultaten van het algoritme steeds hetzelfde soort recht hebben als dat van de input. Ook verwachten we dat bij slechte resultaten het gaat over andere soorten recht dan de input. Na het uitvoeren van de test op een steekproef van rechtszaken blijkt dit ook waar te zijn: bij de beste resultaten gaat het uitsluitend over dezelfde soort recht. Hoe verder we kijken in de resultatenlijst, hoe vaker we verschillen waarnemen.

Opmerking: deze verificatie kan ons enkel vertellen of onze resultaten compleet fout zijn of niet. We kunnen hieruit **niet** afleiden dat onze resultaten wel correct zijn. Door enkel gebruik te maken van deze test kan geen sluitende conclusie gevormd worden.

```
De implementaties van deze test staat in:
src/main.py: zet vlag TEST op True
```

3.2 Dataset uitbreiden

Bij de eerste paar testen met Word2Vec met dataset 200501, merkten we dat de resultaten vaak niet echt relevant waren. Hoewel we konden begrijpen waarom het algoritme een rechtszaak boven de andere verkoos, vonden we de link niet sterk genoeg. We hebben dit probleem opgelost door de dataset uit te breiden van enkel de eerste maand van 2005 naar het volledige jaar. Zo was de

kans groter dat er echt vergelijkbare zaken konden aangeboden worden. Dit leek ook te werken: Word2Vec gaf ons hogere scores en na manuele controle konden wij bevestigen dat de resultaten waren verbeterd. Zo vonden we bijvoorbeeld met als input een rechtszaak, en op de eerste plaats diezelfde zaak in beroep.

Het uitbreiden van de dataset had niet veel impact op de snelheid van het maken van het model. Toen we dit later ook testten met Doc2Vec en BERT kwamen we tot dezelfde conclusie. Echter duurde het bevragen van het model wel merkbaar langer bij BERT. Hier komen we in sectie 3.5 nog op terug.

```
De relevante code staat onder andere in
thesis_master/src/load.py.
```

3.3 Analyse Word2Vec & Doc2Vec

Word2Vec en Doc2Vec hadden vergelijkbare karakteristieken. Beide hadden ongeveer even lang nodig om een model te trainen, maar eens dat gebeurd was, ging het zeer vlot om het model te bevragen. We konden geen merkbaar verschil meten in het geheugenverbruik tijdens het trainen en bevragen van het model van één maand of het volledige jaar. Beide zijn dus een goede keuze als schaalbaarheid en snelheid belangrijk zijn.

3.4 Implementatie BERT

De versie van BERT die wij hanteerden [1], controleerde hoeveel verschil er zat op de semantiek van 2 zinnen. Echter bestaat onze data uit volledige rechtszaken, niet enkel zinnen.

3.4.1 Document als zin beschouwen

Een eerste idee dat we hadden was 2 volledige teksten aan het algoritme geven en die op gelijkenis laten controleren. Hoewel dit ons acceptabele antwoorden kon geven, zagen we ruimte voor verbetering door de antwoorden te vergelijken met Word2Vec.

3.4.2 Documenten splitsen

Dit is de methode die we zullen gebruiken. Hierbij worden beide teksten gesplitst in 2 lijsten van zinnen, zoals beschreven in sectie 2.3.2. Het idee is dan als volgt: voor iedere zin van document 1 wordt de best passende zin uit document 2 gezocht. Dan wordt de gemiddelde score van alle zinnen uit document 1 genomen als score voor BERT.

3.5 Analyse BERT

In tegenstelling tot de 2 andere algoritmes, stelden we geen model op van BERT. We hadden hiervoor de data en computationele kracht niet. Hoewel dit algoritme ons goede resultaten gaf, denken we dat er nog veel ruimte voor verbetering is: indien we een model zouden hebben dat specifiek getraind is voor Nederlandstalige juridische teksten, zou dat nog veel betere resultaten kunnen opleveren. Toen we 2 verschillende versies van Roodkapje aan het algoritme gaven (gewone Nederlandstalige zinnen dus), zagen we dat het merkbaar beter presteerde.

Onze eerste implementatie van BERT uit sectie 3.4.1 gaf ons schaalbare, maar sub-optimale resultaten. We denken niet dat dit een goede implementatie is. De tweede implementatie gaf ons veel betere resultaten. Echter hadden we hier een probleem met schaalbaarheid. Doordat iedere zin

van document 1 moet vergeleken worden met iedere zin uit document i (n documenten, $i \in 0..n$), duurt de uitvoering lang. Na toepassing van de selectieprocedure zoals beschreven in sectie 2.7.1, duurde het ongeveer 1 uur om de top 5 te bepalen voor een gegeven document, met gebruik van GPU-versnelling. We zouden de chunk-size van het algoritme kunnen verhogen, maar dat heeft dan weer grote invloed op het RAM-gebruik. Er moet dus een trade-off gemaakt worden.

3.6 Onze keuze

Indien er niet voldoende rekenkracht beschikbaar is, stellen wij voor om een combinatie van Word2Vec, Doc2Vec en de wettekstenregel uit sectie 2.7.1 te hanteren. Eens het model is opgesteld, kan er makkelijk iteratief aan toegevoegd worden. De bevragingen zijn snel en accuraat. We raden dus niet aan de snelle implementatie van BERT te gebruiken.

Indien tijd minder belangrijk is, of er voldoende rekenkracht is, stellen we voor de combinatie van Word2Vec, Doc2Vec, BERT en de wettekstenregel te gebruiken zoals aangegeven in dit verslag. Dit gaf ons de beste resultaten na manuele inspectie.

4 Conclusie

We zijn in staat om een relatief goed aanbevelingssysteem op te stellen voor juridische documenten. Onze implementatie is niet klaar om gebruikt te worden in de praktijk aangezien er daarvoor feedback nodig is van een jurist. Er is dus wel nog ruimte voor verbetering.

Het is ook nog niet mogelijk om op een lokale pc een rechtszaak in te vullen en dan onmiddellijk een top 5 terug te krijgen. We verwachten wel dat met een server-client-versie men dit programma relatief eenvoudig schaalbaar kan maken, indien de server enkele GPU's ter beschikking heeft. Een alternatief is dat men de dataset eerst manueel zou filteren tot een beperkte lijst van potentieel relevante zaken, en dan op die beperkte dataset het programma uitvoert. Echter verliezen we dan ook deels het nut van het programma.

4.1 Verbeteringen voor toekomstig werk

De pre-processing van de data kan uitgebreid worden: het moet mogelijk zijn om aan de hand van een whitelist en een jurist enkel juridisch relevante zinnen te filteren. Op die manier verlaagt de ruis in de dataset, en verhoogt dus de kwaliteit. Ook zouden er nog technieken kunnen toegepast worden om zinnen te abstraheren: specifieke namen, plaatsen en dergelijke vervangen door abstracte termen. Hiermee zou het makkelijker moeten zijn voor de algoritmes om verbanden te leggen tussen de informatie die wel belangrijk is.

De resultaten zouden nog accurater kunnen gemaakt worden indien er een model voor BERT kan opgesteld worden dat specifiek getraind is voor het gebruik met het Nederlandstalige juridisch lexicon. Aan de hand van een jurist kan ook bepaald worden welk algoritme er de beste resultaten heeft. Vervolgens kunnen dan de gewichten van ieder algoritme aangepast worden in de scoreberekening.

Referenties

[1] Chetan Ambi. Sentence embeddings with sentence-transformers library, 2020.

[2] Saeys Yvan (promotor) Commeine Bert. Artificiële intelligentie voor juridische toepassingen:vergelijkbaarheid van juridische documenten. Master's thesis, Ghent University, 2020.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert. https://github.com/google-research/bert, 2018.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[5] Prof. dr. Yvan Saeys. Natural language processing. https://ufora.ugent.be/d2l/le/content/234669/viewContent/640681/View, 2020.

[6] D. Greenberg. How can i split a text into sentences. https://stackoverflow.com/questions/4576077/how-can-i-split-a-text-into-sentences, 2015.

[7] Rani Horev. Bert explained: State of the art language model for nlp. https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270, 2018.

[8] De Rechtspraak. Open data van de rechtspraak. https://www.rechtspraak.nl/Uitspraken/paginas/open-data.aspx.

[9] Nils Reimers and Iryna Gurevych. Making monolingual sentence embeddings multilingual using knowledge distillation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2020.

[10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.